NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF CHEMICAL ENGINEERING

SECTOR IV: SYNTHESIS AND DEVELOPMENT OF INDUSTRIAL PROCESSES

Zographou Campus, Athens 157 80

LABORATORY OF POLYMER TECHNOLOGY

DIPLOMA THESIS

Development and application of machine learning models for the prediction of critical properties of dynamically crosslinked polymers

DANAI DIMITRA MORFI

SUPERVISORS:

STAMATINA VOUYIOUKA

MICHAEL KAVOUSANAKIS

ATHENS, 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ IV: ΣΥΝΘΕΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΔΙΚΑΣΙΩΝ

Πολυτεχνειούπολη Ζωγράφου, Αθήνα 157 80

ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΟΛΟΓΙΑΣ ΠΟΛΥΜΕΡΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη και εφαρμογή μοντέλων μηχανικής μάθησης για την πρόβλεψη κρίσιμων ιδιοτήτων δυναμικά διασταυρωμένων πολυμερών

ΔΑΝΑΗ ΔΗΜΗΤΡΑ ΜΟΡΦΗ

ΕΠΙΒΛΕΠΟΝΤΕΣ:

ΣΤΑΜΑΤΙΝΑ ΒΟΥΓΙΟΥΚΑ

ΜΙΧΑΗΛ ΚΑΒΟΥΣΑΝΑΚΗΣ

ΑΘΗΝΑ, 2025

Acknowledgements

I would like to express my gratitude to my family and friends that listened to every concern I had during this work and helped me recalibrate whenever I hit a mental wall. I would also like to thank my supervisors deeply, for never letting me wonder what my next steps should be, offering me support and ideas constantly. This past year was challenging, as well as rewarding, and this work was at the center of it. I wish you a pleasant reading experience, and I hope the following acts as an interesting introduction into the fascinating subject of Machine Learning and Polymer Science.

Abstract

The ongoing environmental crisis is driving scientists to develop “greener” polymers, either by adopting more sustainable materials or alternative chemistries. This effort led to the discovery of a fascinating group of dynamically crosslinked polymers, called vitrimers. Vitrimers bridge the gap between traditional thermosets and thermoplastics, combining excellent mechanical abilities with recyclability. To accelerate vitrimer research while minimizing experimental cost, this thesis explores the use of machine learning to predict four key properties: crosslink density, glass transition and topology freezing temperature and gel fraction. This was achieved through processing manually collected data from literature along their respective extracted chemical descriptors. Despite several difficulties during database curation, an experimental data set comprising of roughly 200 observations was created. Using SMILES notation for small molecules and PSMILES, a variation created for polymer repeating units, a variety of descriptors was extracted leveraging a popular cheminformatics package. After rigorous preprocessing- including feature filtering and elimination, handling of missing values, and normalization to achieve a standard value range- two dimensionality reduction methods were applied. First up was Principal Component Analysis (PCA) which involves combining extracted molecular descriptors and features into principal components, dramatically reducing model’s inputs. The second hybrid methodology involved combines Pearson correlation analysis with feature selection algorithms like Mutual Information and Random Forest (CA/FS). Model training was done after applying a 5-fold cross validation split into training and test sets, and validation was carried out through metrics like R^2, root mean squared error and mean absolute error highlighted the impact of both methodologies on results. To further investigate exhibited behavior, plots including parity plots, box plots and error distribution plots were created. More specifically, in T_g prediction Gradient Boosting and Random Forest models achieved cross-validated R^2 values of 0.85, confirming the effectiveness of ensemble-based methods. This also applied in the case of T_v where Random Forest reached an R^2 of 0.70. PCA performed poorly because it only captures linear relationships well, which was not the case for T_g and T_v. Contrastingly, PCA yielded better results when used to predict crosslink density, with the Random Forest model achieving an R^2 of 0.77, suggesting it captures broader structural trends. Prediction of gel fraction was unsuccessful, (R^2 < 0.40), indicating descriptors may not capture relevant information or the target is too variable. Finally, hyperparameter tuning, while an important step in machine learning pipelines, did not improve on results massively, except for the case of the Gaussian Process model whose R^2 values improved from -54.81 to 0.83 when predicting T_g. These findings demonstrate the promise and limitations of ML approaches in polymer informatics for emerging materials like vitrimers and highlight a way forward.

Key words: machine learning, vitrimers, glass transition temperature, topology freezing temperature, crosslink density, database curation, data processing, features, model performance, metrics, optimization

Extended Abstract (in Greek)

Τα πολυμερή αποτελούν ένα πολύ σημαντικό κομμάτι της ζωής των ανθρώπων, με πολλαπλές χρηστικότητες, αλλά και αρνητικές περιβαλλοντικές επιπτώσεις. Οι επιστήμονες, αναζητώντας τρόπους να βελτιώσουν αυτή την κατηγορία υλικών με τον ταχύτερο δυνατό τρόπο, έχουν στραφεί στην αξιοποίηση της μηχανικής μάθησης και της τεχνητής νοημοσύνης. Παρότι η χρήση τέτοιων μεθόδων έχει βρει μεγάλη εφαρμογή στον τομέα των απλών υλικών (Πληροφορική των υλικών-Material Informatics), η αξιοποίησή της για εξαγωγή χρήσιμων πληροφοριών, όπως η πρόβλεψη ιδιοτήτων για τα πολυμερή, βρίσκεται σε πιο πρώιμο στάδιο. Η έλλειψη μιας ενιαίας και συμπαγούς βάσης δεδομένων που να περιέχει πληροφορίες και ιδιότητες για τα πολυμερή, σε συνδυασμό με την ανάγκη εξαγωγής πολλαπλών περιγραφικών μεγεθών για την αναπαράσταση των πολυμερικών δομών, αποτελεί τροχοπέδη στην ανάπτυξη αξιόπιστων μοντέλων μηχανικής μάθησης. Μια άλλη σημαντική πρόκληση είναι η ίδια η φύση των πολυμερών, η οποία χαρακτηρίζεται ως στοχαστική, καθώς δεν μπορεί να οριστεί με τρόπο απόλυτο και ενιαίο. Η μορφή των πολυμερών, τα μεγέθη των διαφορετικών αλυσίδων, και κατ' επέκταση οι ιδιότητες εξαρτώνται πάντα από τις πειραματικές συνθήκες, τα χρησιμοποιούμενα κατά την σύνθεση υλικά, ακόμα και τις συνθήκες που επικρατούν κατά την μέτρηση των ιδιοτήτων. Γι' αυτό τον λόγο, υπάρχει μια εγγενής δυσκολία ως προς την αναπαράστασή τους με τρόπο αναγνώσιμο από τα μοντέλα μηχανικής μάθησης. Στα απλά μόρια, συνήθως γίνεται χρήση του συμβολισμού SMILES, μιας γραμμικής αναπαράστασης της δομής και των δεσμών που απαντώνται σε ένα μόριο. Στα πολυμερή, αυτός ο συμβολισμός δεν είναι άμεσα αξιοποιήσιμος και γι' αυτό έχουν διερευνηθεί τρόποι επέκτασης των SMILES. Στην παρούσα εργασία, χρησιμοποιήθηκε ο συμβολισμός PSMILES, που αναπαριστά την επαναλαμβανόμενη μονάδα ενός πολυμερούς, καθώς και τα σημεία πολυμερισμού.

Στην πορεία της έρευνας για την εύρεση βελτιωμένων πολυμερών, ανακαλύφθηκαν τα δίκτυα προσαρμοσμένων ομοιοπολικών δεσμών (Covalent Adaptable Networks-CANs), μέρος της οποίας είναι τα βιτριμερή που αποτελούν και το ερευνητικό αντικείμενο αυτής της διπλωματικής. Τα βιτριμερή είναι μια νέα κατηγορία πολυμερών που παρουσιάζουν δυναμικούς ομοιοπολικούς σταυροδεσμούς που εναλλάσσονται συνεχώς, ανταποκρινόμενοι στην ύπαρξη κάποιου ερεθίσματος όπως η θερμοκρασία, η αλλαγή στο pH και το φως. Η ειδοποιός διαφορά μεταξύ των βιτριμερών και των υπόλοιπων πολυμερών είναι ότι τα πρώτα ακολουθούν ένα προσεταιριστικό μονοπάτι (associative pathway) όταν σπάνε οι δεσμοί τους, το οποίο σημαίνει ότι προτού σπάσει ένας δεσμός σε ένα σημείο έχει δημιουργηθεί ένας άλλος σε διαφορετικό σημείο, διατηρώντας τον συνολικό αριθμό σταυροδεσμών σταθερό. Τα υπόλοιπα πολυμερή ακολουθούν την αποπροσεταιριστική οδό (dissociative pathway) και γι΄ αυτό δεν παρουσιάζουν τις ιδιαίτερες ικανότητες των βιτριμερών.

Τα βιτριμερή αποτελούν την γέφυρα μεταξύ θερμοσκληρυνόμενων και θερμοπλαστικών πολυμερών, εξαιτίας της δυναμικότητας των δεσμών τους. Όταν εφαρμόζεται σε αυτά κάποιο ερέθισμα ρέουν και παρουσιάζουν ικανότητες αυτό-ίασης και επεξεργασιμότητας και άρα ανακυκλωσιμότητας, όπως τα θερμοπλαστικά, ενώ σε κατάσταση ηρεμίας μοιάζουν με τα θερμοσκληρυνόμενα, αφού παρουσιάζουν ιδιότητες όπως μηχανική σταθερότητα και αντοχή. Έτσι, βρίσκουν εφαρμογή στον κλάδο της αεροναυπηγικής, του 3D printing, της ρομποτικής, της ιατρικής και πολλούς άλλους.

Η ιδιαίτερη ιξωδοελαστική συμπεριφορά των βιτριμερών πρέπει να διερευνηθεί υπό το πρίσμα δύο χαρακτηριστικών θερμοκρασιών: αυτή της υαλώδους μετάπτωσης (glass transition temperature- T_g) και αυτής κάτω από την οποία η τοπολογία του πολυμερικου δικτύου «παγώνει» (topology freezing temperature- T_v). Το T_g αποτελεί το σημείο όπου ένα πολυμερές παύει να είναι εύκαμπτο και γίνεται άκαμπτο, ενώ το T_v είναι η θερμοκρασία στην οποία οι δυναμικοί δεσμοί αρχίζουν να εναλλάσονται και το ιξώδες παίρνει την τιμή 10^{12} Pa·s, σύμφωνα με τη σχέση του Maxwell. Τα βιτριμερή παρουσιάζουν μεταβολή του ιξώδους τους με βάση τη θερμοκρασία, αρχικά σύμφωνα με το μοντέλο Williams-Landel-Ferry (μέχρι το T_v), και στη συνέχεια με εξάρτηση Arrhenius. Οι τιμές των δύο αυτών θερμοκρασιών καθορίζουν σε μεγάλο βαθμό τις ιδιότητες του τελικού υλικού. Το T_g επηρεάζεται από την πυκνότητα των σταυροδεσμών (crosslink density- v_e) και τη δομή των μονομερών, ενώ το T_v εξαρτάται από τον τύπο και τη συγκέντρωση του καταλύτη, την πυκνότητα των λειτουργικών ομάδων και τη χημεία των δυναμικών δεσμών. Οι υπόλοιπες ιδιότητες των βιτριμερών όπως η πυκνότητα σταυροδεσμών, κλάσμα πυκτώματος (gel fraction- *GF*), οι μηχανικές ιδιότητες, κτλ. ελέγχονται από παραμέτρους όπως η φύση των εμπλεκόμενων μορίων, οι συγκεντρώσεις, η θερμοκρασία αντίδρασης και πολλά άλλα.

Η παρασκευή των βιτριμερών μπορεί να γίνει μέσω πολυμερισμού πολυδραστικών μονομερών ή διασύνδεσης έτοιμων θερμοπλαστικών πολυμερών ή ακόμα και πρόσδοσης δραστικότητας σε ένα μακρομόριο αφού έχει ήδη πολυμεριστεί. Υπάρχει μια πληθώρα διαθέσιμων χημικών αντιδράσεων που μπορούν να αξιοποιηθούν για την παραγωγή βιτριμερών, με την αντίδραση μετεστεροποίησης να είναι από τις πιο συχνά χρησιμοποιούμενες.

Ο κλάδος της μηχανικής μάθησης έχει γνωρίσει μεγάλη άνθιση τα τελευταία χρόνια με την δημιουργία νέων και βελτιωμένων μοντέλων με πολλαπλές χρηστικότητες. Η εφαρμογή της στα πολυμερή μπορεί ελαττώσει την ανάγκη για χρονοβόρα και επαναλαμβανόμενα πειράματα, πράγμα ιδιαίτερα ελκυστικό για την επέκταση του γνωστικού πεδίου, ιδιαίτερα στα βιτριμερή όπου παρόμοια έρευνα βρίσκεται σε πολύ πρώιμο στάδιο, με ελάχιστους ερευνητές να ασχολούνται ενεργά. Τα μοντέλα μηχανικής μάθησης χωρίζονται σε δύο βασικές κατηγορίες: επιβλεπόμενη

(supervised) και μη επιβλεπόμενη (unsupervised). Η επιβλεπόμενη μάθηση χωρίζεται σε μεθόδους παλινδρόμησης (regression) για την πρόβλεψη αριθμητικών τιμών μεταβλητών και ταξινόμηση (classification) που διαχειρίζεται διακριτές κατηγορικές μεταβλητές και βασίζεται σε δεδομένα που φέρουν επισήμανση (εξαρτημένες μεταβλητές) για να κάνει προβλέψεις και να διερευνήσει συνδέσεις. Η μάθηση χωρίς επίβλεψη, αντίθετα, ασχολείται με μη επισημασμένα δεδομένα για να αποκαλύψει κρυφές συνδέσεις (ομαδοποίηση) ή να μειώσει τις διαστάσεις του προβλήματος εξαλείφοντας τις περιττές μεταβλητές (μείωση διαστάσεων). Στην παρούσα εργασία εξετάζονται τα μοντέλα παλινδρόμησης, μιας και στόχος είναι η πρόβλεψη της τιμής μιας επιθυμητής ιδιότητας. Πιο συγκεκριμένα, διερευνώνται τα εξής μοντέλα: Linear regression, Random Forest, Support Vector Machine, Gaussian Process Regression, Gradient Boosting και eXtreme Gradient Boosting.

Μια κλασική ροή εργασιών μηχανικής μάθησης περιλαμβάνει τα ακόλουθα βήματα: συλλογή και επεξεργασία δεδομένων, εξαγωγή περιγραφικών μεγεθών και δημιουργία δακτυλικού αποτυπώματος (fingerprinting), επιλογή και εκπαίδευση μοντέλων και τέλος εκτίμηση απόδοσης μοντέλων και συγκριτική τους αξιολόγηση (benchmarking).

Είναι, επομένως, φανερό πως, ενώ η χημεία και η σύνθεση των βιτριμερών αποτελούν ένα πολυπαραγοντικό πρόβλημα, η αξιοποίηση μοντέλων μηχανικής μάθησης μπορεί να συμβάλει ουσιαστικά στη μελλοντική έρευνα. Με τον τρόπο αυτό μειώνεται η εξάρτηση από χρονοβόρους, δαπανηρούς και επαναλαμβανόμενους πειραματικούς κύκλους. Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας ολοκληρωμένης ροής εργασίας μηχανικής μάθησης, μέσω της οποίας θα εκπαιδευτούν μοντέλα ικανά να προβλέπουν τέσσερις κρίσιμες ιδιότητες των βιτριμερών: την πυκνότητα σταυροδεσμών (ν_e), τη θερμοκρασία υαλώδους μετάπτωσης (T_g), τη θερμοκρασία κάτω από την οποία η τοπολογία του πολυμερικου δικτύου «παγώνει» (T_v) και το κλάσμα πηκτώματος (*GF*).

Στην αρχή, δημιουργήθηκε μια πειραματική βάση δεδομένων με στοιχεία που συλλέχθηκαν από περίπου 30 άρθρα από αξιόπιστες πηγές. Σε αυτό το στάδιο αντιμετωπίστηκαν οι μεγαλύτερες δυσκολίες, καθώς η κάθε ερευνητική ομάδα επικεντρωνόταν σε διαφορετικές ιδιότητες του τελικού προϊόντος και παρουσίαζε τα στοιχεία σχετικά με την σύνθεση του διαφορετικά. Μετά από προσεκτικούς υπολογισμούς, η τελική βάση αριθμούσε περίπου 200 παρατηρήσεις, δηλαδή διαφορετικά παραδείγματα συνθέσεων, δοσμένα με ενιαίο τρόπο και με τις λιγότερες δυνατές ελλείψεις. Μερικά παραδείγματα ιδιοτήτων που συλλέχθηκαν, εκτός από τις ποσότητες και την φύση των ενώσεων και των καταλυτών όπως και των τιμών των υπό διερεύνηση ιδιοτήτων που ήταν απαραίτητες για την εκπαίδευση των μοντέλων, ήταν μηχανικές ιδιότητες (μέτρο ελαστικότητας και αντοχή σε εφελκυσμό), ενέργεια ενεργοποίησης, χρόνος χαλάρωσης και άλλα.

Το επόμενο βήμα ήταν η συλλογή των αναγνωριστικών SMILES για τις διάφορες ενώσεις μέσω βάσεων δεδομένων, σαν το PubChem, και σχεδιαστικών εργαλείων, όπως το MolView. Χρησιμοποιώντας τα SMILES σαν δεδομένο εισόδου έγινε εξαγωγή χημικών περιγραφικών μεγεθών (descriptors) μέσω του δημοφιλούς προγραμματιστικού πακέτου RDKit. Η διαδικασία αυτή είναι γνωστή ως fingerprinting και περιλαμβάνει την εύρεση ενός τρόπου να αναπαρασταθεί η χημική δομή ενός μορίου με τρόπο αναγνωρίσιμο από μια γλώσσα προγραμματισμού όπως η Python. Με τον τρόπο αυτό δημιουργείται ένα αριθμητικό διάνυσμα που περιλαμβάνει περιγραφείς διαφόρων τάξεων μεγέθους, το οποίο χρησιμοποιείται σαν είσοδος (ανεξάρτητη μεταβλητή) κατά την εκπαίδευση των μοντέλων. Οι περιγραφείς χωρίζονται σε μιας, δύο, τριών ή πολλαπλών διαστάσεων, με την πολυπλοκότητά αλλά και την περιγραφικότητά τους να βαίνει αυξανόμενη. Το δακτυλικό αποτύπωμα που δημιουργείται λαμβάνοντας δεδομένα από το RDKit χαρακτηρίζεται ως περιγραφικό αποτύπωμα (descriptor- based fingerprint).

Στην συνέχεια εφαρμόστηκε ένας καθαρισμός τόσο στα συλλεχθέντα δεδομένα όσο και στους περιγραφείς, ώστε να εξαλειφθούν τυχόν μη περιγραφικά και ασήμαντα μεγέθη. Στην αρχή, ως εναλλακτική στην ανίχνευση και αφαίρεση ακραίων τιμών που θεωρήθηκε ότι θα ελάττωνε περαιτέρω το μέγεθος της ήδη μικρής βάσεως δεδομένων και θα εξαφάνιζε ενδεχομένως χρήσιμες πληροφορίες, εφαρμόστηκαν δύο φίλτρα. Το πρώτο φίλτρο εξάλειφε ιδιότητες με μικρή μεταβλητότητα, οι οποίες δεν θα χρησίμευαν στην διαδικασία της εκπαίδευσης των μοντέλων, και το δεύτερο ιδιότητες που περιείχαν πολλά μηδενικά που υποδήλωναν την έλλειψη αυτής της ιδιότητας. Με αυτό τον τρόπο ελαττώθηκαν σημαντικά οι διαθέσιμες ιδιότητες. Έπειτα, τα ελλιπή δεδομένα αντιμετωπίστηκαν με την αντικατάστασή τους με τη μέση τιμή της αντίστοιχης ιδιότητας ή με μηδενική τιμή, αναλόγως αν φανέρωναν απουσία μέτρησης μιας ιδιότητας ή απουσία της ιδιότητας συνολικά. Τέλος, εφαρμόστηκε κανονικοποίηση όλων των ιδιοτήτων ώστε να κυμαίνονται οι τιμές τους από -1 έως +1 και να μην επηρεάσει το εύρος τους λανθασμένα τα μοντέλα μηχανικής μάθησης.

Έπειτα, δοκιμάστηκαν δύο διαφορετικές μέθοδοι ελάττωσης των διαστάσεων του προβλήματος, διαδικασία απαραίτητη για να μην οδηγηθούν τα μοντέλα σε υπερπροσαρμογή (overfitting), δηλαδή στην εκμάθηση των δεδομένων εκπαίδευσης και την αποτυχία πρόβλεψης άγνωστων δεδομένων. Η πρώτη μέθοδος ήταν αυτή της ανάλυσης των κύριων συνιστωσών (Principal Component Analysis- PCA) που μπορεί να συνδυάσει εκατοντάδες χαρακτηριστικά (features) σε ένα μικρό αριθμό μεταβλητών (3-4 στην εργασία αυτή). Η δεύτερη υβριδική μέθοδος που εφαρμόστηκε ήταν ένας συνδυασμός της ανάλυσης συσχέτισης Pearson για την απόρριψη γραμμικά συσχετιζόμενων χαρακτηριστικών με αλγορίθμους επιλογής χαρακτηριστικών, όπως οι αλγόριθμοι Mutual Information και Random Forest, οι

οποίοι ανιχνεύουν πιο σύνθετες και λεπτές σχέσεις μεταξύ της μεταβλητής-στόχου και των χαρακτηριστικών. Κατά μέσο όρο, τα τελικά χαρακτηριστικά που χρησιμοποιήθηκαν για την εκπαίδευση των μοντέλων ήταν 4 για την πρώτη και 16 για την δεύτερη μέθοδο. Ο λόγος που δοκιμάστηκαν δύο μέθοδοι ήταν για να συγκριθεί η αποτελεσματικότητά τους κατά την εκπαίδευση των μοντέλων, μιας και οι αρχές λειτουργίας τους είναι πολύ διαφορετικές.

Το τελευταίο βήμα αυτής της ροής εργασίας ήταν ο χωρισμός της τελικά διαμορφωμένης βάσης δεδομένων σε σετ εκπαίδευσης (training) και σετ δοκιμών (test), το οποίο έγινε μέσω της αξιόπιστης μεθόδου της πενταπλής διασταυρούμενης επικύρωσης (5-fold cross validation) που δημιουργεί 5 μικρότερα σετ και εκπαιδεύει τα μοντέλα σε καθένα από αυτά ελέγχοντας την απόδοσή τους με τα υπολειπόμενα δεδομένα. Η διαδικασία αυτή είναι ιδιαίτερα χρήσιμη σε μικρές βάσεις δεδομένων για να αποφευχθεί ο κίνδυνος της υπερπροσαρμογής. Κατά την αρχική εκπαίδευση των μοντέλων που αναφέρθηκαν νωρίτερα, επιλέχθηκαν τυπικές τιμές για τις υπερπαραμέτρους (hyperparameters) των μοντέλων μέσω της βιβλιοθήκης scikit της γλώσσας Python. Η επιλογή κατάλληλων υπερ-παραμέτρων γενικά είναι κρίσιμη για να επιτευχθεί η βέλτιστη προβλεπτική ικανότητα των μοντέλων. Αφότου εκπαιδεύτηκαν τα πρώτα μοντέλα, έγινε προσπάθεια ρύθμισης (tuning) των παραμέτρων μέσω της μεθόδου αναζήτησης πλέγματος (grid search method) που αναζητά τις βέλτιστες τιμές των ορισμένων παραμέτρων.

Τα αποτελέσματα αξιολογήθηκαν μέσω των δεικτών R^2, RMSE και MAE, ενώ χρησιμοποιήθηκαν parity plots, error distribution plots και box plots για την οπτική ανάλυση των επιδόσεων. Για την πρόβλεψη της πυκνότητας διακλαδώσεων (ν_e), το «προεπιλεγμένο» (default) Random Forest πέτυχε την καλύτερη απόδοση με R^2 = 0.77 και RMSE = 0.08. Το tuning επηρέασε αρνητικά το ίδιο μοντέλο, μειώνοντας την τιμή του R^2 σε 0.74. Παρόλα αυτά, άλλα μοντέλα όπως το Gaussian Process παρουσίασαν σημαντική βελτίωση, φτάνοντας από την πλήρη αποτυχία με αρνητική τιμή στο R^2 (R^2 = -54.81) σε R^2 = 0.51. Στην περίπτωση του T_g, το Gradient Boosting ξεχώρισε με R^2 = 0.85 και RMSE = 0.17, ενώ το Random Forest και το XGBoost ακολούθησαν με R^2 = 0.85 και R^2 = 0.83, αντίστοιχα. Παρόλ' αυτά, το XGBoost παρουσίαζε μια πιο ασταθή συμπεριφορά λόγω αυξημένης τυπικής απόκλισης. Το tuning είχε θετικό αντίκτυπο σχεδόν σε όλους τους αλγορίθμους, ενισχύοντας τη σταθερότητα και την προβλεπτική ικανότητα των μοντέλων. Για το T_v, τα καλύτερα αποτελέσματα προήλθαν και πάλι από το Random Forest (R^2 = 0.70), ενώ το XGBoost με tuning κατέγραψε παρόμοια απόδοση με R^2 = 0.70 και RMSE = 0.17. Ωστόσο, παρατηρήθηκε ότι τα μοντέλα προ- ρύθμισης εμφάνισαν συχνά καλύτερη σταθερότητα (χαμηλότερη τυπική απόκλιση), κάτι που υποδηλώνει πως οι τυπικές παράμετροι ήταν σε αρκετές περιπτώσεις επαρκώς ισορροπημένες για το συγκεκριμένο σετ δεδομένων. Αντιθέτως, η πρόβλεψη του *GF* απέτυχε συνολικά, με κανένα μοντέλο

να ξεπερνά το R^2 = 0.40, ενώ το καλύτερο αποτέλεσμα επιτεύχθηκε με Support Vector Machine (R^2 = 0.37), αλλά συνοδευόταν από μεγάλο σφάλμα και αστάθεια. Η αστοχία αυτή αποδόθηκε στην έλλειψη ομοιομορφίας στα πειραματικά δεδομένα και στη διαδικασία μέτρησης του *GF*. Τα parity plots που δημιουργήθηκαν για κάθε στόχο ανέδειξαν το επίπεδο προβλεπτικής ικανότητας κάθε μοντέλου, με τα καλύτερα να παρουσιάζουν τιμές κοντά στη διαγώνιο, ενώ τα error distribution plots πρόσφεραν πληροφορίες για την ύπαρξη προκατάληψης του μοντέλου ή ακραίων τιμών στα αποτελέσματα. Τέλος, τα box plots αποκάλυψαν σημαντικές διακυμάνσεις στις αποδόσεις ανάμεσα στα μοντέλα και στις επαναλήψεις του cross-validation, αναδεικνύοντας τον αντίκτυπο της ποιότητας των δεδομένων και της επιλογής χαρακτηριστικών.

Σε ό,τι αφορά τις μεθόδους μείωσης των διαστάσεων, το PCA βελτίωσε την απόδοση των μοντέλων μόνο στην περίπτωση του v_e που πιθανά ορίζεται από μεγαλύτερες δομικές τάσεις που αντιλαμβάνεται το μοντέλο, ενώ απέτυχε για τις υπόλοιπες τρεις ιδιότητες, μιας και συμπίεσε πιθανά χρήσιμα και περιγραφικά μεγέθη στα 4 τελικά χαρακτηριστικά. Μελλοντικά, θα ήταν πολύ χρήσιμο να επεκταθεί σημαντικά η παρούσα βάση δεδομένων ώστε να βελτιωθούν οι προβλέψεις των μεταβλητών-στόχων. Συνολικά, όμως, η παρούσα εργασία αναδεικνύει την χρησιμότητα και τους περιορισμούς των μοντέλων μηχανικής μάθησης και μπορεί να χρησιμοποιηθεί σαν βάση για περαιτέρω έρευνα.

Table of contents

Acknowledgements 2
Abstract 3
Extended Abstract (in Greek) 5
Table of contents 11
List of tables 13
List of figures 14
Chapter 1: Introduction 17
1.1 Leveraging Machine Learning in Polymer Science 17
1.2 Scope of work 18
Chapter 2: Theoretical background 20
2.1 Vitrimers 20
2.1.1 The emergence of a new class of polymers 20
2.1.2 Vitrimer characteristics 21
2.1.3 Basics of vitrimer chemistry 23
2.1.4 Vitrimer recycling strategies and general applications 25
2.2 Machine Learning 26
2.2.1 Categorization of Machine Learning Applications 26
2.2.2 Overview of machine learning pipeline 28
2.2.3 Fingerprinting methods 30
2.2.4 Algorithms applied in ML pipelines 32
Chapter 3: Methods and Materials 40
3.1 Setting the goal of the ML model 40
3.2 Database curation 40
3.2.1 Data collection 40
3.2.2 Necessary calculations 42
3.2.3 Translation of chemical structure through SMILES 44
3.3 Feature extraction 46
3.4 Data processing 47

3.4.1 Outlier detection and data cleaning 47
3.4.2 Handling of missing values 49
3.4.3 Data normalization 50
3.4.4 Dimensionality reduction 53
3.5 Model training 56
3.5.1 Data splitting and composition 56
3.5.2 Model choice and hyperparameter tuning 57
3.6 Model validation and benchmarking 59
Chapter 4: Results and Discussion 60
4.1 Data processing and feature selection 60
4.1.1 Outlier detection 60
4.1.2 Dimensionality reduction 60
4.2 Model performance results 67
4.2.1 Comparison between dimensionality reduction methods 67
4.2.2 Investigation of hyperparameter tuning on model performance 83
Chapter 5: Conclusions and Future Work 93
References 97
Appendix 105

List of tables

Table 1. List of variables dropped for different cases of target variables. 53
Table 2. Summary of threshold values and resulting features during dimensionality reduction..... 56
Table 3. Models trained with their respective python functions. 57
Table 4. Ranges of hyperparameters used during grid search for all models. 58
Table 5. Features chosen for v_e by MI (left) and RF (right), ranked by importance with their respective scores. 63
Table 6. Summary of threshold values and resulting features during dimensionality reduction..... 65
Table 7. Summary table of final features for model training selected for all variables. 65
Table 8. Mean performance table with metrics for v_e. 69
Table 9. Mean performance table with metrics for T_g. 73
Table 10. Mean performance table with metrics for T_v. 77
Table 11. Mean performance table with metrics for GF. 80
Table 12. Most common parameters chosen across cross validations by the grid search method for v_e. 84
Table 13. Mean performance table with metrics for v_e with hyperparameter tuning and default values. 84
Table 14. Most common parameters chosen across cross validations by the grid search method for T_g. 87
Table 15. Mean performance table with metrics for T_g with hyperparameter tuning and default values. 87
Table 16. Most common parameters chosen across cross validations by the grid search method for T_v. 90
Table 17. Mean performance table with metrics for T_v with hyperparameter tuning and default values. 91

List of figures

Figure 1. Relationship between vitrimers, thermoplastics and thermosets [7]. 20
Figure 2. Difference between dissociative and associative mechanisms [11]. 21
Figure 3. Vitrimer behavior relating to temperature [7]. 23
Figure 4. Exchange reactions used in vitrimer synthesis [12]. 24
Figure 5. Schematic representation of ML categories [2]. 27
Figure 6. Common ML pipeline [3]. 29
Figure 7. Fingerprinting methods: the example of polystyrene [2]. 30
Figure 8. Schematic of a support vector machine model [2]. 34
Figure 9. Schematic of a Decision Tree (left) and a Random Forest (right) [2]. 36
Figure 10. Illustration of BigSMILES [5]. 45
Figure 11. Histogram of Kappa3. 49
Figure 12. Histogram of reaction temperature T (oC). 51
Figure 13. Histogram of Chi0. 52
Figure 14. PCA heatmap of principal components for v_e. 61
Figure 15. PCA heatmap of principal components for T_g. 61
Figure 16. PCA heatmap of principal components for T_v. 62
Figure 17. PCA heatmap of principal components for GF. 62
Figure 18. Model performance comparison (mean+std) for v_e with PCA methodology. 70
Figure 19. Model performance comparison (mean+std) for v_e with CA/FS methodology. 70
Figure 20. Parity plots for v_e with a) Random Forest (left) and b) Gradient Boosting (right) models. 71
Figure 21. Error distribution plot for v_e with Random Forest model. 72
Figure 22. Model performance comparison (mean+std) for T_g with PCA methodology. 74
Figure 23. Model performance comparison (mean+std) for T_g with CA/FS methodology. 74
Figure 24. Parity plots for T_g with a) Random Forest (left) and b) Gradient Boosting (right) models. 75
Figure 25. Error distribution plot for T_g with Gradient Boosting model. 76
Figure 26. Model performance comparison (mean+std) for T_v with PCA methodology. 77
Figure 27. Model performance comparison (mean+std) for T_v with CA/FS methodology. 78
Figure 28. Parity plots for T_v with a) Random Forest (left) and b) Gradient Boosting (right) models. 79
Figure 29. Error distribution plot for T_v with Random Forest model. 80

Figure 30. Model performance comparison (mean+std) for GF with PCA methodology. 81
Figure 31. Model performance comparison (mean+std) for GF with CA/FS methodology. 81
Figure 32. Parity plots for GF with a) PCA Random Forest (left) and b) CA/FS Support Vector Machine (right) models. 82
Figure 33. Error distribution plot for GF with Support Vector Machine model. 83
Figure 34. Model performance comparison (mean+std) for v_e with hyperparameter tuning. 85
Figure 35. Parity plots for v_e with a) default Random Forest (left) and b) tuned Gradient Boosting (right) models. 86
Figure 36. Model performance comparison (mean+std) for T_g with hyperparameter tuning. 88
Figure 37. Parity plots for T_g with a) Gradient Boosting (left) and b) XGBoost (right) models. 89
Figure 38. Error distribution plot for T_g with Gradient Boosting model. 90
Figure 39. Model performance comparison (mean+std) for T_v with hyperparameter tuning. 91

Abbreviations used:

ML:	Machine Learning
AI:	Artificial Intelligence
CANs:	Covalent Adaptable Networks
RF:	Random Forest
SVM:	Support Vector Machines
NN:	Neural Network
QSAR:	Quantitative structure–activity relationship
QSPR:	Quantitative structure–property relationship
GPR:	Gaussian Process Regressor
MFs:	Morgan Fingerprints
GBM:	Gradient Boosting Machines
XGBoost:	EXtreme Gradient Boosting
PCA:	Principal Component Analysis
MI:	Mutual Information
MSE:	Mean Squared Error
RMSE:	Root Mean Squared Error
MAE:	Mean Absolute Error

Chapter 1: Introduction

1.1 Leveraging Machine Learning in Polymer Science

Polymers are a class of materials that made their way into the everyday life of most, if not all, humans. They are used across many different sectors such as packaging, construction, and even medicine. Nevertheless, it is no secret that most conventional polymers are a burden for the environment since they decompose very slowly and tend to accumulate on land or in the sea in vast amounts, disrupting the ecosystem. It is critical for researchers to find new and "green" ways to manufacture polymers or even discover new sustainable ones altogether, which is a very labor-intensive and costly endeavor. As a result, alternative research paths have come into play, such as integrating the rapidly growing field of computer science, machine learning (ML) and Artificial Intelligence (AI) into the material science one. This can create new pathways for exploring novel materials without the need for extensive experimental procedures, since through ML, one can predict properties and behavior of a material through existing data and trained algorithms.

