ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Πλατφόρμα αυτόνομης διαχείρισης μοντέλων μηχανικής μάθησης για την πρόβλεψη παραμέτρων πλοίων, στο πλαίσιο ενός πληροφοριακού συστήματος διαχείρισης στόλου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΗΣΑΪΑ Π. ΒΡΑΚΙΔΗ

Επιβλέπων: Ιάκωβος Στ. Βενιέρης
Καθηγητής Ε.Μ.Π.

Συμμετοχή στην Επίβλεψη:
Δρ. Σοφία Καπελλάκη
Ε.Δ.Ι.Π.

ΕΡΓΑΣΤΗΡΙΟ ΕΥΦΥΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ ΕΥΡΕΙΑΣ ΖΩΝΗΣ

Αθήνα, Νοέμβριος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΕΥΦΥΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ ΕΥΡΕΙΑΣ ΖΩΝΗΣ

Πλατφόρμα αυτόνομης διαχείρισης μοντέλων μηχανικής μάθησης για την πρόβλεψη παραμέτρων πλοίων, στο πλαίσιο ενός πληροφοριακού συστήματος διαχείρισης στόλου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΗΣΑΪΑ Π. ΒΡΑΚΙΔΗ

Επιβλέπων: Ιάκωβος Στ. Βενιέρης
Καθηγητής Ε.Μ.Π.

Συμμετοχή στην Επίβλεψη:
Δρ. Σοφία Καπελλάκη
Ε.Δ.Ι.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 12η Νοεμβρίου 2019.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Ι. Βενιέρης	Δ.-Θ. Κακλαμάνη	Γ. Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2019

(Υπογραφή)

.....

ΗΣΑΪΑΣ Π. ΒΡΑΚΙΔΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Η διπλωματική εργασία αυτή εκπονήθηκε στα πλαίσια των ερευνητικών δραστηριοτήτων του Εργαστηρίου Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης κατά το ακαδημαϊκό έτος 2018-2019. Το εργαστήριο υπάγεται στον Τομέα Συστημάτων Μετάδοσης Πληροφορίας και Τεχνολογίας Υλικών, της σχολής Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κ. Βενιέρη που μου έδωσε την ευκαιρία να ασχοληθώ με τα ζητήματα γύρω από την πρακτική εφαρμογή της Μηχανικής Μάθησης και για την επιστημονική υποστήριξη που μου παρείχε καθ' όλη τη διάρκεια της εκπόνησης της διπλωματικής εργασίας. Επιπλέον θα ήθελα να ευχαριστήσω την καθηγήτρια κ. Δ.-Θ. Κακλαμάνη και τον καθηγητή κ. Γ. Ματσόπουλο για τη συμμετοχή τους στην τριμελή επιτροπή της εξέτασης της διπλωματικής.

Θα ήθελα ιδιαίτερα να ευχαριστήσω την Διδάκτορα Σοφία Καπελλάκη για την πολύ καλή συνεργασία, τη συνεχή παρακολούθηση, τη συμβολή και καθοδήγηση τόσο σε οργανωτικό όσο και σε επιστημονικό επίπεδο.

Ακόμα, θα ήθελα να ευχαριστήσω την εταιρία METIS Cybertechnology και συγκεκριμένα τον Τεχνικό Διευθυντή κ. Σ. Κατσικά, την Υπεύθυνη Λογισμικού Κ. Μερμικλή και τον Υπεύθυνο Τεχνητής Νοημοσύνης Γ. Στρατάκο, για την διάθεση του συστήματος, την παραχώρηση των δεδομένων και την γενικότερη συνεργασία.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου και τους συναδέλφους μου για την ανιδιοτελή βοήθεια και υποστήριξη, με ιδιαίτερη αναφορά στους Α. Χρήστου, Γ. Μανδήτσιο, Ε. Ντάβλη και Θ. Παναγιώτου, καθώς και την οικογένεια μου που χάρη στη στήριξη τους ήταν εφικτή η ολοκλήρωση των σπουδών μου.

Περίληψη

Καθώς η Ναυτιλία εισέρχεται στην εποχή του Digital Shipping, δηλαδή, της εφαρμογής νέων τεχνολογιών ηλεκτρονικών και πληροφορικής σε μεγάλη έκταση πάνω στα πλοία, εντείνεται η ανάγκη ανάπτυξης εργαλείων ώστε να αυξηθεί η απόδοση αυτών και να ελαχιστοποιηθεί η επίπτωσή τους στο περιβάλλον. Τεχνολογίες όπως Machine Learning, Cloud Computing και Internet of Things (IoT) λαμβάνουν εκτεταμένη χρήση για την επίτευξη αυτού του σκοπού.

Η πληροφορία που παράγεται από τα πλοία και το περιβάλλον τους συλλέγεται με τρόπο που μπορεί να υποστηρίξει την εφαρμογή τεχνικών μηχανικής μάθησης, παρουσιάζοντας, όμως, αρκετές προκλήσεις. Καθώς ο αριθμός των αναπτυσσομένων μοντέλων μηχανικής μάθησης αυξάνεται, καθίσταται αδύνατη η εκτέλεση της ροή εργασιών, που απαιτείται, εξ ολοκλήρου από ανθρώπινο δυναμικό.

Στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας πλατφόρμας για την εύκολη δημιουργία και ενσωμάτωση μοντέλων μηχανικής μάθησης σε IoT συστήματα διαχείρισης στόλου, καθιστώντας, με αυτόν τον τρόπο, εφικτή η εκτεταμένη χρήση της μηχανικής μάθησης, για την ανάλυση των δεδομένων των πλοίων και την παραγωγή προβλέψεων. Μέσω της πλατφόρμας θα διαχειρίζεται μεγάλος αριθμός μοντέλων, θα παρακολουθείται η απόδοσή τους και θα προσαρμόζονται στις μεταβολές των καταστάσεων των πλοίων. Θα παρουσιαστεί ο σχεδιασμός της, ο τρόπος με τον οποίο μπορεί να διασυνδέεται σε άλλα συστήματα, καθώς και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξή της. Τέλος, θα προσομοιωθεί η λειτουργία της πλατφόρμας ώστε να αναδειχθεί η συνεισφορά της κατά την εφαρμογή της μηχανικής μάθησης.

Λέξεις Κλειδιά

Μηχανική Μάθηση, Ναυτιλία, Διαδίκτυο των Πραγμάτων, Υπολογιστικό Νέφος.

Abstract

As maritime operations enters the era of Digital Shipping, there is a growing need for technological tools to increase their efficiency and minimize their impact on the environment. Technologies such as Machine Learning, Cloud Computing and Internet of Things (IoT) are extensively used under this scope.

The data acquired from vessels and their environment can support the application of machine learning. This task, though, presents several challenges. As the number of the developing machine learning models increases, the execution of the required workflow entirely by humans becomes infeasible.

The purpose of this thesis is the development of a platform for the effortless creation and integration of machine learning models into IoT fleet management systems, thereby making possible the extensive usage of machine learning under the scope of analyzing ship data and performing predictions. The platform will be able to manage large numbers of models, monitor their performance and enable their adaption to changes in ship conditions. The platform's design, the way the interconnection with other systems is achieved, as well as the technologies used in its development, will be presented. Finally, the platform's operation will be simulated to highlight its contribution to the application of machine learning.

Keywords

Machine Learning, Maritime, Digital Shipping, Internet of Things, Cloud Computing.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Το θαλάσσιο περιβάλλον αποτελεί διαχρονικά σημείο έντονης ανθρώπινης δραστηριότητας πληθώρας τύπων (εμπόριο, μεταφορές, τουρισμός, αλιεία κ.λ.π.). Κατά συνέπεια η βιομηχανία που έχει αναπτυχθεί γύρω από αυτό το περιβάλλον, η ναυτιλιακή βιομηχανία, αποτελεί μία από τις σημαντικότερες και πιο ισχυρές βιομηχανίες. Η επιστράτευση και αξιοποίηση νέων τεχνολογιών στην βιομηχανία αυτή αποτελούσε ανέκαθεν ζήτημα υψηλής προτεραιότητας. Αρκετές τέτοιες προσπάθειες επικεντρώνονται γύρω από τον τομέα των τηλεπικοινωνιών καθώς η επικοινωνία και η μεταφορά δεδομένων μεταξύ θάλασσας και στεριάς αποτελεί σημείο κρίσιμης σημασίας.

Σε συνέχεια της ανάπτυξης και εγκατάστασης δορυφορικών δικτύων που παρέχουν την δυνατότητα διασύνδεσης των πλοίων στο διαδίκτυο με ικανοποιητικές χωρητικότητες και σε προσιτό κόστος, η Ναυτιλία εισέρχεται στην εποχή του Digital Shipping. Με τον όρο αυτόν περιγράφεται η εφαρμογή νέων τεχνολογιών ηλεκτρονικών και πληροφορικής σε μεγάλη έκταση πάνω στα πλοία, ώστε να αυξηθεί η απόδοση αυτών και να ελαχιστοποιηθεί η επίπτωσή τους στο περιβάλλον. Η διαχείριση των πλοίων αλλάζει καθώς ο όγκος της συλλεγόμενης πληροφορίας κατευθείαν από τα συστήματα των πλοίων αυξάνεται με μεγάλους ρυθμούς. Κατά συνέπεια, η ανάγκη δημιουργίας εργαλείων επεξεργασίας και ανάλυσης των δεδομένων σε πραγματικό χρόνο εντείνεται τα τελευταία χρόνια. Πληροφοριακά συστήματα για την αξιοποίηση αυτών των πληροφοριών και την υποστήριξη του ανθρωπίνου δυναμικού που είναι υπεύθυνο για την διαχείριση των πλοίων, αναπτύσσονται ραγδαία.

Λόγω της συνεχούς ροής δεδομένων σχετικά υψηλού ρυθμού δειγματοληψίας που παρέχεται από τέτοια συστήματα, ο όγκος της συλλεγόμενης πληροφορίας μπορεί να επιτρέψει την εφαρμογή τεχνικών μηχανικής μάθησης. Καθώς επίσης, το περιβάλλον που συνθέτεται γύρω από ένα πλοίο είναι αρκετά σύνθετο η μηχανική μάθηση προσφέρει τα μέσα για την αξιόπιστη μοντελοποίησή του. Κατά συνέπεια, η χρήση τεχνικών μηχανικής μάθησης ενδέχεται να επιφέρει αξιοποιήσιμα αποτελέσματα. Ιδιαίτερη σημαντικότητα βρίσκεται στο γεγονός ότι η εφαρμογή αυτών θα μπορεί να γίνεται σε μεγάλη κλίμακα και σε χαμηλό κόστος, σε αντίθεση με τεχνικές μέτρησης συμπεριφοράς μηχανικών μοντέλων σε υδροδυναμικές πισίνες, χρήση CFD αναλύσεων και άλλων μεθοδολογιών που χρησιμοποιούνται στον παρόντα χρόνο.

1.2 Αντικείμενο της διπλωματικής

Αντικείμενο της παρούσας διπλωματικής εργασίας θα αποτελέσει η ανάπτυξη μίας πλατφόρμας, η οποία θα μπορεί να χρησιμοποιηθεί από συστήματα άντλησης και επεξεργασίας δεδομένων πάνω από τα πλοία και θα θέτει στην διάθεση τους τις δυνατότητες που προσφέρει η μηχανική μάθηση, ελαχιστοποιώντας την ανάγκη επαναλαμβανόμενης ανθρώπινης επέμβασης. Ως εκ τούτου, θα σχεδιαστεί και θα αναπτυχθεί σε περιβάλλον υπολογιστικού νέφους μια πλατφόρμα αυτόνομης δημιουργίας, αξιολόγησης, συντήρησης και διαχείρισης μοντέλων μηχανικής μάθησης. Σκοπός της πλατφόρμας είναι η δυναμική μοντελοποίηση παραμέτρων των πλοίων και η ”on demand” πραγματοποίηση προβλέψεων γι αυτές. Επιπλέον, η δυνατότητα συγκρότησης μεγάλου πλήθους μοντέλων μηχανικής μάθησης κρίνεται απαραίτητη, λόγου του πολυπαραμετρικού χαρακτήρα του περιβάλλοντος ενός πλοίου, αλλά και της εξυπηρετησης μεγάλου αριθμού αυτών.

Για τις ανάγκες της παρουσίασης της πλατφόρμας θα αναπτυχθεί ένα μοντέλο για την πρόβλεψη του ρυθμού της κατανάλωσης καυσίμου ενός πλοίου και θα προσομοιωθεί η συμπεριφορά του εντός της πλατφόρμας. Στην συνέχεια θα συγκριθεί η πορεία της απόδοσής του με την απόδοση που θα είχε σε περιβάλλον εκτός πλατφόρμας. Θα χρησιμοποιηθούν δεδομένα ενός πλοίου για διάστημα 9 μηνών.

1.3 Οργάνωση κειμένου

Η παρούσα διπλωματική εργασία δομείται ως εξής:

- Το Κεφάλαιο 1 αποτελεί την εισαγωγή της διπλωματικής, όπου περιγράφεται το αντικείμενο της καθώς και οι στόχοι.
- Στο Κεφάλαιο 2, αναλύεται το υπόβαθρο της εργασίας. Γίνεται μία εισαγωγή στο περιβάλλον της ναυτιλίας και στην έννοια της μηχανικής μάθησης. Περιγράφεται, επίσης, το σύστημα συλλογής δεδομένων που επιλέχθηκε και ο τρόπος λειτουργίας του. Ακολουθεί ο προσδιορισμός του προβλήματος.
- Στο Κεφάλαιο 3, παρουσιάζονται όλες οι τεχνολογίες που μελετήθηκαν και χρησιμοποιήθηκαν στα πλαίσια αυτής της εργασίας.
- Το Κεφάλαιο 4, περιέχει την ανάλυση των απαιτήσεων, των λειτουργιών και των σεναρίων χρήσης της πλατφόρμας. Στη συνέχεια ακολουθεί η σχεδίαση της όπου περιγράφεται η αρχιτεκτονική της και τα δομικά της μέρη. Αναλύεται, επίσης, ο κύκλος ζωής που ακολουθούν τα μοντέλα της μηχανικής μάθησης εντός της πλατφόρμας.
- Στο Κεφάλαιο 5, περιγράφονται αναλυτικά επιλεγμένα τμήματα της ανάπτυξης της πλατφόρμας. Παρουσιάζεται, επίσης, ένα μοντέλο μηχανικής μάθησης και η ανάπτυξή του.
- Στο Κεφάλαιο 6 πραγματοποιείται μία προσομοίωση της λειτουργίας της πλατφόρμας μέσα από τον κύκλο ζωής του μοντέλου του προηγούμενου κεφαλαίου. Παρουσιάζονται, επίσης, τα αποτελέσματα που έχει επιφέρει η χρήση της πλατφόρμας, μέσω των αξιολογήσεων του μοντέλου συγκριτικά με την απόδοση που θα είχε εκτός πλατφόρμας σε βάθος χρόνου.

- Τέλος, στο Κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα που προκύπτουν και οι εν δυνάμει επεκτάσεις της πλατφόρμας.

Κεφάλαιο 2

Υπόβαθρο

2.1 Το περιβάλλον της ναυτιλίας

Οι ναυτιλιακές εταιρείες δρουν σε ένα πολυσύνθετο και ανταγωνιστικό επιχειρηματικό περιβάλλον. Ένα περιβάλλον εντός του οποίου βασικός πυλώνας της λειτουργίας, των στόχων και των ενεργειών τους θα πρέπει να είναι η οργάνωση και η διοίκηση. Ως διοίκηση ορίζουμε το σύνολο των διαδικασιών για τις βασικές λειτουργίες προγραμματισμού, οργάνωσης, διεύθυνσης και ελέγχου, με σκοπό την, κατά το δυνατόν, αποδοτικότερη χρήση πόρων για την μεγιστοποίηση της αποτελεσματικότητας. Πιο συγκεκριμένα, σε ένα ναυτιλιακό επιχειρηματικό περιβάλλον, στον ρόλο της διοίκησής εμπίπτουν ο προγραμματισμός, η οργάνωση των πόρων, εν συνεχεία η καθοδήγηση, η διεύθυνση και η ενεργοποίηση των εκάστοτε διαδικασιών και τέλος ο έλεγχος και η αξιολόγηση του αποτελέσματος. Υψηλή σημαντικότητα, επίσης, εντοπίζεται και στον τρόπο με τον οποίο δομείται μία ναυτιλιακή εταιρία, δηλαδή ο τρόπος με τον οποίο είναι σχεδιασμένα τα διάφορα οργανογράμματα για την εκτέλεση των λειτουργιών της.

Το οργανόγραμμα που συντίθεται γύρω από την διαχείριση ενός πλοίου διαφέρει από εταιρία σε εταιρία, από πλοίο σε πλοίο, από χώρα σε χώρα, από εποχή σε εποχή. Για παράδειγμα, διαφορετικό θα είναι το οργανόγραμμα για ένα πλοίο μεταφοράς χύδην φορτίου και διαφορετικό για ένα επιβατηγό πλοίο. Αν παρακολουθήσει κανείς τα διάφορα οργανογράμματα από τις αρχές του 20ου αιώνα, θα παρατηρήσει την εξέλιξη και την ανάπτυξη των ναυτιλιακών επιχειρήσεων. Η αύξηση των τμημάτων, των πλοίων και των ανθρώπων σε συνδυασμό με την τεχνολογική εξέλιξη, τον όγκο των δεδομένων και τη ταχύτητα της πληροφορίας, έχουν συμβάλει σημαντικά στη δημιουργία ενός πολυσύνθετου ναυτιλιακού οργανογράμματος, σήμερα. Μια ναυτιλιακή εταιρεία απαρτίζεται συνήθως από την γενική διεύθυνση (general management), το τμήμα επιχειρήσεων (operations department), το τεχνικό τμήμα (technical department), το τμήμα πληρωμάτων (crew department), το τμήμα ασφαλίσεων (insurance claims department), το τμήμα λογιστηρίου (accounting department), το τμήμα προμηθειών (purchasing supply department), το τμήμα ποιότητας και ασφάλειας (SQE, Safety Quality department) και το νομικό τμήμα (legal department).

Η γενική διεύθυνση αποτελεί το κυριότερο τμήμα της επιχείρησης καθώς αποτελεί τον πυρήνα των αποφάσεων, της στρατηγικής, της καθοδήγησης και της συνεργασίας με τα υπόλοιπα τμήματα, για την επίτευξη των στόχων. Επίσης, καθορίζει τις σχέσεις με το ευρύτερο ναυτιλιακό χώρο, όπως οργανισμούς, τράπεζες κτλ. Επικεφαλής είναι συνήθως ο πλοιοκτήτης ή αντίστοιχα ένα έμπειρο στέ-

λεχος στο χώρο της ναυτιλίας.

Το τμήμα επιχειρήσεων είναι από τα βασικότερα τμήματα μίας ναυτιλιακής επιχείρησης, μιας και κύρια ασχολία του αφορά τα ταξίδια των πλοίων(vessels). Συγκεκριμένα, καπετάνιοι, πλοίαρχοι, οικονομολόγοι με άριστη γνώση του ναυτιλιακού οικοσυστήματος ασχολούνται με την λειτουργία των πλοίων, την άφιξη τους στα λιμάνια και την επικοινωνία με τους αντίστοιχους πράκτορες (agents). Υποτμήμα του αποτελεί το τμήμα ναυλώσεων (chartering department or brokering department), το οποίο μπορεί είτε να βρίσκεται εντός της ναυτιλιακής εταιρίας είτε να υφίσταται μέσω συνεργασίας της εταιρίας με κάποιο εξωτερικό συνεργάτη που ασχολείται με τις ναυλώσεις πλοίων. Στόχος του συγκεκριμένου τμήματος είναι η αναζήτηση ναύλων για τα πλοία στα πλαίσια του ανταγωνισμού και της διεθνή ναυλαγοράς, όπως επίσης και η τήρηση των υπογραφέντων ναυλοσυμφώνων. Τέλος, το τμήμα επιχειρήσεων είναι υπεύθυνο για την διαχείριση οποιουδήποτε προβλήματος μπορεί να παρουσιαστεί και να αφορά είτε το πλήρωμα, είτε το φορτίο του πλοίου.

Το τεχνικό τμήμα είναι υπεύθυνο για τη σωστή λειτουργία του πλοίο ως «μονάδα». Μηχανικοί, ναυπηγοί, μηχανολόγοι είναι υπεύθυνοι για τη συντήρηση του πλοίου, των εξαρτημάτων και μηχανημάτων. Είναι επίσης υπεύθυνοι για την επισκευή και επιδιόρθωση τυχόν βλαβών που μπορεί να παρουσιάσει το πλοίο. Η εμπειρία έχει δείξει, ότι με συστηματική παρακολούθηση, επέρχονται τα λιγότερα δυνατά κόστη σε τεχνικό, οικονομικό και περιβαλλοντικό επίπεδο. Για παράδειγμα, ένα καράβι το οποίο είναι μερικών ετών και δεν έχει πραγματοποιηθεί η οποιαδήποτε συντήρηση στα μηχανήματα του, μπορεί να φέρει βλάβες οι οποίες να κοστίσουν πολύ περισσότερο από τις επικείμενες συντηρήσεις. Επομένως, το τεχνικό τμήμα είναι ένα επίσης κομβικό τμήμα του ναυτιλιακού οικοσυστήματος.

Όσον αφορά τα υπόλοιπα τμήματα μια ναυτιλιακής επιχείρησης, αποτελούν και αυτά σημαντικά μέρη της. Το τμήμα πληρωμάτων ασχολείται με την επιλογή κατάλληλα εκπαιδευμένου πληρώματος για την επάνδρωση των πλοίων και αποτελείται ως επί το πλείστων, από καπετάνιους. Είναι υπεύθυνο, επίσης, για την διαχείριση εκτάκτων καταστάσεων σχετικά με το πλήρωμα. Το τμήμα ασφαλίσεων ασχολείται με την ασφαλιστική κάλυψη κάθε πλοίου και των εμπλεκόμενων μερών, αποτελείται από νομικούς ή μέλη τα οποία έχουν ειδίκευση στο ναυτικό δίκαιο και στις ναυτασφαλίσεις. Το λογιστήριο είναι υπεύθυνο για την διαχείριση των οικονομικών της εταιρίας καθώς βασική του αρμοδιότητα είναι ο έλεγχος των εσόδων και εξόδων της ναυτιλιακής επιχείρησης. Το εν λόγω τμήμα, απαρτίζεται από οικονομολόγους και λογιστές. Ένα ακόμα τμήμα του οποίου η σημαντικότητα αναδείχτηκε στα μέσα της δεκαετίας του ’90 είναι το τμήμα ποιότητας και ασφάλειας. Στόχος του αποτελεί η τήρηση των κανόνων ποιότητας και ασφάλειας που ορίζονται στον Διεθνή Κώδικα Ασφαλούς Διαχειρίσεως (ISM Code), ο οποίος εκπονείται από τον Διεθνές Ναυτιλιακό Οργανισμό (IMO). Στελεχώνεται κυρίως από άτομα με συνολικές γνώσεις γύρω από τα ζητήματα σε μία ναυτιλιακή εταιρία. Το τμήμα προμηθειών είναι αυτό που αναλαμβάνει τον ανεφοδιασμό των πλοίων με τρόφιμα, ανταλλακτικά και αναλώσιμα υλικά. Στις αρμοδιότητές του εμπίπτει ο έλεγχος και η αγορά των απαραίτητων μηχανημάτων, ανταλλακτικών και εργαλείων και λειτουργεί υπό την διεύθυνση πλοιάρχων σε συνεργασία με το τεχνικό τμήμα. Τέλος, το νομικό τμήμα το οποίο ασχολείται με το νομικό πλαίσιο που αφορά θέματα ναυτικού δικαίου και στελεχώνεται από δικηγόρους και νομικούς.

Σαφές γίνεται ότι γύρω από την λειτουργία ενός πλοίου εμπλέκεται ένα πολύπλοκο σύστημα από ενδιαφερόμενες ομάδες. Η εύκολη και αποδοτική επικοινωνία μεταξύ όλων αυτών των ομάδων, αποτελεί ένα από τα κρισιμότερα ζητήματα στον κλάδο της ναυτιλίας.

2.2 Πλατφόρμες συλλογής και επεξεργασίας δεδομένων

Όπως αναφέρθηκε η Ναυτιλιακή Βιομηχανία δραστηριοποιείται σε ένα αρκετά σύνθετο και απαιτητικό περιβάλλον. Δημιουργούνται έτσι ανάγκες αντίστοιχου χαρακτήρα, οι οποίες συνοψίζονται στα εξής:

- Τροφοδότησή της ομάδας στην ξηρά με αξιόπιστες πληροφορίες που έρχονται σε πραγματικό χρόνο από το πλοίο, για την λήψη καλύτερων και πιο άμεσων αποφάσεων.
- Αξιόπιστες προβλέψεις για την απόδοση του πλοίου και ταχύτερη διάγνωση προβλημάτων.
- Μείωση του κινδύνου αποτυχιών και ατυχημάτων με την παροχή συμβουλών και ειδοποιήσεων στο αρμόδιο προσωπικό σε έγκαιρο χρόνο.
- Εξασφάλιση της συμμόρφωσης με τους περιβαλλοντικούς κανονισμούς.
- Μείωση της γραφειοκρατίας και απλοποίηση των διαδικασιών. Π.χ. η μείωση του χρόνου που απαιτείται για την υποβολή αναφορών.

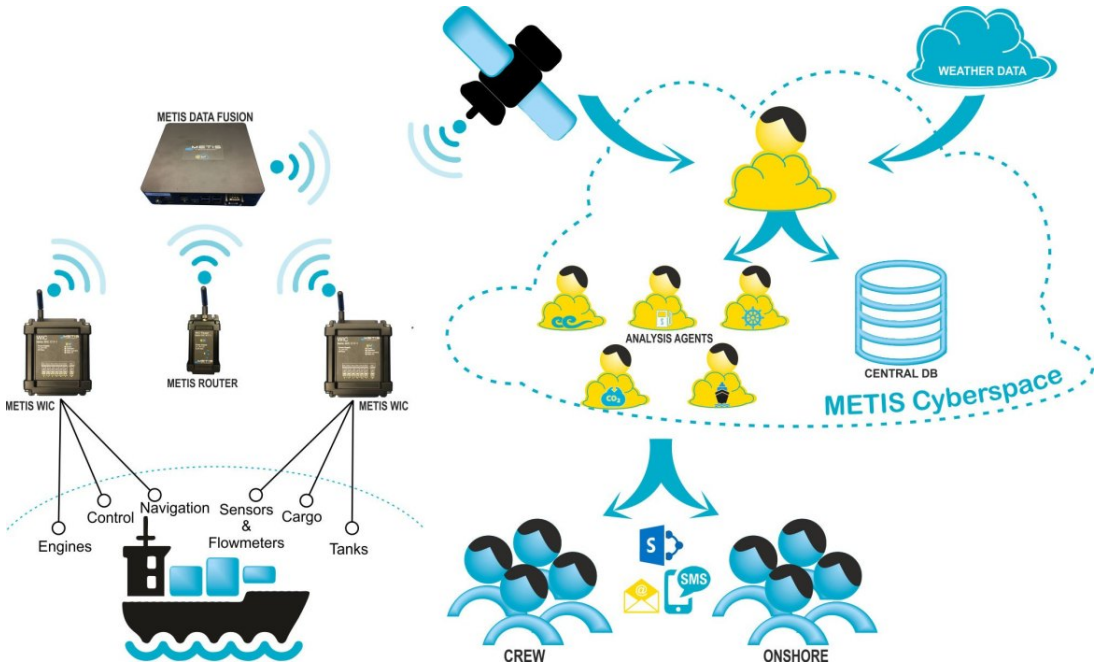
Συστήματα συλλογής και επεξεργασίας δεδομένων μπορούν να καλύψουν τέτοιου είδους ανάγκες, κάνοντας χρήση τεχνολογιών Υπολογιστικού Νέφους, Διαδικτύου των Πραγμάτων και Τεχνητής Νοημοσύνης.

Από τα υπάρχοντα συστήματα για τον σκοπό αυτό έχει επιλεγεί, για τις ανάγκες της διπλωματικής εργασίας, το σύστημα που αναπτύχθηκε από την εταιρία METIS [1]. Το σύστημα αυτό αποτελεί μία συνάθροιση συσκευών συλλογής δεδομένων και Cloud υπηρεσιών, παρέχοντας δεδομένα υψηλής ποιότητας σε πραγματικό χρόνο, γι αυτό και επιλέχθηκε έναντι των υπόλοιπων επιλογών. Εφεξής το σύστημα που έχει αναπτυχθεί από την εταιρία METIS και θα αποτελέσει την πηγή των δεδομένων που θα χρησιμοποιηθούν από την πλατφόρμα θα αναφέρεται ως ”το Σύστημα” και η πλατφόρμα που αναπτύσσεται στο πλαίσιο της διπλωματικής ”η Πλατφόρμα”. Στην συνέχεια θα περιγραφεί εν συντομία ο τρόπος λειτουργίας του Συστήματος για την διεύρυνση της κατανόησης γύρω από το περιβάλλον με το οποίο θα συνεργάζεται η Πλατφόρμα.

Οι συσκευές του Συστήματος συλλέγουν, συγκεντρώνουν και καθαρίζουν τα δεδομένα από οποιοδήποτε εξοπλισμό, όργανο ή αισθητήρα πάνω στο πλοίο και τα μεταδίδουν στο Cloud του Συστήματος μέσω δορυφορικής σύνδεσης. Εκεί λαμβάνουν χώρα οι διάφορες αναλύσεις των δεδομένων από τις υπηρεσίες του Συστήματος. Οι υπηρεσίες αυτές καθιστούν, επίσης, ικανή την δια-λειτουργικότητα με άλλα συστήματα από τα οποία, είτε αντλούνται πρόσθετα δεδομένα είτε διατίθενται αποτελέσματα από τις αναλύσεις. Τα προσφερόμενα εργαλεία και αναλύσεις γίνονται διαθέσιμα στους χρήστες της πλατφόρμας μέσω μίας διεπαφής φυσικής γλώσσας χρησιμοποιώντας κανάλια όπως e-mail, skype etc, καθώς και από έναν διαδικτυακό ιστότοπο.

2.2.1 Συλλογή Δεδομένων

Μια ναυτιλιακή εταιρεία, χρησιμοποιώντας κάποιο σύστημα συλλογής δεδομένων (data acquisition), μπορεί να αποκτήσει πρόσβαση σε σημαντικό όγκο πληροφορίας προερχόμενη από διάφορες πηγές πάνω στο πλοίο αλλά και έξω από αυτό. Αυτή η δυνατότητα δεν υπήρχε πριν από μερικά χρόνια. Η



Σχήμα 2.1: Επισκόπηση του Συστήματος.

συλλογή δεδομένων μπορεί να περιλαμβάνει όλες τις πληροφορίες από τον εξοπλισμό πλοήγησης (όπως αυτόματο πιλότο, γυροσκόπιο, GPS, speedlog, echosounder, ανεμόμετρο, AIS κ.λ.π.), τον έλεγχο και την λειτουργία φορτοεκφόρτωσης, την λειτουργία της κύριας μηχανής, την λειτουργία της ντιζελογεννήτριας, την λειτουργία του λέβητα, την κατανάλωση καυσίμου, το σύστημα παρακολούθησης συναγερμού κ.λ.π..

Αυτό που είναι πολύ σημαντικό σε σχέση με τα μετρηθέντα δεδομένα που συλλέγονται από τον εξοπλισμό του πλοίου (ή ακόμη και από το πλήρωμα), είναι οι αξιόπιστες και αποτελεσματικές διαδικασίες ανάληψης, καθαρισμού και αποθήκευσης που πρέπει να εφαρμόζονται στα πλαίσια των διαδικασιών του συστήματος απόκτησης. Αυτές επιτρέπουν τη λήψη δεδομένων από οποιοδήποτε εξοπλισμό, αισθητήρα και όργανο, αντιμετωπίζοντας τα με κοινό τρόπο, καθώς αυτά παρουσιάζουν διαφορετική μορφή, διαμόρφωση, ακρίβεια και χρονική σήμανση. Επομένως, ένα επαρκές επίπεδο διασύνδεσης του εξοπλισμού και των συστημάτων του πλοίου είναι απαραίτητο. Αυτό θα επιτρέψει σε όλες τις διαφορετικές συσκευές να έχουν μια ενιαία σύντηξη δεδομένων, διευκολύνοντας σε μεγάλο βαθμό τις μετέπειτα αναλύσεις.

Το Σύστημα είναι σχεδιασμένο για να καλύψει τις προαναφερθείσες ανάγκες, γι αυτό και αποτέλεσε το επιλεχθέν σύστημα για την ανάπτυξη της Πλατφόρμας. Το πεδίο εφαρμογής του Συστήματος είναι η διασύνδεση οποιουδήποτε αισθητήρα ή οργάνου στα πλοία σε ένα κοινό δίκτυο για την απόκτηση όλων των χρήσιμων δεδομένων και την προώθησή τους μέσω μίας ενιαίας ροής σε οποιοδήποτε πληροφοριακό σύστημα ανάλυσης δεδομένων. Τρία βασικά χαρακτηριστικά του Συστήματος αποτελούν:

1. Το κύριο δίκτυο για τη μεταφορά των δεδομένων είναι ένα ασύρματο δίκτυο ZigBee με βάση το πρωτόκολλο IEEE 802.15.4.

2. Το πρώτο επίπεδο φιλτραρίσματος και προ-επεξεργασίας των σημάτων στις συσκευές συλλογής δεδομένων είναι αρκετό, ώστε να εξασφαλίσει άρτια δεδομένα σε οποιοδήποτε 3ο πληροφορια-

κό σύστημα.

3. Το σύστημα υπολογίζει το σφάλμα κάθε μέτρησης λαμβάνοντας υπόψη τη μέθοδο δειγματοληψίας και επεξεργασίας των σημάτων.

Το δίκτυο του Συστήματος υποστηρίζεται από συσκευές τριών διαφορετικών τύπων:

- DFS: Το DFS είναι ουσιαστικά είναι ένα σύνολο προγραμμάτων τα οποία τρέχουν σε ένα διακομιστή μέσα στο σκάφος. Σκοπός του είναι να συγκεντρώσει όλα τα δεδομένα από τις συσκευές WIC, να δημιουργήσει πακέτα από αυτές και να τα στείλει στο σύστημα ανάλυσης δεδομένων μέσω της ευρυζωνικής σύνδεσης του σκάφους χρησιμοποιώντας ένα κανάλι sFTP. Ο διακομιστής θα μπορούσε να είναι οποιοσδήποτε υπολογιστής με τις ελάχιστες απαιτήσεις για την εκτέλεση του πακέτου λογισμικού DFS.

- WIC: Είναι συσκευές που λαμβάνουν τα δεδομένα από τον συνδεδεμένο εξοπλισμό του πλοίου υποστηρίζοντας πληθώρα πρωτοκόλλων, εξαγάγουν τις απαιτούμενες πληροφορίες, εκτελούν ένα πρώτο επίπεδο ανάλυσης των σημάτων και μεταδίδουν περιοδικά τα δεδομένα στη μονάδα DFS είτε μέσω του ZigBee ασύρματου δικτύου ή μέσω του LAN δικτύου του πλοίου.

- Router WIC: Το Router WIC είναι μια ηλεκτρονική συσκευή που μπορεί να χρησιμοποιηθεί είτε για την ενίσχυση ή την επέκταση της κάλυψης του ασύρματου δικτύου, είτε για τη διασύνδεση οποιουδήποτε υπολογιστή με το ασύρματο δίκτυο των συσκευών WIC μέσω θύρας USB 2.0.

2.2.2 Υπολογιστικό Νέφος

Στο υπολογιστικό Νέφος του Συστήματος λαμβάνουν χώρα όλες οι διαδικασίες που εκτελούνται πάνω στα δεδομένα που έχουν συλλεχθεί, από την επεξεργασία τους και την ανάλυσή τους, μέχρι την παρουσίαση των αποτελεσμάτων στους χρήστες.

Οι διάφορες αναλύσεις και υπηρεσίες υλοποιούνται από ένα σύνολο microservices τα οποία ονομάζονται Analysis Agents. Κάθε Agent αναλαμβάνει ένα σύνολο από υπηρεσίες σχετικές μεταξύ τους. Προκύπτουν έτσι οι παρακάτω Analysis Agents:

- Diesel Generator Agent. Προσφέρει αναλύσεις και πληροφορίες σχετικά με τις ντιζελογεννήτριες του πλοίου.

- Fuel Oil Consumption Agent. Πληροφορίες σχετικές με την κατανάλωση του πλοίου και τα κόστη γύρω από αυτή.

- Hull Performance Agent. Παρέχει πληροφορίες σχετικά με τις απώλειες ισχύος λόγω των καιρικών συνθηκών, της κατάστασης της θάλασσας και της κατάστασης του κύτους του πλοίου.

- Main Engine Agent. Αναλαμβάνει οτιδήποτε σχετικό με την κύρια μηχανή του πλοίου.

- Manual Import Agent. Επιτρέπει την εισαγωγή εγγράφων όπως bunker notes, noon reports, cargo reports κ.λ.π.. Ο Agent αναλαμβάνει να εξάγει από τα έγγραφα και να εισάγει στο σύστημα αξιοποιήσιμα δεδομένα που υπάρχουν.

- Operation Agent. Παρέχει γενικές πληροφορίες σχετικά με το ταξίδι και την κατάσταση του πλοίου.
- Period Export Agent. Εξάγει δεδομένα του πλοίο και τα στέλνει στον χρήστη μέσω email. Επιπλέον, παρέχει γραφήματα σχετικά με την κατάστασης της θάλασσας, την απόδοση της κυριας μηχανής κ.λ.π..
- Prediction Agent. Παρέχει εκτιμήσεις, με βάση μοντέλα μηχανικής μάθησης, σχετικά με ποσότητες του πλοίου. Αποτελεί τον αμεσότερο ”χρήστη” της πλατφόρμας που πραγματεύεται η διπλωματική εργασία.
- Reporting Agent. Επιτρέπει την εξαγωγή αναφορών σχετικά με τη λειτουργία του πλοίου.
- Vessel Registry Agent. Προσφέρει γενιές πληροφορίες για το πλοίο. Την μορφολογία του, τα ιδιαίτερα χαρακτηριστικά του, ποσότητες για τις οποίες είναι διαθέσιμες μετρήσεις κ.λ.π..
- Weather Agent. Παρέχει πληροφορίες σχετικά με τις καιρικές συνθήκες γύρω από το πλοίο.