Leveraging the power of ML in material science is not a novel concept, which is why a separate field called Material Informatics was created years ago. However, the use of it in polymer science specifically is at a somewhat preliminary stage. That is because a unified polymer properties database has not been curated just yet, and while there are several, none of them encompass all of the properties one might be interested in researching and/or targeting through experiments. At the same time, the data across those databases is not presented in a unified way. That means that for someone to access enough data necessary to train a ML model properly, they must go through the data manually and even alter some for uniformity reasons. Adding to all of this, while researchers have gathered a lot of empirical data on polymers, much of that information is not digitized and can only be found in books or is under proprietary rights, which makes the task of gathering enough data points all the more difficult [1]-[3]. Small data samples paired with the need for a vast amount of features representative of the polymers come with problems such as model overfitting. That is where the model learns the training data well and performs excellently on them, but when exposed to fresh data performs very poorly; in other words, it fails to generalize. At the same time, small datasets come at the cost of model accuracy [2],[4].

It is also important to note that polymer properties can be greatly influenced by the processing step in their manufacturing procedure, the measuring methods and conditions. Additionally, polymer properties are often nonlinearly related, which adds another layer of complexity to the problem at hand [3].

Another hurdle in the application of ML Pipelines in the case of polymers is the fact that they are intrinsically stochastic structures, which is to say, they do not have a well-defined structure and therefore do not follow the same rules as simple molecules as far as their representation goes. Simple molecules (like atoms, organic and inorganic compounds, *etc*.) can be represented using SMILES notation. SMILES is a type of language used by Material Informatics to "translate" the chemical structure of a molecule into a machine-readable format. It is quite popular due to the simple "grammatical" rules it adheres to as well as the fact that it is easily interpretable by humans. SMILES, however, are not directly applicable to polymers, since the length and even chemistry of their macromolecular chains vary. Some chains can be perfectly linear, while others can have branches or be shorter due to side reactions. There have been several alternatives proposed, one of them being BigSMILES, which is an evolution of SMILES that encodes an extra structure into the representation and PSMILES, which adds some extra rules as far as SMILES "grammar" goes [5],[6]. These approaches, alongside their advantages and disadvantages, will be covered later. However, it is important to highlight that there is still no one solution that is being used across the field of Polymer Informatics.

1.2 Scope of work

In recent years, scientists have made some significant breakthroughs in polymer research, with or without leveraging computer science. One example of this is vitrimers, a sub-category of a larger class of polymeric materials called CANs, *i.e*., Covalent Adaptable Networks. The name "vitrimers" was given to them by Leibler *et al*., who noticed their viscoelastic behavior was similar to vitreous silica. These materials are characterized by covalent crosslinks that present reversible, dynamic behavior when exposed to stimuli like heat, light, pressure or pH change. The main difference between vitrimers and other dynamic networks is the nature of the exchange reactions. In the former, bond exchange occurs *via* an associative pathway rather than a dissociative one. This way, there is no distinction between breaking and reconnecting of the bonds, therefore maintaining the overall density of the network stable, which in turn grants them with unique abilities. It should be noted that all polymers with dynamic covalent bonds, either dissociative or associative, are considered CANs [7],[8].

The chemistry behind vitrimers is complex and involves a lot of different variables, such as the choice of compounding molecules and catalysts that will make up the final product as well as their respective amounts. Catalysts are not always necessary, but in many cases, they are used, since they aid in the crosslinking process and grant the final product with dynamicity. There is also the matter of experimental conditions; temperature, mixing and curing time, *etc*., as well as measuring conditions, for example,

in mechanical and rheological tests. It is also critical to emphasize that there are few, if any, robust mathematical expressions that can calculate key vitrimer properties like crosslink density, which characterizes the density of the polymeric network, and topological freezing temperature (T_v), which is a temperature tied to their viscoelastic behavior, distinct from the typical glass transition temperature (T_g). The limited availability of standardized data on this area makes leveraging Machine Learning for vitrimer research interesting but adds yet another layer of difficulty compared to traditional Polymer Informatics. To date, there have been very few reports of such an endeavor, making this work novel. Thus, an effort will be made to present the framework behind this work as thoroughly as possible.

Therefore, this thesis aims to explore and implement an ML scheme for vitrimers, in order to predict critical properties such as crosslink density, gel fraction and thermal transitions (T_v, T_g). A detailed analysis will be presented, and all the necessary steps in training and deploying the model will be laid out. By doing so, this work aspires to contribute to bridging the research gap in vitrimers and laying a solid foundation for future developments.

Chapter 2: Theoretical background

2.1 Vitrimers

2.1.1 The emergence of a new class of polymers

Conventionally, polymers are divided into two broad classes based on their structure; thermoplastics and thermosets. Thermoplastics are non-crosslinked linear or branched polymers, able to flow under heating, which allows them to be reprocessed many times as well as recycled without the heat adversely affecting their properties. On the other hand, thermosets cannot be reprocessed and therefore recycled due to the permanent crosslinks that are formed between chains during the curing process. Once cured, thermosets cannot take a different shape like thermoplastics. This downside comes with significant upsides such as excellent mechanical properties, stability when exposed to heat and chemicals, and structural integrity. Vitrimers, in a way, form a bridge between the two and can, therefore, be considered a third category, adding to the previous ones. That is because their covalent bonds present a reversible and dynamic behavior when exposed to stimuli and allow vitrimers to act as both thermoplastics and thermosets. When a stimulus is applied (most commonly heat), vitrimers can flow, self-heal and return to their original shape, a property known as shape memory, which is what enables them to be repurposed and recycled, similarly to thermoplastic polymers. At service temperatures, the crosslinks are stable, granting vitrimers mechanical stability, durability and other properties usually found in thermoset polymers. The reason behind vitrimers responding to certain stimuli is the continuous exchange between links, which allows chains to rearrange themselves in the network. These properties allow them to be used in a plethora of applications like robotics and 3D printing [8]-[11]. Their unique properties can be seen in Figure 1.

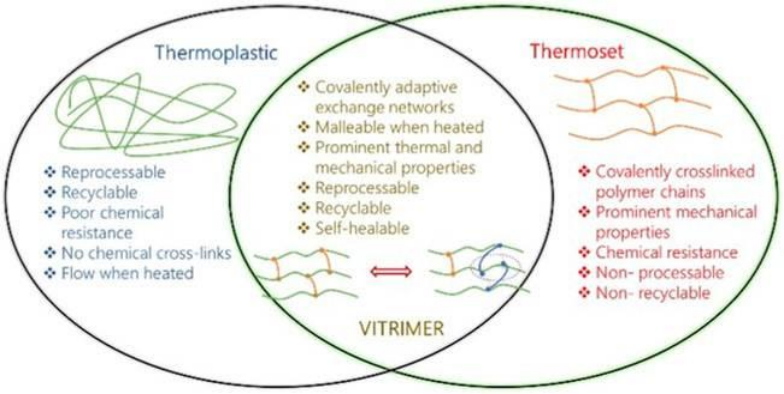


Figure 1. Relationship between vitrimers, thermoplastics and thermosets [7].

As previously mentioned, the essential difference between classic polymers and/or dissociative CANs and vitrimers lies in the way bonds are exchanged in their networks, and specifically, the fact that the latter follow an associative pathway. This means that when bond exchange happens, old crosslinks break only when new ones have already formed, which keeps the total number of crosslinks stable at all times. Following the dissociative pathway, bonds break and reappear in another place, creating two separate steps in the process [11], as seen in Figure 2.

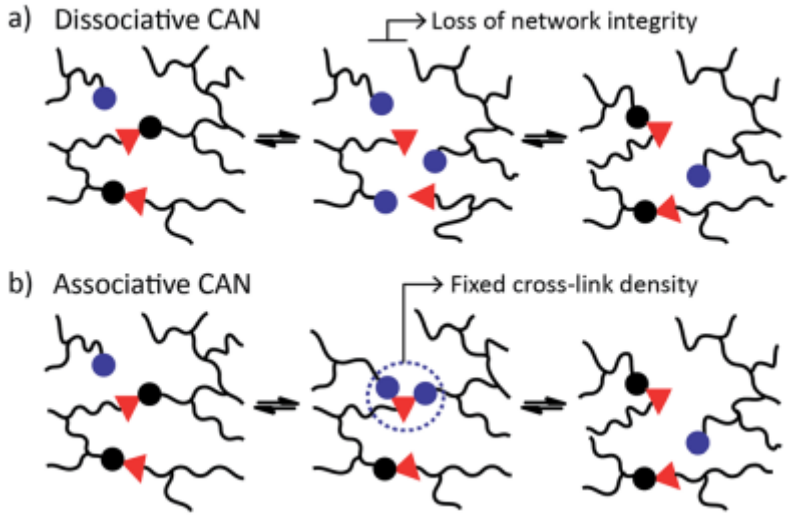


Figure 2. Difference between dissociative and associative mechanisms [11].

2.1.2 Vitrimer characteristics

In order to properly investigate vitrimer viscoelastic behavior, it is crucial to differentiate between important temperatures like T_g, melting temperature (T_m), and T_v. In classic polymers, T_g represents the temperature below which they act like glassy solids. Amorphous and semi-crystalline polymers act as viscoelastic solids above the T_g and T_m, respectively. Moreover, thermoplastics flow for temperatures above the T_g for amorphous and T_m for semi-crystalline ones. A sudden drop in viscosity is noted above the T_g according to the Williams, Landel, and Ferry (WLF) model [10],[11]. T_v is a temperature that describes the effect that topological rearrangements have on a material on a macroscopic level. It is defined as the temperature where dynamic bonds become reversible and viscosity takes the value of 10^{12} Pa·s. That number is based on the Maxwell relation

$$\eta = G\,\tau^* \tag{2.1}$$

where, η (Pa·s) stands for viscosity, G (Pa) for shear modulus, and τ^* (s) for relaxation time. G can be estimated using the following relationship:

$$G = \frac{E'}{2\,(1+\nu)} \sim \frac{E'}{3} \tag{2.2}$$

where, ν is Poisson's ratio (usually equal to 0.5 for rubbers) and E' (Pa) is the tensile modulus. T_v can be measured experimentally by dilatometry and stress relaxation tests like oscillatory shear rheology. It is a crucial vitrimer property because it denotes the moment where the material shifts from viscoelastic solid to viscoelastic liquid [10],[12].

Another important characteristic of vitrimers is their crosslink density, in other words the density of the crosslinks in the polymeric network, which can be calculated through the following equation

$$\nu_e = \frac{E'}{3R(T_g+40)} \tag{2.3}$$

where, E' stands for the storage modulus in the rubbery plateau (MPa) and R for the universal gas constant. Crosslink density is linked to gel point, where the crosslink density is sufficient enough to assume the presence of an infinite molecular weight network [8].

As far as vitrimers' viscosity in relation to heat, it is governed by the relationship between the material's T_v and T_g, with the latter being analogous to that of classic polymers. More often than not, T_g is lower than T_v in vitrimers. They, like other polymers, transition from rubbery to glassy state at T_g. In the beginning, viscosity is described by the WLF model, and for temperatures above T_v, viscosity seems to follow an Arrhenius dependence on temperature (silica-like). In the temperature range between T_g and T_v, bond exchange is so slow that the network is assumed to be fixed, and only after being heated past T_v, the speed of the exchange becomes quantifiable and causes the material to behave like a viscoelastic liquid. This can be observed for both amorphous and semi-crystalline vitrimers, with the melting temperature typically lying between T_g and T_v. It is also sometimes possible for T_v to be lower than T_g, in which case no chain movements are observed until vitrimers are heated past T_g. That is when the rigid glassy material becomes a viscoelastic liquid, first controlled by diffusion, *i.e.*, the WLF model, and then by exchange rate, *i.e.*, the Arrhenius model. In this case, the value of T_g must be extrapolated from creep or stress-relaxation experiments, as there is no way to directly calculate it [9],[10],[12],[13]. This interesting behavior can be observed in Figure 3.

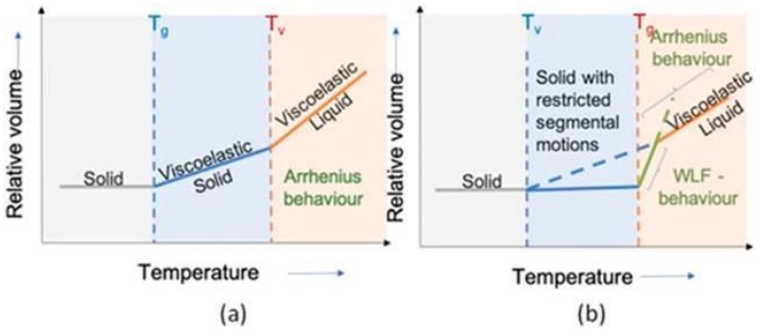


Figure 3. Vitrimer behavior relating to temperature [7].

The importance of T_g and T_v lies in their ability to define the properties of the final product, and it is therefore critical to tune them properly. T_g can be controlled through parameters like crosslink density and rigidity of the monomers, while T_v is influenced by catalyst type and its concentration, dynamic bonds used, and density of functional groups in the monomers. More specifically, increasing catalyst concentration leads to lower T_v values, while the nature of the catalyst affects the activation energy of the exchange reactions directly. It is important to note that crosslink density and stability of the network can be dictated by the percentage of permanent crosslinks, more stable dynamic chemistries, and weaker links (like hydrogen bonding), which, in turn, is a direct result of the choice of monomers. At the same time, the parameters mentioned above, including the stoichiometry of the monomers, can affect the reversibility of the dynamic bonds. Stress relaxation and creep can be manipulated through catalyst concentration since it controls the kinetics behind the exchange reactions and mechanical properties, in general, are reliant on the same parameters as T_g. In reality, vitrimers can fully relax stress because of their dynamic behavior, except for temperatures below T_g where polymeric chains can move slightly, if at all, and thus prevent stress relaxation. Above T_g, stress relaxation is dependent on the kinetics of the exchange reaction and, therefore, can be described by the Arrhenius law [8],[12],[13].

2.1.3 Basics of vitrimer chemistry

Two main synthetic routes can be taken to create vitrimers: polymerizing monomers that have multifunctionality or crosslinking already formed thermoplastics. On the other hand, another possible route is functionalizing the material after it has polymerized, which is especially interesting in the case of existing polymers. There is an abundance of reactions that can be employed to synthesize polymers, which can be reviewed in Figure 4. In the current work, vitrimer chemistry will not be thoroughly presented since it does not have a direct impact on the ML model training process. Nevertheless, some groups of molecules and their corresponding dynamic exchange reactions commonly found in the bibliography will be presented [12].

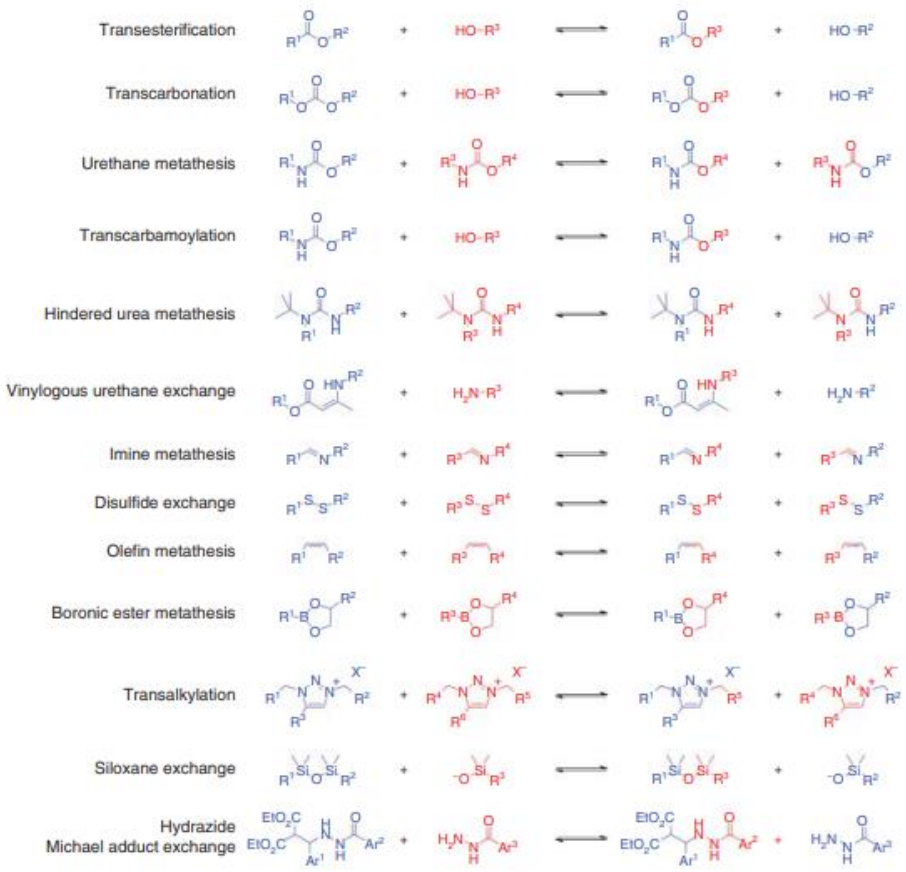


Figure 4. Exchange reactions used in vitrimer synthesis [12].

Esters and carbonates

By utilizing transesterification reactions, epoxy/acid systems can lead to the synthesis of vitrimers by mixing stoichiometric amounts of bi- and multifunctional monomers. On the contrary, in epoxy/anhydride systems, stoichiometry must be considered more carefully, and the polymerization process seems to be more intricate. The reaction involves dehydration, condensation, and hydrolysis of carboxyl and hydroxyl groups. These kinds of systems are usually favored since they are easy to upscale and the introduction of a catalyst can control reaction kinetics, as well as T_v, even though work has been done to eliminate catalysts altogether. It is also interesting how organic molecules, like lignin, can be chosen as precursor molecules for transesterification reactions [9],[11],[12].

Urethanes, vinylogous urethanes and ureas

Urethane groups are highly stable under thermodynamic stress and therefore difficult to process. Studies have shown that catalysts are required to promote urethane metathesis and transcarbamoylation reactions, while dynamic exchange between free

amine and vinylogous urethane groups has been found to happen without the introduction of a catalyst. Vinylogous urethanes and their transamination reactions are being studied as an alternative to transesterification-based polymers. A common way to create such networks is by reacting acetoacetates and amines in a slightly off-stoichiometric ratio, without adding catalysts [11],[12].

Imine exchange

Imine links, or otherwise known as Schiff bases, are a thoroughly studied class of bonds that is now investigated under the light of dynamic behavior. Imine metathesis polymers exhibit unique qualities like malleability after water introduction, and much research is being done to determine how to impart more functionality into these molecules and expand their capabilities, seeing that their mechanical and thermal behavior is not up to par, in contrast to their rheological profile. Aldehydes and mixes of di- and triamines or biobased materials like fructose and vanillin can be used in these reactions and provide a greener alternative for the synthesis of such vitrimers [10],[11],[12].

It is clear that the chemistry behind vitrimers is very rich, making vitrimers a multifactorial "problem". Other variables one needs to consider are processing conditions, choice of compounds and catalyst, respective ratios to be used, *etc.* ML can facilitate research by providing insights (such as vitrimer final properties) without requiring expensive and repetitive experiments.

2.1.4 Vitrimer recycling strategies and general applications

As previously stated, vitrimers, amongst their many benefits, provide an eco-friendlier and more circular alternative to classic polymers, and especially thermosets, as far as recycling goes. Their dynamic bonds allow them to be recycled several times without extreme changes in their properties from one cycle to the next. Usually, the properties under investigation are T_g, elastic modulus of the rubbery plateau, stability under thermal stress, tensile properties and gel content [13].

There are two main recycling strategies: mechanical and chemical recycling. In the former case, vitrimers are ground into powder or pellets and then recycled through hot pressing, injection, or extrusion molding, *etc.* Recycled materials exhibit changes in properties according to the parameters used in the recycling process, such as and not limited to temperature, pressure, and time. Research has been done both in hot pressing and injection molding to investigate the influence of different parameters on the final product [8],[9].

The second recovery method is chemical recycling, where vitrimers are dissolved in solvents or monomers to aid in their depolymerization. That is when the polymeric network is broken down into oligomers or even monomers, which can then be

polymerized all over again and produce new vitrimers [9]. The first method, therefore, relies on repurposing the polymer, and the second on using its building blocks to create something novel. There are benefits and drawbacks to both; mechanical recycling is often favored for its simplicity and ease of use but sometimes needs to operate under harsh conditions. In contrast, chemical methods tend to offer higher efficiency, enabling more precise results. The trade-off, however, is that these methods often come with higher energy consumption and additional time and cost considerations due to the need for specialized chemicals [9].

Adding to these, vitrimer chemistry is paving the way for introducing dynamic bonds in existing thermosets, enabling their reprocessing and recycling, thereby addressing one of their most significant issues [11]. Thermosets sometimes contain dynamic bonds that are activated at very high temperatures due to high activation energy. Through "vitrimerization", a catalyst is infused into the polymers, lowering the activation energy. That can be done either by swelling the thermoset in a solution of the catalyst or grinding it together with the catalyst [8].

Vitrimers are versatile materials that can be used in several applications as mentioned earlier. Besided robotics and 3D printing, they are also utilized in aeronautics, electronic devices, equipment used in automation, and many more. The self-healing ability of vitrimers is of paramount importance since polymeric materials can generally sustain damage, which causes loss in functionality. Several types of healing and welding can be applied, but each comes with its problems. In vitrimers, damage can be reversed, both on the surface and internally, as far as properties go, due to the dynamic covalent bonds. At the same time, they can be used to weld objects together since they form interfacial crosslinks and promote strong adhesion. Additionally, their "shape memory" property makes them ideal in biomedicine, sensors, and aircrafts, and they make for good composites with the introduction of glass, cellulose, and carbon fibers, as well as carbon nanotubes, graphene, and inorganic fillers. Finally, the ability of vitrimers to flow makes them a strong candidate for additive manufacturing, which is to say, 3D printing. 3D printed objects from vitrimers can be sophisticated and geometrically complex, opening doors to interesting applications like optical materials and smart actuators [8],[9],[12].

2.2 Machine Learning

2.2.1 Categorization of Machine Learning Applications

In general, machine learning uses functions that, when correctly tuned, can predict an output value by using data, or otherwise known as features, that were given to them as an input. Functions can be tuned through different parameters. The process of

scientists using data and machine learning to predict certain properties is called inverse design. This way, researchers can avoid experimenting "blindly", since models can provide insights on hidden patterns in data, and offer solutions never thought before [14].

Two main categories of machine learning can be observed, supervised, which is further divided into regression and classification, and unsupervised, which includes clustering and dimensionality reduction. The main difference between the two is whether the data is labeled or not. In supervised learning, data has labels and the outputs provided by the models can be continuous numerical values, in the case of regression, or discrete categorical values, like the ones from classification models. The model depends on the labels to make predictions and explore connections. It should be noted that labels are another name for dependent variables, while features used for training represent the independent variables. On the contrary, in unsupervised learning, the output, for whatever reason, is not labeled, and so the model is asked to make predictions in order to either uncover unseen connections in data (clustering) or reduce the dimensions of a problem by eliminating unnecessary variables (dimensionality reduction). Models like Random Forest (RF), Support Vector Machine (SVM) and Neural Networks (NN) can perform both supervised and unsupervised learning [2],[15]. The aforementioned categories can be observed in Figure 5.

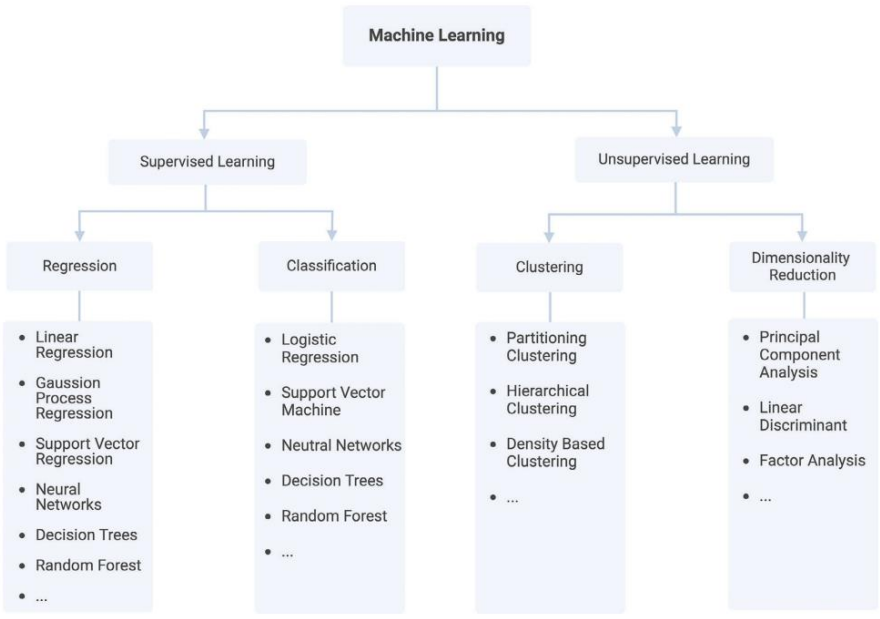


Figure 5. Schematic representation of ML categories [2].

In order to predict material properties, it is necessary to use supervised learning, and more specifically regression models. Some examples include Linear regression,

Random Forests, , Gradient Boosting, Neural Networks and others. The models Support Vector Machines used in this work will be described in detail in the following chapters.

Semi-supervised learning and active learning are also noteworthy categories that are gaining popularity in recent years. Semi-supervised learning includes both supervised and unsupervised techniques to get the most out of small datasets, while active learning is based on Bayesian optimization that drives data acquisition to get useful outcomes out of initially very small datasets [14].

There are many tools available for material and polymer informatics, like RDKit and Scikit Learn, two Python libraries that make the process of implementing ML easier, by providing ready-made functions that perform training, validation and analysis. Indeed, Python is one of the most accessible programming languages and widely used by programmers and material scientists alike, due to its ease of use and extensive documentation [15].

2.2.2 Overview of machine learning pipeline

It is crucial to provide an overview of a common ML pipeline, which is to say the necessary steps that need to be taken in order to train and validate a model. First of all, it is important for the goal of the study to be set so that the objectives, and therefore the training process, are clear. Afterwards, we need to work on curating a database that will be used during model training, which will include the input variables (features) and the output variable (target). In this stage, information can be included from both literature (data collection) and lab experiments.

Next up is the data processing step, during which data that could potentially hurt the model is eliminated from the data set. This can be done through outlier detection, which is the process of locating values that are significantly different to the others and excluding them. During this step it is also necessary to treat any missing values, either by completely eliminating them from the data set or imputing them. This can be done in multiple ways, but two common strategies are assuming missing data points have a zero value and substituting missing values with the average value derived from the rest of the relevant data points. So, for example, if there are T_g values missing from a dataset, the mean average of the existing values can be calculated and used to impute the missing ones. Finally, during data processing, it is important to normalize all variables, through different schemes, so they can have the same range. This is very important later on because non-normalized data can create biases during the model training, where variables with larger ranges can be interpreted as more important than variables with smaller ranges. Normalizing all variables in a uniform way forces the model to treat all variables equally, and it can be done through linear, z-score or log scaling.

The fourth step is the featurization step otherwise known as fingerprinting step. This is where the chemistry behind the physical problem is translated into a machine-readable quantity (feature) through different procedures. After featurization, researchers must select the appropriate models and then train them. Before training a model, though, it is usually necessary to apply a dimensionality reduction scheme, in order to eliminate redundant features, and avoid overfitting. During the training step, it is important to find the suitable hyperparameters that will lead to optimized results, which is usually done through trial and error or through readily available algorithms.

In the end, we benchmark our results. This means comparing how models behave when trained with different data sets, different fingerprints, different hyperparameters and even different dimensionality reduction schemes. This is usually done through the calculation of metrics like mean squared error, R^2, and others, and visualizing the results provided by the models with residual plots, parity plots, *etc*. Residual plots visualize predicted values against residuals, while parity plots, actual values and predicted values. After every step has been taken, it is possible to set up the final model and use it in the way it was intended [3]. Figure 6 provides a quick overview of the workflow described previously.

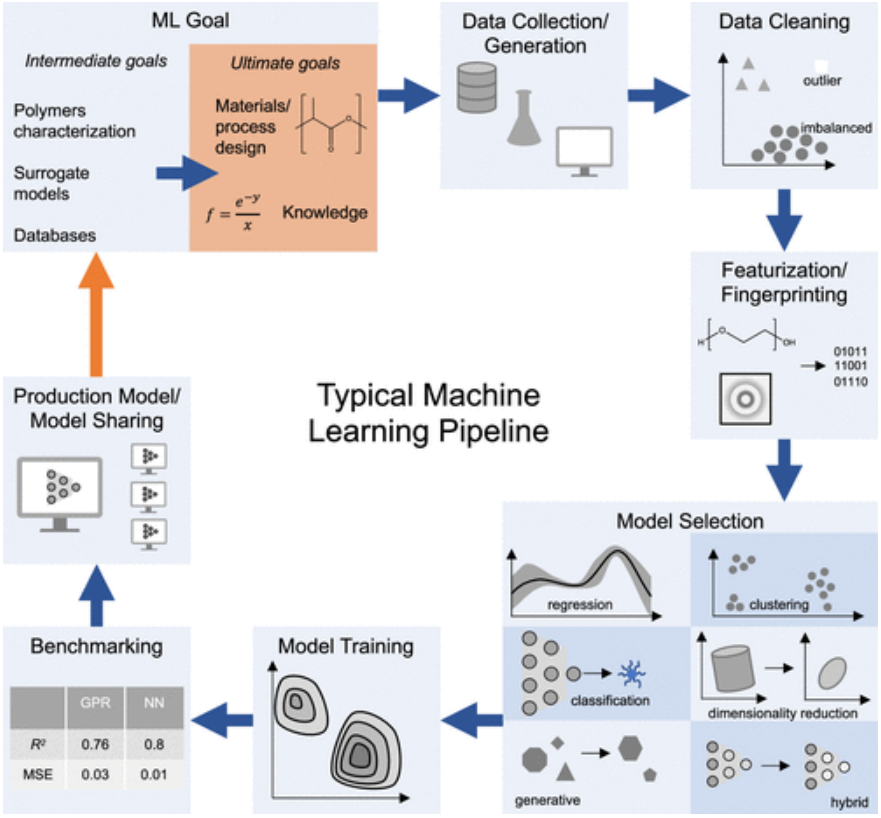


Figure 6. Common ML pipeline [3].

2.2.3 Fingerprinting methods

Creating a numerical vector, also known as a fingerprint, is the backbone of every material informatics project because it transforms the input features and the chemistry and/or structure of a molecule into a computer-readable input. Based on these curated fingerprints, models can be trained to predict a target value, usually a property of the final molecule, based on the chemistries of its precursor molecules [6]. This process is also referred to as quantitative structure–activity relationship (QSAR) and, similarly, quantitative structure–property relationship (QSPR). Both utilize statistical and mathematical methods to explore the connection between molecular structure and properties, which creates the need to establish a way to describe the molecular structure in a unique and measurable way [16].

Several fingerprinting schemes are being used across material informatics problems like molecular fingerprints, descriptor-based fingerprints, deep learning-based fingerprints, graph-based representations, *etc*. There is no "turnkey" solution that can be applied across all problems, and it is therefore critical for scientists to choose their fingerprinting method carefully. In Figure 7, the example of polystyrene is being used to showcase different available fingerprints [2].

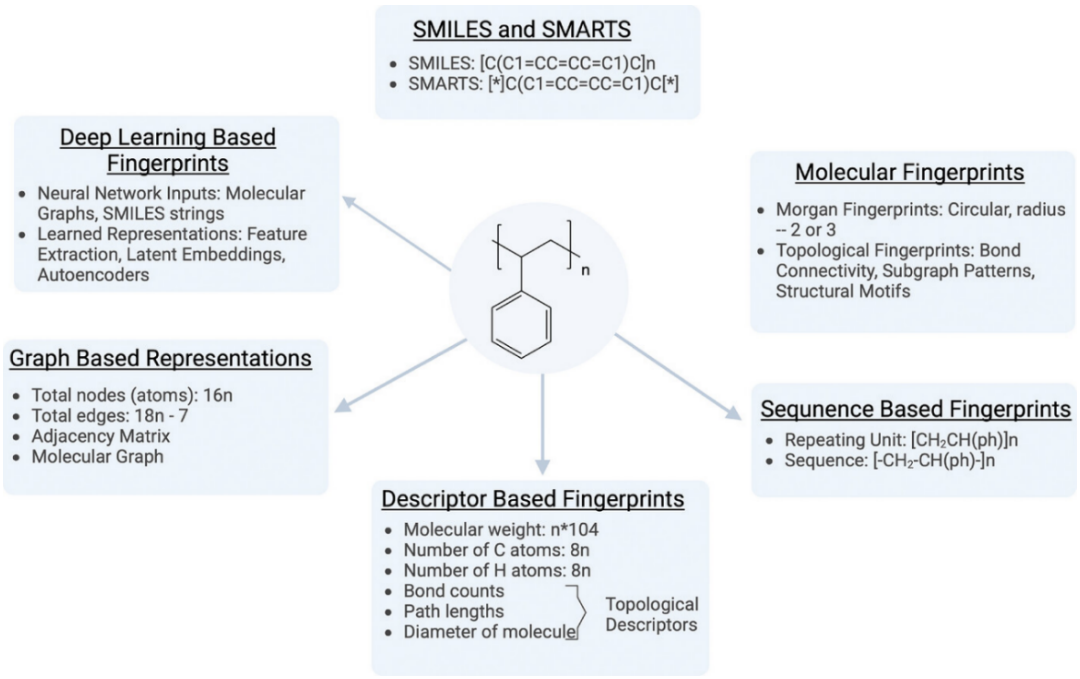


Figure 7. Fingerprinting methods: the example of polystyrene [2].

Feature extraction is a necessary step in descriptor-based fingerprinting. Here, chemical descriptors, like the number of bonds, hydrogen acceptors and donors, and many

more, are obtained from the chemical structure of a molecule. Molecular descriptors were created by combining several theories, like quantum chemistry, information theory, graph theory, and so on, and can be processed through cheminformatics and used in QSAR/QSPR and other computer-based tasks [18].

It is clear that the structural complexity of a molecule cannot be fully described using such features, because each descriptor can only describe a small part of the information contained in a single molecule. The only way to achieve as much representation as possible is through high-dimensional descriptors or multiple descriptors, which adds -sometimes unnecessary- complexity to a model. Most of the time, researchers extract thousands of descriptors and, during the pre-processing stage eliminate all unnecessary ones through dimensionality reduction schemes, ending up with a significantly reduced number of final features. In general, descriptors can be categorized in the following way [16]-[18]:

1. Zero-dimensional (0D): Descriptors that draw information from the chemical formula are called 0D descriptors, and they represent bulk properties, like atom counts, molecular weight, and sums or averages of properties found in the atoms of a molecule.

2. One-dimensional (1D): Descriptors that break molecules into substructures, like functional groups, and count their occurrence are called 1D descriptors. They are usually binary, using 0 or 1 to signal the absence or existence of a structure, or occurrence frequencies, counting the times a certain group appears in a molecule. These, together with 0D descriptors, are also known as constitutional descriptors, which only consider chemical composition and not topology.

3. Two-dimensional (2D): These descriptors reveal more information on the connectivity of atoms than 1D descriptors, where the knowledge of the entire molecular structure is not necessary. Here, bonds between atoms are considered, in addition to taking note of which and how many atoms exist, and the molecule is represented through a graph. The atoms are the vertices of the graph, and the bonds are their edges. Graph representation makes the calculation of several descriptors easy. There are two main categories considered: topological indices (TIs) that encode properties related to topology, like adjacency, and topochemical indices that represent chemical properties of the target atoms, like whether they are hydrogen bond donors or acceptors, their mass and polarizability, *etc*. TIs are sensitive to different structures, which is to say they are sensitive to size, symmetry, branches in the molecule, presence of cyclical structures, and others.

4. Three-dimensional (3D): 3D descriptors, commonly referred to as geometrical descriptors, are similar to 2D descriptors, but add another layer of complexity

by considering the molecule as a 3D object in space, which means they consider its x-y-z coordinates. Therefore, for 3D descriptors, the position of an atom is as important as the atoms it is connected to. They are very useful, especially in challenging subjects like modeling biological properties, but are inherently complex and tricky to calculate correctly. An example of this are Coulomb matrices.

5. Higher dimensions: 4D and higher dimension descriptors, like Comparative Molecular Field Analysis (CoMFA), exist but are computationally intensive and should be considered for each machine learning problem separately. These descriptors can capture properties that originate from electron distribution.

Descriptors can also be categorized as classical molecular descriptors and as binary fingerprints. The first represents chemical features through a single value and can be used in combination with other descriptors, which may potentially be measured on a different scale; some could be integers (atom counts), others binary values (presence or absence of a group), and others continuous (molecular weight). The second only represents structures in a binary form, as a series of bits, and therefore needs to be used as a whole, for it to be interpretable [17].

Structural keys and molecular fingerprints are examples of binary fingerprints. With both, an effort is made to represent a molecule by capturing parts of the structure. Usually, structural keys have a list of already defined structural features, such as atom pairs, functional groups, *etc.*, used to differentiate between molecules and derive useful information from them. The most popular is MACCS keys, with 960 structural features. On the other hand, molecular fingerprints explore a molecular structure through every possible path, following a set of rules. It is important to choose the kind of patterns that will be identified by the fingerprint. In most cases, two types can be considered, paths and atom-centered, which are also called circular, and two classes will be calculated, path and extended connectivity fingerprints. The latter is also referred to as Morgan/ circular fingerprint, and it is commonly used in ML problems. Users can define the radius of the patterns to identify and optionally set up parameters to tune the discrimination power of substructures. Some properties calculated are the number of neighboring atoms, formal charge, atom aromaticity, and many others. Molecular fingerprints are Boolean vectors that define whether a feature of the structure exists or not [18].

2.2.4 Algorithms applied in ML pipelines

2.2.4.1 Machine Learning Algorithms

The selection of a specific algorithm is influenced by the kind of mathematical techniques it uses, or the type of loss function it optimizes and whether those suit the problem at hand. Because of this, an important step that can be taken in the training

process, if possible, is exploring the connection between the data and the mathematical background of the model, which is called inductive bias [15]. Below, models used for this work will be presented.

Linear regression

Linear regression is one of the simplest, yet effective algorithms for modeling when the target value and the features have a linear or close to linear relationship. It is possible for nonlinear relationships to also exist and be explored through linearization of the variables. In other words, variables like x^2, x^3, *etc*., can be transformed into x_1, x_2, *etc*. and be used in a linear relation like

$$y = b_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots + a_k \cdot x_k. \quad (2.4)$$

Most linear regression models apply a regularization process, where the target is to minimize overfitting through limiting the values of the relation coefficients optimized during training. Popular models used for this are Ridge and LASSO regression. The coefficients are optimized through a process whose goal is to reduce the ordinary least squares value of the model [2],[15].

Support Vector Machine (SVM)

Support vector machine algorithms are very commonly used ML models for classification and regression, and present similarities to regular linear regression. While linear regression works to minimize the ordinary least squares error, SVM minimizes the sum of squared errors (w^2) made during prediction while trying to find a line that best fits the data. More specifically, SVM minimizes the following equation

$$\frac{1}{2}\|w^2\| + C\sum_{l=1}^{l}(\xi_i + \xi_i^*) \quad (2.5)$$

where, C is a positive constant used to show the trade-off between training error and model smoothness, ξ_i, ξ_i^* are slack variables, introduced to handle the observations and l is the length of the dataset. By doing so, SVM retrieves a mathematical equation, known as a hyperplane. The hyperplane is used to separate groups of data (clusters) by emphasizing the data points that are nearest to the decision barrier between classes. These classes are also called support vectors. This is usually a linear process, but it can be applied to non-linear data as well, by using a kernel function to mathematically transform the data and add dimensions. This model can also be used in regression tasks when the hyperplane is chosen in order to include as many observations as possible and provide a representative boundary between different classes [2],[6],[15].

One particular strength of this model lies in its robustness and customizability through kernel functions. These include linear, radial bases function (RBF), polynomial, sigmoid and other kernels. It is a good choice for smaller datasets with multiple features. In the presence of large datasets, the computational complexity of SVM hurts the model's

predictive accuracy [6],[14]. SVM is also indifferent to the presence of some errors, as long as they fall under a margin, called epsilon tube, that is considered around the fitted line. The support vectors mentioned earlier lie on the edge, or outside of this tube, which can be seen in Figure 8 [2].

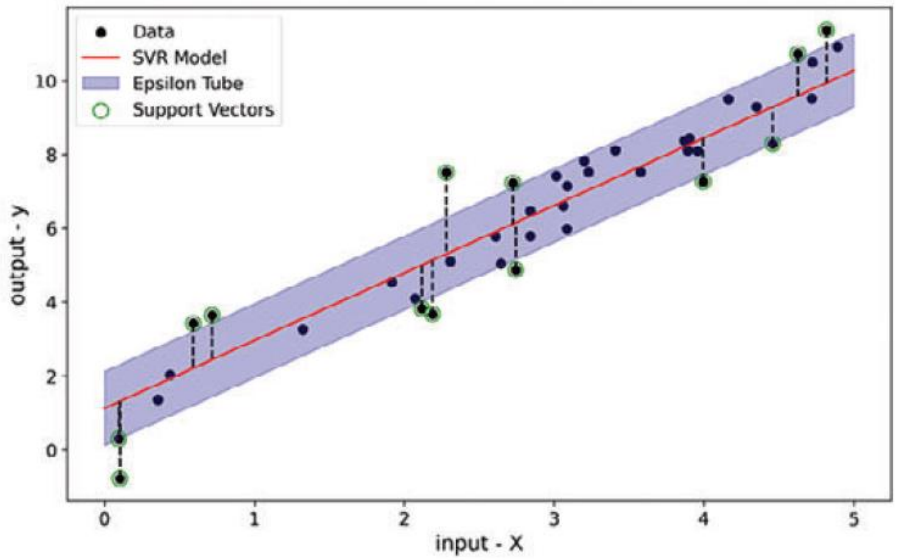


Figure 8. Schematic of a support vector machine model [2].

Gaussian Process Regression (GPR)

Gaussian Process regression is a subcategory of the Gaussian Process model, which is used for both classification and regression. It is a non-parametric approach, and it measures how similar data points are, using kernel functions, like white, RBF, Matern kernel, *etc*., to predict new data. Kernels can be set by users if there is previous knowledge or updated during the training process based on the model's performance and are used to measure the distance between two points. The optimal predicted function is modeled as the distribution of all possible functions that fit the data, using Bayesian statistics to estimate uncertainty. The distribution is of Gaussian nature, and it is a flexible surface over the space that features hold. This model, in contrast to the previous ones, does not assume linearity but rather a flexible, smooth function [2],[6],[15].

In GPR, every predicted value is accompanied by an uncertainty value or variance, which is important for model interpretation and understanding where the model makes poor predictions. It is best suited for small data sets, where it captures relationships between data well. It is not recommended for high dimensional problems and big datasets due to its high computational cost [15]. It is also important to note that GPR's results rely heavily on choosing the right kernel [2].

Random Forest (RF) algorithm

The Random Forest algorithm is an evolution of another algorithm known as decision trees. Decision trees are used in both supervised and unsupervised ML, and it is an algorithm that creates structures resembling trees with branches, which split from nodes. A node that splits the tree based on some criterion is called a decision node, while a terminal node is called a leaf. The logic behind the branching of such a tree is based on successive "if/else" questions asked at each node for the various variables. Using a specific value, each variable is compared and accordingly classified into subsequent branches. When used in regression tasks, the mean squared error (MSE) and other values are measured to evaluate how much the variable under computation varies at each node, and the node ultimately takes on the average value computed at that point. This process is repeated multiple times until homogeneity is achieved in the labels assigned by the tree, a property known as leaf purity.

Decision trees are not ideal for training models based on small datasets because they are prone to high variability, which ultimately undermines the model's accuracy. Thus, characteristics such as high noise can lead to overfitting and, consequently, poor generalization of the model. However, their advantages—such as capturing non-linear relationships well and interpretability—are usually harnessed through the so-called ensemble methods. These are methods that use many machine learning models together or use one model many times, combining their results to produce a final, weighted output. One such example is the Random Forest algorithm, which can be defined as a "forest" of decision trees [2],[15].

The main advantage of Random Forest is that it combats the high variability inherent to decision trees by creating multiple trees. Each one works on a random portion of the database (a technique known as bootstrapping), and at each node, only a subset of the total features is examined, usually the square root of the total number of features, again selected randomly. In this way, each tree explores different connections within the data and contributes to the overall result in a unique manner.

Thus, the final value given by a Random Forest in regression tasks is the average of the values from all individual trees. This approach increases the robustness of the model, as it is not as heavily affected by potential fluctuations in the data. Some additional advantages include its ability to handle missing data and its capability to accept multiple input types, both categorical and numerical. It is clear that the best predictions occur when the forest has great diversity, meaning the trees are uncorrelated with each other. The main drawback is increased complexity compared to decision trees, making it more difficult to interpret and draw conclusions from. A schematic representation of such a model is shown in Figure 9 [2],[15],[19].

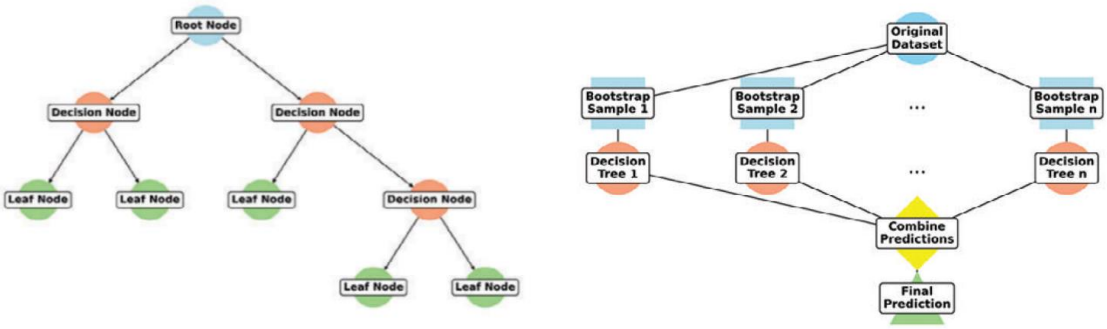


Figure 9. Schematic of a Decision Tree (left) and a Random Forest (right) [2].

Gradient Boosting Machines (GBM)

Gradient boosting machines are an ensemble method, like Random Forests, but they operate on a different ensembling strategy. More specifically, the idea behind them is adding new weak, base-learner models in every iteration, and training them based on the error of the whole ensemble up to that point. This means that the new models need to be correlated with the negative gradient of the loss function, which can be chosen by the researchers. A common choice would be the classic squared-error loss, which would make the training a consecutive error-fitting process. The goal is to improve the estimate of the model as much as possible, since GBM treats boosting as a numerical optimization problem [20],[21].

One of the main advantages of Gradient Boosting is its customizability, which makes it suitable and successful for a variety of different tasks and data sets. Choosing an appropriate loss function can be done through trial and error, but for continuous target variables there are four first order loss functions to be considered. An example of such a loss function is the Laplacian L1 loss function $\Psi(y,f)_{L1}$:

$$\Psi(y,f)_{L1} = |y - f| \tag{2.6}$$

where, y is the continuous target and f is the function the model tries to estimate.

The base learner models can belong in one of three categories: linear models, like ordinary linear regression and ridge penalized linear regression, smooth models, such as P-splines and radial basis functions) and decision trees- either stumps or trees with arbitrary interaction depth. Decision tree stumps are trees with only one split. Decision trees based GBM is usually the most common choice, since it is the most reasonable, computationally speaking [21].

eXtreme Gradient Boosting (XGBoost)

The extreme Gradient Boosting algorithm improves on the GBM algorithm by expanding the loss function to the second order, using Taylor expansion and adding a regularization term. That is why it is also called Newton boosting method. It uses the

Gradient Boosting methodology and predicts values based on several complement functions, further expanding the previous algorithm's capabilities. Adding to that, XGBoost perfects the details of feature processing and the iteration process while it is implemented. It is a powerful tree-based algorithm that, like GBM, is considered an ensemble method. As with every model, there is a trade-off between model accuracy and overfitting [20],[22].

2.2.4.2 Dimensionality Reduction Algorithms

Dimensionality reduction is an important step in every ML project, necessary to avoid the pitfall of overfitting and time-consuming computations. One can avoid overfitting by reducing the amount of features a model is trained on, or in other words the dimensions of the problem, which leads to overall less complexity of the model. Researchers often face the issue of choosing the appropriate input features for model training. Indeed, there are a number of molecular descriptors that can be used to represent materials and especially polymers, so it is important for the right ones to be selected to avoid suboptimal model performance [2]. Dimensionality reduction is also useful for removing noise from datasets, since scarce or non-homogeneous data points are excluded during this process, making the final model more robust [20].

These kinds of algorithms fall under the unsupervised learning category and one of the most commonly used ones is principal component analysis. That being said, there are a lot more algorithms that can be harnessed for this work, like t-distributed stochastic neighbor embedding (t-SNE), uniform manifold approximation and projection (UMAP), and diffusion maps [15]. There are also alternative paths that can be taken for dimensionality reduction, such as running a correlation analysis solely on features and then checking the correlation between the remaining features and the target variable and reducing the model's inputs through feature selection algorithms. Such examples are the Mutual Information method, the Random Forest algorithm and others. It must be noted that feature selection needs to follow the correlation analysis to avoid selecting features that are already interconnected.

It is worth mentioning that algorithms used for feature selection are divided into three main categories: filters, embedded, and wrappers. Filters rely on the correlation that features may have with the response variable and are typically used during preprocessing, while embedded methods are applied during model training, aiming to improve model performance by selecting the most relevant features. Wrappers are based on statistical measures to rank features according to their importance. Additionally, the objectives among these algorithms may vary- some focus on identifying the minimum number of features necessary, while others aim to find all potentially useful ones. Filters are less computationally expensive than wrappers, as the latter require training and testing models repeatedly for each possible subset of features. Nevertheless, wrapper methods provide better results [23],[24].

Principal Component Analysis (PCA)

Principal component analysis is a technique used to transform high-dimensional problems into much smaller ones by clustering features together, while making an effort to preserve the relationship between data as much as possible. In fact, PCA maps N-dimensional features to K-dimensional features that are mutually orthogonal using the main principal components. PCA focuses first on descriptors with the maximum amount of variability, and this is why it is especially important to have scaled and normalized all values to the same range, so that no features are considered more important than others due to having a seemingly larger range [25]. To create the principal components, PCA forces linear connections between features, which is as much a strength as it is a weakness in some cases. On the one hand, PCA is easy to use and very effective in making model inputs as simple as possible, but on the other hand, by introducing linearity in complex non-linearly connected data, it can lead to the loss of important information, and, in the end, reduce the model's accuracy. At the same time, PCA makes result interpretation more difficult because it combines features together in a "black box" manner. This means that, in reality, one cannot know exactly which features end up informing the model's predictive power and final results, which is something that can be mitigated by introducing more algorithms to interpret PCA's results, or by plotting the PCA's results in scatter plots, which visualize potential groups of variables that inform the so called "principal components". Another option is heat maps, where it is shown exactly how much each feature contributes to each principal component. [2],[15],[20].

Pearson correlation analysis

Often, when using multiple descriptors without *a-priori* knowledge of their influence on the final result, it is important to make sure they are all equally important and not redundant. It is, in fact, critical to eliminate correlated features because models pick up on those internal connections and learn from them, which may lead to overfitting. One way to do this is by computing the correlations between different features using Pearson coefficients, Spearman coefficients, *etc.* Another option is creating a Pearson correlation map to visualize the connections between descriptors and, potentially, eliminating some of them based on how strongly they are related to others [26].

The Pearson correlation coefficient R can be used to measure the linear relation between two variables. R is characteristic of data association, has a range from -1 to 1, and is usually computed in the following way:

$$R = \frac{cov(x,y)}{[std.x \cdot std.y]} \tag{2.7}$$

where, *x* and *y* are the two variables under consideration. When R takes the value of -1 it means the variables are perfectly negatively correlated, and adversely, the value of +1 shows perfect positive correlation. By introducing Bayesian analysis of the linear correlation, better predictions can be achieved, as shown in literature [27].

Mutual Information (MI) method

Mutual Information is a filter method that has its roots in statistics and is useful in identifying even non-linear relationships between data, as it detects any correlation they might have with each other. Another advantage of this method is that it does not require a specific database size to function, so it can be used equally well on small databases, like the database curated for this work, as well as on large databases with thousands of entries. It is, of course, important for the data to be normalized on the same scale for their relationships to be correctly diagnosed.

The Mutual Information between two numbers is a non-negative value and is equal to zero only when the variables under examination are completely independent. This method is often used for feature selection and can rely on the calculation of entropy, meaning uncertainty, based on the k-nearest neighbors algorithm, which essentially calculates the distances between 2 neighboring points and thus determines how scarce the data are [28],[29]. Another area in which it finds application is in Independent Component Analysis (ICA), which is used to separate a multivariate system into multiple independent variables [29].

Random Forest (RF) algorithm

This algorithm, which is considered a wrapper method, beyond its direct use in machine learning as the primary predictive model, is also frequently applied in the process of selecting the final features that will be used as inputs during the training step of a model. Various sub-methods can be employed, such as the Boruta algorithm, RFE, RRF, *etc*. [23]. The way feature importance is assessed is as follows: in each tree, there is a portion of data and features that are not examined (out-of-bag individuals). At the end of the training process of the entire forest, it becomes evident which features truly matter if their absence in a particular tree prevents it from making accurate predictions [19].

Chapter 3: Methods and Materials

3.1 Setting the goal of the ML model

The objective behind this machine learning pipeline is to train models to make predictions on four key vitrimer properties and compare their predictive accuracy while testing different dimensionality reduction methods and hyperparameters. The vitrimer properties under investigation are crosslink density, T_g and T_v and gel fraction. Similar workflows will be applied for all four properties to draw useful conclusions about the chemistry behind the problem, and how it can affect the predictions made by ML models, which only handle numerical data and possess no domain knowledge.

A usual ML workflow has been followed, but unique steps, especially during dimensionality reduction, have been taken, in order to discover the best way to handle this intricate problem, which involves chemistries of not only one, but multiple components.

3.2 Database curation

3.2.1 Data collection

As was previously stated, data collection for building databases for Polymer Informatics is a laborious task with several problems. There are still not many such databases available, but some interesting and noteworthy examples are Polymer Genome/ Khazana, which is not a library of properties, but a predictive tool created by Ramprasad Research Group, and PolyInfo, which gives access to several properties for different polymers. Some more reliable sources are PubChem and ChemSpider, which contain information on some polymers and polymer properties as well as all sorts of molecules [2].

In the current work, an effort was made to collect data points from different articles available online, with two main considerations in mind. First, the sources needed to be reliable and present several measurements to create a consistent database with few missing values. Secondly, the materials used in the experiments needed to be diverse with the aim of including and exploring different chemistries. Almost 50 articles were considered, out of which 31 were finally used to draw useful data points, as referenced [30]-[62]. Despite the extensive literature, the problem of missing values proved challenging. Not all research groups are indeed interested in measuring the same vitrimer properties and presenting results relating to every one of their specimens. In

the earlier versions of the database, many properties were included, a lot of which were later removed due to limited data availability.

The properties in the original dataset were: compound name, compound molecular weight *MW* (g/mol), number of functional groups, compound molar ratio (% mol), reactive end-groups ratio (-), reaction temperature *T* (°C), catalyst type its molar ratio (% mol) and its MW, gel fraction *GF* (%), swelling ratio *SR* (%), crosslink density v_e (mol/m^3), glass transition temperature T_g (°C), relaxation time τ^* (s), pre-exponential factor of relaxation time A (s^{-1}), activation energy E_a (kJ/mol), topology freezing temperature T_v (°C), melting temperature T_m (°C), zero shear modulus G_0 (MPa), decomposition temperature at 5% weight loss $T_{d,5\%}$ (°C) and finally decomposition temperature T_d (°C). The final three were eliminated in the next version of the dataset because few groups measured them, and they were ultimately deemed to be of little physical importance for the model.

GF and *SR* can be calculated through insolubility tests, and they represent the insoluble fraction of a vitrimer and the amount of it that swells, respectively. Both can give insight into the crosslinking degree and density. The same can be said for G_0, which expresses the elastic response of the material in the rubbery plateau region. Additionally, relaxation time shows the time required for a vitrimer to fully relax stress, and its pre-exponential factor can be calculated through the Arrhenius equation as follows:

$$\tau_0\ (\mathrm{s}^{-1}) = \tau^* \cdot \exp(-E_a \cdot \frac{1000}{\mathrm{RT}}) \tag{3.1}$$

where, R=8.314 *J/mol·K* and *T (K),* is the temperature where the relaxation time τ^* was measured in [29]. Equation (3.1) can be used because, as previously mentioned, the Arrhenius equation can be used to fit the temperature-dependent behavior of the characteristic relaxation time. This measurement is helpful to calculate, since it is not temperature dependent like relaxation time, and thus can be consistently represented in the database.

Aside from the molecular weight and the number of functional groups, some other properties like intrinsic viscosity, density, and polarity were considered for the database but were not included in the end. The reasons behind this choice were their temperature dependence, the manual labor required to fill them in, making the database counterintuitive and harder for future expansion, and, finally, the lack of immediate connection to the final vitrimer properties. Either way, compound chemistry will be portrayed thoroughly through specific descriptors that will be introduced in the next chapter.

Finally, the choice to add Young's modulus *E* (MPa) and tensile strength *σmax* (MPa) was made, alongside the effort to represent recycled vitrimer properties through the

two, for three consecutive cycles. In the end, *E* and *σmax* for recycled samples were dropped from the dataset because a limited number of groups recycled their produced material and took note of the respective values, making the overall results in the dataset very scarce. Adding to that, out of the groups that actually investigated recycling, many of them followed different experimental procedures and focused on different properties and characteristics of the final material, such as texture, appearance, mechanical properties, and even relaxation times. This made it very hard to present all these findings in a uniform and consistent way. After this, relaxation time τ^* was also eliminated due to its temperature dependence, as was compound molar ratio, since reactive end-group ratio expresses the same value, more concisely.

To sum up, the final measures that remain in the dataset after all the changes are the following:

- Compound name, MW, number of functional groups and reactive end-groups ratio,
- Catalyst type, MW and molar ratio,
- *T*,
- GF and SR that yield v_e (mol/m^3),
- T_g and T_v and T_m
- A and E_a,
- *E* and σ_{max}.

The dataset initially comprised 222 observations. However, the final number used for training was reduced to 209 due to the exclusion of certain data points. These excluded cases involved systems composed of four components, where information on the fourth component was often limited or incomplete. This resulted in several missing values and compromised the overall homogeneity of the dataset, justifying their removal from the analysis.

3.2.2 Necessary calculations

Relaxation time and pre-exponential factor

It is worth mentioning that some relaxation time values were not directly reported and instead were only presented through figures. In order to extract relaxation times and the corresponding temperatures where they were measured, WebPlotDigitalizer, an online tool that helps extract numerical information from images, was used [62]. The user uploads the graph image, define the scales (i.e. normal, logarithmic, *etc.*) and calibrate the axes by assuming two sets of values- here x_0, y_0 and x_{final}, y_{final} which were easy to deduce. Afterwards, coordinates are provided for any chosen point in the image, which can then be used as an input for equation (3.1). More specifically, in the plots, *x* represented *1000/T* (K^{-1}) and *y*, τ^*.