Όλες αυτές οι υπηρεσίες και αναλύσεις παρέχονται στον τελικό χρήστη μέσω μηνυμάτων σε μορφή ερωταπαντήσεων, μέσω αναλυτικών αναφορών, ή και μέσω ειδοποιήσεων όταν εντοπίζονται από τους διάφορους Agents πιο σημαντικές ή έκτακτες καταστάσεις. Για την ανάπτυξη της διεπαφής με την παραπάνω μορφή έχει σχεδιαστεί ένα Chat Bot το οποίο με χρήση επεξεργασίας φυσικής γλώσσας και μηχανικής μάθησης και μέσω διάφορων καναλιών όπως Skype, Slack κ.λ.π. είναι σε θέση να αναπαραστήσει έναν προσωπικό εικονικό βοηθό για τον κάθε χρήστη - υπάλληλο της ναυτιλιακής εταιρίας.

2.3 Μηχανική Μάθηση

Η μάθηση είναι ένα σύνθετο βιολογικό και πνευματικό φαινόμενο που έχει μελετηθεί από διάφορους κλάδους της επιστήμης όπως ψυχολογία, παιδαγωγική, φυσιολογία, ιατρική, βιολογία και άλλοι. Οι διαδικασίες της μάθησης είναι τόσο ποικιλόμορφες και διαφορετικές, ώστε η ένταξη τους σε μία και μοναδική κατηγορία δεν μπορεί να είναι βάσιμη και πλήρης. Παρότι έχει διεξαχθεί πληθώρα σχετικών μελετών, η μάθηση παραμένει μια διαδικασία η οποία δεν έχει ερμηνευτεί και κατανοηθεί πλήρως και κατά τρόπο παραδεκτό από όλους, όσους ασχολούνται με αυτή [2]. Στο πεδίο της επιστήμης των υπολογιστών προσπάθειες μίμησης της διαδικασίας της μάθησης έχουν ξεκινήσει από τις αρχές της δεκαετίας του 50. Όχι πολύ αργότερα, σχηματίζεται, ως κλάδος της Τεχνητής Νοημοσύνης, το πεδίο της Μηχανικής Μάθησης.

Η Μηχανική Μάθηση είναι η επιστημονική μελέτη των αλγορίθμων και των στατιστικών μοντέλων που χρησιμοποιούν τα υπολογιστικά συστήματα, για την αποτελεσματική εκτέλεση μιας συγκεκριμένης εργασίας χωρίς να χρησιμοποιούν ρητές οδηγίες, αλλά εντοπίζοντας μοτίβα και εξάγοντας συμπεράσματα από σύνολα δεδομένων. Οι αλγόριθμοι μηχανικής μάθησης δημιουργούν μαθηματικά μοντέλα τα οποία απορρέουν από την επεξεργασία ενός δείγματος δεδομένων, το οποίο αποκαλείται συχνά ”training set”, προκειμένου να διεξάγουν προβλέψεις ή να λάβουν αποφάσεις χωρίς να έχουν σχεδιαστεί ρητά για την εκτέλεση αυτών των εργασιών [3]. Αυτή η διαδικασία αντιστοιχίζεται με την

λειτουργία της μάθησης στους ζωντανούς οργανισμούς, καθώς από την αλληλεπίδραση με εξωτερικά ερεθίσματα και την επεξεργασία αυτών προκύπτει ”γνώση” και με βάση αυτή προσαρμόζεται η συμπεριφορά τους. Θα μπορούσαμε να πούμε, γενικά, ότι μια μηχανή μαθαίνει όταν μεταβάλλονται οι παράμετροι, οι οποίες ορίζουν την συμπεριφορά που θα έχει όταν λαμβάνει κάποιο ερέθισμα, με τέτοιο τρόπο ώστε να είναι αναμενόμενο να ανταποκρίνεται καλύτερα σε μελλοντικά ερεθίσματα. Ένα σύστημα Μηχανικής Μάθησης, επομένως, δέχεται εξωτερικά ερεθίσματα μέσω των εισόδων του, και παράγει κάποια ή κάποιες εξόδους. Οι είσοδοι και οι έξοδοι που αναπαριστούν την σύνδεση του συστήματος με τον εξωτερικό κόσμο είναι ή μετατρέπονται σε αριθμητικές ποσότητες.

Γιατί όμως είναι χρήσιμο οι μηχανές να μαθαίνουν; Γιατί να μην κατασκευάσουμε μηχανές που θα επιτελούν την εργασία που θέλουμε εξ’αρχής, χωρίς να προϋποθέτουν μάθηση; Θα μπορούσαμε να πούμε ότι αναπτύσσοντας συστήματα τα οποία μαθαίνουν θα μας βοηθούσε να κατανοήσουμε τον τρόπο με τον οποίο ο άνθρωπος και οι ζωντανοί οργανισμοί μαθαίνουν. Αυτό είναι σημαντικό αλλά δεν είναι όμως το κυριότερο. Υπάρχουν πολλά σχεδιαστικά πλεονεκτήματα με την εισαγωγής της έννοιας της μάθησης στα υπολογιστικά συστήματα:

- Μερικά προβλήματα δεν μπορούν να οριστούν σαφώς παρά μόνο με τη χρήση παραδειγμάτων. Μπορούμε να ορίσουμε παραδείγματα της μορφής εισόδου-επιθυμητής εξόδου αλλά όχι σαφείς σχέσεις εισόδου-εξόδου. Θα θέλαμε οι μηχανές να μπορούν να προσαρμόσουν την εσωτερική τους δομή ώστε να παράγουν σωστή έξοδο για μεγάλο αριθμό εισόδων και επομένως να προσεγγίσουν τη σχέση εισόδου-εξόδου που ορίζεται από τα παραδείγματα.

- Σε μεγάλες ποσότητες δεδομένων υπάρχουν πολλές σχέσεις και συσχετίσεις. Μια μηχανή μου μαθαίνει μπορεί να τις εντοπίσει και να τις εξάγει.

- Οι μηχανές που σχεδιάζονται και αναπτύσσονται ρητά για να επιτελούν μια συγκεκριμένη εργασία πολλές φορές αδυνατούν να φέρουν τα επιθυμητά αποτελέσματα, καθώς τα χαρακτηριστικά του περιβάλλοντος στα οποία πρόκειται να λειτουργήσουν δεν είναι απαραίτητα γνωστά κατά τη σχεδίασή της.

- Η ποσότητα της πληροφορίας μπορεί να είναι τόσο μεγάλη σε όγκο που να είναι αδύνατο να κωδικοποιηθεί από έναν άνθρωπο. Μηχανές που μαθαίνουν, σταδιακά μπορούν να συλλάβουν αυτή τη πληροφορία.

- Το περιβάλλον που ενεργούν τα συστήματα αλλάζει. Οι μηχανές που προσαρμόζονται στα νέα περιβάλλοντα ελαχιστοποιούν τον χρόνο επανασχεδίασης ενός συστήματος.

Επομένως υπάρχουν προβλήματα, για τα οποία με προσεγγίσεις μηχανικής μάθησης μπορούν να προκύψουν μοντέλα, που φέρνουν αρκετά ικανοποιητικά αποτελέσματα. Το γεγονός, όμως, ότι τα μοντέλα αυτά απορρέουν από την επεξεργασία ενός συνόλου δεδομένων, καθιστά την απόδοσή τους ευάλωτη σε κακής ποιότητας δεδομένα (garbage in garbage out) αλλά και ανεπαρκή επεξεργασιά. Προκειμένου ένα μοντέλο μηχανικής μάθησης να θεωρηθεί ότι αποδίδει ικανοποιητικά, αλλά και να είναι εφικτή η ανάπτυξη και η χρήση του, θα πρέπει να χαρακτηρίζεται από τις παρακάτω ιδιότητες:

1. Γενίκευση: Γενίκευση είναι η ικανότητα μιας μηχανής να επιτυγχάνει σωστή απόκριση και σε παραδείγματα που είναι αρκετά διαφορετικά από αυτά από τα οποία έχει εκπαιδευτεί.

2. Επεκτασιμότητα: Το σύστημα θα πρέπει να αναπροσαρμόζεται εύκολα σε τυχόν νέα δεδομένα μάθησης και νέες κατηγορίες

3. Χαμηλό υπολογιστικό κόστος: Παρόλο το γεγονός ότι οι υπολογιστικές δυνατότητες των Η/Υ αυξάνονται δραματικά, η πολυπλοκότητα των αλγορίθμων μάθησης πρέπει να είναι χαμηλή. Στις περισσότερες εφαρμογές είναι απαραίτητο οι μηχανές να μαθαίνουν να αποφασίζουν γρήγορα. Έτσι κι αλλιώς η αύξηση των υπολογιστικών δυνατοτήτων αυξάνεται ταυτόχρονα με τον όγκο της διαθέσιμης πληροφορίας με αποτέλεσμα να αντισταθμίζονται.

4. Ανεξαρτησία από το πρόβλημα: Θα πρέπει μία μηχανή να μπορεί να αντεπεξέλθει σε όσο το δυνατόν μεγαλύτερη ποικιλία προβλημάτων χωρίς να χρειάζεται πολύς κόπος επανασχεδίασης και παραμετροποίησής της.

5. Μικρή ευαισθησία στο θόρυβο: Θορυβώδη δεδομένα εκπαίδευσης θα πρέπει να επηρεάζουν κατά το ελάχιστο την ποιότητα της απόκρισης εξόδου.

Οι αλγόριθμοι Μηχανικής Μάθησης χωρίζονται σε 2 γενικές κατηγορίες: τους αλγορίθμους επιβλεπόμενης μάθησης (supervised learning) και μη επιβλεπόμενης μάθησης (unsupervised learning). Οι αλγόριθμοι επιβλεπόμενης μάθησης εκπαιδεύονται πάνω σε δείγματα δεδομένων εισόδου γνωρίζοντας την επιθυμητή έξοδο για κάθε δείγμα. Στη συνέχεια χρησιμοποιούν τη γνώση που απέκτησαν σε καινούρια δεδομένα εισόδου, επιχειρώντας να κάνουν προβλέψεις ή εκτιμήσεις για τις αντίστοιχες εξόδους. Αντίθετα, οι αλγόριθμοι μη επιβλεπόμενης μάθησης δεν εκπαιδεύονται με επιθυμητές εξόδους αλλά προσπαθούν να βρουν τη δομή ή τις σχέσεις μεταξύ των δεδομένων εκπαίδευσης συνήθως χωρίζοντας τα δεδομένα σε ομάδες (clustering). Στη συνέχεια, παίρνοντας ως είσοδο ένα καινούριο δείγμα δεδομένων το τοποθετούν στην κατάλληλη ομάδα.

Κάθε δείγμα δεδομένων το οποίο επεξεργάζονται οι αλγόριθμοι Μηχανικής Μάθησης έχει κάποια χαρακτηριστικά (features). Ένα χαρακτηριστικό είναι μια μετρήσιμη ιδιότητα των δεδομένων η οποία μπορεί να έχει διακριτές ή συνεχείς τιμές. Παραδείγματος χάριν, για ένα σύνολο δεδομένων όπου το κάθε δείγμα αντιπροσωπεύει ένα κτήριο, μπορεί να υπάρχουν τα χαρακτηριστικά ύψος με συνεχείς τιμές και χρώμα με διακριτές τιμές. Τα χαρακτηριστικά των δεδομένων είναι αυτά που δίνουν την απαραίτητη γνώση στους αλγορίθμους Μηχανικής Μάθησης προκειμένου να εκτελέσουν την επιθυμητή εργασία (πρόβλεψη επιθυμητής εξόδου, ομαδοποίηση, κ.λπ.) καθώς μέσω αυτών οποιαδήποτε πληροφορία μετατρέπεται σε αριθμητικές ποσότητες.

Το πεδίο της Μηχανικής Μάθησης έχει εφαρμογή μεταξύ άλλων στην όραση υπολογιστών, επεξεργασία φυσικής γλώσσας, βιοϊατρική, οικονομικά, μετεωρολογία κ.λ.π..

2.4 Προσδιορισμός προβλήματος

Όπως αναλύθηκε παραπάνω γύρω από τα πλοία υφίσταται ένα αρκετά σύνθετο και πολυπαραμετρικό περιβάλλον. Ο όγκος των δεδομένων, η φύση τους, αλλά και η πολυπλοκότητα των συσχετίσεων που τα συνδέουν επιφέρουν αρκετές προκλήσεις κατά την προσπάθεια μοντελοποίησης τους. Από της βασικότερες εξ αυτών είναι η συνεχής προσαρμογή στις αλλαγές του περιβάλλοντος και των συσχετίσεων των παραμέτρων ενός πλοίου. Για παράδειγμα τα βρώμικα ύφαλα, οι στρεβλώσεις της προπέλας,

τα ελαττώματα του πηδαλίου, κ.λ.π. είναι χαρακτηριστικά που προκύπτουν και μεταβάλλονται συνεχώς, επηρεάζοντας την απόδοση ενός πλοίου. Παράλληλα ο αριθμός των πλοίων που συνδέονται στην πλατφόρμα αυξάνεται.

Παρουσιάζεται, έτσι, η ανάγκη δημιουργίας υποδομών οι οποίες θα μπορέσουν να υποστηρίξουν την εύκολη δημιουργία και ενσωμάτωση μοντέλων μηχανικής μάθησης στο σύστημα. Η σχεδίαση αυτών θα πρέπει να επιτρέπει την εύκολη άντληση δεδομένων από το σύστημα και να καθιστά ικανή την συνεχή διαθεσιμότητα των μοντέλων για χρήση από τις υπόλοιπες υπηρεσίες του. Η δυνατότητα προσαρμογής και ανανέωσης των μοντελοποιήσεων που έχουν επιτευχθεί είναι απαραίτητη, όπως και ο ενδελεχής και συστηματικός έλεγχος της απόδοσης των μοντέλων, για την διασφάλιση της ποιότητας των προβλέψεων και εκτιμήσεων. Επιπλέον, υψηλές απαιτήσεις υπάρχουν όσον αφορά την διαχείριση και την συντήρηση του μεγάλου αριθμού μοντέλων μηχανικής μάθησης που θα προκύψει.

Κεφάλαιο 3

Τεχνολογίες

3.1 Java

Η Java [4] είναι μία γενικού σκοπού, υψηλού επιπέδου, αντικειμενοστραφής γλώσσα προγραμματισμού. Ο μεταγλωττισμένος κώδικάς της μπορεί να τρέξει σε οποιαδήποτε πλατφόρμα υποστηρίζει Java χωρίς ανάγκη για τοπική μεταγλώττιση. Αυτό επιτυγχάνεται με την χρήση μιας εικονικής μηχανής (Java Virtual Machine, JVM) σχεδιασμένη γι αυτόν τον σκοπό. Οι εφαρμογές που είναι γραμμένες σε αυτή τη γλώσσα μεταγλωττίζονται μέσω του μεταγλωττιστή της (javac), ο οποίος παράγει έναν αριθμό από αρχεία .class. Στη συνέχεια, η εικονική μηχανή JVM μεταφράζει αυτά τα αρχεία στη γλώσσα μηχανής που υποστηρίζεται από το εκάστοτε λειτουργικό σύστημα και επεξεργαστή.

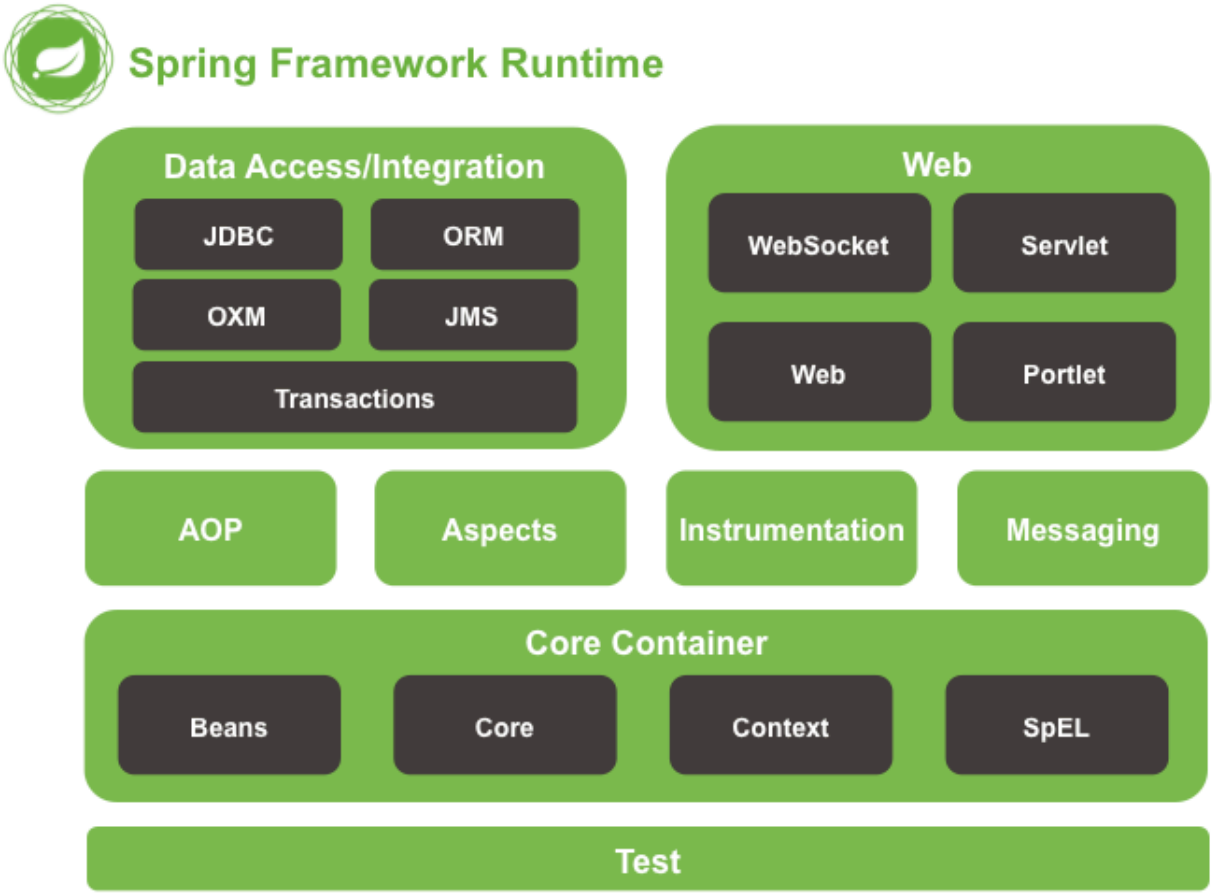
Εδώ και πολλά χρόνια η Java είναι μία από τις δημοφιλέστερες και περισσότερο χρησιμοποιούμενες γλώσσες στον κόσμο, συγκεκριμένα για διαδικτυακές υπηρεσίες εξυπηρετητή-πελάτη, με αναφερόμενη ύπαρξη εννιά εκατομμυρίων προγραμματιστών που την χρησιμοποιούν σήμερα. Αρχικά αναπτύχθηκε από τον James Gosling στην εταιρία Sun Microsystems και εκδόθηκε το 1995. Το συντακτικό της είναι επηρεασμένο από τις C και C++, αλλά διατηρεί λιγότερες λειτουργίες χαμηλού επιπέδου από αυτές. Η δημιουργία της και η ανάπτυξή της έχει διαμορφωθεί πάνω στη βάση πέντε πρωταρχικών απαιτήσεων:

- Πρέπει να είναι «απλή, αντικειμενοστραφής, και γνώριμη»
- Πρέπει να είναι «ευέλικτη και ασφαλής»
- Πρέπει να είναι «ανεξάρτητη αρχιτεκτονικής και μεταφέρσιμη»
- Πρέπει να εκτελείται με «υψηλή απόδοση»
- Πρέπει να είναι «μεταφρασμένη (interpreted), να υποστηρίζει νήματα εκτέλεσης και να είναι δυναμική»

Η Java είναι μία από τις σημαντικότερες γλώσσες της εποχής μας και χρησιμοποιείται ευρέος στη βιομηχανία λογισμικού.