Molecular weight (MW)

For novel molecules, MW was calculated by summing the atomic weights of their constituent atoms. In the case of polymers, the MW of the repeating unit was used instead of the full polymer MW. This approach ensured a consistent representation across polymers that may differ only in polymerization degree. Additionally, using full polymer molecular weights would have resulted in an extremely wide range of values, potentially destabilizing the model during training. Finally, in some cases, epoxy equivalent weights (EEW) were provided and by following the following equation they were transformed into molecular weights for uniformity:

$$MW = EEW \cdot Number\ of\ epoxy\ groups \tag{3.2}$$

Reactive end- groups ratio

Reactive end-groups ratio was calculated by simply dividing the mol of a reactive group present in a molecule with the mol of another. A choice had to be made as to which functional group would be the numerator and which the denominator to maintain uniformity in the data set. There are many possible functional groups like epoxy (-COC), amine (-NH_2/-NHR/-NR_2), carboxylic acid (-COOH), hydroxyl (-OH), ester (-RCOOR'), anhydride (-RCOCR') and more.

Epoxy groups were always used as the denominator, and when the composition was not based on molecules with epoxy end groups, amines were used as the denominator. Whenever it was possible, the following relation was applied:

$$Reactive\ end-\ groups\ ratio = \frac{mol\ of\ functional\ group\ of\ Compound2}{mol\ of\ functional\ group\ of\ Compound1} = \frac{functional\ group2\ equivalent}{functional\ group1\ equivalent} \tag{3.3}$$

This was not always possible due to the existence of a third compound that had to be included in the calculation. Initially, compound1 referred to the crosslinker or reactive resin used in the experiments, but this was also not applicable universally either. In general, one of the biggest challenges in this work was properly representing this exact value.

An added layer of difficulty came from the fact that many research groups chose to present the quantities they used in their materials through mass and not mol. In these cases, $n = \frac{m}{MW}$ was used to transform mass to mol. It is especially important to remember that the reactive group ratio is a division between functional groups and not entire molecules, which means that the mass of the functional group had to be estimated by the molecule's mass given by the researchers.

Catalyst molar ratio

This ratio was usually provided by researchers, and it referred to how much catalyst was added in relation to one of the compounds. To make everything more concise, it was decided that given values would be divided by the number of functional groups. This way it would be similar to reactive end group ratio. Of course, the same applied for these calculations as before when quantities or percentages were given in mass instead of mol.

Whenever possible, average values are used in the data set. The same is done in the case of mixtures; a fifty-fifty contribution was assumed, unless stated otherwise, and the final properties were calculated by equally weighing the properties of the precursor molecules.

Finally, all calculations made are highlighted in red in the excel spreadsheet named "Draft4_new2" provided in the link in the Appendix.

3.2.3 Translation of chemical structure through SMILES

3.2.3.1 Explanation of SMILES representation used

SMILES is a 2D graph-based representation and the most common way to represent chemical structures in a way that computers can interpret. Molecule graphs are translated into linear notation, where atoms and connectivity are defined [17]. Since there are limitations in the case of polymers, scientists have been exploring alternatives. One such example is BigSMILES, where pieces of polymers are expressed by a list of repeating units between curly brackets. The repeating units can be written in the SMILES language, with the addition of bonding descriptors that showcase how these units are connected. This structure provides the opportunity to represent complex macromolecules like homopolymers, copolymers- both random and block-, non-linear polymers, and so on, with the addition of some rules and operators to the existing SMILES [5]. Different structures can be seen in Figure 10. However, despite their many capabilities, BigSMILES cannot yet be used in existing Cheminformatics packages like RDKit, which is usually leveraged to extract chemical descriptors for molecules based on their structure [6].

Another interesting approach and one that will be used in the current work is PSMILES, proposed by Ramprasad Research Group, which represents the repeating unit of a polymer, using SMILES rules as well as some additional ones that can be found in the Polymer Genome site [64]. The most important addition is that of an asterisk (*) that highlights the polymerization points and allows for the repetition of many units in the structure to better capture the molecule's complexity. The advantage of this approach over simply using monomer SMILES is the fact that it provides information for polymerization and, because of that, increases the prediction accuracy in ML

applications. Traditional monomer SMILES, when used in small datasets, provide adequate accuracy and are generally easier to represent, while PSMILES require processing before being fed to a model, and there are few, if any, ready-made and available to this day. At the same time, regular SMILES make differentiating between small molecules and the repeating unit of a polymer almost impossible and cannot account for the possible existence of different polymerization points in polymeric chains. For all these reasons, the representation of polymers was chosen to be through SMILES of the repeating unit (PSMILES), using just one unit to avoid unnecessary complexity at this stage. Adding to that, it has been shown that including many repetitions of the repeating unit in PSMILES, does not improve the model's ability to make predictions [6].

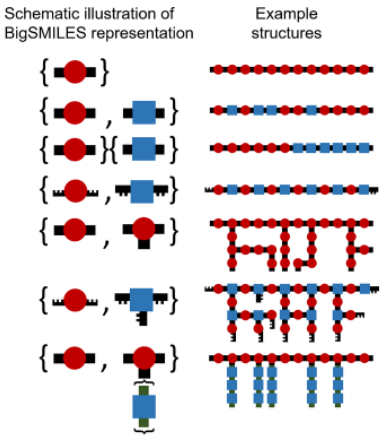


Figure 10. Illustration of BigSMILES [5].

3.2.3.2 SMILES collection

For simple molecules (e.g., acids, catalysts), SMILES identifiers were retrieved from databases such as PubChem and ChemSpider and copied into an excel spreadsheet. These SMILES were then verified for validation purposes by visualizing the corresponding structures using the SMILES Mapper website- a useful tool that generates molecular structures from user-provided SMILES strings [65].

To generate and visualize repeating units for polymers, Polymer Genome guidelines and a publicly available Python Package were used [66]. The resulting structures were also added to the spreadsheet for consistency and clarity.

A different approach was followed for more complex molecules, such as novel compounds whose synthesis was described in the literature. When molecular structures

were not explicitly provided by the research groups, chemistry knowledge, along with informed assumptions, was used to deduce plausible structures. Corresponding SMILES identifiers were then constructed based on the rules defined by Daylight [67].

In cases where structures were available as images, the MolView website was used to reconstruct and extract the remaining SMILES. MolView is a powerful tool for molecular representation that also supports the export of related data such as SMILES, molecular weight, hydrogen donors/acceptors, and more [68]. This functionality was particularly helpful for handling complex or unconventional chemistries, where manually deriving SMILES using structural rules would have been time-consuming and error prone. Other molecular drawing websites and software are also available and can be used instead of MolView, like eMolecules or Structure Search by ChemSpider.

In a later stage, using a suitable Python package, the SMILES strings were rendered as images via RDKit, to make the database even more user-friendly and to allow quick and easy visualization, even of complex compounds [69].

3.3 Feature extraction

Molecular descriptor calculations are time consuming if done manually, because of their complex mathematical basis, but can be done quickly and easily by available softwares. RDKit, a popular cheminformatics python package, was chosen as the feature extraction tool for its ease of use and ability to extract a multitude of descriptors. More specifically, it used to be able to calculate up to 117 0D to 2D descriptors, but this number has expanded to about 200 [17]. PaDEL - Descriptor is another package available for extracting chemical descriptors, with a web-based interface [70]. It is simpler to use than RDKit and has more descriptors available. Nevertheless, it has less functionalities, and RDKit can create results that can be directly fed to a machine learning pipeline, making it more suitable. RDKit can receive a SMILES string as an input and print out a table of descriptors requested as an output. For the objective of this thesis, keeping in mind that the connection behind compound chemistry and final vitrimer properties is not fully explored, all available RDKit descriptors were requested, alongside the Morgan Fingerprint. Usually, when domain knowledge is available, more specific descriptors can be obtained in the beginning, to cut down on calculation time.

Requesting these descriptors was done in an effort to combine two fingerprinting methods together, *i.e.*, descriptor-based and molecular fingerprinting. While it is rather common practice for Morgan Fingerprints to be employed in material informatics problems, here it created a high dimensional latent space- 2047 bits, *i.e.*, 2047 columns

in the excel spreadsheet were reserved just for that. Therefore, the number of features far outweighed the number of datapoints available, which is considered bad practice that can lead to overfitting or underperforming of the model due to added complexity. This is one of the reasons why MFs are usually found in problems where the available database is quite large [2]. The dimensionality reduction methods analyzed previously and utilized in this thesis would not have been able to deal with the Morgan Fingerprint adequately, which is why in the end it was eliminated from the database. In general, though, it has been proven that using Morgan Fingerprint with frequencies is better for prediction than using SMILES as an input, which only works for natural language models that read input the way a human would. That in turn is better than using RDKit-based descriptors, which is still good for most cases [6].

In the current work, a combination of features extracted from RDKit that describe the compounds used to synthesize different vitrimers, alongside features that express how much of each compound was used, will be the basis on which models will be trained. Two separate databases were created first, one with the descriptors or otherwise referred to in the current work as features (feature database), and one with the data collected from literature (experimental database).

3.4 Data processing

The data processing step was carried out as described below in three stages: first for the experimental database, then for the feature database, and finally for the combination of the two. The last database was the one that was used to train the models and draw conclusions from.

3.4.1 Outlier detection and data cleaning

Outliers are values that deviate significantly from most of the values within a given feature or variable and may pose a potential risk to the models training process, especially when they occur in large numbers. Models often try to fit them with the rest of the values, which leads to overfitting and poor generalization. However, outliers are not inherently problematic; in some cases, they may carry meaningful information- especially when they arise from distinct chemistries rather than measurement errors or noise. Python has functions that perform outlier detection automatically, but there are also statistical methods for this, like the Z-score method, where the *z* value is calculated as the number of standard deviations a data point deviates from the mean, and then a threshold for outlier elimination is set- usually for *z* greater than 3 [71].

However, outlier detection and removal were not applied in this work, for two main reasons. First of all, the data set was manually curated from literature, where reported

values are the direct result of experimental validation and can be considered meaningful and trustworthy, even if some seemed rare or different than the majority of other values. Secondly, the small size of the data set was of major consideration, so removing additional points risked narrowing it down even further, which could jeopardize the model training process and its consequent generalization. At the same time, it is known that some models, like tree-based models and Support Vector Machines, are not heavily influenced by the existence of some outliers, so long as they are not extreme. Adding to that, outliers in small data sets can be a result of a skewed distribution and therefore be harmless [25]. As a result, aggressive preprocessing, like outlier removal, was avoided and instead other techniques were utilized to clean the data set as much and as thoroughly as possible, without introducing data sparsity.

Three measures were taken to ensure that the database was properly processed before model training. These also doubled as a means to reduce dimensions because they eliminated unnecessary descriptors.

Firstly, care was taken to remove the features that presented no variance in their values, or in other words, the ones that had a standard deviation value of zero. The same was done for features with very little variance, like Kappa3 (a topological index that quantifies shape), which is shown in Figure 11. Some features with low standard deviation values persisted even though a strict threshold fraction of 0.000001 was set to eliminate them. This fraction was multiplied with the range of values of each feature $(\max - \min)$, which produced the corresponding thresholds. Larger fraction values left unnecessary features intact, and smaller values did not seem to reduce the features further due to precision and floating-point errors.

These two measures were taken to ensure the data set only included meaningful, multivariant features, which could be used during the training process. Leaving uninformative features in the data set would not only keep the dimensions of the problem high but also introduce unnecessary noise. Lastly, in an effort to address the remaining features, a "scarcity" filter was placed. Specifically, any column in which more than 90% of the values were zeros was considered information-scarce and was subsequently removed. It is important to emphasize that these techniques were applied exclusively to features derived from RDKit. In contrast, features collected directly from the literature were left untouched, as zeros in those cases often represent meaningful quantities (*e.g.,* a temperature of 0 °C) rather than the absence of a property.

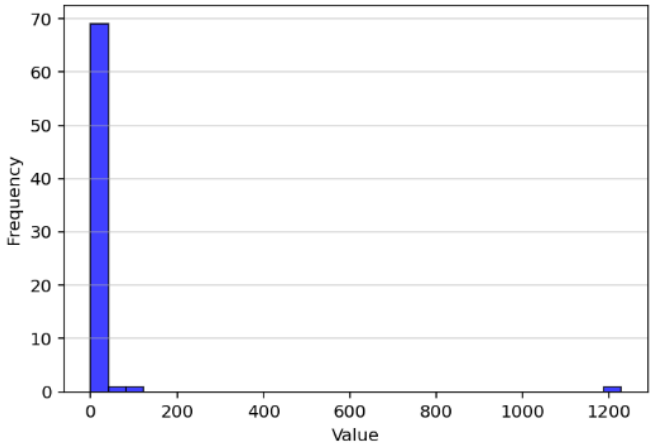


Figure 11. Histogram of Kappa3.

3.4.2 Handling of missing values

Another important step during processing of the data set is to handle missing values in a suitable way. Missing data is inevitable in machine learning projects, so it is important to deal with them correctly, since many ML models or dimensionality reduction schemes do not tolerate them. Some algorithms ignore missing values, and therefore distort the final result, and others simply do not work. Here, a choice was made when curating the data set to represent missing values with a dash, because zero values carry physical meaning, as was previously mentioned. This dash was then replaced by a "NaN" (Not a Number) value across both data sets, which was then imputed with the mean average value derived from the rest of the values of a single property in the case of the experimental data set, and with a zero value in the case of the feature data set. In the former, missing values were not indicative of a property that did not exist but rather a property not measured. Imputing the average value seemed fairer than imputing a zero value which would imply an absence. Adversely, for features, assuming a zero value for missing values seemed more reasonable since descriptors like, for example, aromatic rings were not applicable across all molecules. This is a common methodology in relevant literature as well.

A similar way of thinking was applied in the case of missing molecules. Most experiments were carried out by using two compounds and one catalyst to create the final vitrimer, but some included even three compounds or no catalyst. For those cases, the empty cells that would normally contain the name of the compound or catalyst were filled with the label "No molec.", which was also added as a datapoint in the feature data set. The feature cells corresponding to "No molec." were left empty, and during the value imputation step, the empty cells were filled with zero to reflect the absence of these values. Similarly, when imputing certain property values in cases of missing molecules, such as the molar ratio, a zero value was used and for molecular

weight, a dash. The rationale behind this choice was that the molar ratio would remain zero and thus contribute to the column's mean without being altered, while the molecular weight would be converted to a "NaN" value and be subsequently replaced with the column's mean—since a zero molecular weight has no physical meaning.

3.4.3 Data normalization

In order to choose the appropriate normalization method, it is important to visualize features through histograms, where the values of the features are plotted against the frequency of said values, and identify the kind of distribution they follow. Histograms are also very useful in identifying outliers, because their frequency of appearance is much less than that of other features. In the current work, z-score scaling, which is done through the following relation

$$x_{new} = \frac{x - x_{mean}}{std.x} \tag{3.4}$$

was not applied, because no features followed the normal distribution, which is necessary for this type of scaling.

For features with relatively uniform distributions, which means almost even bars in the histograms, and few, non-extreme outliers, linear scaling was applied using

$$x_{new} = 2 \cdot \frac{x - x_{min}}{x_{max} - x_{min}} - 1. \tag{3.5}$$

This equation transforms data from their natural range to a -1,1 range. This range was chosen to improve model performance and training stability. This scaling is particularly beneficial for algorithms sensitive to the magnitude of input features, such as Support Vector Machines. It is also true that some models are benefited by a zero-centered input [72],[25].

Linear scaling to a 0,1 range is also possible, by eliminating 2 and 1 from the equation. Centering data around the mean, and not the minimum value, is another option, which is done by changing x_{min} to x_{mean}. This was avoided because it does not consistently produce a fixed output range. An example of a feature normalized with this method can be seen in Figure 12.

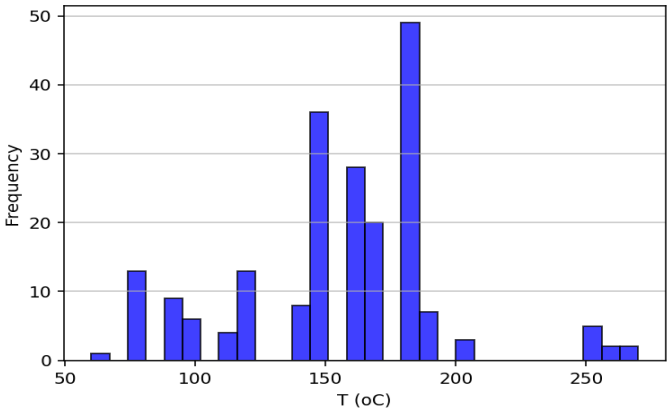


Figure 12. Histogram of reaction temperature T (oC).

Features that were found to adhere to a pattern where higher *x* values are associated with lower *y* values and *vice versa,* were normalized using log scaling. These features are considered as skewed, so their values were transformed using their respecting logarithms,

$$x_{new} = logx. \tag{3.6}$$

This transformation could also be done through square root or inverse, but log is the most common one. Of course, this methodology does not completely normalize feature distributions, but it does improve them [25].

Afterwards, equation (3.5) was applied, in order to achieve the -1 to 1 range for all features. Eliminating skewness followed by scaling to a uniform range has been found beneficial before proceeding to model training, especially for algorithms like PCA. In fact, this methodology was applied to many features in the data set, like Chi0 (an atomic connectivity index of zero order), which is depicted in Figure 13. Chi0 is an example of a right-skewed distribution, where most of the points lie on the left side of the distribution, which means they have small values [25]. It is necessary to note that log transformation was only applied in the feature data set. Some variables in the literature data set were found to be slightly skewed but not enough to alter the final model results significantly. For this reason, only linear transformation was applied.

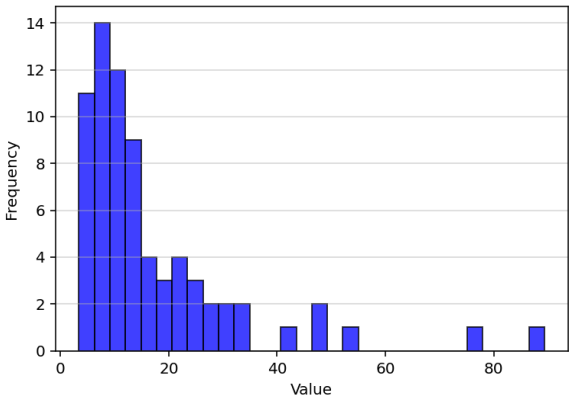


Figure 13. Histogram of Chi0.

While histograms are powerful tools, it was important to define a more automated way to draw conclusions about the choice of normalization method, that did not rely on simple observation and arbitrary decisions. Therefore, it was decided to measure skewness as a means to define the features that required log instead of linear scaling. The more skewed the distribution, the bigger the skewness statistic becomes [25]. For an absolute value of the skewness statistic equal or greater than 0.5, features were considered skewed, and therefore, were normalized with log scaling. The threshold can be adjusted to be either stricter, so more features are considered skewed, or less strict. This specific value seemed to be a good balance between scaling slightly unsymmetrical features and scaling some skewed ones linearly.

It is important to note that the nature of this transformation does not allow for non-positive numbers. A choice had to be made between eliminating negative data points by considering their absolute value and therefore losing potentially important information about the chemistry behind them and applying regular normalization to features that contained negative values. The latter was chosen, since it is not something that would hurt the predictive accuracy of the model in a significant way. Of course, it is better for models to be trained on data that has been normalized according to their respecting distributions, but it would be far more harmful to consider all values as positive. Another possible route would be shifting the distribution of skewed data entirely, instead of avoiding log scaling for features that contained negative values. This could be done through using the absolute values and then adding a number like 1, but that would also potentially distort the original information. Adding to this, some models, like Random Forest can handle even skewed and sparse data, without requiring any pre-processing [25].

Python has ready-made functions, like *MinMaxScaler()* or *MaxAbsScaler()*, that can perform all types of scaling and normalization in an automated way [72]. However, to

have as much control as possible over the outcomes of the processing step, this manual approach was chosen.

Finally, another thing worth mentioning is that, instead of using *log(x)* for log scaling, *log(x+1)* was applied to avoid errors for zero values [73]. In the same spirit as earlier, eliminating zero values could hurt the model by losing meaningful information, so a compromise was made.

3.4.4 Dimensionality reduction

After removing irrelevant features from the dataset, two different dimensionality reduction approaches were explored. The first involved applying PCA to the remaining features. The second was a custom dimensionality reduction strategy that combined correlation analysis with feature selection techniques. Specifically, Pearson correlation coefficients were used to identify and eliminate highly correlated features, followed by the application of both Mutual Information and Random Forest algorithms for feature selection. It is important to highlight that before moving forward with either of these methodologies, the following lists of variables summarized in Table 1 were removed from the data set, because they are properties of the final vitrimers and were only collected for informational purposes. Additionally, during the initial dimensionality reduction process, these data points were used to ensure the dataset was properly adjusted, as correlations between them and the target variables began to emerge, as expected.

Table 1. List of variables dropped for different cases of target variables.

Target variable: v_e	Target variable: T_g	Target variable: T_v	Target variable: GF
'T_{m2} (°C)',	'T_{m2} (°C)',	'T_{m2} (°C)',	'T_{m2} (°C)',
GF (%)',	GF (%)',	GF (%)',	'T_g (°C)',
'SR (%)',	'SR (%)',	'SR (%)',	'SR (%)',
'T_g (°C)',	'v_e (mol/m^3)',	'v_e (mol/m^3)',	'v_e (mol/m^3)',
'*A* (s-1)',	'*A* (s-1)',	'*A* (s-1)',	'*A* (s-1)',
'E_a (kJ/mol)',	'E_a (kJ/mol)',	'E_a (kJ/mol)',	'E_a (kJ/mol)',
'T_v (°C)',	'T_v (°C)',	'T_g (°C)',	'T_v (°C)',
'*E* (MPa)',	'*E* (MPa)',	'*E* (MPa)',	'*E* (MPa)',
'σ_{max} (MPa)'	'σ_{max} (MPa)'	'σ_{max} (MPa)'	'σ_{max} (MPa)'

As previously discussed, PCA is one of the most widely used methods in machine learning pipelines. However, it is inherently linear and may struggle to capture non-linear relationships within the data. For this reason, the alternative strategy was developed and evaluated alongside PCA, with the goal of identifying which method better suits the characteristics of this data set and improves model performance during validation.

This combined correlation analysis and feature selection (CA/FS) methodology is particularly valuable because it addresses both linear relationships- through Pearson correlation analysis- and non-linear dependencies, by leveraging the strengths of filter-based feature selection (MI) and wrapper-based methods (RF).

Principal Component Analysis (PCA)

PCA works by combining multiple features into the minimum possible number of final uncorrelated features called principal components, which are combinations of the previous ones. An important step in PCA, other than correctly setting the target variable and excluding it from the analysis, is to determine how many principal components will be created. Overly restricting the number of dimensions through PCA can lead to significant loss of information, which may in turn reduce the model's predictive performance. This can be especially intense in data sets where features are connected to each other and to the target variable non-linearly, since PCA works with linear connections. Contrarily, allowing for a big number of principal components can lead to overfitting, like high-dimensional problems usually do.

For these reasons, in all four possible cases (different target variables), the number of principal components was set at four. Any less than that led to significant loss in predictive ability, while adding more components (up to six) did not make a positive difference in the results. Scikit-learn in python has a function, *pca.fit_transform(),* that performs the principal component analysis. By extracting the four requested components in excel format, alongside the target variable, the final data set was created, which was then used to train the models.

Finally, a heatmap was created to determine the top thirty features that informed the four principal components used in model training, to allow for result interpretation. This way, it is also possible to compare them to the top features selected by the CA/FS scheme and observe if there are any similarities between them.

Correlation analysis/Feature Selection (CA/FS)

As a first step in reducing the problem's dimensions, Pearson coefficients for every candidate feature were calculated, using the *pd.dataframe.corr()* function from pandas (a python library created for data handling), where *pd.dataframe* takes the name of the dataframe used in a specific problem. The standard correlation analysis is Pearson's,

but different methods can be utilized like Kendall Tau correlation by setting the method inside the brackets as follows: *pd.dataframe.corr(method='kendall')*. To use Pearson, it is not necessary to set the method manually. The difference between the two lies mainly on the fact that Pearson assumes linear, normally distributed variables, while Kendall Tau monotonic variables with no specific distribution.

Pearson correlation is known to perform less reliably than Kendall's tau when variables follow different distributions, but this issue was addressed during the earlier scaling and normalization step [74],[75]. A threshold value of 0.8 was set for both experimental and feature data sets, since correlated features usually need to have a value of at least 0.6, and there was no need for aggressive reduction of the features at this stage. For every pair of features that had a Pearson correlation coefficient above 0.8, the second feature was eliminated from the data set. Following this, the two data sets were merged and another round of correlation analysis was performed, same as before.

The next step was to perform feature selection with the Mutual Information algorithm and the Random Forest algorithm. Calculating MI scores is not always straightforward, but it was made possible by scikit's function *mutual_info_regression(X_numeric, y, random_state=42)*, where *X_numeric* is an array with the features under inspection, *y* is the target variable and *random_state* is a parameter used in many algorithms to control the shuffling of data [72]. Here, the logic of rejecting features is essentially based on the lack of dependency between variables, thereby ranking by feature importance, in contrast to Pearson correlation where features were rejected due to the existence of interdependency. If the computed values were higher than a threshold of 0.25, the corresponding features were included in the data set.

In parallel to this, a simple RF algorithm was deployed to also perform feature selection. Usually, RF algorithms during the feature selection process apply forward selection and/or backward elimination to decide on the usefulness of certain features. Since the goal here is solely removing redundant features, and not training the model to perform predictions, feature importance will be calculated using the *random_forest_regressor* from the scikit-learn library and setting *the number of estimators* to 100 and the *random state* to 42. The first method is characterized as performance-based, while the second as test-based, since the former relies on selecting features to improve the model, whereas the latter examines them in a more statistical and isolated manner [23]. Features with a calculated importance below 0.002 - an empirically determined threshold- were eliminated.

Finally, using the conclusions drawn from choosing features with MI and RF separately, a final choice of features was made based on both algorithms. In other words, the features that were deemed important by both algorithms were included in the final

data set alongside the target variable. Threshold values used in each step for each target variable are summarized in Table 2.

Table 2. Summary of threshold values and resulting features during dimensionality reduction.

Target variable	Final Pearson correlation threshold	Mutual Information threshold	Random Forest threshold
T_g	0.8	0.5	0.008
T_v	0.8	0.3	0.004
GF	0.8	0.25	0.003

3.5 Model training

3.5.1 Data splitting and composition

Before training models, it is necessary to apply a data splitting technique to separate the data set into at least two different sets: one for training and one for testing. Testing sets are kept from the models during training and are only used to draw conclusions about their predictive accuracy and performance. This is how model metrics like determination coefficient (R^2) are calculated.

Scikit's function *KFold(n_splits=5, shuffle=True, random_state=42)* was ultimately utilized to split the data into training and testing sets using 5-fold cross-validation. This process is particularly useful for assessing whether the model generalizes well, during model validation. The 5-fold method essentially means that the training set is divided into five subsets, each containing 20% of the data, and in each iteration, one of these subsets is used as the testing set. This way, models are trained and validated for five different subsets of the data set. This is a common technique to ensure randomness in the data used for training the model, as well as for evaluating whether the model suffers from underfitting or overfitting, especially in smaller data sets. As mentioned, underfitting signals the inability of the model to make predictions while overfitting blocks the model from generalizing well. Both are obviously unwanted and using cross-validation mitigates the risk of them occurring [15].

The final evaluation is based on the accuracy scores calculated during each iteration. In larger datasets, a good practice would be to split the dataset into a training set

(typically 80%) and a testing set (typically 20 %) and then apply k-fold cross-validation within the training set. However, due to the limited amount of data in this case, such a split into training/validation/test sets would be inefficient and unhelpful [6],[15].

3.5.2 Model choice and hyperparameter tuning

The models chosen for this work are commonly used in regression problems, especially Linear Regression, Random Forest, Support Vector Regression and Gaussian Process Regression. Another common one is Neural Networks, but it usually requires a large, diverse database because it is prone to overfitting. This is why NN was not selected for training in the current work. Additionally, to further leverage the strengths of tree-based models and ensemble-based methods, Gradient Boosting and XGBoost were also explored. The scikit library has built in functions for all the models, except for XGBoost, which has a library of its own. Those functions alongside the basic, initial parameters that were used during training are presented in Table 3 for easy overview. There is extensive documentation available for all the possible hyperparameters that can be set in every model [72],[76]. Choosing the right parameters is very important because they directly affect the ability of a model to make predictions.

Table 3. Models trained with their respective python functions.

Model	**Python function with hyperparameters**
Linear Regression	*LinearRegression()*
Random Forest	*RandomForestRegressor(random_state=42)*
Support Vector Regression	*SVR()*
Gaussian Process Regression	*GaussianProcessRegressor(kernel=RBF(), random_state=42)*
Gradient Boosting	*GradientBoostingRegressor(random_state=42)*
XGBoost	*XGBRegressor(random_state=42, verbosity=0)*

Hyperparameter tuning is an important step in machine learning pipelines, which optimizes models' performance. Instead of manually tuning hyperparameters, which would take both time and effort and would probably still be inefficient due to the sheer volume of available hyperparameters, the grid search method was applied. This allows the investigation of a range of values for desired hyperparameters in an automated way, where the hyperparameters are chosen based on which result in the best outcomes for criteria like R^2, mean squared error (MSE), *etc.* During the grid search,

scores of cross-validation for hyperparameters are calculated and used to determine the final values that need to be used in model training [22].

The scikit function *GridSearchCV()* with a 3-fold cross validation was utilized to compute the desired hyperparameters. A 3-fold validation is a good starting point that shows balance between fair predictions and moderate computational times. For each model, a tailored set of hyperparameters was selected based on common practice and recommended ranges found in the scikit-learn documentation and applied machine learning literature [72],[76],[77]. The grids that are presented in Table 4 were designed to balance model complexity, training time, and generalization performance, focusing on the most influential parameters for each algorithm.

Table 4. Ranges of hyperparameters used during grid search for all models.

Model	**Range of hyperparameters**
Random Forest	n_estimators: [100, 200] max_depth: [5, 10, None] min_samples_split: [2, 5]
Gradient Boosting	n_estimators: [100, 200] learning_rate: [0.05, 0.1] max_depth: [3, 5] max_features: ['auto', 'sqrt']
Support Vector Machine	C: [0.1, 1, 10] epsilon: [0.01, 0.1] kernel: ['rbf', 'linear']
Gaussian Process	kernel: [ConstantKernel(1.0)*RBF(10)+WhiteKernel(0.1), ConstantKernel(0.1)*RBF(1)+WhiteKernel(0.01)]
XGBoost	n_estimators: [100, 200] learning_rate: [0.05, 0.1] max_depth: [3, 5, 7]
Linear Regression	None

More specifically, for Random Forest and Gradient Boosting models, the parameters *n_estimators*, *max_depth*, and *min_samples_split* (or *learning_rate* and *max_features*, respectively) were tuned, as they directly control the model's capacity to learn patterns

while managing overfitting. In the case of Support Vector Regression, variations in the *C* and *epsilon* parameters allowed exploration of different levels of regularization and error tolerance, while both linear and RBF kernels were tested to account for both linearity and non-linearity between data. SVR models always try to balance model complexity and data fitting. For Gaussian Process Regression, two contrasting kernel structures were selected to test how assuming smooth or noisy data affects the model. In the case of XGBoost, which includes built-in regularization, deeper trees (up to depth 7) were included in the grid to take advantage of its robustness. Finally, no hyperparameter tuning is available for Linear Regression.

Hyperparameter tuning was applied to all target variables except gel fraction, as none of the models were able to predict it adequately from the start. Also. tuning was performed only on the dimensionality reduction scheme that yielded the best predictive performance in preliminary testing.

3.6 Model validation and benchmarking

In regression problems, there are several metrics that can be used to assess the model's performance. Here, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Coefficient of Determination (R^2) were considered. RMSE and MAE measure the difference between predicted and actual values, while R^2 gives insight into how much of the variance that occurs in the dependent variable can be explained by the independent variable. In fact, R^2 shows the distance between the predicted values and the regression line. Lower values of RMSE and MAE and R^2 values close to 1 indicate a model with very good performance [2]. These metrics were calculated automatically through Python using the following functions: *np.sqrt(mean_squared_error()*, *mean_absolute_error()* and *r2_score()*, respectively, and the final results were stored in excel for easier handling and comparison between models, dimensionality reduction schemes and hyperparameter tuning efforts.

Chapter 4: Results and Discussion

4.1 Data processing and feature selection

4.1.1 Outlier detection

The three measures that were taken instead of outlier detection- dropping univariant features and features with very little variance and applying a scarcity filter- proved to be very useful in cleaning up the feature data set. More specifically, they helped reduce the number of descriptors from the original 217 to 131, which goes to show just how important they were in eliminating redundant values.

4.1.2 Dimensionality reduction

Using PCA as the sole dimensionality reduction technique resulted in the extraction of four principal components, each representing a combination of features. To better understand which original variables contributed most significantly to these components, heatmaps were generated to visualize the top 30 most influential features for each target variable. Heatmaps for crosslink density, glass transition temperature, topology freezing temperature and gel fraction are shown in Figures 14 to 18, respectively. The way heatmaps work is the following: warmer colors (*i.e.* red) indicate a stronger connection between the feature (*y*- axis) and the principal component (*x*-axis), while colder (*i.e.* blue) indicate a lack of influence of the principal component by the respective feature. The values given (for example 0.18 in the first column and line) help quantify how much a feature "participates" in a principal component.

A clarification of the feature naming convention is essential: features ending in ".1" correspond to the first compound used in each experiment, those ending in ".2" to the second compound, ".3" to the third compound, and features ending in "." (with no number) represent the catalyst.

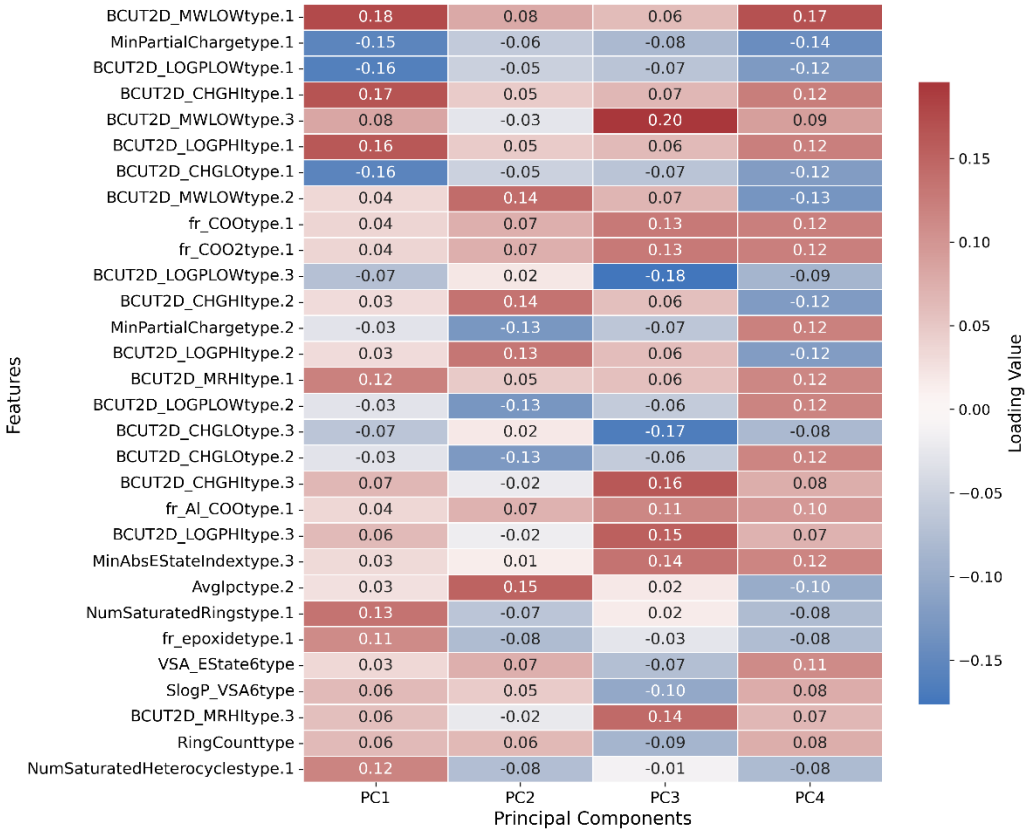


Figure 14. PCA heatmap of principal components for v_e.

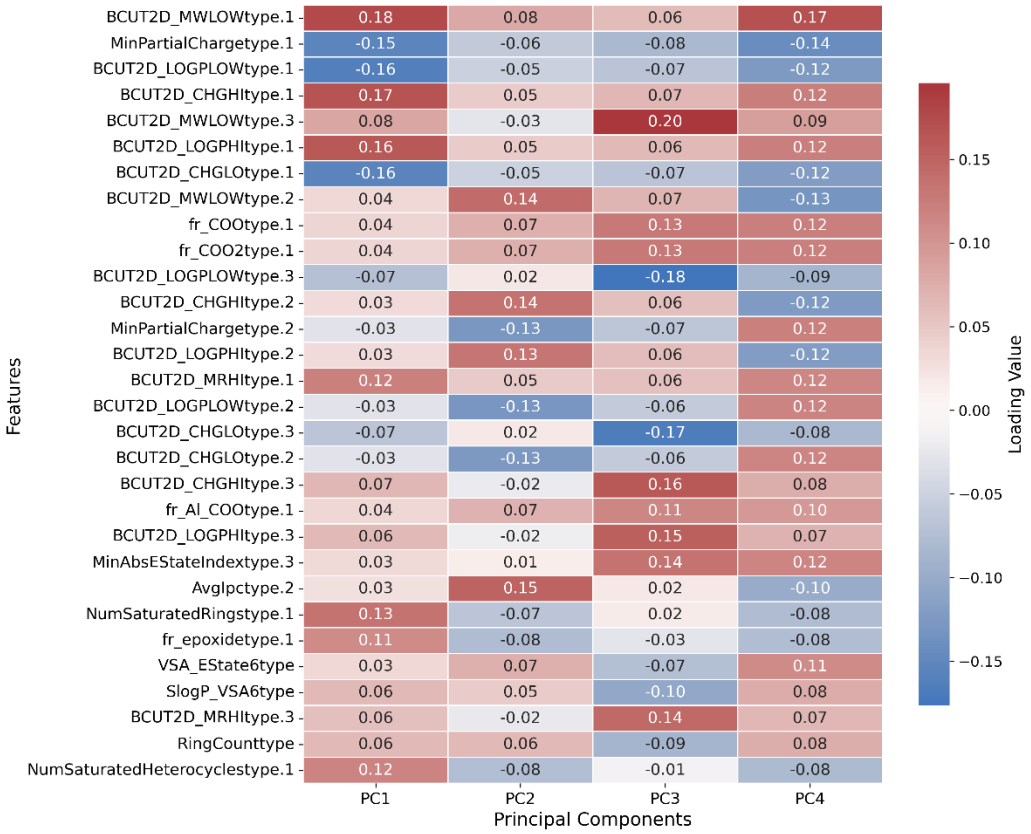


Figure 15. PCA heatmap of principal components for T_g.

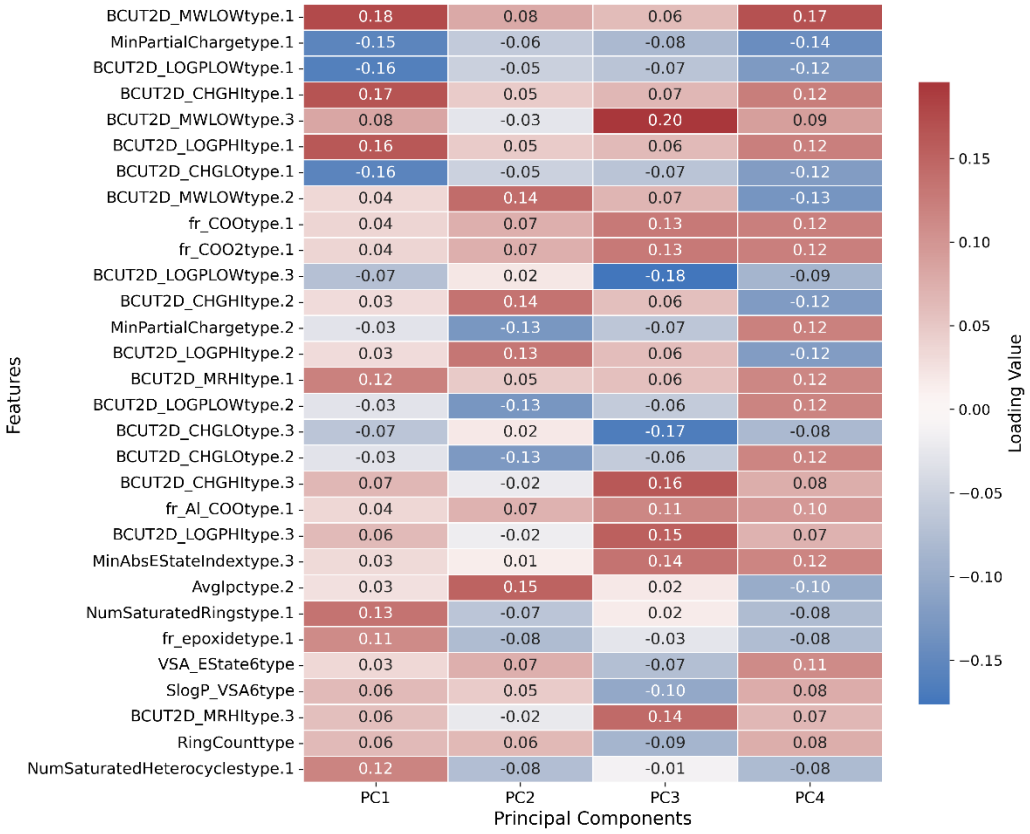


Figure 16. PCA heatmap of principal components for T_V.

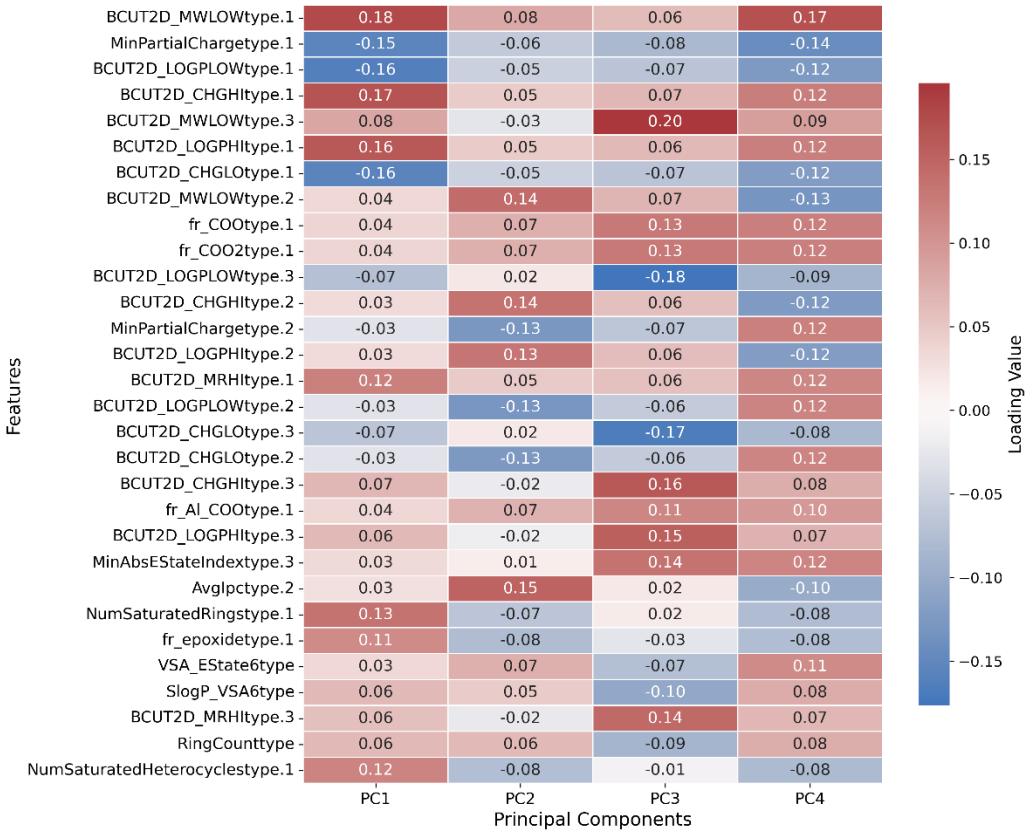


Figure 17. PCA heatmap of principal components for GF.

The top 30 contributing features to the 4 final principal components are the same across all four target variables. This is interesting and it gives insight into how PCA really works. PCA does not consider the target variable and therefore does not examine the connections between input and output. This is why the heatmaps are identical across targets. PCA is based solely on the variance that each feature presents, which is both a strength and a weakness. This helps reduce the dimensions dramatically, but it can create situations where influential features (based on domain knowledge) are not considered important due to low variance. At the same time, PCA does not capture non- linear connections well, which can also lead to some problems. The absence of features like "Reactive end group ratio", "Catalyst molar ratio", "T", *etc.*, which are known to be important for the properties of the final vitrimer shows that PCA indeed fails to capture what is already known theoretically. Considering the fact that PCA relies on variance, it makes sense that features collected manually from literature were not influential because some values were missing and had to be imputed, which reduced their variance.

As far as the custom-made CA/FS dimensionality reduction scheme goes, applying Pearson's correlation analysis with a threshold of 0.8 for the case of crosslink density as the target value helped reduce the descriptors in the feature data set from 131 to only 36. Contrarily, no features were dropped in the experimental data set. After applying the analysis once more on the merged data set, the initial 155 features went down to 112. These features were the target for the next step of this dimensionality reduction scheme that involved combining MI and RF algorithms to perform feature selection.

In the Mutual IInformation algorithm, the threshold, 0.25, was chosen by experimenting with the number of final features selected. For crosslink density, the number was 42. These features and their MI scores can be seen in Table 5.

On the other hand, choosing a threshold of 0.002 in Random Forest resulted in 28 features that can be seen in Table 5 alongside features chosen by Mutual Information.

Table 5. Features chosen for v_e by MI (left) and RF (right), ranked by importance with their respective scores.

MI algorithm		**RF algorithm**	
Feature	**Score**	**Feature**	**Score**
AvgIpctype.1	0.618	EState_VSA2type.2	0.712
MaxAbsEStateIndextype.1	0.573	PEOE_VSA10type	0.109
MinEStateIndextype.1	0.523	Reactive end-groups ratio	0.025
MinAbsEStateIndextype.1	0.517	FractionCSP3type.1	0.023

BalabanJtype.1	0.512	MW1 (g/mol)	0.010
SPStype.1	0.512	MinEStateIndextype.2	0.009
T (oC)	0.502	MW (g/mol)	0.008
MW2 (g/mol)	0.495	T (oC)	0.006
PEOE_VSA14type.1	0.414	Catalyst molar ratio (%mol)	0.005
FractionCSP3type.1	0.406	qedtype	0.005
Catalyst molar ratio (%mol)	0.406	SPStype.1	0.004
SPStype	0.405	EState_VSA5type	0.004
Kappa3type	0.397	SMR_VSA6type	0.004
MW1 (g/mol)	0.394	BalabanJtype	0.004
MinAbsEStateIndextype	0.389	MaxAbsEStateIndextype	0.004
MaxAbsEStateIndextype	0.384	fr_NH0type.3	0.003
VSA_EState8type.1	0.370	VSA_EState4type	0.003
AvgIpctype	0.370	BalabanJtype.1	0.003
MW (g/mol)	0.362	PEOE_VSA2type.1	0.003
VSA_EState8type	0.357	SlogP_VSA1type	0.003
qedtype	0.356	MinPartialChargetype	0.003
BCUT2D_MRLOWtype	0.352	MinAbsPartialChargetype.1	0.003
MinEStateIndextype	0.343	SPStype	0.002
EState_VSA2type.1	0.336	EState_VSA7type	0.002
Reactive end-groups ratio	0.328	AvgIpctype.1	0.002
BalabanJtype	0.322	AvgIpctype	0.002
VSA_EState8type.3	0.305	Kappa3type	0.002
FractionCSP3type	0.301	SMR_VSA10type	0.002
SMR_VSA10type.1	0.295	-	-
MinEStateIndextype.3	0.293	-	-
MinAbsPartialChargetype.1	0.287	-	-
MaxPartialChargetype	0.281	-	-
BCUT2D_MRLOWtype.1	0.278	-	-
SMR_VSA6type.1	0.271	-	-
MinAbsPartialChargetype.3	0.266	-	-
SMR_VSA10type.3	0.255	-	-
EState_VSA6type.3	0.253	-	-
VSA_EState8type.2	0.253	-	-
FractionCSP3type.3	0.252	-	-
SlogP_VSA3type	0.252	-	-
EState_VSA4type.3	0.251	-	-
MinAbsPartialChargetype	0.250	-	-

Mutual Information seems to consider more features important than Random Forest does. MI explores intricate connections between inputs and output, while RF uses features in its training process and prioritizes them based on how truly important they are for prediction accuracy. This is why in most cases, RF was stricter than MI, which can be observed in Table 5, as well as Table 6. Moreover, MI tends to interpret non-

linear connections better than RF and this is another reason why more MI- derived features are shown in Table 5.

Besides this, it is apparent that MI and RF attribute different importance to the same features. MI scores always refer to a pair of values: a feature and the target variable. RF looks at features more holistically and picks up on how some features can be influential when combined with others, which is why some low rated MI features are highly rated in RF. Finally, RF is a more robust model, capable of handling noise and outliers better than MI. For this reason, RF may disregard some noisy features that MI considers important.

The last step of this process was to combine both algorithms and extract the final features. In other words, the 16 features that had an MI value above 0.25 and an RF value above 0.002 were the ones that made up the final data set used for model training. Table 6 summarizes thresholds and final feature numbers for the other three variables, while Table 7 includes the names of the final features selected per variable, for easier comparison between this method and PCA.

Table 6. Summary of threshold values and resulting features during dimensionality reduction.

	Final Pearson correlation		**Mutual Information**		**Random Forest**		**MI & RF combination**
Target variable	**Final features**	**Threshold set**	**Final features**	**Threshold set**	**Final features**	**Threshold set**	**Final features**
T_g	115	0.8	32	0.5	26	0.008	17
T_v	115	0.8	33	0.3	27	0.004	16
GF	115	0.8	28	0.25	35	0.003	16

In general, the thresholds set were quite similar across targets, except for T_g where stricter limits had to be applied to reduce features and bring down the dimensions of the problem. In the case of T_g, it seems that more features were correlated to it and could chemically describe it. This is reasonable since the connection between polymer chemistry and T_g is a well explored one in research.

Table 7. Summary table of final features for model training selected for all variables.

Final features for v_e	**Final features for T_g**	**Final features for T_v**	**Final features for *GF***
AvgIpctype	AvgIpctype.1	AvgIpctype.1	AvgIpctype.1

AvgIpctype.1	BCUT2D_MRLOWtype.1	BalabanJtype.1	BCUT2D_MRLOWtype.1
BalabanJtype	BalabanJtype.1	FractionCSP3type.1	Catalyst molar ratio (%mol)
BalabanJtype.1	EState_VSA2type.1	Kappa3type	FractionCSP3type.1
Catalyst molar ratio (%mol)	FractionCSP3type.1	MW2 (g/mol)	Kappa3type
FractionCSP3type.1	Kappa3type	MinAbsEStateIndextype.1	MW (g/mol)
Kappa3type	MW (g/mol)	MinAbsPartialChargetype.1	MW1 (g/mol)
MW (g/mol)	MW1 (g/mol)	MinEStateIndextype.1	MW2 (g/mol)
MW1 (g/mol)	MW2 (g/mol)	PEOE_VSA8type	MaxAbsEStateIndextype.1
MaxAbsEStateIndextype	MaxAbsEStateIndextype.1	SMR_VSA4type	MinAbsPartialChargetype.1
MinAbsPartialChargetype.1	MinAbsEStateIndextype.1	SMR_VSA6type	MinAbsPartialChargetype.3
Reactive end-groups ratio	MinEStateIndextype.1	SMR_VSA6type.1	MinEStateIndextype
SPStype	PEOE_VSA14type.1	SPStype	MinEStateIndextype.1
SPStype.1	SMR_VSA10type.1	SPStype.1	PEOE_VSA14type.1
T (°C)	SPStype.1	SlogP_VSA3type	SMR_VSA10type.3
qedtype	T (°C)	T (°C)	T (°C)
	VSA_EState8type.1		

Here, contrastingly to PCA, almost no common features are selected across all four target variables. There are similarities but universally, the only common features are "T (°C)" and "AvgIpctype.1", which are the reaction temperature and a topological index that refers to the first compound. Other than that, it is apparent that all targets are strongly connected to the nature and characteristics of the first compound, which makes sense since this was usually the main building block of the final vitrimer.

The variety of features can be explained by the fact that this custom methodology takes the target variable into account and explores connections between features through this lens. This means that some features may matter for one target and not matter for another, like electronic descriptors (*Estate* indices) for T_v and not crosslink density, which is influenced more heavily by topological descriptors (*Balaban* indexes), and so on [18]. Additionally, some properties such as gel fraction may be subject to higher experimental variability, which can obscure weaker associations and reduce the robustness of certain features during selection. This introduces additional differentiation in the final sets of descriptors.

As mentioned, each target variable is governed by different underlying mechanisms. Some examples are the following:

- Crosslink density is closely related to the number and nature of reactive end-groups, monomer structure, and steric accessibility.

- T_g is influenced by chain flexibility, intermolecular interactions, and molecular weight.
- T_v varies depending on dynamic bonds and activation energy, which is influenced by the nature and quantity of catalyst used.
- Gel fraction depends on the extent of crosslinking reactions.

Based on Table 7, crosslink density seems to be heavily influenced by the catalyst and depend only on the nature and quantity of the first compound. This makes sense since the first compound was usually the crosslinker used in the experiments and the catalyst can influence the extent of crosslinking reactions and therefore network formation.

T_g also seems to be influenced mainly by the first component's characteristics. The presence of "MW" and "MW2" signals that the size of the catalyst and the second compound is also important in terms of this characteristic temperature. Similarly to crosslink density, the catalyst affects the extent of the crosslinking reactions, shifting T_g indirectly. The effect of the molecular weights of the catalyst and the second compound can be explained through the network that is created. The second compound in combination with the first can create bigger or smaller chains and the catalyst dictates how many links will be created.

The main features used to predict T_v exhibit similar trends to T_g, with one differentiating quality. Here, chemical descriptors of the catalyst seem to be very important in making informed predictions. This is probably due to the catalyst directly influencing bond exchange efficiency, which in turn affects T_v.

Finally, features for gel fraction overlap with the ones for crosslink density and T_g with the addition of 2 chemical descriptors from the third compound. This suggests a discrepancy because chemical understanding suggests that gel fraction and crosslink density represent similar physical characteristics. Thus, they would be expected to rely on similar features. This suspicion was later verified during model training, where models indeed failed to predict gel fraction.

4.2 Model performance results

4.2.1 Comparison between dimensionality reduction methods

As mentioned, two different dimensionality reduction methods were explored and the features dictated by them were used for model training. Afterwards, RMSE, MAE and R^2 were calculated for each model and each method. A comparison between the two methods and the resulting metrics is made in Tables 8 through 11. The R^2 values that are depicted in the tables refer to the average score across folds in cross validation and it estimated how well models predict unseen data. There is also the overall R^2, which is

calculated using the entire data set. This is not an entirely trustworthy metric and that is why it will only be used as a reference point in parity plots later on.

It is worth noting that PCA consistently performed better than the CA/FS method only for crosslink density. This likely stems from the fact that crosslink density is more closely related to more general structural patterns -such as molecular size and functional group density- which are effectively captured by the principal components. In contrast, the other target variables could be more sensitive to specific chemical substructures and localized interactions, which may be lost during PCA transformation. As a result, the second method, which retains more chemically interpretable features, led to improved model performance for those cases.

Furthermore, using the predicted values each model calculated as well as the true values from the original data set, three types of plots were created to better visualize the performance of each model. Box plots that show the average values of all metrics calculated per model, with their respective standard deviation, were created for a quicker overview of the results. High standard deviations show that the model is unstable during different cross validation folds. This means that even if the respective metric is good, the predictions are not fully trustworthy because they are dependent on the data set that is used for training the model.

Models with negative R^2 values (*) were not represented in the box plots, since they are of no use in predicting the target. Such a value bears no physical meaning and it only shows that models completely fail to make predictions. R^2 values are expected to be between 0 and 1.

Additionally, parity plots were created for all target variables and models, but only the models with the highest two R^2 scores will be presented here. The rest of the parity plots can be found by following the link in the Appendix. To make them more interpretable, the normalization that was applied in the beginning of the workflow was undone, using equation (3.5). In interpreting parity plots, it is important to remember that the closer the points are at the diagonal line, the better the predictions are.

Finally, error distribution plots were plotted per model. The latter plots residuals ($y_{pred} - y_{true}$) against their frequency of occurrence. The error distribution plots presented here will only refer to the best model per target. Comparing between models for the same target variable is possible using the rest of the plots in the Appendix.

They are interesting tools that help check for model bias and diagnose outliers. More specifically, a zero- centered distribution means the model is unbiased, which means it does not consistently over/under- predicts certain values. At the same time, the wider the distribution is, the higher the variance is in prediction. Finally, big tails in distribution highlight outliers.

Crosslink density

For the prediction of crosslink density, several trends can be observed across the different models and feature selection methods. In general, ensemble models such as Random Forest, Gradient Boosting, and XGBoost achieved the best performance, with R^2 values exceeding 0.60 when PCA-derived features were used. More specifically, Random Forest with PCA proved to be the best performing model with an R^2 of 0.77. Though in a typical regression this might seem like an inadequate result, relevant literature shows that R^2 values for ML models typically are between 0.6 and 0.9, making this a very promising result [6].

On the contrary, Gaussian Process models performed extremely poorly, showing a highly negative R^2 value. The same goes for Linear Regression when using the CA/FS methodology. These negative values suggest a mismatch of the models with the structure of the data, and so they are deemed incapable of predicting this variable.

When comparing the two dimensionality reduction methods, PCA consistently led to better or equal performance across most models, particularly in terms of R^2 and RMSE. This indicates that PCA captures the most predictive variance for this specific target, as previously mentioned.

These findings are summarized in Table 8 and visualized in the accompanying boxplots for a clearer comparison across models and error metrics.

Table 8. Mean performance table with metrics for v_e.

Model	R^2 Mean		RMSE Mean		MAE Mean	
	PCA	CA/FS	PCA	CA/FS	PCA	CA/FS
Linear Regression	0.22	-0.24 (*)	0.13	0.15	0.07	0.09
Random Forest	0.77	0.57	0.08	0.09	0.03	0.03
Support Vector Machine	0.50	0.32	0.11	0.12	0.07	0.07
Gaussian Process	-54.81 (*)	-54.81 (*)	0.86	0.86	0.79	0.79
Gradient Boosting	0.73	0.67	0.09	0.09	0.03	0.03
XGBoost	0.60	0.60	0.10	0.10	0.03	0.03

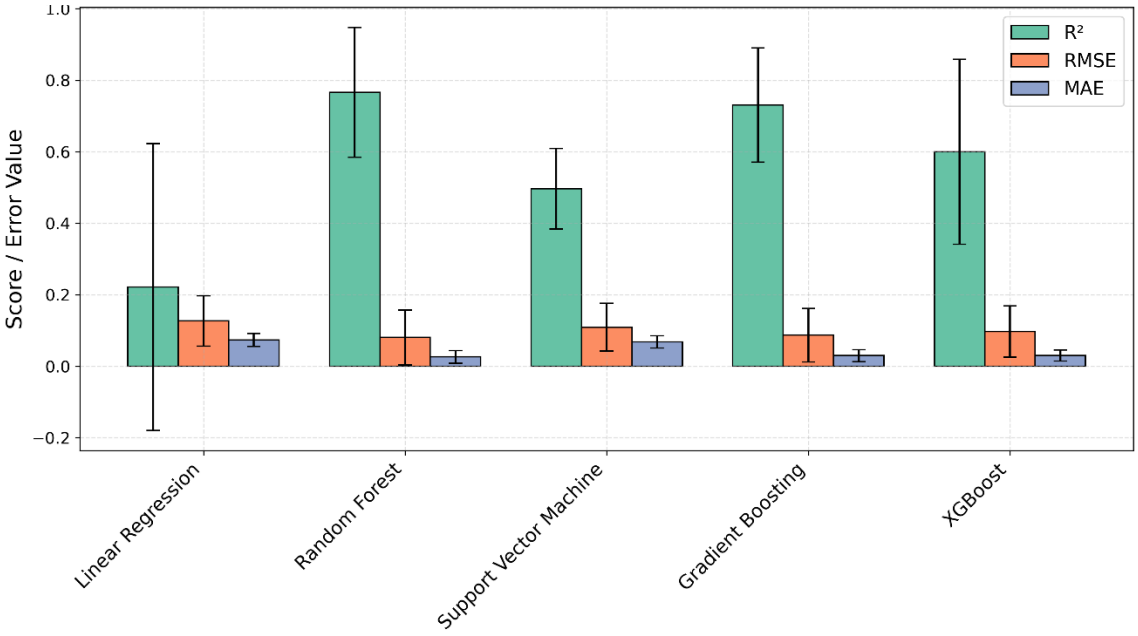


Figure 18. Model performance comparison (mean+std) for v_e with PCA methodology.

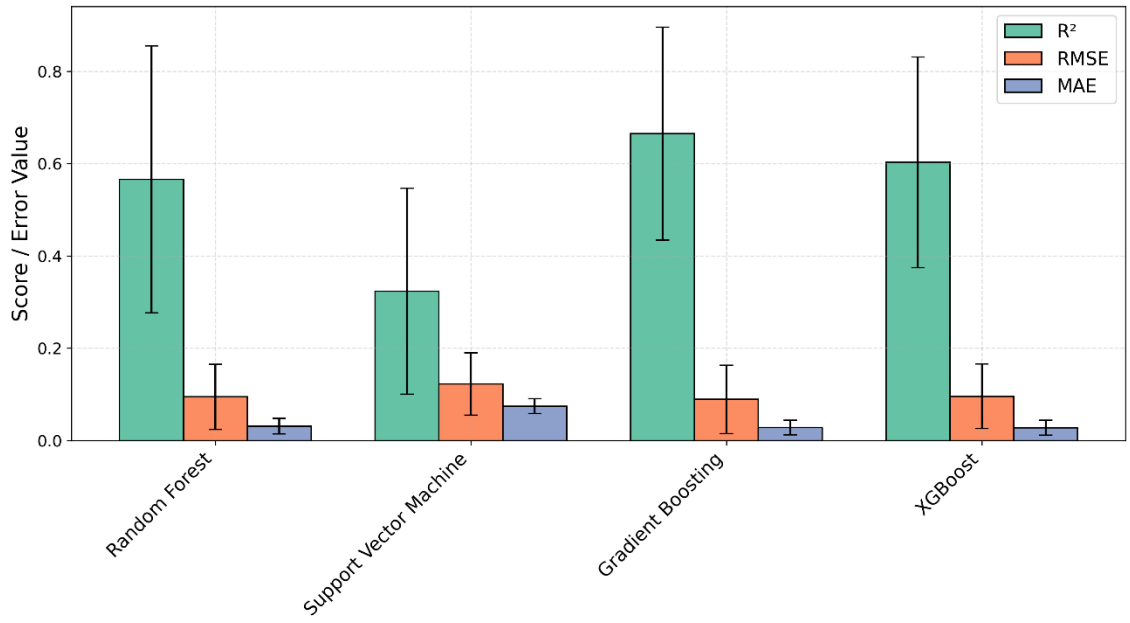


Figure 19. Model performance comparison (mean+std) for v_e with CA/FS methodology.