3.1.1 Spring Framework

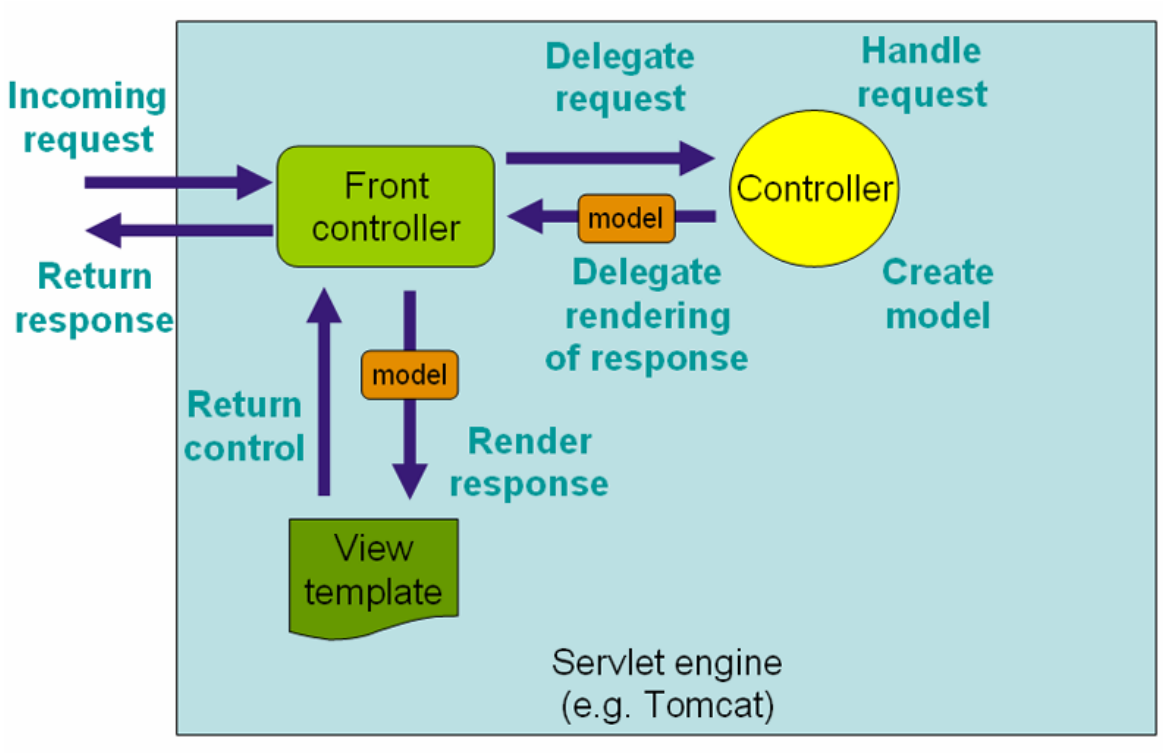
Το Spring Framework είναι ένα δημοφιλές περιβάλλον εργασίας για την ανάπτυξη μεγάλων εφαρμογών σε Java. Χειρίζεται ζητήματα υποδομής των εφαρμογών έτσι ώστε οι χρήστες του να επιστιάζουν στην ανάπτυξη της επιχειρησιακής λογικής της εφαρμογής τους. [5] Το Spring Framework αποτελείται από χαρακτηριστικά που οργανώνονται σε περίπου 20 modules. Αυτά τα modules ομαδοποιούνται στα Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging, and Test όπως φαίνεται στο Σχήμα 3.1. Η χρήση των modules στο σύνολό τους δεν είναι απαραίτητη, το οποίο σημαίνει ότι κάθε εφαρμογή μπορεί να έχει εξαρτήσεις μόνο από ένα υποσύνολο των modules του framework.



Σχήμα 3.1: Επισκόπηση του Spring Framework.

Ιδιαίτερο ενδιαφέρον παρουσιάζουν η δυνατότητες που παρέχει το Spring Framework σχετικά με την πρόσβαση σε δεδομένα, καθώς προσφέρει υλοποιήσεις για τις συνήθεις λειτουργίες πάνω σε βάσεις δεδομένων, αλλά και την ανάπτυξη εφαρμογών διαδικτύου μέσω του Spring MVC Framework.

Το Spring MVC Framework είναι μια υλοποίηση του Model View Controller Design Pattern. Στο επίκεντρό βρίσκεται ένα DispatcherServlet το οποίο είναι υπεύθυνο για το χειρισμό όλων των HTTP αιτήσεων και απόκρισεων. Η ροή εργασιών για την επεξεργασία ενός αιτήματος του DispatcherServlet απεικονίζεται στο Σχήμα 3.2. Πρόκειται για μια έκφραση του προτύπου σχεδίασης ”Front Controller” το οποίο είναι ένα μοτίβο που μοιράζεται το Spring Web MVC με πολλά άλλα κορυφαία web frameworks[5].



Σχήμα 3.2: Η ροή εργασιών για την επεξεργασία αιτήσεων στο Spring Web MVC.

Μετά από μια εισερχόμενη αίτηση HTTP στο DispatcherServlet, η ακολουθία των συμβάντων που λαμβάνουν χώρα για την εξυπηρέτηση του αιτήματος περιγράφεται από τα κάτωθι βήματα:

- Αφού λάβει ένα αίτημα HTTP, το DispatcherServlet ανατρέχει στο HandlerMapping για να καλέσει τον κατάλληλο Controller.
- Ο Controller λαμβάνει το αίτημα και καλεί τις κατάλληλες μεθόδους εξυπηρέτησης, με βάση την HTTP μέθοδο (POST, GET, κτλπ) που χρησιμοποιήθηκε στο αίτημα. Η μέθοδος εξυπηρέτησης με βάση την καθορισμένη επιχειρησιακή λογική θα θέσει δεδομένα σε ένα μοντέλο και θα επιστρέψει το όνομα μιας προβολής στο DispatcherServlet.
- Το DispatcherServlet θα λάβει από το ViewResolver την καθορισμένη προβολή για το αίτημα.
- Μόλις οριστικοποιηθεί η προβολή, το DispatcherServlet μεταβιβάζει τα δεδομένα του μοντέλου στην προβολή, η οποία επιστρέφεται τελικά στην απόκριση του αιτήματος.

Όλα τα προαναφερθέντα στοιχεία (HandlerMapping, Controller και ViewResolver) είναι μέρη του WebApplicationContext το οποίο ανήκει στο Core Container του Spring Framework.

3.2 Python

Η Python [6] είναι μία ευρέως διαδεδομένη αντικειμενοστραφής, διερμηνευόμενη, υψηλού επιπέδου, γενικού σκοπού γλώσσα προγραμματισμού, με δυναμική σημασιολογία. Δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε δημοσίως το 1991. Η φιλοσοφία σχεδίασής της δίνει έμφαση στην αναγνωσιμότητα του κώδικα, και το συντακτικό της επιτρέπει στους προγραμματιστές να

εκφράζονται χρησιμοποιώντας λιγότερες γραμμές κώδικα από για παράδειγμα γλώσσες σαν τη C++ ή την Java. Η γλώσσα παρέχει δυνατότητες για παραγωγή καθαρού και σαφή κώδικα σε μικρή ή μεγάλη κλίμακα. Η Python υποστηρίζει πλήθος προγραμματιστικών μοντέλων, μεταξύ άλλων αντικειμενοστραφή, προστακτικό και συναρτησιακό προγραμματισμό. Χρησιμοποιεί δυναμικό σύστημα τύπων και αυτόματο διαχειριστή μνήμης και έχει μεγάλη ποικιλία βιβλιοθηκών. Οι διερμηνευτές της Python είναι διαθέσιμοι σε πολλά λειτουργικά συστήματα, επιτρέποντας έτσι στον κώδικα Python να τρέχει σε μεγάλη ποικιλία συστημάτων.

3.2.1 Scikit-Learn

Το Scikit-learn [7] είναι μια open-source (διανέμεται με BSD license) βιβλιοθήκη της Python για μηχανική μάθηση βασισμένη στις βιβλιοθήκες NumPy, SciPy, και matplotlib. Παρέχει απλά και αποδοτικά εργαλεία για εξόρυξη και ανάλυση δεδομένων με πολλές υλοποιήσεις αλγορίθμων μηχανικής μάθησης για προβλήματα ταξινόμησης (classification), ομαδοποίησης (clustering) και παλινδρόμησης (regression). Επίσης μέσω των μεθόδων που περιλαμβάνει μπορεί να γίνει εύκολη αξιολόγηση της απόδοσης των αλγορίθμων που δοκιμάζονται αλλά και να πραγματοποιηθεί προ-επεξεργασία των δεδομένων εισόδου στους αλγορίθμους προκειμένου να αυξηθεί η αποδοτικότητα τους.

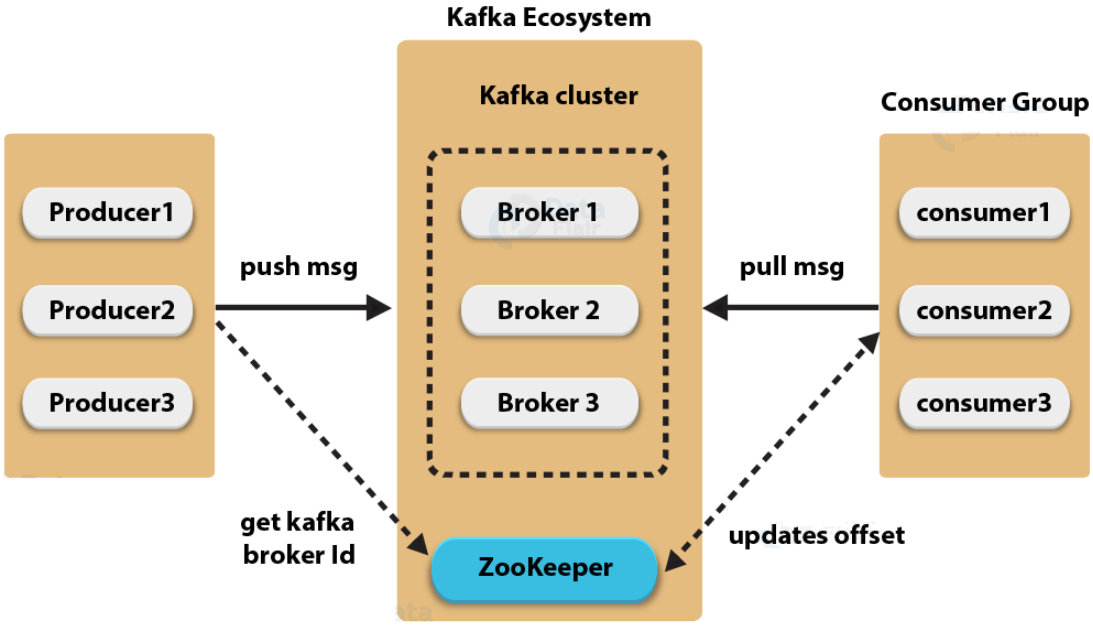
Αποτελεί μία από τις πιο δημοφιλείς βιβλιοθήκες μηχανικής μάθησης με εκατομμύρια χρήστες. Συνδυάζει υψηλή ευχρηστία λόγω της απλής και συνεπούς διεπαφής της, μεγάλη κάλυψη σε αλγορίθμους, συνεχής συντήρηση και αναβάθμιση καθώς και αναλυτικά εγχειρίδια, λόγω της μεγάλης κοινότητας που την υποστηρίζει.

3.2.2 Pandas

Το Pandas αποτελεί μια βιβλιοθήκη της Python πλούσια σε δομές δεδομένων και εργαλεία για επεξεργασία δομημένων σύνολων δεδομένων. Ξεκίνησε να αναπτύσσεται από τον Wes McKinney το 2008, λόγω της ανάγκης για ένα υψηλής απόδοσης, ευέλικτο εργαλείο για την εκτέλεση ποσοτικών αναλύσεων οικονομικών στοιχείων. Έχει ευρεία χρήση στην στατιστική, στα οικονομικά, στις κοινωνικές επιστήμες και πολλούς άλλους τομείς. Η βιβλιοθήκη παρέχει ολοκληρωμένες και ευκατανόητες ρουτίνες για την εκτέλεση κοινών χειρισμών και αναλύσεων πάνω σε τέτοια σύνολα δεδομένων. Σκοπεύει να αποτελέσει θεμελιώδες επίπεδο στο μέλλον της υπολογιστικής στατιστικής στην Python. Χρησιμεύει ως μία ισχυρή επέκταση των υφιστάμενων επιστημονικών πακέτων της Python, ενώ παράλληλα υλοποιεί και βελτιώνει τα είδη των εργαλείων για χειρισμό δεδομένων που βρίσκονται σε άλλες γλώσσες προγραμματισμού για υπολογιστική στατιστική, όπως π.χ. η R. [8]

3.3 Apache Kafka

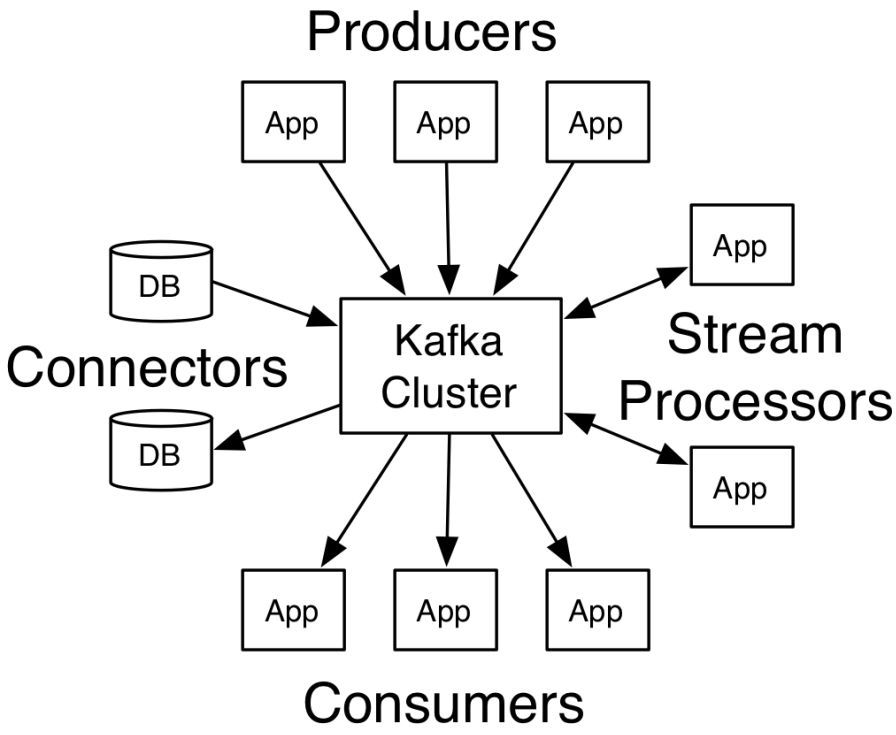
Η πλατφόρμα Apache Kafka παρέχει ένα ενιαίο σύστημα διαχείρισης τροφοδοσίας δεδομένων σε πραγματικό χρόνο και είναι γραμμένη σε Scala και Java. Πετυχαίνει υψηλή απόδοση και χαμηλή καθυστέρηση στη διαβίβαση της πληροφορίας, κάτι που την καθιστά ως ένα από τα πιο χρήσιμα εργαλεία για συστήματα που διαχειρίζονται μεγάλο όγκο μηνυμάτων και συμβάντων ή που κλιμακώνονται γρήγορα.



Σχήμα 3.3: Η αρχιτεκτονική του Apache Kafka.

Η πλατφόρμα έχει κατανεμημένη αρχιτεκτονική και αναπτύσσεται σε δική της συστάδα (cluster) ενός ή πολλών εξυπηρετητών. Κατά την λειτουργία της η πλατφόρμα αποθηκεύει μηνύματα που δέχεται από εφαρμογές οι οποίες καλούνται ”Παραγωγοί” (Producers). Αυτά διατηρούνται μέσα στον cluster του συστήματος σε συγκεκριμένες θέσεις ανάλογα με το θέμα (topic) στο οποίο τα έχει προωθήσει ο παραγωγός και την διαμέριση (partition) που έχει επιλεχτεί. Σε επόμενο στάδιο, εφαρμογές υπό τον ρόλο του ”Καταναλωτή” (Consumer) αναζητούν αυτά τα μηνύματα στις διαμερίσεις του συστήματος. Συγκεκριμένα, κάθε Καταναλωτής μπορεί να εγγραφεί σε συγκεκριμένα θέματα (topics) και να λάβει μηνύματα μόνο από αυτά. Κύριες έννοιες του πυρήνα του συστήματος αποτελούν τα θέματα (topics), οι καταγραφές (records) και οι ”Διαμεσολαβητές” (Brokers). Τα θέματα αποτελούνται από ροές καταγραφών και αναπαριστούν μία κατηγορία τέτοιων, ενώ οι Διαμεσολαβητές είναι υπεύθυνοι για τη σωστή διαχείριση των καταγραφών και για την αντιγραφή τους. Στο Σχήμα 3.3, φαίνεται μια αφαιρετική αναπαράσταση της αρχιτεκτονικής της πλατφόρμας Kafka.

Η πρόσβαση και διαχείριση της πλατφόρμας από τους χρήστες γίνεται μέσω συγκεκριμένων διεπαφών (API) όπως φαίνεται στο Σχήμα 3.4. Τέτοιες είναι η Διεπαφή του Παραγωγού (Producer API) που επιτρέπει την παραγωγή μηνυμάτων σε ένα η παραπάνω θέματα, η Διεπαφή του Καταναλωτή (Consumer API) που επιτρέπει την εγγραφή μιας εφαρμογής σε ένα θέμα και την κατανάλωση της ροής μηνυμάτων που βίσκεται σε αυτό, η Διεπαφή της Ροής Δεδομένων (Stream API) για τη μετατροπή και τροποποίησή τους και η Διεπαφή Διασύνδεση (Connector API) για την διασύνδεση με άλλες εφαρμογές.