As shown in Table 8, Random Forest outperformed all other configurations, achieving an R^2 of 0.77 and RMSE of 0.08, while maintaining a very low MAE of 0.03. Notably, Gradient Boosting also performed competitively with R^2 = 0.73 and similar RMSE/MAE values. These two models show the greatest promise in predicting the crosslink density and will be later tuned to see if they can achieve an even better performance. It is worth noting that even though XGBoost is the third better model, the high standard deviation it presents in R^2 shows it is adequate but not reliable.

In contrast, Gaussian Process models produced nonsensical R^2 values (≈ –54.8), for both dimensionality reduction methods, which goes to show that the model itself is not necessarily incapable of making predictions, but the kernel that is chosen during training is not the appropriate one. Linear regression is inappropriate for predicting this target since it has a low R^2 and a high standard deviation in this value.

As far as the CA/FS strategy is concerned, all models perfomed significantly worse, with Gradient Boosting being the best one here, with R^2 = 0.67, but very high standard deviation. The rest of the models also seem to be unstable in their predictions, which is why this strategy is considered ineffective in reducing dimensions and helping models perform better.

Parity plots for Random Forest and Gradient Boosting, respectively, can be seen in Figure 20 where the predicted values from the best performing models across both methodologies are plotted against the actual values. The two are visually similar, but not much can be said about the distribution of values. The existence of some extreme values (for example crosslink density=36000 mol/m^3) has made it hard to see clearly how the lower values are distributed.

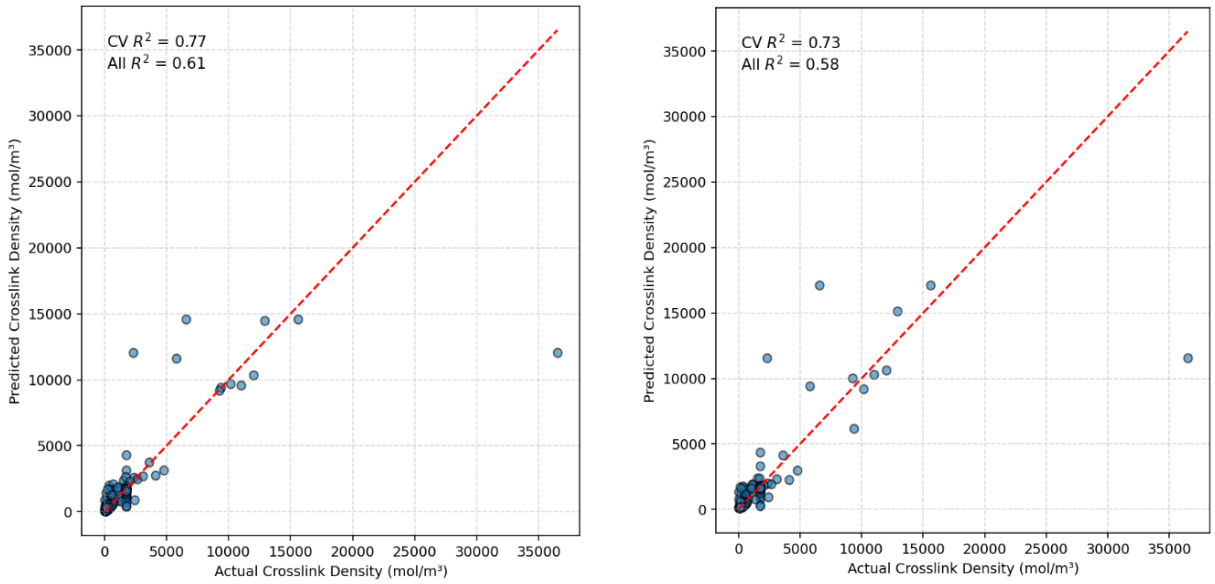


Figure 20. Parity plots for v_e with a) Random Forest (left) and b) Gradient Boosting (right) models.

Both models seem to perform reasonably well, with values being close to the diagonal. However, a noticeable overestimation occurs for samples with high experimental values, particularly above 5000 mol/m^3. Despite the spread, most predictions remain clustered around the lower density range, indicating that both models perform better in this region of the data space. The overestimation is reflected in the lower overall R^2

values, especially for the Gradient Boosting model (R^2 = 0.58). The Random Forest model achieves a slightly better fit (R^2 = 0.61) and higher cross-validation score (R^2 = 0.77), suggesting more stable performance across folds. The difference between the two types of R^2 can be explained by the fact that during cross-validation, each fold has a slightly different training/testing split. If one fold fits particularly well, it can inflate the average CV R^2 — especially in small datasets. The lower overall R^2 value is indicative of the fact that overfitting was avoided during cross validation, and it is not a reason for concern.

The extreme values that can be seen in Figure 20 were not considered outliers, even though they differed significantly from the rest of the values. This is because they simply reflect different crosslinking densities achieved during experimentation. At the same time, their exclusion risked introducing bias into the models, since a high variance in this value is to be expected. Since vitrimer properties remain generally unmodeled, every piece of information is considered important.

Finally, an error distribution plot was created for Random Forest, which performs the best both in terms of R^2 and RMSE/MAE. This plot reinforces what has already been said about the performance of RF.

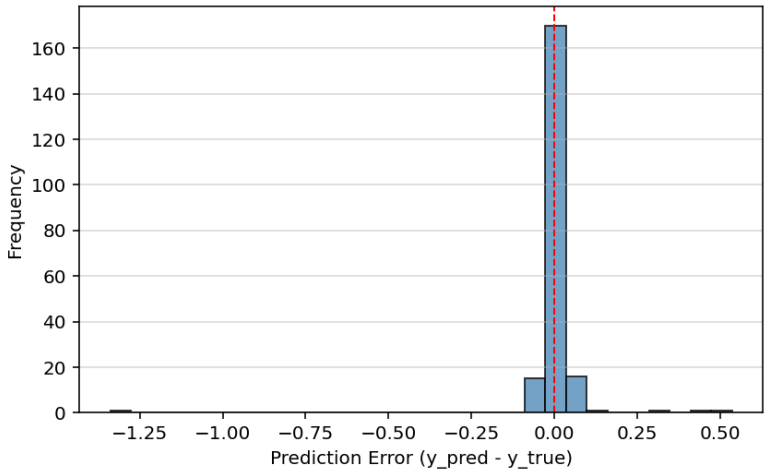


Figure 21. Error distribution plot for v_e with Random Forest model.

More specifically, the distribution presented in Figure 21 is tight, which shows how little errors are made during predictions, and centered around zero, which proves the model is performing well. The only thing worth mentioning is that there is a bit of residual values present, especially to the far left of the plot, suggesting that the model failed to predict them. The exclusion of these datapoints could potentially improve model performance. At the same time, adding more data points, especially in the range where the model fails, and widening the range of values could benefit model performance.

Glass transition temperature

For the prediction of T_g, the main trend that can be observed is that models seem to perform really well, and especially the ones trained with the CA/FS method. In general, ensemble models such as Random Forest, Gradient Boosting, and XGBoost achieved the best performance, with R^2 values exceeding 0.8. More specifically, Random Forest with the CA/FS method proved to be the best performing model with an R^2 of 0.85. In similar research, polymer T_g has been successfully predicted (R^2 = 0.90, RMSE = 34.28) using a Random Forest model with molecular descriptors as features instead of RDKit-based descriptors [6]. This goes to show how well the results actually are, compared to existing research.

Contrary to crosslink density where Gaussian Process was by far the worst performing model, here its predictive accuracy significantly improved, making it almost equally good as the top performers. Even Linear Regression performed adequately suggesting that the relationship between descriptors and T_g is not very complex.

When comparing the two dimensionality reduction methods, PCA consistently led to worse performance across all models and all metrics. This indicates that PCA fails to capture the chemically important descriptors and produces principal components with little connection to the target variable. CA/FS, on the other hand, could distinguish which features mattered for predicting T_g and kept them during training, which lead to accurate results.

These findings are summarized in Table 9 and visualized in the accompanying boxplots for a clearer comparison across models and error metrics.

Table 9. Mean performance table with metrics for T_g.

Model	R^2 Mean		RMSE Mean		MAE Mean	
	PCA	CA/FS	PCA	CA/FS	PCA	CA/FS
Linear Regression	0.08	0.58	0.44	0.30	0.36	0.24
Random Forest	0.78	0.85	0.21	0.18	0.12	0.10
Support Vector Machine	0.16	0.69	0.42	0.26	0.29	0.18
Gaussian Process	-0.24 (*)	0.77	0.52	0.22	0.41	0.12
Gradient Boosting	0.77	0.85	0.22	0.17	0.13	0.09
XGBoost	0.75	0.83	0.23	0.18	0.12	0.09

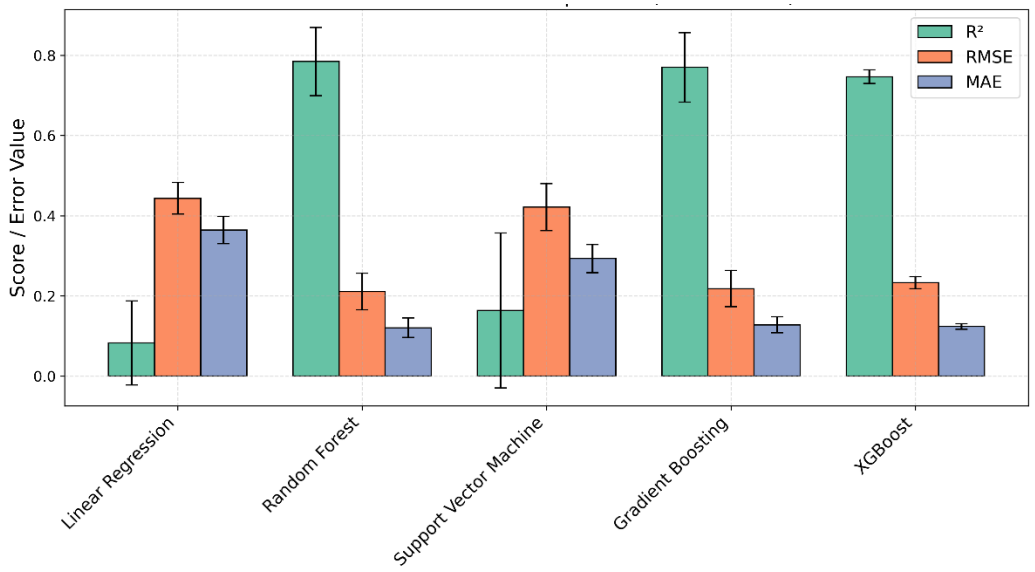


Figure 22. Model performance comparison (mean+std) for T_g with PCA methodology.

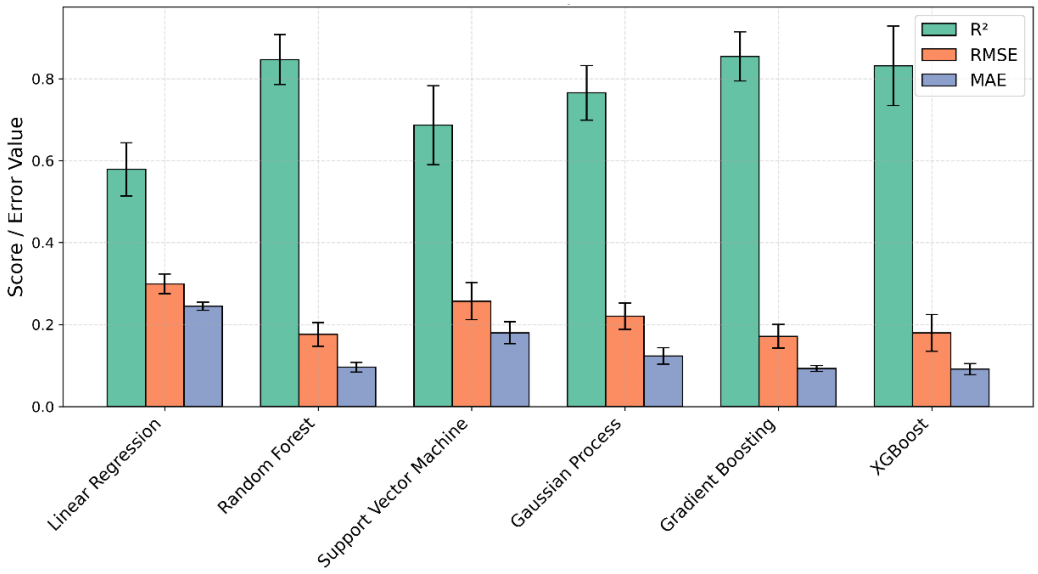


Figure 23. Model performance comparison (mean+std) for T_g with CA/FS methodology.

According to Table 9, Gradient Boosting outperformed all other configurations, achieving an R^2 of 0.85 and RMSE of 0.17, while maintaining low MAE of 0.09. Notably, Random Forest also performed competitively with R^2 = 0.85 and marginally higher RMSE/MAE values. Both have low standard deviation in R^2 values, which is very promising as far as generalization goes. These two models show the greatest promise in predicting the T_g and hyperparameter tuning later will prove if they can perform

even better than this. XGBoost is the third best model (R^2 = 0.83, RMSE = 0.18), with a slightly higher standard deviation.

Furthermore, Gaussian Process performs well with CA/FS (R^2 = 0.77) but produces nonsensical R^2 values (R^2 = -0.24) when trained on PCA derived features, as does Support Vector Machine (R^2 = 0.69 versus R^2 = 0.16). Linear Regression also performs adequately in the first case (R^2 = 0.58) but fails when using PCA (R^2 = 0.08).

All in all, PCA destabilizes performance across all models, decreasing R^2 and increasing RMSE and MAE values, as well as standard deviations. The best model was Random Forest with an R^2 value of 0.75. PCA for predicting T_g will not be investigated further.

Parity plots for Random Forest and Gradient Boosting, respectively, can be seen in Figure 24 where the predicted values from the best performing models across both methodologies are plotted against the actual values. The two are visually very similar, which is to be expected since the two models have the same R^2 value.

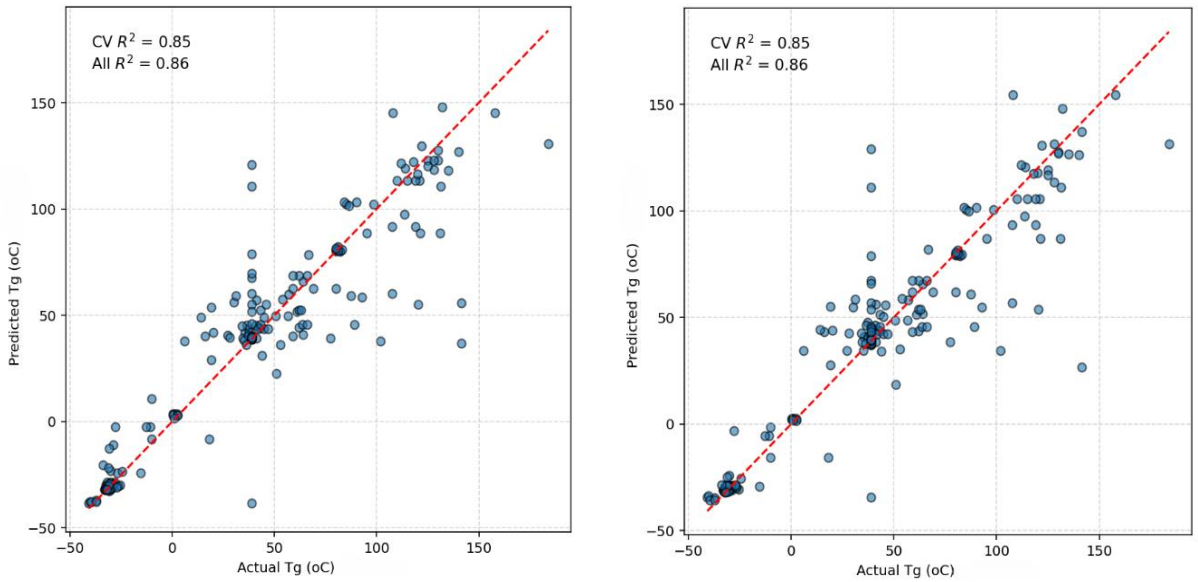


Figure 24. Parity plots for T_g with a) Random Forest (left) and b) Gradient Boosting (right) models.

The behavior of both models is very similar, with most accurate predictions happening for T_g values near -40 and 40 °C, as well as some close to 120 °C, where most points aggregate close to the diagonal. Some underestimations occur on the right side of the line, and there is a vertical line around 48 °C, probably indicating imputed values. These values have the same *x* but different *y* coordinates, suggesting that the models made different predictions for the same value. CV and overall R^2 values (0.85 and 0.86, respectively) are identical in the two models. The slightly higher overall R^2 value seems to indicate that cross validation, while mitigating overfitting, marginally hindered the

models from achieving the best performance possible. Additionally, the fact that the two types of R^2 values are so close could mean that the models are very stable across different folds and are, therefore, indifferent to the kind of data they are given.

Finally, an error distribution plot was created for Gradient Boosting, since it has a lower RMSE and MAE than Random Forest, which makes it slightly better.

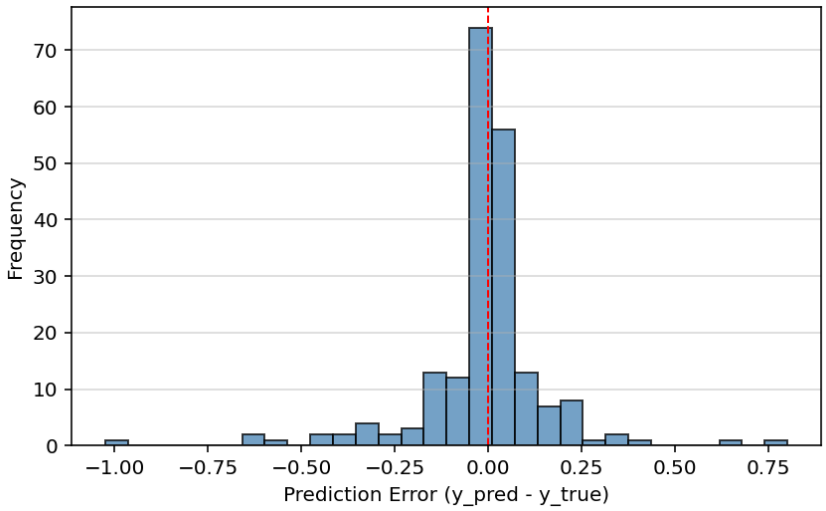


Figure 25. Error distribution plot for T_g with Gradient Boosting model.

The observed distribution is narrow, close to normal and centered around zero, all of which are indicative of a good model. There is a systematic underappreciation of predicted values, which can be seen by their higher frequency of appearance in contrast to accurate predictions. These errors are not that significant, so they are no cause for concern, and they could represent the imputed values. The tails of the distribution, however, highlight some observations, although few, that the model fails to predict correctly, either by overestimating them (right) or underestimating them (left). This could simply be a matter of such values lying at the ends of the overall range in the original data set and could be mitigated by adding similar datapoints that the model could be trained on.

Topology freezing temperature

When predicting T_v, ensemble models were clearly superior, with R^2 values exceeding 0.68. More specifically, CA/FS Random Forest proved to be the best performing model with an R^2 of 0.70.

Similarly to T_g, Gaussian Process was amongst the better performing models. Linear Regression performed better than it did in predicting crosslink density, which could mean that the relationship between descriptors and T_v is not that complex.

When comparing the two dimensionality reduction methods, PCA consistently led to worse performance across all models and all metrics, just like previously. PCA, once more, produces principal components with little connection to the target variable. This is why it was so important to have a second, completely different strategy available, in case PCA harmed model training.

These findings are summarized in Table 10 and visualized in the accompanying boxplots for a clearer comparison across models and error metrics.

Table 10. Mean performance table with metrics for T_V.

Model	R^2 Mean		RMSE Mean		MAE Mean	
	PCA	CA/FS	PCA	CA/FS	PCA	CA/FS
Linear Regression	-0.03 (*)	0.33	0.32	0.26	0.20	0.19
Random Forest	0.63	0.70	0.19	0.17	0.11	0.09
Support Vector Machine	0.33	0.62	0.26	0.19	0.17	0.14
Gaussian Process	-0.24 (*)	0.61	0.35	0.20	0.27	0.10
Gradient Boosting	0.63	0.69	0.19	0.17	0.11	0.09
XGBoost	0.53	0.69	0.22	0.17	0.12	0.09

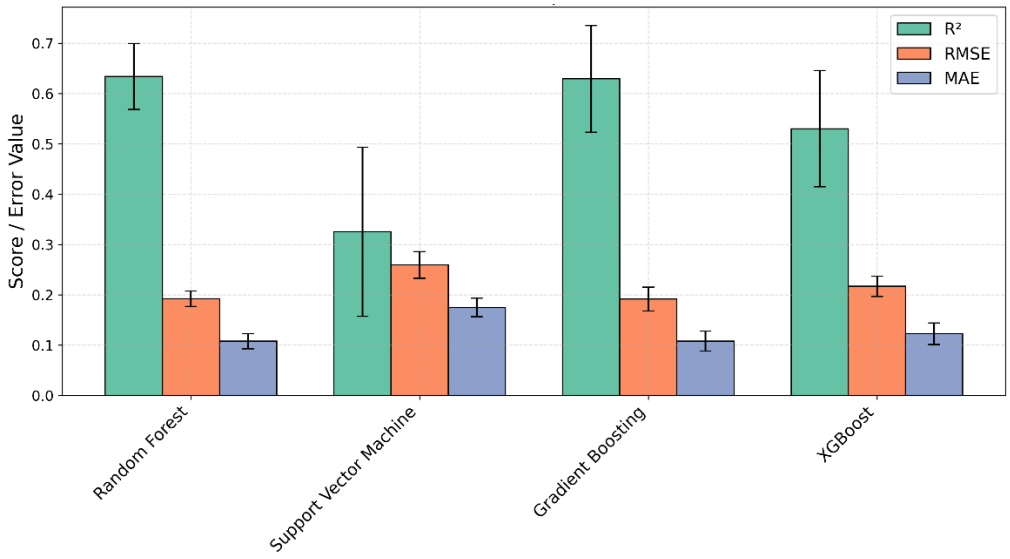


Figure 26. Model performance comparison (mean+std) for T_V with PCA methodology.

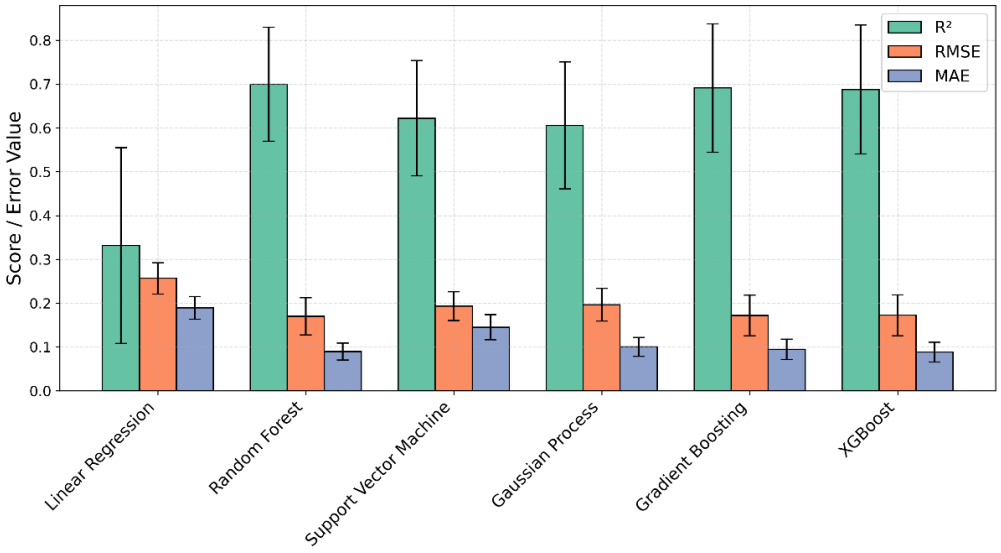


Figure 27. Model performance comparison (mean+std) for T_v with CA/FS methodology.

As seen in Table 10, Random Forest outperformed Gradient Boosting and XGBoost by 0.01 unit, achieving an R^2 of 0.70 and RMSE of 0.17, while maintaining low MAE of 0.09. Notably, Gaussian Process performs well enough with the CA/FS method (R^2 = 0.61) but produces nonsensical R^2 values (R^2 = -0.24) when trained on PCA derived features, as does Support Vector Machine (R^2 = 0.62 versus R^2 = 0.33). Linear Regression does not perform well with either methodology with an R^2 value ranging from -0.03 (PCA) to 0.33 (CA/FS), proving that the relationship between descriptors and T_v is not linear, and is more complex than that of T_g. Standard deviations across all models and both methodologies are non-negligible, which suggests that models' accuracy depends on which data set they are trained on. This could be improved by adding more data and re- training the models so they can make more informed decisions.

All in all, PCA does not improve model performance because it decreases R^2 and increases RMSE and MAE values. Interestingly, it seems that standard deviations for PCA are generally lower (except for SVM). This could hint at PCA's inherent robustness, which leads to more standard results, even if they are inaccurate. The two best models were Random Forest and Gradient Boosting with R^2 values of 0.63. PCA for predicting T_v will not be investigated further.

Parity plots for Random Forest and Gradient Boosting, respectively, can be seen in Figure 28 where the predicted values from the best performing models across both methodologies are plotted against the actual values. Gradient Boosting and XGBoost presented the same exact behavior when predicting T_v. The two are visually similar.

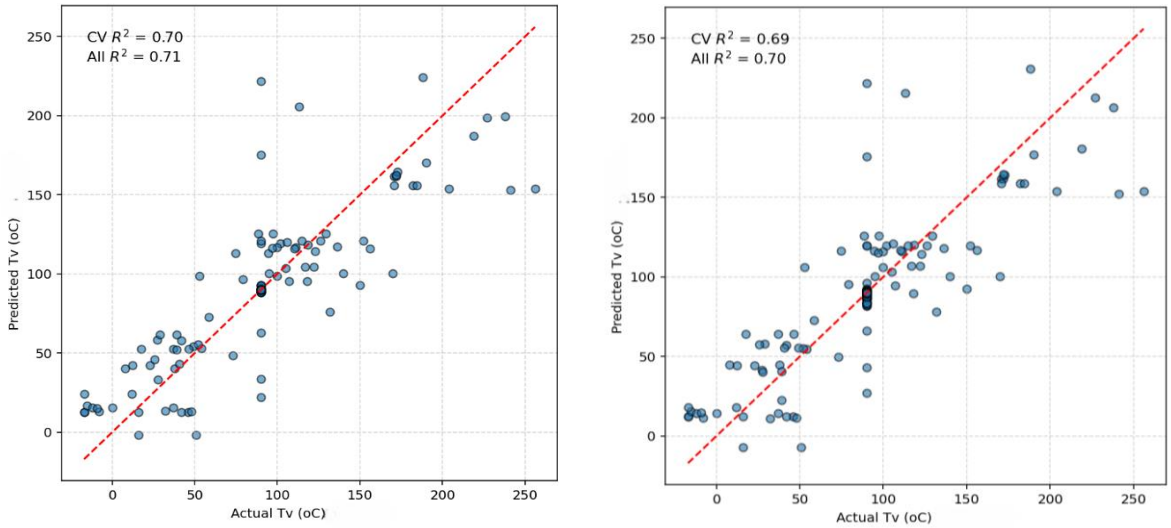


Figure 28. Parity plots for T_v with a) Random Forest (left) and b) Gradient Boosting (right) models.

The behavior of both models is very similar, with most accurate predictions happening in the range between 20 and 120 °C, where points lie close to the diagonal. Values to the far right of the plot are not predicted accurately and there are some other values far from the diagonal on the left side. These "failures" might be due to little information being provided for this kind of T_v values, which made it hard for models to make good predictions. CV and overall R^2 values are very close for both models. RF has CV R^2 = 0.70 and overall R^2 = 0.71 and GB, 0.69 and 0.70, respectively. The slightly higher overall R^2 value seems to indicate that cross validation, while mitigating overfitting, marginally hindered the models from achieving the best performance possible. Additionally, the fact that the two types of R^2 values are so close could mean that the models are very stable across different folds and are, therefore, indifferent to the kind of data they are given.

Finally, an error distribution plot was created for Random Forest since it has a slightly higher R^2 value than Gradient Boosting.

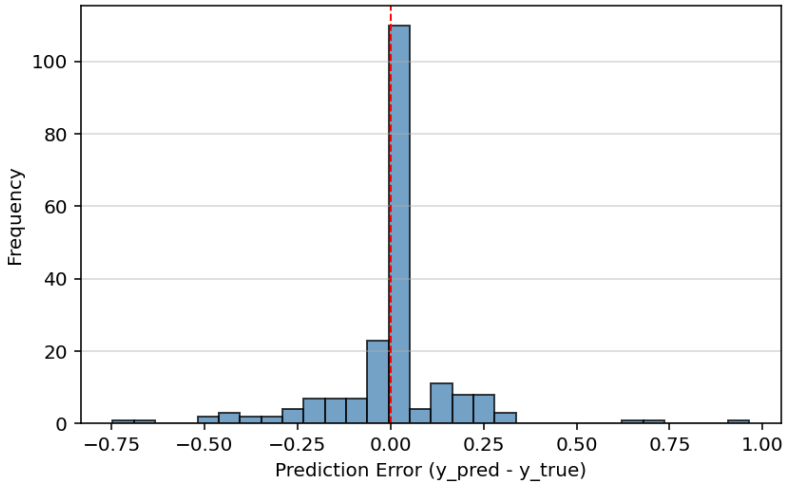


Figure 29. Error distribution plot for T_v with Random Forest model.

More specifically, the distribution presented in Figure 29 is relatively tight, which shows little prediction errors, and mostly centered around zero, which is indicative of good performance. The only thing worth mentioning is that there are some values to the far right and far left that the model failed to capture, which could be attributed to either outliers or lack of sufficient data for the model to be trained on. The latter seems to be more reasonable because the "extreme" values are not the only ones that the model fails to predict. Hence, adding more data points would benefit model performance.

Gel fraction

For the prediction of gel fraction, the same trend can be observed across the different models and feature selection methods. In general, all models failed to capture this target variable completely, since an R^2 of at least 0.60 would be necessary to characterize a model as adequate. Here, the best performing model was Support Vector Machine with an R^2 of only 0.37. Ensemble models who performed well for the other targets could not predict this variable. The worse performing model was once again Gaussian Process with a negative R^2 value.

When comparing the two dimensionality reduction methods, neither PCA nor CA/FS consistently led to better performance across most models, proving that using the current data set makes it impossible to accurately predict gel fraction.

These findings are summarized in Table 11 and visualized in the accompanying boxplots for a clearer comparison across models and error metrics.

Table 11. Mean performance table with metrics for GF.

Model	R^2 Mean		RMSE Mean		MAE Mean	
	PCA	CA/FS	PCA	CA/FS	PCA	CA/FS

Linear Regression	0.10	0.16	0.30	0.29	0.19	0.20
Random Forest	0.34	0.22	0.25	0.26	0.13	0.12
Support Vector Machine	0.24	0.37	0.28	0.25	0.16	0.15
Gaussian Process	-1.64 (*)	-0.41 (*)	0.47	0.34	0.42	0.24
Gradient Boosting	0.28	0.02	0.26	0.29	0.13	0.14
XGBoost	-0.04 (*)	0.00	0.29	0.29	0.14	0.13

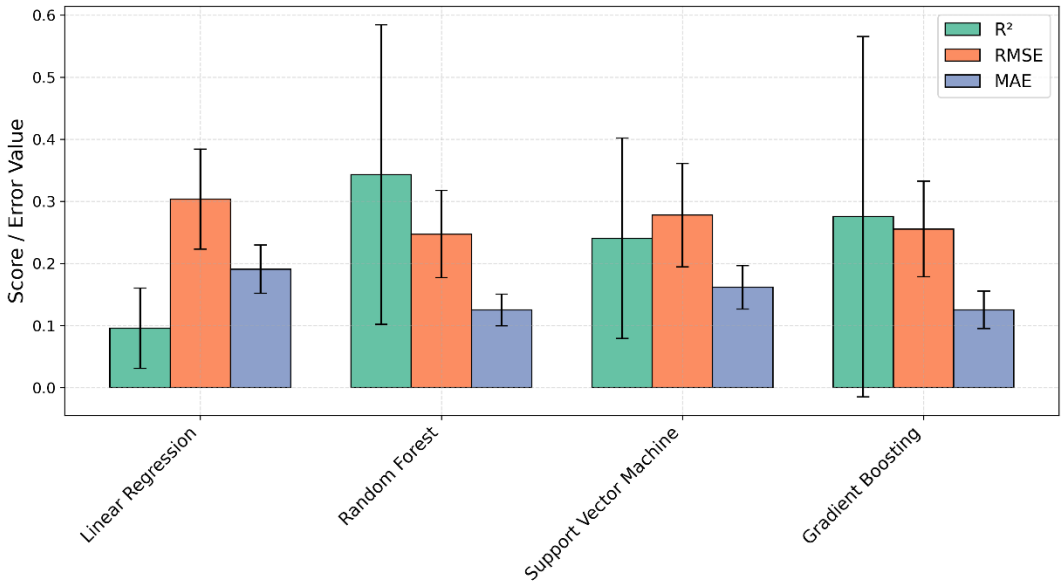


Figure 30. Model performance comparison (mean+std) for GF with PCA methodology.

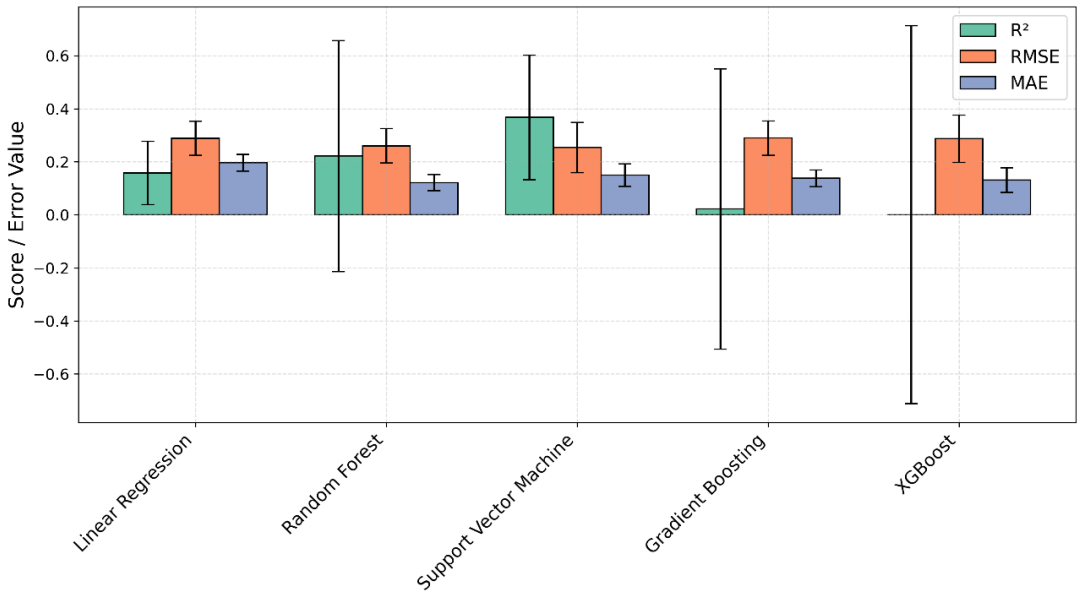


Figure 31. Model performance comparison (mean+std) for GF with CA/FS methodology.

As shown in Table 11, all models achieved R^2 values below 0.4 and RMSE/MAE values similar to other target variables. The standard deviations for all metrics, and especially

R^2 are very high- higher than all other target variables- which shows significant instability of the models during different folds in cross-validation. This suggests that the predictions made are unreliable, besides also being poor. What is especially interesting is how high the standard deviation is across both methodologies for all ensemble models, which usually outperform all other models. These results are not surprising because gel fraction is not an entirely subjective measure. Not all research groups perform the same experimental procedure to measure it, which can result in a variety of values, not all of which are precise or indicative of the phenomena behind crosslinking and swelling. Consequently, this makes it harder for models to predict it accurately and repeatedly.

Parity plots for Random Forest with PCA and Support Vector Machines with the CA/FS method will be presented since no single methodology stood out as the better one and in general, models failed to predict the target completely.

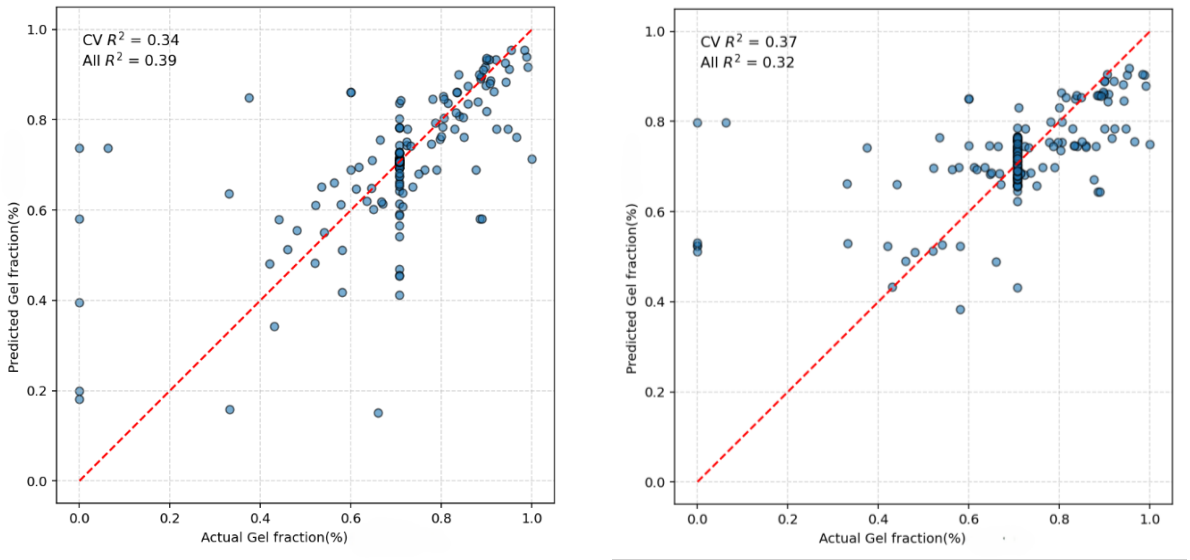


Figure 32. Parity plots for GF with a) PCA Random Forest (left) and b) CA/FS Support Vector Machine (right) models.

Values in both parity plots are quite scattered and far from the diagonal line. This phenomenon seems to be even more prominent in SVM, even though it has a slightly higher cross-validation score (R^2 = 0.37). This is likely because RF has a higher overall score (R^2 = 0.39), which means it makes better predictions across the entire data set. The best predictions seem to be made for gel fraction values over 0.7. Nevertheless, both models have a lot of extreme values they cannot predict, especially in the left side of the plot. These could be considered outliers, but it could also just be a case of different experimental procedures and measurements.

At the same time, in both plots there is an area with aggregated values that form a vertical line. These datapoints are most likely the imputed ones, since the actual value remains stable but the predicted varies. Value imputation corrupts the variability of values and can mess with model performance, but is, as previously mentioned, necessary in data- scarce sets.

Finally, an error distribution plot was created for Support Vector Machine, since it has the highest R^2 value.

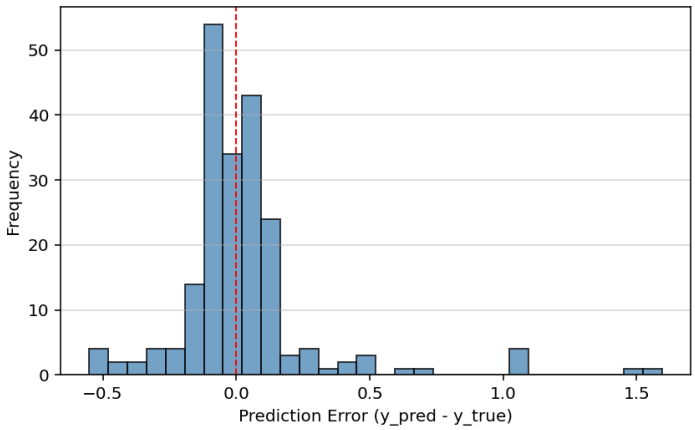


Figure 33. Error distribution plot for GF with Support Vector Machine model.

This error plot reinforces what has already been said about the model underperforming. More specifically, the wide, right- skewed distribution proves that the model is not good and the tail of the distribution highlights a lot of values that cannot be predicted properly. What is also notable is that the residual with a value of about –0.1 has the highest frequency of appearance, which means the model consistently underestimates results by 0.1 units.

4.2.2 Investigation of hyperparameter tuning on model performance

In this section, the impact of hyperparameter tuning on model performance for predicting crosslink density, glass transition temperature and topology freezing temperature is evaluated. The reason gel fraction is not investigated is that during initial training all models across both dimensionality reduction methods failed to produce accurate predictions. It was, therefore, assumed that hyperparameter tuning would not improve their performance.

The general direction that was followed during hyperparameter optimization was to encourage generalizability through chosen parameters over finding parameters that would generate the best possible model performance. The criteria used to compare tuned and untuned models were the same as previously, so R^2, RMSE and MAE and

supplementary box plots, as well as parity plots and error prediction plots. The goal is to assess whether the optimization process meaningfully improves predictive accuracy and model stability, and to identify which algorithms generalize best.

Crosslink density

For the prediction of crosslink density, most frequently selected hyperparameters during grid search are summarized in Table 12, while Table 13 presents a comparative analysis of performance metrics for default models and models with tuning, accompanied by box plots for visualization.

In this case, performance improvement varies depending on the model. More to the point, Gaussian Process improved largely, going from a negative R^2 value to a positive one, while Gradient Boost and XGBoost also improved slightly as far as their R^2 values go. Random Forest and Support Vector Machine's performance worsened, which suggests the default parameters were better. Hyperparameter tuning seemed largely effective in improving predictions for crosslink density.

Table 12. Most common parameters chosen across cross validations by the grid search method for v_e.

Model	Most Common Params
Linear Regression	{}
Random Forest	{'max_depth': None, 'min_samples_split': 5, 'n_estimators': 200}
Support Vector Machine	{'C': 0.1, 'epsilon': 0.01, 'kernel': 'rbf'}
Gaussian Process	{'kernel': 0.316**2 * RBF(length_scale=1) + WhiteKernel(noise_level=0.01)}
Gradient Boosting	{'learning_rate': 0.1, 'max_depth': 5, 'max_features': 'sqrt', 'n_estimators': 100}
XGBoost	{'learning_rate': 0.05, 'max_depth': 3, 'n_estimators': 100}

Table 13. Mean performance table with metrics for v_e with hyperparameter tuning and default values.

Model	R^2 Mean		RMSE Mean		MAE Mean	
	HP TUNING	DEFAULT	HP TUNING	DEFAULT	HP TUNING	DEFAULT
Linear Regression	0.22	0.22	0.13	0.13	0.07	0.07
Random Forest	0.74	0.77	0.09	0.08	0.03	0.03
Support Vector Machine	0.44	0.50	0.12	0.11	0.06	0.07
Gaussian Process	0.51	-54.81 (*)	0.10	0.86	0.05	0.79
Gradient Boosting	0.75	0.73	0.08	0.09	0.03	0.03
XGBoost	0.64	0.60	0.09	0.10	0.04	0.03

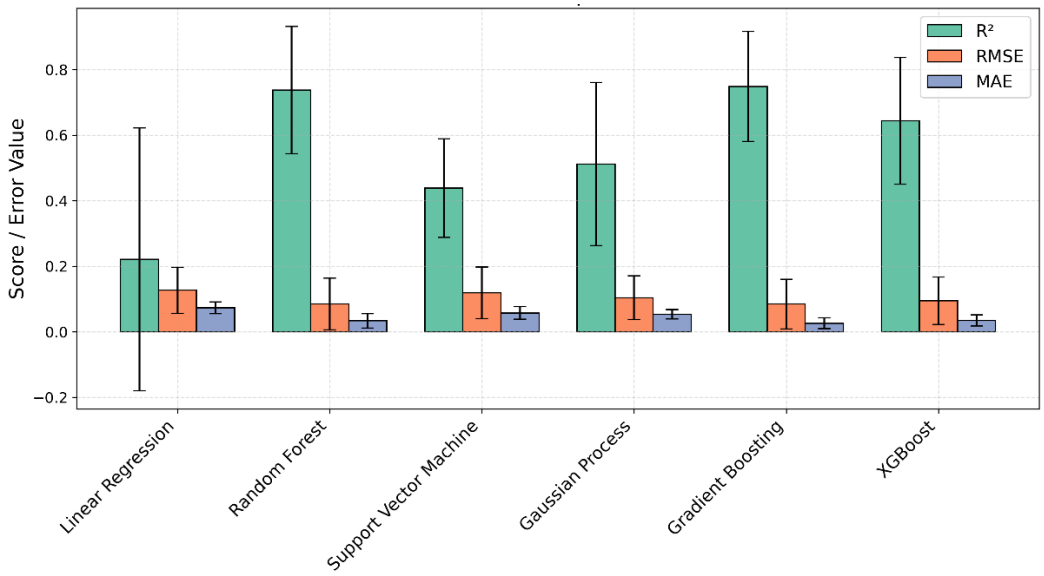


Figure 34. Model performance comparison (mean+std) for v_e with hyperparameter tuning.

The general trend here, aside from favoring generalization, is using parameters that work well in data sets where moderate to little amount of noise is present. This is evident through parameters like *"noise level"*, *"epsilon"*, *"max depth"*, *etc.,* which have low values. Also, conservative values for "n estimators" shows that stable and slow convergence was chosen. Kernel- based algorithms like SVM and GP use the *"RBF"* kernel, which reveals the existence of non- linear relationships.

Here, the best performing model is Gradient Boosting with an R^2 value of 0.75 and RMSE of 0.08, which is very low. Even so, it is outperformed marginally by the default Random Forest which has R^2 value of 0.77 and the same RMSE. What is interesting is that both have roughly the same standard deviation (compared with Figure 18), which means they are equally reliable across different training sets. The most prominent example of good tuning is Gaussian Process, where the choice of the appropriate kernel dramatically changed the model's performance, taking its R^2 value from -54.81 to 0.51. However, 0.51 is still not a great R^2 value, especially combined with a high standard deviation, which means the model is also affected by the actual data that is trained on. It is known that GP is especially sensitive to noisy data and seeing that for crosslink density some values had to be imputed, it makes sense that the model had some trouble predicting it. As far as SVM goes, it is likely that it required higher values of *"C"* and *"epsilon"* to better perform, since restricting these generally leads to better generalization over performance.

In conclusion, tuning the models who predict the crosslink density favors their performance in general, but hurts the performance of the best model, RF. This is probably due to the fact that while trying to improve all three metrics, grid search

suggested a combination of parameters that was not as good as the default variables. Further manual tuning could be explored.

Parity plots for the default Random Forest and tuned Gradient Boosting will be presented.

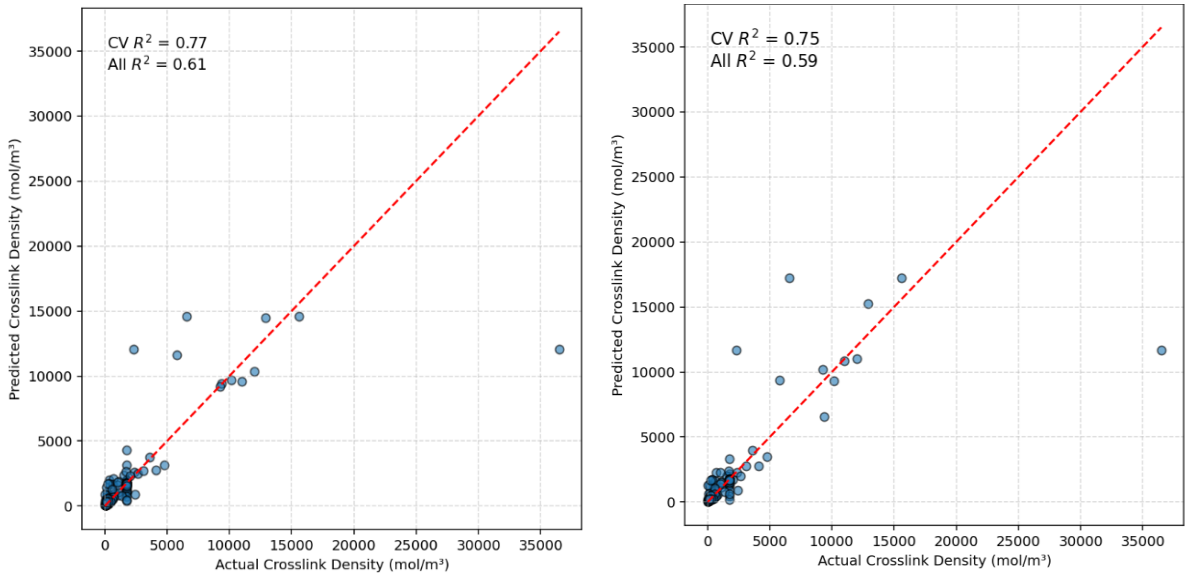


Figure 35. Parity plots for v_e with a) default Random Forest (left) and b) tuned Gradient Boosting (right) models.

The two models show very similar behavior, with very little points being far from the diagonal and most observations clustered in the lower left region of the plot, where the best predictions are made. The higher overall R^2 of RF (R^2 = 0.61 versus R^2 = 0.59) in combination with the higher CV R^2 value, solidify the statement that the default RF has better and more stable predictive accuracy than the tuned GB and should be used to make further predictions. The same that was said about outliers in Chapter 4.2.1 applies here as well. It is obvious that the best performing model would greatly benefit from the addition of more datapoints and especially examples of higher crosslinking densities, since the model itself is already performing optimally.

Since the best model of the two is the untuned RF, the error plot has already been presented in Figure 21, so no further analysis is necessary.

Glass transition temperature

For the prediction of glass transition temperature, the most frequently selected hyperparameters during grid search are summarized in Table 14, while Table 15 presents a comparative analysis of performance metrics for default models and models with tuning, accompanied by box plots for visualization.

Hyperparameter tuning improved predictive capacity of all models except Random Forest, where it was slightly damaged. R^2 values were higher for tuned than default models and, in some cases, RMSE/MAE values decreased. Here, tuning the models seemed to optimize performance consistently. Ensemble- based algorithms once again proved superior to the others, with Gradient Boosting and XGBoost having the best R^2 value, followed closely by Random Forest.

Table 14. Most common parameters chosen across cross validations by the grid search method for T_g.

Model	Most Common Params
Linear Regression	{}
Random Forest	{'max_depth': 5, 'min_samples_split': 2, 'n_estimators': 100}
Support Vector Machine	{'C': 1, 'epsilon': 0.1, 'kernel': 'rbf'}
Gaussian Process	{'kernel': 1**2 * RBF(length_scale=10) + WhiteKernel(noise_level=0.1)}
Gradient Boosting	{'learning_rate': 0.1, 'max_depth': 3, 'max_features': 'sqrt', 'n_estimators': 100}
XGBoost	{'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}

Table 15. Mean performance table with metrics for T_g with hyperparameter tuning and default values.

Model	R^2 Mean		RMSE Mean		MAE Mean	
	HP TUNING	DEFAULT	HP TUNING	DEFAULT	HP TUNING	DEFAULT
Linear Regression	0.58	0.58	0.30	0.30	0.24	0.24
Random Forest	0.84	0.85	0.18	0.18	0.10	0.10
Support Vector Machine	0.69	0.69	0.26	0.26	0.17	0.18
Gaussian Process	0.83	0.77	0.19	0.22	0.11	0.12
Gradient Boosting	0.86	0.85	0.17	0.17	0.10	0.09
XGBoost	0.85	0.83	0.17	0.18	0.10	0.09

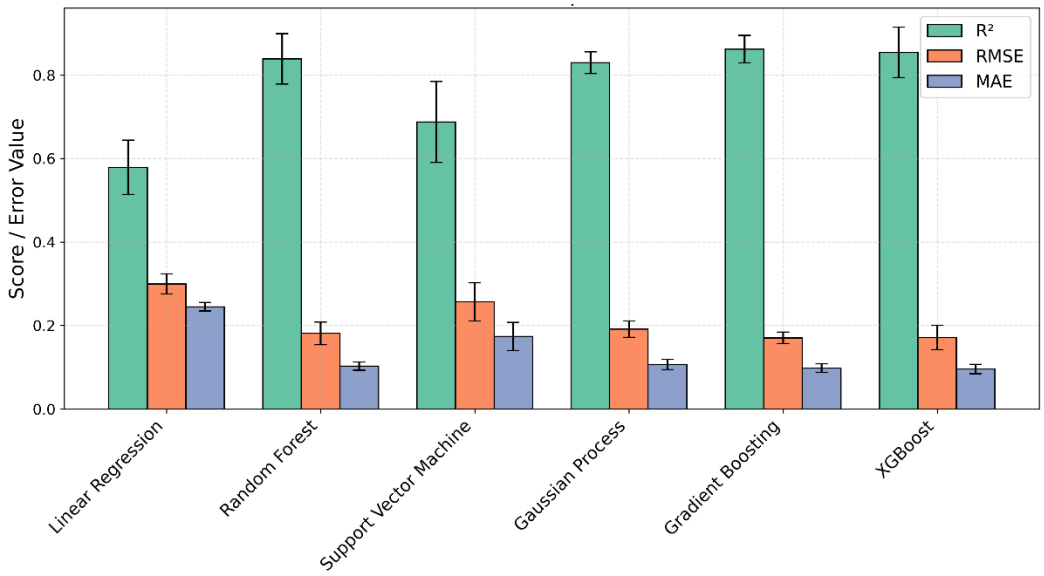


Figure 36. Model performance comparison (mean+std) for T_g with hyperparameter tuning.

The general trend here, aside from favoring generalization, is using parameters that work well in data sets where moderate amounts of noise is present. This is evident through parameters like *"noise level", "epsilon", etc.*, which have mid-range values. Also, low values for *"n estimators"* and *"max depth"* create swallow trees, which means that the relationship between features and T_g is not very complex, but it is not linear either, a fact that is backed up by the choice of the *"RBF"* kernel in kernel-based algorithms.

Here, the best performing model is Gradient Boosting with an R^2 value of 0.86 and RMSE of 0.17, which is the lowest in comparison to other models. XGBoost comes close with an R^2 value of 0.85, but a higher standard deviation. This trend is observed for other target variables as well, with XGBoost showing good but slightly unstable performance, possible due to the parameters set or existing noise in the training sets. The next best models are Random Forest (R^2 = 0.84) and Gaussian Process (R^2 = 0.83), that seems to be very robust, which is evident by their very low standard deviation. This is the first case where Gaussian Process is amongst the highest performing models, which is interesting considering it failed most of the time, as shown by its negatively calculated R^2 values. T_g is one of the best documented values in the original data set and has very little imputed values, especially in relation to the others. At the same time, during dimensionality reduction, it was observed that many features were strongly related to this target variable, which means the signal that had to be interpreted by the model was very strong. This is probably why GP, a model with known constraints when it comes to noisy signals, performed well, in contrast to previous attempts. Support Vector Machines maintained the same performance before and after hyperparameter investigation, with only MAE value dropping marginally. Finally, Random Forest's

performance worsened slightly, which is what happened for the case of crosslink density as well.

In conclusion, tuning the models who predict the T_g optimizes their performance almost entirely, making them strong candidates for future predictions, and especially Gradient Boosting.

Parity plots for Gradient Boosting and XGBoost are presented in Figure 37, since they have the highest R^2 values, followed closely by Random Forest.

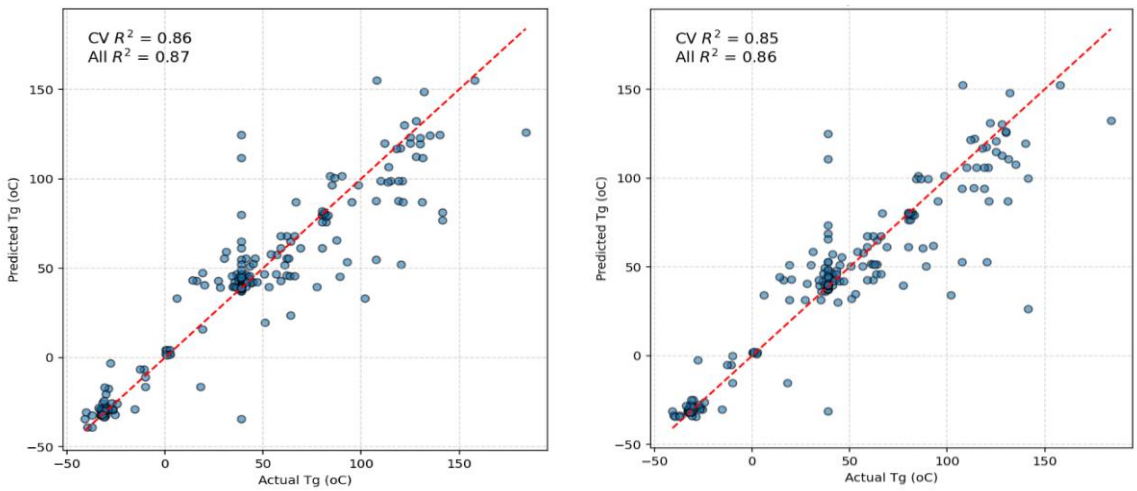


Figure 37. Parity plots for T_g with a) Gradient Boosting (left) and b) XGBoost (right) models.

The two parity plots are very similar, with both models accurately predicting T_g values close to –40 °C and 40 °C, where points lie very close to the diagonal. These values were the most frequent ones in the original data set. Both models seem to underestimate slightly T_g values in the range between 55 and 150, with some extreme value after 150 °C. Additionally, a vertical line is observed in both plots, which probably represents imputed values. Both GB and XGBoost have slightly higher overall R^2 values, 0.87 and 0.86 respectively, versus CV R^2 values, 0.86 and 0.85. This means that the chosen hyperparameters favored reducing overfitting over model performance, allowing the performance during training with the entire data set to be slightly better.

Finally, an error distribution plot was created for Gradient Boosting (see Fig. 38), since it is slightly better than XGBoost.

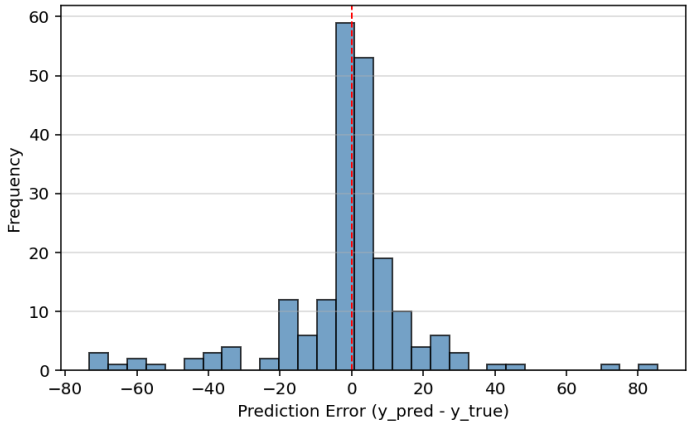


Figure 38. Error distribution plot for T_g with Gradient Boosting model.

The observed distribution is narrow, close to normal and centered around zero. All these are evidence of a well-performing model. The only thing of concern is that there is a systematic underappreciation of predicted values, which can be seen by their higher frequency of appearance in contrast to accurate predictions. These errors are not that significant, so they are no cause for concern, and they could represent the imputed values. The tails of the distribution, however, highlight some observations that the model fails to predict correctly, either by overestimating them (right) or underestimating them (left). This could simply be a matter of such values lying at the ends of the overall range in the original data set and could be mitigated by adding similar datapoints that the model could be trained on.

Topology freezing temperature

For the prediction of topology freezing temperature, most frequently selected hyperparameters during grid search are summarized in Table 16, while Table 17 presents a comparative analysis of performance metrics for default models and models with tuning, accompanied by box plots for visualization.

Interestingly, in this case, the default models generally outperform their tuned counterparts, as reflected by higher R^2 values and slightly lower MAE values across most algorithms—except for the Support Vector Machine and XGBoost, where hyperparameter tuning resulted in marginal improvements. This observation suggests that the default hyperparameters were already well-suited to the dataset and prediction task, and that the grid search process did not significantly enhance model performance.

Table 16. Most common parameters chosen across cross validations by the grid search method for T_v.

Model	Most Common Params
Linear Regression	{}

Random Forest	{'max_depth': 5, 'min_samples_split': 2, 'n_estimators': 200}
Support Vector Machine	{'C': 0.1, 'epsilon': 0.1, 'kernel': 'rbf'}
Gaussian Process	{'kernel': 1**2 * RBF(length_scale=10) + WhiteKernel(noise_level=0.1)}
Gradient Boosting	{'learning_rate': 0.1, 'max_depth': 3, 'max_features': 'sqrt', 'n_estimators': 200}
XGBoost	{'learning_rate': 0.05, 'max_depth': 3, 'n_estimators': 100}

Table 17. Mean performance table with metrics for T_v with hyperparameter tuning and default values.

Model	R^2 Mean		RMSE Mean		MAE Mean	
	HP TUNING	DEFAULT	HP TUNING	DEFAULT	HP TUNING	DEFAULT
Linear Regression	0.33	0.33	0.26	0.26	0.19	0.19
Random Forest	0.68	0.70	0.17	0.17	0.10	0.09
Support Vector Machine	0.47	0.62	0.23	0.19	0.15	0.14
Gaussian Process	0.63	0.61	0.19	0.20	0.12	0.10
Gradient Boosting	0.68	0.69	0.17	0.17	0.10	0.09
XGBoost	0.70	0.69	0.17	0.17	0.10	0.09

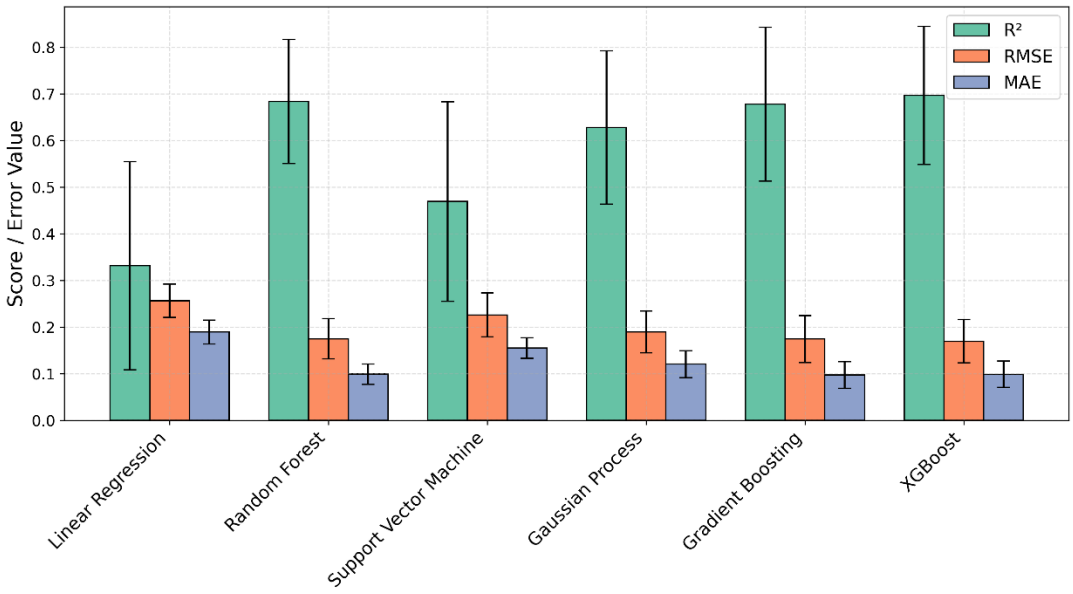


Figure 39. Model performance comparison (mean+std) for T_v with hyperparameter tuning.