Σχήμα 3.4: Αναπαράσταση των διεπαφών του Apache Kafka.

3.4 PostgreSQL

Η PostgreSQL αποτελεί μία σχεσιακή βάση δεδομένων ανοικτού κώδικα. Συντηρείται από την ομάδα ανάπτυξης PostgreSQL, ενώ βελτιώνεται και τροποποιείται από τους υπεύθυνους ανάπτυξης λογισμικού σε όλο τον κόσμο. Προέρχεται από το πακέτο της POSTGRES που αναπτύχθηκε στο πανεπιστήμιο Καλιφόρνιας, Berkeley [9]. Η ανάπτυξη της ήδη διαρκεί πάνω από 20 χρόνια και βασίζεται σε μια αποδεδειγμένα καλή αρχιτεκτονική η οποία έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία, την ακεραιότητα δεδομένων και την ορθή λειτουργία.

Η PostgreSQL είναι ένα αντικείμενο-σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (ORDBMS) [10]. Εισήγαγε πολλές ιδέες και χαρακτηριστικά τα οποία έγιναν διαθέσιμα σε αντίστοιχα εμπορικά συστήματα πολύ αργότερα. Υποστηρίζει ένα μεγάλο μέρος του προτύπου SQL και προσφέρει πολλά σύγχρονα χαρακτηριστικά όπως:

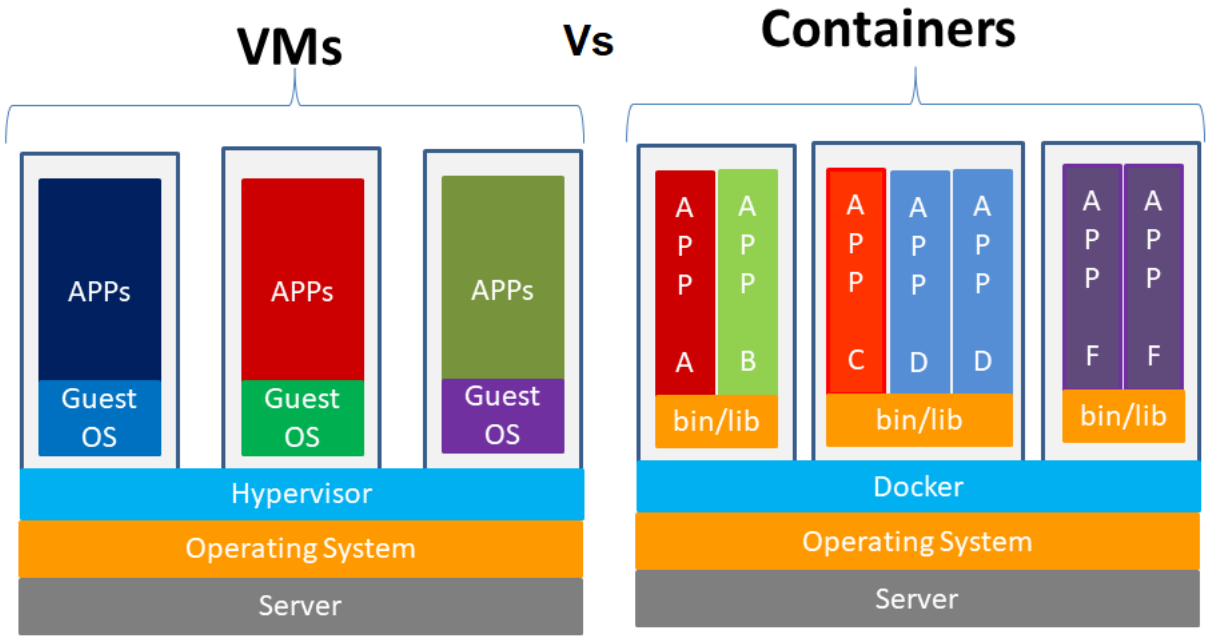
- Εκτέλεση σύνθετων ερωτημάτων.
- Υποστήριξη ξένων κλειδιών.
- Υποστήριξη σκανδαλιστών (triggers).
- Υλοποίηση όψεων (views).
- Ακεραιότητα δοσοληψιών (transactional integrity).

Επίσης, η PostgreSQL μπορεί να διευρυνθεί και από τους χρήστες της. Για παράδειγμα μπορεί κανείς να ορίσει τύπους δεδομένων, συναρτήσεις, τελεστές, συναθροιστικές συναρτήσεις, μεθόδους, κ.α.

Η PostgreSQL βασίζεται στο μοντέλο πελάτη/εξυπηρετητή ενώ μπορεί να διαχειριστεί μεγάλο αριθμό ταυτόχρονων χρηστών καθώς και μεγάλο όγκο δεδομένων. Διαθέτει περιβάλλοντα προγραμματισμού για C, C++, Java, Perl, Python, Ruby και λειτουργεί σε όλα τα λειτουργικά συστήματα Linux, Unix, MacOS και Windows. Έχει σχεδιαστεί για να ικανοποιήσει τις ανάγκες των χρηστών, από βασική χρήση SQL μέχρι την ανάπτυξη πολύπλοκων βάσεων δεδομένων.

3.5 Docker

Το Docker είναι μία πλατφόρμα λογισμικού ανοικτού κώδικα για την υλοποίηση Εικονοποίησης (Virtualization) σε επίπεδο Λειτουργικού Συστήματος. Προσφέρει αυτοματοποιημένες διαδικασίες για την ανάπτυξη εφαρμογών σε απομονωμένες Περιοχές Χρήστη (User Spaces) που ονομάζονται Software Containers. Το λογισμικό χρησιμοποιεί τεχνολογίες του πυρήνα Linux όπως τα cgroups και οι χώροι ονομάτων πυρήνα (kernel namespaces), για να επιτρέπει σε ανεξάρτητα και απομονωμένα software containers να εκτελούνται στο ίδιο λειτουργικό σύστημα [11]. Έτσι αποφεύγεται η χρήση επιπλέον υπολογιστικών πόρων που θα απαιτούσε μια εικονική μηχανή (virtual machine) για την διαδικασία προσομοίωσης ενός Guest Λειτουργικού Συστήματος. Αυτό συμβαίνει επειδή τα Containers χρησιμοποιούν τον πηρύνα του Host Λειτουργικού Συστήματος (βλ. Σχήμα 3.5). Η δημιουργία ενός τέτοιου container μπορεί να γίνει με βάση κάποιο Docker Image. Ένα Image είναι ένα σύνολο immutable πακεταρισμένων αρχείων στα οποία είναι αποθηκευμένο ένα στιγμιότυπο κάποιου περιβάλλοντος. Έτσι, οι εφαρμογές πακετάρονται μαζί με όλα τα συστατικά στοιχεία του περιβάλλοντος στο οποίο θα τρέξουν σε ένα Docker Image. Στην συνέχεια, με βάση αυτό, το Docker αναλαμβάνει το χτίσιμο του περιβάλλοντος και την εκτέλεση της εφαρμογής.



Σχήμα 3.5: Σύγκρηση μεταξύ Εικονικών Μηχανών και Containers.

Το Docker προσφέρει πολλά πλεονεκτήματα όπως η αποδοτικότερη διαχείρισή των εφαρμογών μέσω των μεμονωμένων containers. Λόγω της αρχιτεκτονικής τους τα containers προσφέρουν καλύτερη υπολογιστική πυκνότητα μεταξύ των εφαρμογών. Ευνοεί την δημιουργία ευέλικτων αγωγών

παράδοσης λογισμικού για ταχύτερη και ασφαλέστερη κυκλοφορία νέων λειτουργιών. Σε αντίθεση με τα Virtual Machines, τα container δεν συνθέτουν ένα πλήρες λειτουργικό σύστημα, αλλά προσφέρουν μία προβολή του λειτουργικού συστήματος που εκτελείται ήδη [11]. Αυτό τα καθιστά αποτελεσματικά, ελαφριά, αυτοδύναμα συστήματα και εγγυάται ότι το λογισμικό θα τρέχει πάντα το ίδιο, ανεξάρτητα από το πού είναι αναπτυγμένο. Το Docker αυτοματοποιεί τις επαναλαμβανόμενες ενέργειες για την δημιουργία και την διαμόρφωση του περιβάλλοντος ανάπτυξης. Όταν μια εφαρμογή είναι αποκεντρωμένη, η πολυπλοκότητα αυτή ωθείται στα containers που έχουν εύκολη ανάπτυξη, φορητότητα και λειτουργούν ανεξάρτητα. Αυτό αποδεσμεύει τους προγραμματιστές που χρησιμοποιούν το Docker από την ευθύνη της σύνθεσης του περιβάλλοντος ανάπτυξης, συμβάλλοντας έτσι και στην αύξηση της αποδοτικότητας τους.

3.6 Kubernetes

Το Kubernetes είναι μια φορητή, επεκτάσιμη πλατφόρμα ανοιχτού κώδικα για τη διαχείριση containerized υπηρεσιών και φόρτου εργασίας. Σχεδιάστηκε αρχικά από την Google και συντηρείται από το Cloud Native Computing Foundation. Διαχειρίζεται και οργανώνει την υπολογιστική, δικτυακή και αποθηκευτική υποδομή εκ μέρους των χρηστών του, παρέχοντας βασικούς μηχανισμούς για την αυτόματη ανάπτυξη, συντήρηση και κλιμάκωση εφαρμογών. [12] Παράλληλα, καθιστά δυνατή τη φορητότητα μεταξύ παροχέων υπηρεσιών υποδομής.

Ο Kubernetes αντιπροσωπεύει την κατάσταση ενός συστήματος χρησιμοποιώντας αντικείμενα. Για παράδειγμα το αντικείμενο του Kubernetes το οποίο υποστηρίζει την λειτουργία των Containers ονομάζεται Pod. Ένα Kubernetes αντικείμενο ορίζεται από ένα κομμάτι κώδικα γραμμένο στη γλώσσα YAML και αποτελείται από δύο μέρη:

- Το κομμάτι Spec, το οποίο συντάσει ο χρήστης και αντιπροσωπεύει την επιθυμητή κατάσταση του συστήματος. Το Kubernetes αποθηκεύει αυτό το αντικείμενο και προσπαθεί συνεχώς ώστε να εκπληρώσει την οδηγία του χρήστη.

- Το κομμάτι Status, το οποίο ανανεώνει ο Kubernetes και εκφράζει την πραγματική κατάσταση του συστήματος. Στο παράδειγμα του Pod, το πεδίο Status θα περιέχει πληροφορία σχετικά με το αν το Pod είναι σε λοειτουργία, αν έχει δρομολογηθεί, κ.λ.π..

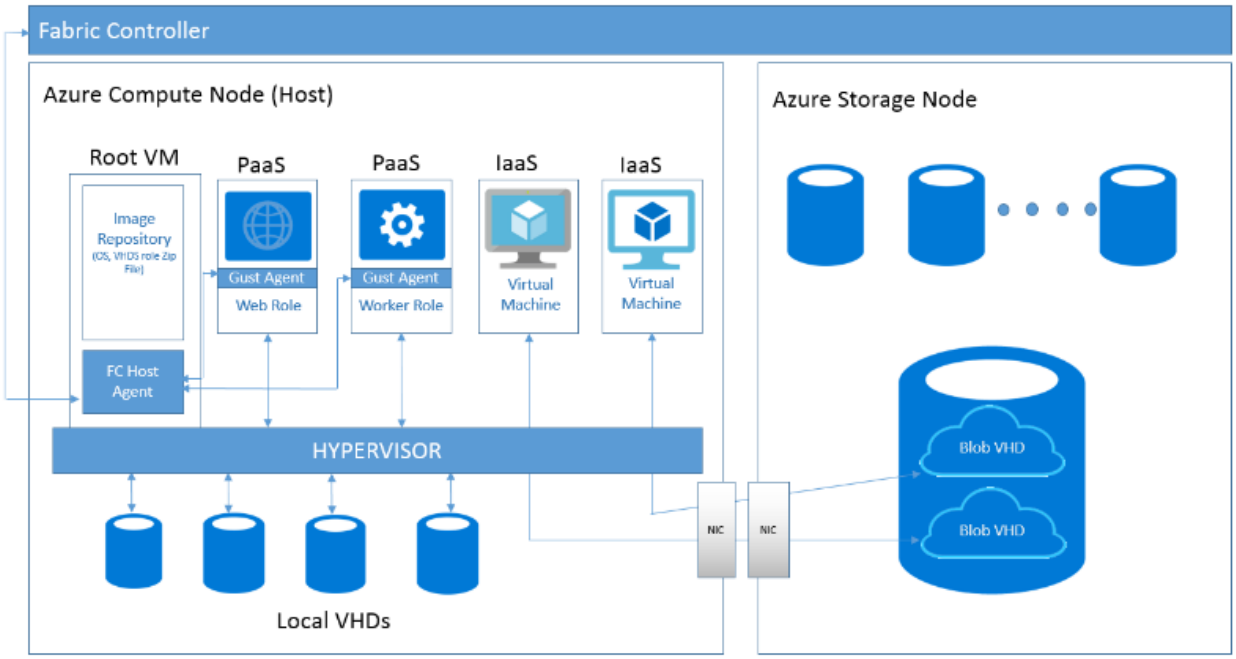
Ο χρήστης, δηλαδή, δηλώνει την επιθυμητή κατάσταση του αντικειμένου (Spec) και το Kubernetes προσπαθεί να την πραγματοποιήσει. Η λειτουργικότητα αυτή του Kubernetes υλοποιείται από κάποιες διεργασίες οι οποίες ονομάζονται Controllers.

3.7 Microsoft Azure

Το Microsoft Azure είναι μια υπηρεσία παροχής υπολογιστικού νέφους που έχει δημιουργήσει η Microsoft και επιτρέπει στους χρήστες της να αναπτύσσουν, να τεστάρουν και να διαχειρίζονται εφαρμογές και υπηρεσίες μέσω των data centers της Microsoft. Κυκλοφόρησε στις 1 Φεβρουαρίου του 2010 ως ”Windows Azure” και στην συνέχεια μετονομάστηκε σε Microsoft Azure. Προσφέρει

”Λογισμικό ως Υπηρεσία” (SaaS), ”Πλατφόρμα ως Υπηρεσία” (PaaS) και ”Υποδομή ως Υπηρεσία” (IaaS) υποστηρίζοντας παράλληλα πληθώρα γλωσσών προγραμματισμού, εργαλείων και δομών. Η Microsoft απαριθμεί πάνω από 600 υπηρεσίες Azure [13] διαφόρων ειδών σχετικά με διαχείρισή υποδομών και πόρων, αποθήκευση δεδομένων, ανάπτυξη εφαρμογών, ασφάλεια, IoT, τεχνητή νοημοσύνη κ.λ.π..

Χρησιμοποιεί ένα ειδικό λειτουργικό σύστημα με το όνομα Microsoft Azure το οποίο αποτελεί μία τροποποιημένη έκδοση του λειτουργικού συστήματος Windows Server 2008. Ως πυρήνας του λειτουργεί το Azure Fabric Controller. Το Azure Fabric Controller παρέχει, αποθηκεύει, διανέμει, παρακολουθεί και διαχειρίζεται τις εικονικές μηχανές (VMs) αλλά και τους φυσικούς διακομιστές που απαρτίζουν το Azure. Αυτοματοποιεί σχεδόν ολοκληρωτικά την εσωτερική λειτουργικότητα της πλατφόρμας, ακόμα και την εγκατάσταση νέου υλικού. Παρέχει τις δυνατότητες για κλιμάκωση και αξιοπιστία των υπηρεσιών που λειτουργούν εντός του Azure, διασφαλίζοντας ότι οι υπηρεσίες και το περιβάλλον δεν αποτυγχάνουν σε περίπτωση αποτυχίας ενός ή περισσοτέρων διακομιστών στα data center της Microsoft.



Σχήμα 3.6: Αναπαράσταση της αρχιτεκτονικής του Microsoft Azure.

Κεφάλαιο 4

Ανάλυση - Σχεδίαση

4.1 Απαιτήσεις πλατφόρμας

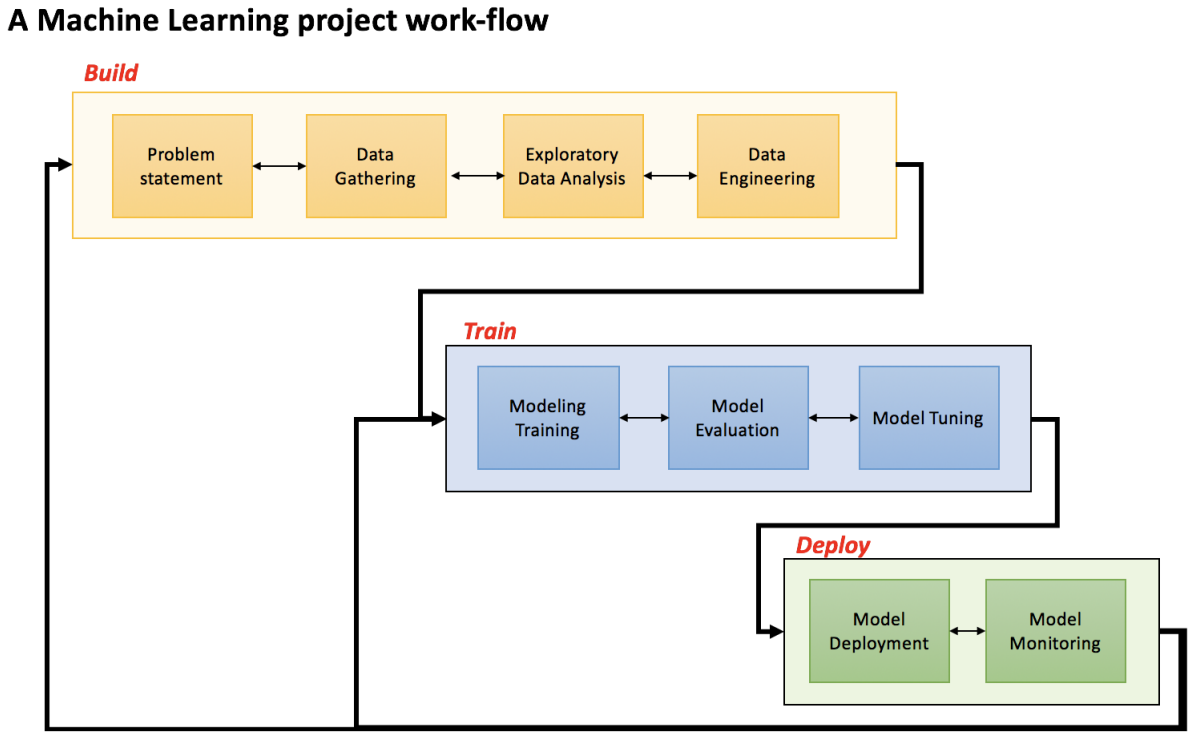
Η τυπική ροή εργασίας ενός έργου Μηχανικής Μάθησης περιλαμβάνει όπως φαίνεται και στο Σχήμα 4.1 τα παρακάτω στάδια:

- Ανάπτυξη: Εδώ ορίζεται το πρόβλημα, συλλέγονται τα δεδομένα, γίνεται ανάλυση και στη συνέχεια καθαρίζονται και μετασχηματίζονται τα δεδομένα στην επιθυμητή μορφή.
- Εκπαίδευση: Χρησιμοποιείται ένας ή περισσότεροι αλγόριθμοι Μηχανικής Μάθησης και εκπαιδεύονται επαναληπτικά διάφορα μοντέλα. Με βάση τις επιδόσεις του κάθε μοντέλου θα γίνει επιλογή μεταξύ των αλγορίθμων, όπως επίσης και των παραμέτρων του (hyperparameter tuning).
- Ενσωμάτωση: Προκείμενου το μοντέλο να χρησιμοποιηθεί, θα πρέπει να ενσωματωθεί σε κάποιο μεγαλύτερο σύστημα του οποίου τις υπηρεσίες θα εξυπηρετεί.

Η όλη διαδικασία είναι ιδιαίτερα επαναληπτική και οποιαδήποτε αλλαγή μπορεί να οδηγήσει στην επιστροφή της διαδικασίας σε οποιοδήποτε προηγούμενο στάδιο. Καθώς τα προβλήματα που προσεγγίζονται με μηχανική μάθηση αλλά και τα δεδομένα που χρησιμοποιούνται πληθαίνουν, παρουσιάζεται έντονα η ανάγκη ύπαρξης κάποιας αυτοματοποίησης της διαδικασίας.

Στο πλαίσιο του Συστήματος, από την διασυνδεσιμότητα με το οποίο θα λαμβάνονται τα δεδομένα, η Πλατφόρμα που θα αναπτυχθεί θα πρέπει να υποστηρίζει τις ακόλουθες λειτουργίες:

- Δυνατότητα ενσωμάτωσης με τις cloud υπηρεσίες, για την εισροή δεδομένων.
- Συνεχής διαθεσιμότητα των μοντέλων προς χρήση.
- Δυνατότητα εισαγωγής πολλαπλών και διαφορετικών μοντέλων.
- Δυνατότητα επιθεώρησης των μοντέλων και αξιολόγησης της επίδοσής τους.
- Δυνατότητα περιοδικής επανεκπαίδευση και αξιολόγηση των μοντέλων.
- Δυνατότητα επανεκπαίδευσης υπό συνθήκες.



Σχήμα 4.1: Τυπική ροή εργασίας ενός έργου Μηχανικής Μάθησης.

- Διαχείριση του υπολογιστικού φόρτου εργασίας.
- Μόνιμη αποθήκευση της ”εκπαιδευμένης” κατάστασης των μοντέλων.

4.2 Χρήση της πλατφόρμας

Αρχικά αφού οριστεί κάποιο πρόβλημα, κριθεί, δηλαδή, απαραίτητο να πραγματοποιούνται προβλέψεις για κάποια ποσότητα ενός πλοίου είτε από συγκεκριμένη είσοδο είτε από κάποια που έχει προκύψει έπειτα από σχετική έρευνα, θα πρέπει να δηλωθούν αυτά τα στοιχεία στο σύστημα. Επιπλέον θα δηλωθεί ένα όνομα για το μοντέλο καθώς και με ποια υπηρεσία μηχανικής μάθησης (MLService) θα αναπτύσσεται. Τα MLService αφορούν τον διαφορετικό τρόπο με τον οποίο κάποιο μοντέλο θα εκπαιδεύεται και θα χρησιμοποιείται. Η διαφορά μπορεί να εντοπίζεται στην γλώσσα προγραμματισμού που θα χρησιμοποιηθεί στον κώδικα του μοντέλου, στο περιβάλλον στο οποίο θα γίνει η εκπαίδευσή του κ.λ.π.. Με αυτόν τον τρόπο θα καταχωρηθεί μία νέα οντότητα ενός μοντέλου στην βάση δεδομένων της πλατφόρμας, χωρίς ακόμα να είναι ολοκληρωμένο αλλά αρχικοποιημένο υπό μία έννοια. Αυτή η ενέργεια πραγματοποιείται με την συμπλήρωση μίας φόρμας, η οποία είναι διαθέσιμη στην Γραφική Διεπαφή Χρήστη στην αντίστοιχη ”καρτέλα” κάποιου πλοίου.

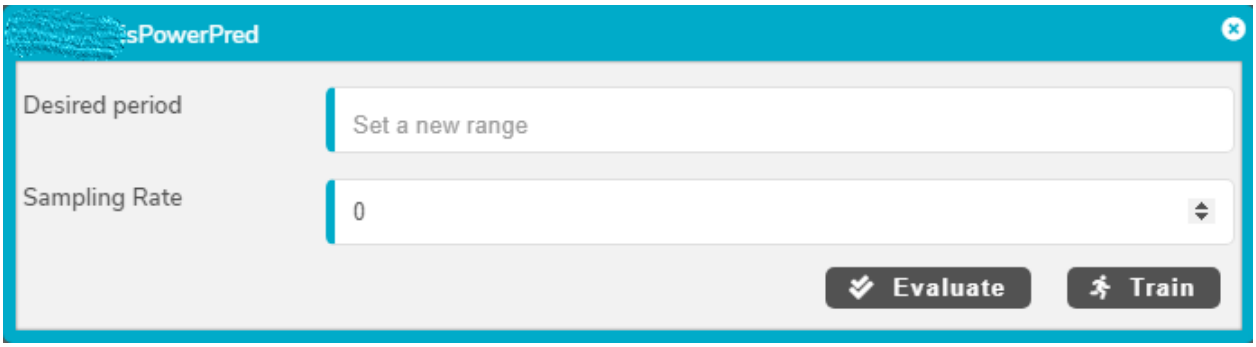
Όπως φαίνεται και στο Σχήμα 4.2 μπορεί να επιλεχθούν διαφορετικά σήματα για την είσοδο της εκπαίδευσης και για την είσοδο της πρόβλεψης. Η εισαγωγή σημάτων εισόδου για την πρόβλεψη είναι προαιρετική καθώς αν αφεθεί κενή θα χρησιμοποιηθούν τα σήματα που δηλώθηκαν στην είσοδο εκπαίδευσης. Ο λόγος που συμβαίνει αυτό είναι για να υπάρχει ευελιξία σε σχέση με τον τρόπο που θα γίνει η εκπαίδευση. Για παράδειγμα μπορεί στην εκπαίδευση του μοντέλου να χρησιμοποιηθούν κάποια επιπλέον σήματα για το φιλτράρισμα κάποιον γραμμών του συνόλου δεδομένων που θα χρη-

Σχήμα 4.2: Φόρμα εισαγωγής νέου μοντέλου στην πλατφόρμα.

σιμοποιηθεί, ώστε να προκύψουν καλύτερα αποτελέσματα. Αυτά τα σήματα δεν θα χρειάζεται να υπάρχουν στην είσοδο για την πρόβλεψη.

Προκειμένου να καταστεί το μοντέλο έτοιμο για εκπαίδευση θα πρέπει να παρασχεθούν στο σύστημα όλες οι υλοποιήσεις των διαδικασιών που θα ακολουθηθούν για την εκπαίδευση, την πρόβλεψη και την αξιολόγηση του μοντέλου. Αυτό πραγματοποιείται μέσω της γλώσσας python και πιο συγκεκριμένα μέσω μίας κλάσης που ορίζεται σε ένα python module. Απαιτείται λοιπόν από τον χρήστη η συγγραφή ενός αρχείου .py στο οποίο θα ορίζεται μία κλάση, η οποία θα πληρεί τις προδιαγραφές ενός συγκεκριμένου interface. Ένα σύνολο τριών μεθόδων, δηλαδή, με συγκεκριμένο όνομα, οι οποίες θα δέχονται ως παραμέτρους, αλλά και θα επιστρέφουν συγκεκριμένες δομές. Αυτές οι μέθοδοι θα υλοποιούν τον επιθυμητό τρόπο εκπαίδευσης, πρόβλεψης και αξιολόγησης του μοντέλου. Το αρχείο αυτό θα αποθηκεύεται σε συγκεκριμένη διεύθυνση του file system στον cluster που φιλοξενεί την πλατφόρμα.

Εφόσον όλα τα παραπάνω έχουν πραγματοποιηθεί, υπάρχει η δυνατότητα μέσω της διεπαφής να αιτηθεί η διαδικασία εκπαίδευσης του μοντέλου. Για την εκτέλεση της εκπαίδευσης θα πρέπει να οριστεί από τον χρήστη μία χρονική περίοδος από την οποία θα αντληθούν τα δεδομένα καθώς και ο επιθυμητός ρυθμός δειγματοληψίας όπως φαίνεται στο Σχήμα 4.3. Σημειώνεται εδώ ότι τα δεδομένα έχουν υποστεί ήδη μία προ-επεξεργασία από το σύστημα συλλογής δεδομένων, το οποίο σημαίνει ότι η πλατφόρμα έχει πρόσβαση σε καθαρισμένα, φιλτραρισμένα και συγχρονισμένα δεδομένα.



Σχήμα 4.3: Αποστολή αιτήματος για εκπαίδευση ή αξιολόγηση ενός μοντέλου.

Σε συνέχεια της εκπαίδευσής του μοντέλου, με βάση τους δείκτες που προκύπτουν μαζί με την εκπαίδευση του ή με περαιτέρω αξιολόγησή του μπορεί να κριθεί και να μαρκαριστεί ως έτοιμο για χρήση. Καθώς υπάρχει διαθέσιμο στο σύστημα ένα μοντέλο μαρκαρισμένο ως έτοιμο, παρέχονται από την πλατφόρμα οι παρακάτω τρόποι χρήσης και αξιοποίησης του:

- Πρόβλεψη σε δοθείσα είσοδο. Ένα ή περισσότερα διανύσματα εισόδου παρέχονται στην πλατφόρμα η οποία τα τροφοδοτεί στο μοντέλο και διεξάγει τις αντίστοιχες προβλέψεις.

  Αυτό το σενάριο χρήσης παρέχει την δυνατότητα εξέτασης της συμπεριφοράς του πλοίου σε συνθήκες που εν δυνάμει μπορεί να εκτεθεί. Δυνατότητα με σημαντικό αντίκτυπο στον προγραμματισμό και την λήψη αποφάσεων σχετικά με ταξίδια του πλοίου.

- Εκτίμηση παρελθοντικών καταστάσεων. Μετά από αίτημα για την εκτίμηση της ποσότητας πρόβλεψης ενός μοντέλου για κάποια χρονική περίοδο ή κάποια χρονικά σημεία στο παρελθόν, η πλατφόρμα συλλέγει τα απαραίτητα δεδομένα από την υπηρεσία Cloud και τα τροφοδοτεί στο μοντέλο ώστε να διεξάγει τις αντίστοιχες προβλέψεις.

  Με αυτόν τον τρόπο παρέχεται μία εκτίμηση της συμπεριφοράς που είχε το πλοίο (για την εκάστοτε παράμετρο του) σύμφωνα με την μοντελοποίηση που έχει πραγματοποιηθεί. Αυτή η εκτίμηση της αναμενόμενης συμπεριφοράς του πλοίου μπορεί να συγκριθεί με τις πραγματικές μετρήσεις, ώστε να διεξαχθούν συμπεράσματα για την ομαλή λειτουργία των οργάνων του πλοίου. Επίσης μπορεί να παρασχεθεί κάλυψη σε πιθανή αδυναμία άντλησης πραγματικών δεδομένων από μετρήσεις.

Παράλληλα, δυνατή είναι και η αξιολόγηση του μοντέλου πάνω σε δεδομένα χρονικών περιόδων εκτός αυτής στην οποία εκπαιδεύτηκε. Όπως φαίνεται και στο Σχήμα 4.3 διαλέγοντας μία χρονική περίοδο και έναν ρυθμό δειγματοληψίας υπάρχει και η επιλογή για την αξιολόγηση του μοντέλου με δεδομένα αυτής της περιόδου. Από την αξιολόγηση του μοντέλου θα εκτιμηθούν οι εξής δείκτες για την ακρίβεια του μοντέλου.

- Explained Variance Score
- Mean Absolute Error
- Mean Squared Error
- Mean Squared Log Error
- Median Absolute Error
- R2 Score
- Root Mean Squared Error

Προκειμένου να εξασφαλιστεί ότι τα μοντέλα της πλατφόρμας δεν θα απαρχαιώνονται με το πέρασμα του χρόνου, υποστηρίζεται η δυνατότητα να οριστεί μία χρονική περίοδος με βάση την οποία θα γίνονται αυτόματα επανεκπαιδεύσεις του εκάστοτε μοντέλου. Έτσι τα μοντέλα στην πλατφόρμα ενημερώνονται και προσαρμόζονται στις μεταβολές της κατάστασης του πλοίου αυτόματα χωρίς την μεσολάβηση κάποιου χρήστη. Όταν πραγματοποιηθεί μία επανεκπαίδευση, η προηγούμενη κατάσταση του μοντέλου αρχειοθετείται. Σχηματίζεται έτσι ένα αρχείο για κάθε μοντέλο με παλιότερες καταστάσεις στις οποίες είχε βρεθεί. Υποστηρίζεται η δυνατότητα, αν για κάποιο λόγο κριθεί αναγκαίο, κάποιο μοντέλο να επιστρέψει σε κάποια παλιότερη κατάσταση που είχε. Ταυτόχρονα μπορούν να εξαχθούν μέσω της πλατφόρμας αναφορές σχετικά με την μεταβολή της κατάστασης του πλοίου

όπως αυτή απορρέει μέσω των μοντέλων από τις διάφορες αρχειοθετημένες καταστάσεις στις οποίες είχε βρεθεί.

Με αντίστοιχο τρόπο μπορούν να οριστούν και περιοδικές αξιολογήσεις των μοντέλων, με σκοπό την παρακολούθηση της απόδοσής τους. Αν η απόδοση ενός μοντέλου εμφανιστεί χειρότερη από την προηγούμενη κατά τρόπο στον οποίο το Mean Squared Error αποκλίνει από αυτό της προηγούμενης αξιολόγησης κατά 10%, τότε η πλατφόρμα θα προωθήσει ένα event στο Σύστημα για να ενημερώσει τους άμεσα ενδιαφερόμενους για το γεγονός αυτό.

4.3 Δομές Δεδομένων

4.3.1 Quantity Predictor

Από τις βασικότερες ανάγκες της πλατφόρμας αποτελεί η αναπαράσταση πληροφοριών για τα μοντέλα πέραν των ίδιων. Η διαθεσιμότητά τέτοιων πληροφοριών θα εξυπηρετήσει σε τέτοιο βαθμό την χρήση τους που σε διαφορετική περίπτωση θα ήταν σχεδόν αδύνατη. Παρουσιάζεται, επομένως, η ανάγκη της δόμησης των πληροφοριών σχετικά με τα μοντέλα έτσι ώστε να είναι εύκολη και ευέλικτη η αναπαράστασή τους.

Η βασική δομή ”κάτω” από την οποία συγκεντρώνεται όλη η πληροφορία σχετικά με την πρόβλεψη κάποιας συγκεκριμένης ποσότητας ενός πλοίου αποτελεί ο Quantity Predictor. Κατά συνέπεια ένας Quantity Predictor αφορά συγκεκριμένο πλοίο και συγκεκριμένη ποσότητα πρόβλεψης, με βασικό πεδίο ένα σύνολο δομών σχετικά με μοντέλα σχεδιασμένα για την πρόβλεψη της συγκεκριμένης ποσότητας. Ο βασικός σκοπός της δομής του Quantity Predictor είναι η ομαδοποίηση των μοντέλων που διεξάγουν προβλέψεις για κοινό ζεύγος ποσότητας και πλοίου.

Όπως αναφέρθηκε, έχει οριστεί μία δομή που αντιπροσωπεύει τα μοντέλα με όνομα Model και βρίσκεται κάτω από έναν Quantity Predictor. Στη δομή αυτή περιέχεται όλη η πληροφορία που σχετίζεται με κάποιο μοντέλο. Τα βασικά πεδία της δομής αυτής είναι:

- Model Name: Το όνομα του μοντέλου
- ML Service: Αναγνωριστικό του τρόπου με τον οποίο αναπτύχθηκε το μοντέλο.
- Model Path: Η διεύθυνση στην οποία βρίσκεται το αρχείο του μοντέλου.
- Usable: Ένα flag που δηλώνει αν η κατάσταση του μοντέλου είναι τέτοια, ώστε να μπορεί να χρησιμοποιηθεί.
- Retrain Interval: Η χρονική περίοδος με την οποία θα εκτελούνται επανεκπαιδεύσεις για το μοντέλο.
- Evaluation Interval: Η χρονική περίοδος με την οποία θα εκτελούνται αξιολογήσεις για το μοντέλο.
- Το Retrain: Ένα flag με το οποίο ενεργοποιούνται η περιοδικές εκπαιδεύσεις του μοντέλου.
- Το Reevaluate: Ένα flag με το οποίο ενεργοποιούνται η περιοδικές αξιολογήσεις του μοντέλου.

- Active Version: Το αναγνωριστικό της έκδοσης του μοντέλου που έχει οριστεί ως ενεργή. Δηλαδή αυτή που θα χρησιμοποιηθεί.

Προκειμένου να παραχθούν προβλέψεις από τα μοντέλα θα πρέπει αυτά να τροφοδοτηθούν με κάποια συγκεκριμένη είσοδο. Είσοδο για τα μοντέλα αποτελούν δεδομένα από συγκεκριμένες ποσότητες οι οποίες έχουν οριστεί από πριν και τα μοντέλα έχουν εκπαιδευτεί με βάση αυτές. Όπως έχει αναφερθεί υπάρχουν περιπτώσεις στις οποίες εξυπηρετεί το σύνολο των ποσοτήτων εισόδου για την εκπαίδευση ενός μοντέλου να είναι διαφορετικό από αυτό για την διεξαγωγή της πρόβλεψη. Έτσι προκύπτουν δύο πανομοιότυπες δομές οι οποίες αντιπροσωπεύουν μία ποσότητα. Για το σύνολο των ποσοτήτων που θα χρησιμοποιηθεί ως είσοδος εκπαίδευσης έχει οριστεί η δομή Model Feature και για το σύνολο των προβλέψεων. η δομή Prediction Input.

Κάθε φορά που εκπαιδεύεται ένα μοντέλο, αυτό αποκτά μία κατάσταση η οποία αντιπροσωπεύει την κατάσταση του πλοίου την δεδομένη χρονική περίοδο. Καθώς όμως πραγματοποιούνται εκ νέου εκπαιδεύσεις ώστε τα μοντέλα να ακολουθούν τις μεταβολές στην κατάσταση των πλοίων, είναι σκόπιμο οι μοντελοποιήσεις των παλιότερων καταστάσεων να αρχειοθετούνται αντί να διαγράφονται. Η δομή για να αναπαριστά την εκάστοτε μοντελοποίηση της κατάστασης ενός πλοίου ονομάζεται Model Version. Σημειώνεται ότι κάθε Model Version δεν αφορά διαφορετικό μοντέλο αλλά μία διαφορετική έκδοσή του, καθώς το μόνο που διαφέρει είναι τα δεδομένα με τα οποία έχει εκπαιδευτεί. Νεότερα δεδομένα αντιστοιχούν σε μοντελοποιήσεις νεότερων καταστάσεων του πλοίου. Τα πεδία που περιέχονται σε ένα Model Version είναι οι χρονικές σημάνσεις της αρχής και του τέλους της περιόδου από την οποία θα συλλεχθούν τα δεδομένα καθώς και ο ρυθμός δειγματοληψίας αυτών και ο αριθμός των στιγμιοτύπων τους.