The general trend in ensemble methods is using hyperparameters that create relatively weak and gradual learners that prevent overfitting. This is shown through low values in parameters like *"max depth"*, *"learning rate"*, *"n_estimators"*, *etc.* Kernel- based algorithms like SVM and GP use the *"RBF"* kernel, which means that linear kernels were tested and dropped due to the existence of non-linear relationships.

Here, the best performing model is XGBoost with an R^2 value of 0.70 and RMSE of 0.17, like Random Forest and Gradient Boosting, but a slightly higher standard deviation in R^2 than RF. In general, though, it can be considered a good model. Notwithstanding, the default RF that has a similar R^2 value is preferred due to its lower standard deviation. While tuned RF and GB have the same exact metrics (R^2 = 0.68, RMSE = 0.17, MAE = 0.10), the standard deviation of the latter is larger, suggesting a more unstable model. The drop in SVM's performance is significant and indicative of the grid search's inappropriate combination of parameters. Finally, it is safe to say that trying to optimize RF through the grid search method did not improve predictions of either crosslink density, or T_g or T_v. The simplicity of the default parameters overtook the more complicated tuning approach.

In conclusion, it seems that default models outperform the others in this case. A possible explanation could be that grid search, while systematic, can become inefficient or overly strict when applied to relatively small data sets. By attempting to optimize across multiple performance metrics simultaneously, it may favor parameter combinations that generalize less effectively. Another reason behind this behavior could be that parameter combinations are not as important here as is the need for additional observations, on which the models could be trained.

Parity plots and error distribution plots have already been covered in Chapter 4.2.1 for this target variable, so no further analysis will be made.

Chapter 5: Conclusions and Future Work

This chapter aims to provide a comprehensive summary of the research conducted, revisiting the objective that guided this study, which was deploying and training ML models that predict key properties of dynamically crosslinked polymers, otherwise known as vitrimers. The main motivation behind this was the existence of a research gap in ML utilization for vitrimer property prediction. To this day, there are very few researchers that have trained and deployed models that can help produce helpful insights for the expansion of domain knowledge in vitrimers. On the other hand, research on polymers and especially smaller molecules is quite mature, even though the former presents greater difficulties than the latter. This chapter will also provide suggestions for future work, based on the limitations encountered during the research process.

By applying and comparing two distinct dimensionality reduction approaches- Principal Component Analysis (PCA) and the hybrid correlation analysis/feature selection (CA/FS)- alongside a suite of regression models, the study has yielded important insights into the relationship between data preprocessing choices and model performance. While no outliers were removed to preserve the small and manually curated dataset, careful preprocessing—such as zero-variance and scarcity filters—ensured that only meaningful, variant features were retained. The two dimensionality reduction methods captured different aspects of the problem: PCA emphasized features with variance, while feature selection highlighted chemically interpretable variables. This, in turn, favored the prediction of some properties over others.

To be more precise, the only property positively affected by PCA was crosslink density, which is probably influenced by bigger structural patterns and not specific variables like the quantity of catalyst used or the ratio between compounds. On the other hand, predictions of the remaining three targets (T_g, T_v and GF) were mostly inaccurate when done through PCA- derived features. These variables required more nuanced chemical descriptors that were lost during feature combination done to create the four principal components. The general trend across all variables, though, was a high dependance on the nature (and sometimes quantity) of the first compound used, as well as the catalyst.

Overall, the best model performance was noted in predicting T_g, with both Gradient Boosting and Random Forest models achieving cross-validated R^2 values of 0.85, confirming the effectiveness of ensemble methods and suggesting that T_g is more predictable from molecular descriptors, especially when using the CA/FS method. For T_v, ensemble models proved better once more, with the Random Forest model reaching an R^2 of 0.70. Notably, PCA performed poorly in this case, likely due to its inability to

capture target-relevant nonlinear relationships and its exclusion of chemically meaningful features.

In the case of crosslink density, PCA yielded better results than the CA/FS strategy, with the Random Forest model reaching an R^2 of 0.77, outperforming all other models. This suggests that crosslink density is governed by broader structural trends that PCA better captures, while getting rid of non- informative features.

However, gel fraction prediction proved to be unsuccessful, with all models performing poorly (R^2 < 0.40) regardless of preprocessing or algorithm choice. The best model (SVM) achieved an R^2 of only 0.37, indicating that either the descriptors used fail to capture relevant information for this target, or the target itself is too variable or noisy for reliable learning with the current data set.

All in all, ensemble- based models outperformed kernel- based models, due to their higher robustness and better handling of noisy data, which makes them especially appealing for smaller data sets. Random Forest managed to predict three out of four targets relatively accurately, suggesting its superiority to the others. On the contrary, kernel- based models like SVM and GP generally underperformed because they depend heavily on appropriate kernel selection. As seen, tuned GP outperformed the default counterpart, across every target variable and especially when predicting T_g with an R^2 value of 0.83, proving that proper parameter selection was key to its performance. Another noteworthy model was XGBoost which was strong at making predictions, but not as well-behaved as the rest of the top performers. XGBoost presented with a high standard deviation, meaning it was unstable across different folds in the training, making it a somewhat unreliable choice for further predictions.

Finally, hyperparameter optimization- despite being a common strategy for performance optimization- did not consistently improve results. In many cases, default models outperformed their counterparts, especially for Random Forest and SVM. This outcome may stem from the limitations of grid search when applied to small or noisy datasets, where tuning may overfit to particular cross validation splits.

While this thesis has addressed key objectives and provided insights into the research topic, several opportunities remain for further exploration and development. The limitations encountered during this study, as well as new experimental studies performed in vitrimers point toward promising directions for future work.

More specifically, after performing all the necessary steps for data processing, model training and optimization, the final predictive capacity across all different algorithms still has room for improvement. Firstly, there is a need for outlier detection and elimination to be performed to mitigate the negative effect extreme values might have

during model training. Outliers mostly hurt a model's generalizability, but they can also affect overall performance by drawing attention away from values that are indeed informative. It must be said that at least as far as the experimental data set is concerned, outlier detection should be lenient, so as to not restrict the range of the experimentally derived values who are important for the problem at hand. As previously mentioned, these observations are drawn from reliable literature, so there is merit in including values that are more extreme than the average. All in all, caution needs to be exerted during this elimination step. At the same time, a different fingerprinting scheme could be tested, such as including Morgan Fingerprints in the training process.

Secondly, and even more importantly, future work should focus on expanding the existing database until it has, at least, doubled in size. This will inevitably improve models' performance through diversifying the potential training pool. It would be especially interesting to add data points from experiments that include a third compound for which limited information was found in this work. This could perhaps even change the final features used for model training, which could, in turn, affect performance. This would also allow for outlier detection and elimination without risking shrinking the data set too much, as it would here. Harnessing the power of natural language processing models to expand the database would be an exciting step, since these models can mine data from literature faster and more effectively than humans.

Besides this, further investigation into hyperparameter tuning is due. More to the point, a different tuning algorithm could be explored instead of grid search, which is randomized grid search. The two are similar in principle, but the latter is fed with a distribution of potential hyperparameters rather than discrete values and it makes choices by randomly sampling them. This can lead to better hyperparameters but is a more computationally intensive procedure. Basic grid search should also be further improved, by adding more possible values and hyperparameters that the algorithm can investigate. Finally, seeing that RF and SVM models consistently failed when tuned through grid search, a manual tuning approach could be utilized.

Another possibility would be to train models, using the current workflow, to predict a different property instead of gel fraction, which is similar to crosslink density and was calculated for comparison. It would be helpful to predict some mechanical property, like Young's modulus or tensile strength. In the existing database, several such measurements can be found. Unfortunately, there were not enough for training, so expanding them would be a great first step. It would be indeed very interesting to have models that predict one crosslinked network property (crosslink density or T_g), one rheological property (T_v) and one mechanical (like tensile strength).

The final step in such research would be to create an interface that would harness the best performing model for each target value and make predictions based on inputs

given by a user. These inputs would have to include SMILES notations for each compound used as well as basic information like reactive end- group ratio, catalyst molar ratio and probably desired reaction temperature. The model would then be able to make informed predictions about the requested target variable, driving vitrimer research forward.

References

[1] Wu, Stephen, et al. *Potentials and Challenges of Polymer Informatics: Exploiting Machine Learning for Polymer Design*. arXiv:2010.07683, arXiv, 15 Oct. 2020. *arXiv.org*, https://doi.org/10.48550/arXiv.2010.07683.

[2] Ge, Wei, et al. "Machine Learning in Polymer Research." *Advanced Materials*, vol. 37, no. 11, Mar. 2025, p. 2413695. *DOI.org (Crossref)*, https://doi.org/10.1002/adma.202413695.

[3] Martin, Tyler B., and Debra J. Audus. "Emerging Trends in Machine Learning: A Polymer Perspective." *ACS Polymers Au*, vol. 3, no. 3, June 2023, pp. 239–58. *DOI.org (Crossref)*, https://doi.org/10.1021/acspolymersau.2c00053.

[4] Wang, Anthony Yu-Tung, et al. "Machine Learning for Materials Scientists: An Introductory Guide toward Best Practices." *Chemistry of Materials*, vol. 32, no. 12, June 2020, pp. 4954–65. *DOI.org (Crossref)*, https://doi.org/10.1021/acs.chemmater.0c01907.

[5] Lin, Tzyy-Shyang, et al. "BigSMILES: A Structurally-Based Line Notation for Describing Macromolecules." *ACS Central Science*, vol. 5, no. 9, Sept. 2019, pp. 1523–31. *DOI.org (Crossref)*, https://doi.org/10.1021/acscentsci.9b00476.

[6] Tao, Lei, et al. "Benchmarking Machine Learning Models for Polymer Informatics: An Example of Glass Transition Temperature." *Journal of Chemical Information and Modeling*, vol. 61, no. 11, Nov. 2021, pp. 5395–413. *DOI.org (Crossref)*, https://doi.org/10.1021/acs.jcim.1c01031.

[7] Sharma, Harsh, et al. "Self-Healable Fiber-Reinforced Vitrimer Composites: Overview and Future Prospects." *RSC Advances*, vol. 12, no. 50, 2022, pp. 32569–82. *DOI.org (Crossref)*, https://doi.org/10.1039/D2RA05103F.

[8] Zheng, Jie, et al. "Vitrimers: Current Research Trends and Their Emerging Applications." *Materials Today*, vol. 51, Dec. 2021, pp. 586–625. *DOI.org (Crossref)*, https://doi.org/10.1016/j.mattod.2021.07.003.

[9] Zhang, Haochuan, et al. "Recycling Strategies for Vitrimers." *International Journal of Smart and Nano Materials*, vol. 13, no. 3, July 2022, pp. 367–90. *DOI.org (Crossref)*, https://doi.org/10.1080/19475411.2022.2087785.

[10] Jin, Yinghua, et al. "Malleable and Recyclable Thermosets: The Next Generation of Plastics." *Matter*, vol. 1, no. 6, Dec. 2019, pp. 1456–93. *ScienceDirect*, https://doi.org/10.1016/j.matt.2019.09.004.

[11] Denissen, Wim, et al. "Vitrimers: Permanent Organic Networks with Glass-like Fluidity." *Chemical Science*, vol. 7, no. 1, Dec. 2015, pp. 30–38. *pubs.rsc.org*, https://doi.org/10.1039/C5SC02223A.

[12] Van Zee, Nathan J., and Renaud Nicolaÿ. "Vitrimer Chemistry and Applications." *Macromolecular Engineering*, edited by Nikos Hadjichristidis et al., 1st ed., Wiley, 2022, pp. 1–38. *DOI.org (Crossref)*, https://doi.org/10.1002/9783527815562.mme0029.

[13] Van Zee, Nathan J., and Renaud Nicolaÿ. "Vitrimers: Permanently Crosslinked Polymers with Dynamic Network Topology." *Progress in Polymer Science*, vol. 104, May 2020, p. 101233. *ScienceDirect*, https://doi.org/10.1016/j.progpolymsci.2020.101233.

[14] Cencer, Morgan M., et al. "Machine Learning for Polymeric Materials: An Introduction." *Polymer International*, vol. 71, no. 5, May 2022, pp. 537–42. *DOI.org (Crossref)*, https://doi.org/10.1002/pi.6345.

[15] Meyer, Travis A., et al. "A User's Guide to Machine Learning for Polymeric Biomaterials." *ACS Polymers Au*, vol. 3, no. 2, Apr. 2023, pp. 141–57. *DOI.org (Crossref)*, https://doi.org/10.1021/acspolymersau.2c00037.

[16] Mauri, Andrea, et al. "Molecular Descriptors." *Handbook of Computational Chemistry*, Springer, Cham, 2017, pp. 2065–93. *link.springer.com*, https://doi.org/10.1007/978-3-319-27282-5_51.

[17] Grisoni, Francesca, et al. "Molecular Descriptors for Structure–Activity Applications: A Hands-On Approach." *Computational Toxicology: Methods and Protocols*, edited by Orazio Nicolotti, Springer, 2018, pp. 3–53. *Springer Link*, https://doi.org/10.1007/978-1-4939-7899-1_1.

[18] Todeschini, Roberto, and Viviana Consonni. *Molecular Descriptors for Chemoinformatics*. 1st ed., Wiley, 2009. *DOI.org (Crossref)*, https://doi.org/10.1002/9783527628766.

[19] Reif, David M., et al. "Feature Selection Using a Random Forests Classifier for the Integrated Analysis of Multiple Data Types." *2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, 2006, pp. 1–8. *IEEE Xplore*, https://doi.org/10.1109/CIBCB.2006.330987.

[20] Li, Chen, and Kun Zheng. "Methods, Progresses, and Opportunities of Materials Informatics." *InfoMat*, vol. 5, no. 8, Aug. 2023, p. e12425. *DOI.org (Crossref)*, https://doi.org/10.1002/inf2.12425.

[21] Natekin, Alexey, and Alois Knoll. "Gradient Boosting Machines, a Tutorial." *Frontiers in Neurorobotics*, vol. 7, Dec. 2013. *Frontiers*, https://doi.org/10.3389/fnbot.2013.00021.

[22] Shahani, Niaz Muhammad, et al. "Developing an XGBoost Regression Model for Predicting Young's Modulus of Intact Sedimentary Rocks for the Stability of Surface and Subsurface Structures." *Frontiers in Earth Science*, vol. 9, Oct. 2021. *Frontiers*, https://doi.org/10.3389/feart.2021.761990.

[23] Iranzad, Reza, and Xiao Liu. "A Review of Random Forest-Based Feature Selection Methods for Data Science Education and Applications." *International Journal of Data Science and Analytics*, Feb. 2024. *Springer Link*, https://doi.org/10.1007/s41060-024-00509-w.

[24] Chen, Peipei, et al. "On the Interaction Between Feature Selection and Parameter Determination in Fuzzy Modelling." *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications*, edited by Jesús Medina et al., Springer International Publishing, 2018, pp. 150–61. *Springer Link*, https://doi.org/10.1007/978-3-319-91479-4_13.

[25] Kuhn, Max, and Kjell Johnson. "Data Pre-Processing." *Applied Predictive Modeling*, edited by Max Kuhn and Kjell Johnson, Springer, 2013, pp. 27–59. *Springer Link*, https://doi.org/10.1007/978-1-4614-6849-3_3.

[26] Ramakrishna, Seeram, et al. "Materials Informatics." *Journal of Intelligent Manufacturing*, vol. 30, no. 6, Aug. 2019, pp. 2307–26. *Springer Link*, https://doi.org/10.1007/s10845-018-1392-0.

[27] Cleophas, Ton J., and Aeilko H. Zwinderman. "Bayesian Pearson Correlation Analysis." *Modern Bayesian Statistics in Clinical Research*, edited by Ton J. Cleophas and Aeilko H. Zwinderman, Springer International Publishing, 2018, pp. 111–18. *Springer Link*, https://doi.org/10.1007/978-3-319-92747-3_11.

[28] Ross, Brian C. "Mutual Information between Discrete and Continuous Data Sets." *PLOS ONE*, vol. 9, no. 2, Feb. 2014, p. e87357. *PLoS Journals*, https://doi.org/10.1371/journal.pone.0087357.

[29] Kraskov, Alexander, et al. *Estimating Mutual Information*. arXiv:cond-mat/0305641, arXiv, 28 May 2003. *arXiv.org*, https://doi.org/10.48550/arXiv.cond-mat/0305641.

[30] Panagiotopoulos, Christos, et al. "Recycling/Upcycling of Physically Aged Poly(Butylene Succinate) into PBS Vitrimers through Zn(II)-Catalyzed Melt Vitrimerization." *Macromolecules*, vol. 57, no. 12, June 2024, pp. 5747–68. *DOI.org (Crossref)*, https://doi.org/10.1021/acs.macromol.4c00322.

[31] In house data

[32] Zhou, Yanwu, et al. "Tuning PBT Vitrimer Properties by Controlling the Dynamics of the Adaptable Network." *Polymer Chemistry*, vol. 10, no. 1, Dec. 2018, pp. 136–44. *pubs.rsc.org*, https://doi.org/10.1039/C8PY01156G.

[33] Niu, Xiling, et al. "Using Zn^{2+} Ionomer To Catalyze Transesterification Reaction in Epoxy Vitrimer." *Industrial & Engineering Chemistry Research*, vol. 58, no. 14, Apr. 2019, pp. 5698–706. *DOI.org (Crossref)*, https://doi.org/10.1021/acs.iecr.9b00090.

[34] Debnath, Suman, et al. "Catalyst-Free Partially Bio-Based Polyester Vitrimers." *ACS Applied Polymer Materials*, vol. 2, no. 2, Feb. 2020, pp. 1006–13. *DOI.org (Crossref)*, https://doi.org/10.1021/acsapm.0c00016.

[35] Giebler, Michael, et al. "Epoxy-Anhydride Vitrimers from Aminoglycidyl Resins with High Glass Transition Temperature and Efficient Stress Relaxation." *Polymers*, vol. 12, no. 5, May 2020, p. 1148. *PubMed*, https://doi.org/10.3390/polym12051148.

[36] Hayashi, Mikihiro, and Ryoto Yano. "Fair Investigation of Cross-Link Density Effects on the Bond-Exchange Properties for Trans-Esterification-Based Vitrimers with Identical Concentrations of Reactive Groups." *Macromolecules*, vol. 53, no. 1, Jan. 2020, pp. 182–89. *DOI.org (Crossref)*, https://doi.org/10.1021/acs.macromol.9b01896.

[37] Taplan, Christian, et al. "Fast Processing of Highly Crosslinked, Low-Viscosity Vitrimers." *Materials Horizons*, vol. 7, no. 1, Jan. 2020, pp. 104–10. *pubs.rsc.org*, https://doi.org/10.1039/C9MH01062A.

[38] Kar, Goutam Prasanna, et al. "Scalable Upcycling of Thermoplastic Polyolefins into Vitrimers through Transesterification." *Journal of Materials Chemistry A*, vol. 8, no. 45, Nov. 2020, pp. 24137–47. *pubs.rsc.org*, https://doi.org/10.1039/D0TA07339C.

[39] Zhao, Xiao-Li, et al. "Sustainable Epoxy Vitrimers from Epoxidized Soybean Oil and Vanillin." *ACS Sustainable Chemistry & Engineering*, vol. 8, no. 39, Oct. 2020, pp. 15020–29. *DOI.org (Crossref)*, https://doi.org/10.1021/acssuschemeng.0c05727.

[40] Qi, Min, et al. "Epoxidized Soybean Oil Cured with Tannic Acid for Fully Bio-Based Epoxy Resin." *RSC Advances*, vol. 8, no. 47, July 2018, pp. 26948–58. *pubs.rsc.org*, https://doi.org/10.1039/C8RA03874K.

[41] Memon, Hafeezullah, et al. "Correlating the Thermomechanical Properties of a Novel Bio-Based Epoxy Vitrimer with Its Crosslink Density." *Materials Today Communications*, vol. 29, Dec. 2021, p. 102814. *ScienceDirect*, https://doi.org/10.1016/j.mtcomm.2021.102814.

[42] Tangthana-umrung, Kanokporn, et al. "Epoxy Homopolymerization as a Tool to Tune the Thermo-Mechanical Properties and Fracture Toughness of Vitrimers." *Macromolecules*, vol. 54, no. 18, Sept. 2021, pp. 8393–406. *DOI.org (Crossref)*, https://doi.org/10.1021/acs.macromol.1c00861.

[43] Wu, Jianqiao, et al. "Natural Glycyrrhizic Acid: Improving Stress Relaxation Rate and Glass Transition Temperature Simultaneously in Epoxy Vitrimers." *Green Chemistry*, vol. 23, no. 15, Aug. 2021, pp. 5647–55. *pubs.rsc.org*, https://doi.org/10.1039/D1GC01274F.

[44] Yin, Yue, et al. "Preparation of Poly(Butylene Succinate) Vitrimer with Thermal Shape Stability via Transesterification Reaction." *Journal of Applied Polymer Science*, vol. 138, no. 39, Oct. 2021, p. 51010. *DOI.org (Crossref)*, https://doi.org/10.1002/app.51010.

[45] Nicolas, Simon, et al. "Shape Memory Epoxy Vitrimers Based on Waste Frying Sunflower Oil." *Journal of Applied Polymer Science*, vol. 138, no. 36, Sept. 2021, p. 50904. *DOI.org (Crossref)*, https://doi.org/10.1002/app.50904.

[46] Xue, Bailiang, et al. "Sustainable Alternative for Bisphenol A Epoxy Resin High-Performance and Recyclable Lignin-Based Epoxy Vitrimers." *Industrial Crops and Products*, vol. 168, Sept. 2021, p. 113583. *ScienceDirect*, https://doi.org/10.1016/j.indcrop.2021.113583.

[47] Chen, Mao, et al. "The Crucial Role in Controlling the Dynamic Properties of Polyester-Based Epoxy Vitrimers: The Density of Exchangeable Ester Bonds (υ)." *Macromolecules*, vol. 54, no. 21, Nov. 2021, pp. 10110–17. *DOI.org (Crossref)*, https://doi.org/10.1021/acs.macromol.1c01289.

[48] Salaeh, Subhan, et al. "Vitrimer-like Material Based on a Biorenewable Elastomer Crosslinked with a Dimeric Fatty Acid." *European Polymer Journal*, vol. 151, May 2021, p. 110452. *ScienceDirect*, https://doi.org/10.1016/j.eurpolymj.2021.110452.

[49] Krishnan, Baiju P., et al. "Design, Synthesis and Characterization of Vitrimers with Low Topology Freezing Transition Temperature." *Polymers*, vol. 14, no. 12, June 2022, p. 2456. *PubMed*, https://doi.org/10.3390/polym14122456.

[50] Kimura, Takahiro, and Mikihiro Hayashi. "One-Shot Transformation of Ordinary Polyesters into Vitrimers: Decomposition-Triggered Cross-Linking and Assistance of Dynamic Covalent Bonds." *Journal of Materials Chemistry A*, vol. 10, no. 34, Aug. 2022, pp. 17406–14. *pubs.rsc.org*, https://doi.org/10.1039/D2TA04110C.

[51] Hubbard, Amber M., et al. "Vitrimer Transition Temperature Identification: Coupling Various Thermomechanical Methodologies." *ACS Applied Polymer Materials*,

vol. 3, no. 4, Apr. 2021, pp. 1756–66. *DOI.org (Crossref)*, https://doi.org/10.1021/acsapm.0c01290.

[52] Hubbard, Amber M., et al. "Recyclability of Vitrimer Materials: Impact of Catalyst and Processing Conditions." *ACS Omega*, vol. 7, no. 33, Aug. 2022, pp. 29125–34. *DOI.org (Crossref)*, https://doi.org/10.1021/acsomega.2c02677.

[53] Van Lijsebetten, Filip, et al. "Recyclable Vitrimer Epoxy Coatings for Durable Protection." *European Polymer Journal*, vol. 176, Aug. 2022, p. 111426. *ScienceDirect*, https://doi.org/10.1016/j.eurpolymj.2022.111426.

[54] Roig, Adrià, et al. "Vitrimeric Epoxy-Amine Polyimine Networks Based on a Renewable Vanillin Derivative." *ACS Applied Polymer Materials*, vol. 4, no. 12, Dec. 2022, pp. 9341–50. *DOI.org (Crossref)*, https://doi.org/10.1021/acsapm.2c01604.

[55] Bandegi, Alireza, et al. "Vitrimerization of Rigid Thermoset Polyurethane Foams: A Mechanochemical Method to Recycle and Reprocess Thermosets." *Polymers for Advanced Technologies*, vol. 33, no. 10, Oct. 2022, pp. 3750–58. *DOI.org (Crossref)*, https://doi.org/10.1002/pat.5827.

[56] Manarin, Eleonora, et al. "Bio-Based Vitrimers from 2,5-Furandicarboxylic Acid as Repairable, Reusable, and Recyclable Epoxy Systems." *ACS Applied Polymer Materials*, vol. 5, no. 1, Jan. 2023, pp. 828–38. *DOI.org (Crossref)*, https://doi.org/10.1021/acsapm.2c01774.

[57] Zhang, Yunfan, et al. "Bio-Based Vitrimers from Divanillic Acid and Epoxidized Soybean Oil." *RSC Sustainability*, vol. 1, no. 3, May 2023, pp. 543–53. *pubs.rsc.org*, https://doi.org/10.1039/D2SU00140C.

[58] Chang, Qianyong, et al. "Catalyst-Free, Self-Healing, and Recyclable Tung Oil-Based Epoxy Vitrimers with Dynamic Ester Bonds." *Macromolecular Chemistry and Physics*, vol. 224, no. 22, Nov. 2023, p. 2300272. *DOI.org (Crossref)*, https://doi.org/10.1002/macp.202300272.

[59] Sougrati, Lisa, et al. "From Lignins to Renewable Aromatic Vitrimers Based on Vinylogous Urethane." *ChemSusChem*, vol. 16, no. 23, Dec. 2023, p. e202300792. *DOI.org (Crossref)*, https://doi.org/10.1002/cssc.202300792.

[60] Zheng, Jiawei, et al. "Fabrication of Recyclable and Biodegradable PBAT Vitrimer via Construction of Highly Dynamic Cross-Linked Network." *Polymer Degradation and Stability*, vol. 219, Jan. 2024, p. 110602. *ScienceDirect*, https://doi.org/10.1016/j.polymdegradstab.2023.110602.

[61] Fabrizio, Luciano, et al. "Upcycling of PET from Recycled Food Packaging Trays via Vitrimers Chemistry." *Polymer*, vol. 266, Jan. 2023, p. 125618. *ScienceDirect*, https://doi.org/10.1016/j.polymer.2022.125618.

[62] Tzouma, Evanthia, et al. "Stress Relaxation Behavior and Electrically Activated Dynamic Exchange in Carbon Nanotube-Modified Epoxy Vitrimers." *ACS Applied Polymer Materials*, vol. 5, no. 1, Jan. 2023, pp. 172–81. *DOI.org (Crossref)*, https://doi.org/10.1021/acsapm.2c01412.

[63] WebPlotDigitalizer website https://automeris.io/ (accessed 10/24)

[64] Polymer Genome Guidelines https://www.polymergenome.org/guide (accessed 12/24)

[65] SMILES Mapper website https://kyll.dk/SMILES-Mapper/ (accessed 12/24)

[66] PSMILES Python package https://colab.research.google.com/github/Ramprasad-Group/psmiles/blob/main/tests/test_book.ipynb (accessed 01/25)

[67] Daylight SMILES website https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html (accessed 12/24)

[68] MolView website https://molview.org/ (accessed 01/25)

[69] Material Informatics Python package https://github.com/sp8rks/MaterialsInformatics (accessed 01/25)

[70] PaDEL Descriptor website http://yapcwsoft.com/dd/padeldescriptor/ (accessed 11/24)

[71] Chikodili, Nwodo Benita, et al. "Outlier Detection in Multivariate Time Series Data Using a Fusion of K-Medoid, Standardized Euclidean Distance and Z-Score." *Information and Communication Technology and Applications*, edited by Sanjay Misra and Bilkisu Muhammad-Bello, Springer International Publishing, 2021, pp. 259–71. *Springer Link*, https://doi.org/10.1007/978-3-030-69143-1_21.

[72] Scikit-learn documentation https://scikit-learn.org/stable/user_guide.html (accessed 02/25)

[73] NumPy documentation https://numpy.org/doc/stable/reference/generated/numpy.log1p.html (accessed 02/25)

[74] Pandas documentation https://pandas.pydata.org/docs/index.html (accessed 02/25)

[75] Cravero, Fiorella, et al. "Feature Selection for Polymer Informatics: Evaluating Scalability and Robustness of the FS4RV$_{DD}$ Algorithm Using Synthetic Polydisperse Data Sets." *Journal of Chemical Information and Modeling*, vol. 60, no. 2, Feb. 2020, pp. 592–603. *DOI.org (Crossref)*, https://doi.org/10.1021/acs.jcim.9b00867.

[76] XGBoost documentation https://xgboost.readthedocs.io/en/stable/ (accessed 04/25)

[77] Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2022.

Appendix

Below, a table with Python functions mentioned in this work will be presented, alongside their respective objectives.

Table I. List of mentioned Python functions with respective functionalities

Python function	Functionality
MinMaxScaler()	Transforms features by scaling them to a uniform range using minimum and maximum values.
MaxAbsScaler()	Transforms features by scaling them using their maximum absolute value. It does not shift data.
pca.fit_transform()	Fits data and then transforms it to principal components.
pd.dataframe.corr()	Computes pearson correlation coefficients.
mutual_info_regression()	Estimates Mutual Information score for a continuous variable.
random_forest_regressor()	Deploys a Random Forest algorithm that fits given data and makes predictions.
Kfold()	Splits data into k- given subsets using consecutive folds.
LinearRegression()	Deploys a Linear regression algorithm that fits given data and makes predictions.
SVR()	Deploys a Support Vector Regressor algorithm that fits given data and makes predictions.
GaussianProcessRegressor()	Deploys a Gaussian Process algorithm that fits given data and makes predictions.
GradientBoostingRegressor()	Deploys a Gradient Boosting algorithm that fits given data and makes predictions.
XGBRegressor()	Deploys an eXtreme Gradient Boosting algorithm that fits given data and makes predictions.
GridSearchCV()	Deploys the grid search algorithm which explores several potential hyperparameters for models, using a “fit” and “score” method.
np.sqrt(mean_squared_error())	Calculates the root square of calculated mean squared error.
mean_absolute_error()	Calculates the mean absolute error.
r2_score()	Calculates the R^2 score.

All code, excel files and respective plots and results produced during the course of this work can be found in GitHub, following this link: https://github.com/DanaiMorfi/ML-application-in-vitrimers