Τελευταίο σύνολο πληροφοριών που σχετίζεται με ένα μοντέλο αποτελεί μία αξιολόγηση που έχει διεξαχθεί γι αυτό. Πιο συγκεκριμένα, καθώς οι αξιολογήσεις πραγματοποιούνται για συγκεκριμένες εκδόσεις των μοντέλων αυτές συνδέονται με κάποια έκδοση του μοντέλου και όχι γενικά με αυτό. Στοιχεία αυτής τις δομής αποτελούν η περίοδος με δεδομένα από την οποία έχει διεξαχθεί η αξιολόγηση, ο αριθμός της έκδοσης του μοντέλου που αφορά και ένα σύνολο από μετρικές που περιέχουν πληροφορία για την ακρίβειά του και για το μέγεθος του σφάλματός των προβλέψεων. που παράγει. Όπως αναφέρθηκε και παραπάνω το σύνολο των μετρικών που συνθέτουν μία αξιολόγηση ενός μοντέλου είναι:

- Explained Variance Score
- Mean Absolute Error
- Mean Squared Error
- Mean Squared Log Error
- Median Absolute Error
- R2 Score
- Root Mean Squared Error

4.3.2 Task

Για την περιγραφή εργασιών Μηχανικής Μάθησης έχει οριστεί η δομή Task. Όπως θα αναλυθεί και στην συνέχεια, η ανάγκη για την περιγραφή μίας εργασίας απορρέει από το γεγονός ότι αυτή δεν εκτελείται από την εφαρμογή που λαμβάνει το αίτημα αλλά προωθείται σε άλλες εφαρμογές για εκτέλεση.

Υπάρχουν 6 είδη Task τα οποία αντιπροσωπεύουν εργασίες σχετικές με την Μηχανική Μάθηση:

1. Train: Εκπαίδευση κάποιου μοντέλου.
2. Evaluate: Αξιολόγηση μοντέλου.
3. Predict Period: Πρόβλεψη πάνω σε κάποια χρονική περίοδο.
4. Predict Points: Πρόβλεψη πάνω σε κάποια χρονικά σημεία.
5. Predict Values: Πρόβλεψη πάνω σε κάποια δοθείσα είσοδο.
6. Model Comparison: Σύγκριση μεταξύ εκδόσεων των μοντέλων. Η σύγκριση πραγματοποιείται παράγοντας ένα διάγραμμα για την κάθε έκδοση. Για το διάγραμμα αυτό επιλέγεται μία ποσότητα της εισόδου για να μπει στον άξονα X καθώς και ένα εύρος τιμών γι αυτή. Στην συνέχεια επιλέγονται σταθερές τιμές για τα υπόλοιπα πεδία της εισόδου. Παράγεται έτσι ένας αριθμός προβλέψεων για την Y συντεταγμένη αρκετών σημείων ώστε να σχεδιαστεί ένα γράφημα. Το γράφημα αυτό αποδίδει οπτικά την σχέση των δύο ποσοτήτων (στον X και στον Y) κρατώντας σταθερά τα άλλα πεδία της εισόδου. Μέσα από τα γραφήματα των διαδοχικών εκδόσεων των μοντέλων μπορεί να παρατηρηθεί η μεταβολή της κατάστασης ενός πλοίο σε σχέση με αυτές τις ποσότητες.

Κατά την δημιουργία μίας οντότητας Task παράγεται και αποδίδεται σε αυτό ένα μοναδικό αναγνωριστικό κλειδί (Task ID). Άλλα πεδία πληροφοριών που συνθέτουν την δομή είναι το αναγνωριστικό του μοντέλου πάνω στο οποίο θα διεξαχθεί η δεδομένη εργασία, χρονικές σημάνσεις για την δημιουργία του, την έναρξη της εκτέλεσης του και την ολοκλήρωσή του, καθώς και ένα πεδίο για την περιγραφή της κατάστασής του. Η περιγραφή της κατάστασης του Task αφορά το στάδιο στο οποίο βρίσκεται η εκτέλεσή του. Τέτοια στάδια μπορεί να είναι:

- INITIALIZED(0): Το Task έχει δημιουργηθεί και αναμένει την έναρξη της εκτέλεσής του.
- DATA RETRIEVING(1): Έχει ξεκινήσει η εκτέλεση του Task και βρίσκεται στην φάση συλλογής δεδομένων.
- DATA PROCESSING(2): Έχει ξεκινήσει η διαδικασία της επεξεργασίας των δεδομένων που συλλέχθηκαν.
- COMPLETED(3): Το Task έχει ολοκληρωθεί επιτυχώς.
- ERROR(4): Η εκτέλεση του Task διακόπηκε λόγω κάποιου σφάλματος.
- NO DATA(5): Η εκτέλεση του Task δεν είναι δυνατή λόγω έλλειψης επαρκούς ποσότητας δεδομένων.

- EXECUTING IN BATCH(6): Η εκτέλεση του Task λαμβάνει χώρα εκτός cluster, χρησιμοποιώντας πόρους της υπηρεσίας Azure Batch.

- KILLED(7): Η εκτέλεση του Task διακόπηκε από κάποιο σήμα που λήφθηκε από την εφαρμογή.

Για την εκτέλεση κάποιου Task απαραίτητο είναι ένα σύνολο ορισμάτων τα οποία όμως διαφέρουν μεταξύ των διάφορων ειδών Task. Η πληροφορία αυτή συνθέτεται σε ένα μήνυμα με το οποίο αιτείται η εκτέλεση του Task. Παράλληλα μετά το πέρας της εκτέλεσης υπάρχουν διάφορα πεδία το ίδιου του Task, αλλά και του μοντέλου στο οποίο απευθύνεται τα οποία πρέπει να ενημερωθούν. Το σύνολο των πεδίων που χρειάζεται να ενημερωθούν μετά την ολοκλήρωση κάποιου Task, επίσης, διαφέρει από είδος σε είδος. Στον πίνακα 4.5 παρουσιάζονται τα δεδομένα που περιέχονται σε κάθε μήνυμα για την αίτηση εκτέλεσης κάθε είδους Task, καθώς και τα πεδία τα οποία θα χρειαστεί να ενημερωθούν μετά την εκτέλεση του.

Πίνακας 4.1: Πληροφορία εισόδου/εξόδου των διαφόρων ειδών Task.

Τύπος Task	Πεδία Μηνύματος	Πεδία για Ενημέρωση
Train	Task ID Αναγνωριστικό του Task. Task Type Όνομα του τύπου του Task. Model ID Αναγνωριστικό του μοντέλου. From Αρχή της περιόδου. To Τέλος της περιόδου. rowsNumber Αριθμός στιγμιοτύπων των δεδομένων. Sampling Rate Ρυθμός δειγματοληψίας.	Task Status Ενημερώνει την κατάσταση του Task. Model Version Δημιουργεί μία νέα καταχώρηση στις εκδόσεις του μοντέλου. Evaluation Δημιουργεί μία νέα καταχώρηση με την πρώτη αξιολόγηση της έκδοσης.
Evaluate	Task ID Αναγνωριστικό του Task. Task Type Όνομα του τύπου του Task. Model ID Αναγνωριστικό του μοντέλου. From Αρχή της περιόδου. To Τέλος της περιόδου. Sampling Rate Ρυθμός δειγματοληψίας.	Task Status Ενημερώνει την κατάσταση του Task Evaluation Δημιουργεί μία νέα καταχώρηση με την νέα αξιολόγηση της έκδοσης.
Predict Period	Task ID Αναγνωριστικό του Task. Task Type Όνομα του τύπου του Task. Model ID Αναγνωριστικό του μοντέλου. From Αρχή της περιόδου. To Τέλος της περιόδου. Sampling Rate Ρυθμός δειγματοληψίας.	Task Status Ενημερώνει την κατάσταση του Task Task Output Τα αποτελέσματα των προβλέψεων.
Predict Points	Task ID Αναγνωριστικό του Task. Task Type Όνομα του τύπου του Task. Model ID Αναγνωριστικό του μοντέλου. Time Points Τα χρονικά σημεία.	Task Status Ενημερώνει την κατάσταση του Task Task Output Τα αποτελέσματα των προβλέψεων.
Predict Values:	Task ID Αναγνωριστικό του Task. Task Type Όνομα του τύπου του Task. Model ID Αναγνωριστικό του μοντέλου. Input Vectors Τα διανύσματα εισόδου.	Task Status Ενημερώνει την κατάσταση του Task Task Output Τα αποτελέσματα των προβλέψεων.
Model Comparison	Task ID Αναγνωριστικό του Task. Task Type Όνομα του τύπου του Task. Model ID Αναγνωριστικό του μοντέλου. Constant Input Σταθερή είσοδος. Ranged Input Εύρος τιμών στον άξονα X. Model Versions Οι εκδόσεις των μοντέλων.	Task Status Ενημερώνει την κατάσταση του Task Task Output Η διεύθυνση στην οποία έχει αποθηκευτεί το διάγραμμα.

4.4 Σχεδίαση

4.4.1 Περιβάλλον λειτουργίας

Η πλατφόρμα θα μπορεί να χρησιμοποιηθεί από τα υπόλοιπα μέρη του Συστήματος συμπεριλαμβανομένου και της γραφικής διεπαφής χρήστη. Αυτό σημαίνει ότι η πλατφόρμα θα συμπεριφέρεται σαν ένα μαύρο κουτί, επικοινωνώντας με το περιβάλλον της μέσω μίας διεπαφής. Όπως είναι σαφές, την διεπαφή αυτή θα αποτελέσει ένα REST API.

Το περιβάλλον στο οποίο θα λειτουργεί η πλατφόρμα αποτελείται από έναν cluster υπολογιστών, καθώς αποτελεί μέρος μίας γενικότερης cloud υπηρεσίας. Η διαχείριση του cluster και των πόρων του αναλαμβάνεται από το λογισμικό Kubernetes. Παράλληλα η εφαρμογή θα αναπτυχθεί στο πλαίσιο της τεχνολογίας Docker. Τα διάφορα μέρη της πλατφόρμας, επομένως, θα λειτουργούν ως containerised εφαρμογές. Δύνεται έτσι η δυνατότητα εύκολης κλιμάκωσης γενικότερα της εφαρμογής, αλλά και των επιμέρους μερών της. Η παράλληλη χρήση του Docker και του Kubernetes προσδίδει πολλά οφέλη. Ένα από τα πιο σημαντικά είναι η δυνατότητα της δυναμικής ανάθεσης πόρων μεταξύ των Containers. Το γεγονός αυτό απλοποιεί πολύ τον τρόπο με τον οποίο μπορεί να κλιμακωθεί η πλατφόρμα όπως θα αναλυθεί και παρακάτω.

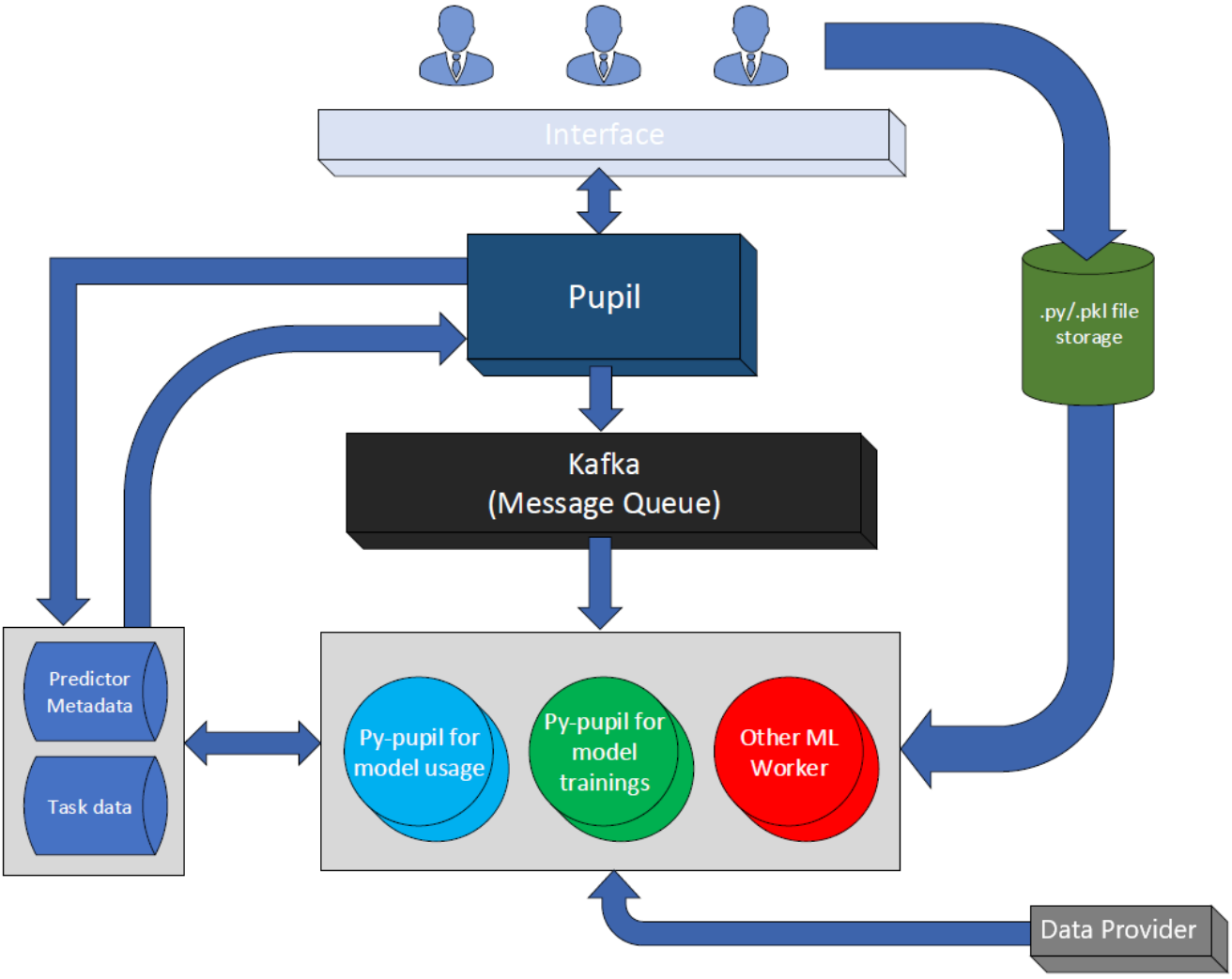
4.4.2 Αρχιτεκτονική

Ένας από τους στόχους της πλατφόρμας είναι να μην περιορίσει τους χρήστες της σε συγκεκριμένες γλώσσες προγραμματισμού, πακέτα, βιβλιοθήκες κ.λ.π. σχετικά με την υλοποίηση των αλγορίθμων μηχανικής μάθησης αλλά και γενικότερα. Για την επίτευξη ενός πρώτου επιπέδου αυτού του σκοπού, την ευκολότερη διαχείριση του φόρτου εργασίας, αλλά και για άλλους λόγους, που θα αναλυθούν παρακάτω, ακολουθείται το δημοφιλές σχεδιαστικό πρότυπο master-slave. Συγκεκριμένα τον ρόλου του master αναλαμβάνει μια εφαρμογή, στην οποία έχει δοθεί το όνομα Pupil. Ο ρόλος του slave μπορεί να υπάρξει μέσα από διαφορετικές εφαρμογές αναλόγως την γλώσσα προγραμματισμού που θα υποστηρίζει. Στην συγκεκριμένη εργασία έχει αναπτυχθεί σε γλώσσα python μία εφαρμογή για τον ρόλο του slave με το όνομα Py-Pupil.

Ο Pupil αναλαμβάνει εξολοκλήρου την επικοινωνία με το εξωτερικό περιβάλλον της πλατφόρμας. Αναλαμβάνει να παρέχει πληροφορίες σχετικά με τα μοντέλα που υπάρχουν στο σύστημα αλλά και να δέχεται αιτήματα για την εκτέλεση ενεργειών πάνω στα μοντέλα. Ο ρόλος των slave εφαρμογών είναι να εκτελούν αυτές τις ενέργειες και να προωθούν την έξοδο που προέκυψε αν υπάρχει. Η οντότητα με την οποία αντιπροσωπεύεται μία ενέργεια προς εκτέλεση ονομάζεται Task και θα αναλυθεί περισσότερο στην συνέχεια.

Για να παρθεί η απόφαση για τον τρόπο με τον οποίο θα προωθούνται τα αιτήματα στις slave εφαρμογές έπρεπε να ληφθούν υπ' όψιν οι παρακάτω ιδιαιτερότητες των Task. Βασική ιδιαιτερότητα των Task είναι ότι για την εκτέλεση τους απαιτείται μη αμελητέος χρόνος, ο οποίος μπορεί να είναι είτε κάποια δευτερόλεπτα κυρίως για την συλλογή δεδομένων, είτε και μέρες αν πρόκειται για κάποια εκπαίδευση. Επομένως δεν θα ήταν δυνατή η χρήση επικοινωνίας μέσω REST APIs. Ο λόγος είναι ότι αυξάνοντας τον χρόνο διατήρησης μίας HTTP σύνδεσης αυξάνεται και το ρίσκο διακοπής της,

άρα και αδυναμίας απόκτησης της εξόδου μετά το τέλος της εκτέλεσης του Task. Επιπλέον κρατώντας ανοιχτές HTTP συνδέσεις για πολύ χρόνο σπαταλώνται πόροι, καθώς για κάθε HTTP σύνδεση δεσμεύεται κάποιο μέρος της μνήμης και του server και του client. Μία ακόμα ιδιαιτερότητα, η οποία αφορά κυρίως τα Task για τις εκπαιδεύσεις, είναι ότι δεν είναι απολύτως απαραίτητο να εκτελούνται ταυτόχρονα. Αντιθέτως σε περιπτώσεις στις οποίες έρχονται πολλά αιτήματα για εκπαιδεύσεις θα ήταν άστοχο να δεσμευτεί μεγάλη ποσότητα πόρων για την ταυτόχρονη εκτέλεση τους. Για να ικανοποιηθούν οι παραπάνω ιδιαιτερότητες, ο τρόπος με τον οποίο επιτυγχάνεται η προώθηση των Task στις slave εφαρμογές είναι μέσο ενός message broker και συγκεκριμένα το kafka. Δημιουργείται έτσι μία ουρά εκτέλεσης των Task, μέσω της ουράς των μηνυμάτων. Η διαχειριστική εφαρμογή Pupil αναλαμβάνει τον ρόλο του Producer και οι εφαρμογές slave τον ρόλο του Consumer. Ο Pupil λαμβάνοντας κάποιο αίτημα μέσω της REST διεπαφής του για την εκτέλεση κάποιου Task, παράγει ένα μήνυμα, το οποίο περιέχει πληροφορίες για την εκτέλεση του Task, σε κάποιο topic του kafka. Στην συνέχεια αυτό το μήνυμα καταναλώνεται από κάποια slave εφαρμογή που είναι εγγεγραμμένη στο συγκεκριμένο topic και αναλαμβάνει την εκτέλεσή του. Στο Σχήμα 4.5 φαίνονται οι ρόλοι και η ροή λειτουργίας της πλατφόρμας.



Σχήμα 4.4: Η ροή λειτουργίας της πλατφόρμας.

Καθότι η εκτέλεση των Task είναι ασύγχρονη, κρίνεται απαραίτητο η διαχειριστική εφαρμογή να μην χάσει ολοκληρωτικά την επίγνωσή της για την ύπαρξή του. Σημαντικό, επίσης, είναι να υπάρχει η δυνατότητα ενημέρωσης για την πρόοδο της εκτέλεσης του Task. Η ενημέρωση μπορεί να απευθύνεται είτε σε κάποιον χρήστη, είτε σε κάποια άλλη εφαρμογή η οποία αναμένει την ολοκλήρωση της εκτέλεσης προκειμένου να λάβει το αποτέλεσμα που προέκυψε. Για την κάλυψη των παραπάνω αναγκών, δημιουργείται ένας πίνακας στην βάση δεδομένων της πλατφόρμας, η οποία αναπαριστά ένα Task. Κάθε εγγραφή σε αυτόν τον πίνακα αντιπροσωπεύει ένα Task το οποίο αιτήθηκε. Υπεύθυνος για την δημιουργία τέτοιον καταχωρήσεων, όπως είναι σαφές, αποτελεί η διαχειριστική εφαρμογή Pupil. Η δημιουργία και αρχικοποίηση των εγγραφών γίνεται με την λήψη κάποιου αιτήματος για την εκτέλεση κάποιου Task. Στην συνέχεια η slave εφαρμογή που έχει αναλάβει την εκτέλεση του Task ενημερώνει την εγγραφή στην βάση με την πρόοδο της εκτέλεσης του.

Κατά το τέλος της εκτέλεσης κάποιου Task, είτε παράγεται κάποια έξοδος, είτε παρουσιάζεται η ανάγκη ενημέρωσης κάποιων μεταβλητών σχετικά με την κατάσταση κάποιου μοντέλου λόγω της εκτέλεσης του Task. Για την πιθανή έξοδο των Task υπάρχει μία στήλη στον αντίστοιχο πίνακα στην βάση δεδομένων ώστε να αποθηκεύεται εκεί σε μορφή JSON String. Οι slave εφαρμογές, όπως γίνεται σαφές, αναλαμβάνουν να ενημερώνουν εγγραφές στην βάση δεδομένων. Εκτός από τον πίνακα για τα Task, αναλαμβάνουν να ενημερώνουν και την κατάσταση (όπως αυτή περιγράφεται στην βάση δεδομένων, το οποίο θα αναλυθεί παρακάτω) των μοντέλων όταν αυτή μεταβάλλεται μετά την εκτέλεση κάποιου Task.

Καθώς τα Task τοποθετούνται σε μία ουρά για εκτέλεσης, παρουσιάζεται το ενδεχόμενο κάποιο Task να περιμένει την ολοκλήρωση κάποιου άλλου για να ξεκινήσει. Υπάρχει, δηλαδή, η περίπτωση κάποιο Task για πρόβλεψη να περιμένει την ολοκλήρωση μίας εκπαίδευσης ενός μοντέλου για να εκτελεστεί. Αυτό είναι ένα σενάριο που θα πρέπει να σιγουρευτούμε ότι θα αποφευχθεί. Ο τρόπος με τον οποίο αποφεύγεται αυτό το σενάριο είναι χωρίζοντας τα Task σε δύο ομάδες και αναθέτοντας σε διαφορετικές slave εφαρμογές την εκτέλεσή των Task της κάθε ομάδας. Στην μία ομάδα τοποθετούνται τα Task για τις εκπαιδεύσεις μοντέλων, τα οποία μπορεί να είνια εξαιρετικά χρονοβόρα. Στην άλλη ομάδα τοποθετούνται όλα τα υπόλοιπα, οπού εκτός από το ότι απαιτούν σημαντικά λιγότερο χρόνο για την εκτέλεσή τους, χρειάζεται να εκτελούνται άμεσα. Η ανάθεση διαφορετικών slaves για κάθε ομάδα Task πραγματοποιείται με την δημιουργία δύο topics στο kafka και την εγγραφή διαφορετικών slave εφαρμογών στο κάθε ένα. Για την εκτέλεση κάποιου Task μίας ομάδας, η διαχειριστική εφαρμογή Pupil παράγει το μήνυμα στο αντίστοιχο kafka topic.

Τέλος, η κλιμάκωση της εφαρμογής μπορεί να επιτευχθεί μέσω της δημιουργίας νέων instances των εφαρμογών. Η διαδικασία αυτή, πέρα από το ότι είναι αρκετά απλή με την χρήση του Docker και του Kubernetes, αλλά και λόγω της αυτόματης και δυναμικής διαχείρισης των πόρων της υποδομής ελαχιστοποιείται η σπατάλη των πόρων. Με την κλιμάκωση της διαχειριστικής εφαρμογής Pupil, δίνεται η δυνατότητα για εξυπηρέτηση παραπάνω HTTP αιτημάτων. Με την κλιμάκωση των slave εφαρμογών αυξάνεται η απόδοση της εκτέλεσης των Task της εκάστοτε ομάδας.

4.4.3 Pupil

Pupil έχει ονομαστεί η διαχειριστική εφαρμογή (Master) της πλατφόρμας. Αναλαμβάνει την αλληλεπίδραση με το περιβάλλον της πλατφόρμας, το οποίο μπορεί να είναι είτε κάποιος χρήστης είτε

άλλες εφαρμογές. Η επικοινωνία με το περιβάλλον επιτυγχάνεται μέσω ενός Rest API. Εκτελεί τις βασικές CRUD διεργασίες στην βάση δεδομένων. Διαχειρίζεται την κυκλοφορία των Task παράγοντας μηνύματα στον message broker, τα οποία ωθούνται στην αντίστοιχη ουρά εργασιών με την ομάδα του Task. Επιπλέον, πραγματοποιεί την περιοδική παραγωγή αιτημάτων Task για την προγραμματισμένη επανεκπαίδευση ή αξιολόγηση μοντέλων.

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την εφαρμογή είναι η Java, ενώ βασικό εργαλείο για την ανάπτυξη πολλών χαρακτηριστικών αποτέλεσε το Spring Framework. Η οργάνωση του REST API της εφαρμογής γίνεται με την χρήση της κλάσης Controller που προσφέρει το Spring Framework. Τα αντικείμενα Controller διαχειρίζονται τα HTTP αιτήματα που απευθύνονται στην εφαρμογή. Κάθε ένα μπορεί να υποστηρίξει μία ομάδα από entpoints. Στο πλαίσιο της εφαρμογής Pupil έχουν αναπτυχθεί 4 Controllers:

1. InfoController: Εξυπηρετεί όλα τα αιτήματα για πληροφορίες σχετικά με τα μοντέλα και την κατάστασή τους.

2. QuantityPredictorController: Εξυπηρετεί όλα τα αιτήματα για την δημιουργία, την επεξεργασία και την διαγραφή εγγραφών στη βάση σχετικά με τα μοντέλα.

3. ModelController: Εξυπηρετεί όλα τα αιτήματα για την εκτέλεση κάποιου Task.

4. TaskController: Εξυπηρετεί τα αιτήματα για την άντληση της εξόδου κάποιου Task και της κατάστασής του.

Ένα ακόμα χαρακτηριστικό του Pupil, όπως αναφέρθηκε, είναι η δυνατότητα του προγραμματισμού περιοδικών εκτελέσεων Task σε συγκεκριμένα μοντέλα. Τα Task που μπορούν να προγραμματιστούν από τον χρήστη για περιοδικές εκτελέσεις είναι Task εκπαίδευσης και αξιολόγησης. Για κάθε μοντέλο όπως αναφέρθηκε διατηρούνται δύο flags που ενεργοποιούν τις περιοδικές εκτελέσεις εκπαιδεύσεων και αξιολογήσεων. Επιπλέον, μπορούν να οριστούν και δύο χρονικές περίοδοι για την εκτέλεση του αντίστοιχου Task μετά το πέρας της από την τελευταία εκτέλεση. Για την πυροδότηση των εκτελέσεων αυτών των Task ο Pupil διατηρεί δύο scheduled υπηρεσίες, οι οποίες εκτελούνται με περίοδο μίας ώρας και ανιχνεύουν την ύπαρξη τυχών Task τα οποία πρέπει να εκτελεστούν. Ο τρόπος με τον οποίο γίνεται η ανίχνευση είναι αντλώντας και φιλτράροντας με τις κατάλληλες συνθήκες (flag setted και τρέχουσα ημερομηνία μείον ημερομηνία τελευταίας εκτέλεσης μεγαλύτερη ή ίση της περιόδου.) τα μοντέλα από την βάση.

Κατά την πυροδότηση της εκτέλεσης κάποιου Task ο Pupil θα ενημερώσει τους ενδιαφερόμενους χρήστες μέσω του event system του Συστήματος. Στην περίπτωση προγραμματισμένης αξιολόγησης ενημέρωση θα υπάρξει μόνο αν η απόδοση του μοντέλου προκύψει να είναι σημαντικά μικρότερη. Για την παραγωγή κάποιου event αρκεί η παραγωγή ενός μηνύματος σε ένα συγκεκριμένο topic του kafka το οποίο εξυπηρετεί αυτό τον σκοπό για όλο το Σύστημα.

Καθώς για την πρόβλεψη κάποιας ποσότητας μπορεί να υπάρχουν παραπάνω από ένα μοντέλο μηχανικής μάθησης, κρίνεται απαραίτητη η ύπαρξη της δυνατότητας αυτόματης επιλογής μοντέλων. Αυτό το χαρακτηριστικό αναπτύσσεται μέσω της ModelSelection υπηρεσίας του Pupil. Η υπηρεσία αυτή προσφέρει διαφορετικές στρατηγικές για την επιλογή κάποιου μοντέλου όπως π.χ. αυτό με την καλύτερη αξιολόγηση, το καλύτερο εκ των οποίων παίρνουν ένα συγκεκριμένο σύνολο σημάτων για

είσοδο κ.λ.π.. Η επιλογή επιτυγχάνεται ορίζοντας τα κριτήρια της κάθε στρατηγικής σε μία λογική έκφραση και φιλτράροντας τα μοντέλα με βάση αυτή.

4.4.4 Py-Pupil

Για τον ρόλο του slave έχει αναπτυχθεί μία εφαρμογή στην γλώσσα προγραμματισμού Python. Σημειώνεται ότι αυτός ο ρόλος μπορεί να αναπτυχθεί σε οποιασδήποτε γλώσσα προγραμματισμού μπορεί να υποστηρίξει διασύνδεση με PostgreSQL και Apache Kafka. Βασικός του σκοπός αποτελεί η εκτέλεση των Task. Είναι εγγεγραμμένος σε ένα topic του Kafka και καταναλώνει επαναλαμβανόμενα το πρώτο στην ουρά μήνυμα, το οποίο περιγράφει ένα Task προς εκτέλεση. Αναλαμβάνει, επίσης, την φόρτωση, την ενσωμάτωση και την εκτέλεση του κώδικα των μοντέλων που είναι αποθηκευμένος στο file storage. Ενημερώνει τον αντίστοιχο πίνακα στην βάση δεδομένων για την πρόοδο του εκάστοτε Task, την έξοδό του άμα υπάρχει, αλλά και την εγγραφή του αντίστοιχου μοντέλου. Δημιουργεί και αποθηκεύει binary αρχεία μετά την εκπαίδευση μοντέλων για την μόνιμη αποθήκευσή τους και τα φορτώνει για να τα χρησιμοποιήσει.

Η εφαρμογή ελέγχει περιοδικά για νέα μηνύματα στο topic που είναι εγγεγραμμένη. Όταν εντοπίσει κάποιο καινούριο μήνυμα το αποκωδικοποιεί από binary σε μορφή JSON String. Στην συνέχεια το JSON String αναλύεται και μετατρέπεται σε μία δομή dictionary της python και προωθείται στις συναρτήσεις του module message_handling για διαχείριση. Στο module message_handling έχει οριστεί, αρχικά, ένα mapping με κλειδιά τα String των αναγνωριστικών των τύπων των Task και με τιμές αναφορές στις αντίστοιχες συναρτήσεις για την διαχείριση του εκάστοτε τύπου Task. Καθώς το αναγνωριστικό του τύπου του Task περιέχεται στο μήνυμα και βρίσκεται σε πεδίο με συγκεκριμένο κλειδί στο dictionary που δημιουργείται, η ανάκτηση και κλήση της αντίστοιχης συνάρτησης είναι απλή διαδικασία. Κατά την διαχείριση του εκάστοτε Task επεξεργάζονται τα στοιχεία στο dictonary του μηνύματος και διαμορφώνονται τα ορίσματα, ώστε καλεστεί με αυτά η κατάλληλη συνάρτηση από το execute module στην οποία ιείται η εκτέλεση του Task.

Κατά την εκτέλεση κάποιου Task συχνό βήμα αποτελεί η συλλογή δεδομένων. Αυτή πραγματοποιείται από το αντικείμενο DataBringer. Στην κλάση του DataBringer έχουν αναπτυχθεί διάφορες μέθοδοι οι οποίες λαμβάνουν ως ορίσματα αναγνωριστικά ποσοτήτων των πλοίων, χρονικούς προσδιορισμούς, ρυθμούς δειγματοληψίας κ.λ.π. και με βάση αυτά αιτούνται δεδομένα από το Σύστημα. Τα δεδομένα αυτά καθώς λαμβάνονται από τον DataBringer οργανώνονται σε μία δομή DataFrame της βιβλιοθήκης Pandas. Ως ονόματα των στηλών τοποθετούνται τα αναγνωριστικά των ποσοτήτων και ως index χρησιμοποιούνται οι χρονικές σημάνσεις κάθε στιγμιότυπου των δεδομένων.

Καθώς έχουν συλλεχθεί τυχών δεδομένα που χρειάζονται, έρχεται η σειρά της χρήσης του μοντέλου. Πρωταρχική ανάγκη αποτελεί η εισαγωγή του κώδικα του μοντέλου και η εκτέλεσή του ώστε να οριστεί ο τύπος (κλάση) του και να αναγνωρίζεται από τον διερμηνευτή. Αυτό συμβαίνει κατά την λειτουργία της εφαρμογής (runtime) το οποίο σημαίνει ότι η εισαγωγή των modules των μοντέλων γίνεται δυναμικά. Για την δυναμική εισαγωγή των modules χρησιμοποιείται η βιβλιοθήκη importlib και πραγματοποιείται σε τρεις φάσεις. Αρχικά διαβάζεται το .py αρχείο και δημιουργείται ένα ModuleSpec αντικείμενο το οποίο χρησιμοποιείται για την δημιουργία του αντικειμένου του module. Χρησιμοποιώντας το αντικείμενο αυτό εκτελείται το module και στην συνέχεια προστίθεται στο dictonary sys.modules του συστήματος. Στην περίπτωση που το module είχε εισαχθεί ξανά στο

σύστημα θα αντικατασταθεί με το νέο.

Καθώς κάθε module που περιέχει κώδικα για ένα μοντέλο θα πρέπει να αρχικοποιεί ένα αντικείμενο με συγκεκριμένο όνομα, αυτό μπορεί να χρησιμοποιηθεί για την δημιουργία του, όταν εκτελείται Task εκπαίδευσης. Καλώντας την train μέθοδο του και δίνοντας ως όρισμα το pandas.DataFrame που έχει δημιουργηθεί εκκινείται η διαδικασία για την εκπαίδευση του μοντέλου. Μετά την ολοκλήρωση της παρουσιάζεται η ανάγκη της μόνιμης αποθήκευσης του αντικειμένου αυτού. Αυτή η διαδικασία πραγματοποιείται μέσω του πακέτου pickle με το οποίο θα γίνει κωδικοποίηση του αντικειμένου σε binary μορφή. Παράγεται έτσι ένα αρχείο με κατάληξη .pkl στο οποίο δίνεται συγκεκριμένο όνομα με το αναγνωριστικό του μοντέλου και τον αύξοντα αριθμό της έκδοσης του και αποθηκεύεται στο file storage της εφαρμογής. Για την χρήση του μοντέλου αρκεί να φορτωθεί από το .pkl αρχείο και να αποκωδικοποιηθεί πάλι σε python αντικείμενο, έτσι καλώντας τις μεθόδους predict και evaluate γίνεται εφικτή η χρήση του.

4.4.5 Azure Batch

Πολλές φορές δημιουργούνται μοντέλα τα οποία για την εκπαίδευσή τους απαιτούν μεγάλα ποσά υπολογιστικών πόρων. Αυτές οι ανάγκες σε πόρους καθιστούν την εκτέλεση της εκπαίδευσής τους εντός του cluster επιζήμια για την ορθή λειτουργία του συστήματος αν αφεθεί ανεξέλεγκτη και υπερβολικά αργή αν οριστούν όρια στους πόρους που θα ανατεθούν για την συγκεκριμένη διαδικασία. Για να λυθεί το πρόβλημα αυτό θα πρέπει να υπάρξει σημαντική αύξηση των προσφερόμενων πόρων από την υποδομή του cloud χωρίς όμως η ανάγκη αυτή να είναι διαρκής. Η υπηρεσία Microsoft Azure που προσφέρει την υποδομή για το cloud του συστήματος μπορεί να παράσχει πόρους για την εκτέλεση high-performance computing (HPC) εργασιών, οι οποίοι να αποδεσμεύονται μετά το πέρας της εργασίας, μέσω του Azure Batch. Η χρήση του και η εκτέλεση τέτοιον εργασιών μέσω του Azure Batch, πραγματοποιείται μέσω του API που προσφέρει, την συγγραφή CLI Scripts ή της γραφικής διεπαφής του. Στην συγκεκριμένη περίπτωση γίνεται χρήση του μέσω του python και Java API.

Όταν κριθεί ότι η εκπαίδευση κάποιου μοντέλου θα απαιτήσει παραπάνω πόρους από όσους μπορεί να διαθέσει ο cluster του cloud, τότε δηλώνεται σαν MlService το αναγνωριστικό PY_PUPIL_BATCH. Αυτό θα διαφοροποιήσει την ροή εκτέλεσης των εκπαιδεύσεων αυτων των μοντέλων. Σε αυτήν την περίπτωση, λοιπόν, κατά την διαχείριση του μηνύματος από τον Py-Pupil, δεν θα καλεστεί η συνήθης διαδικασία για την εκτέλεση της εκπαίδευσης αλλα μια διαδικασία που θα προετοιμάσει και θα εκκινήσει την εκτέλεση στο Azure Batch. Κατά την προετοιμασία της εκτέλεσης θα δημιουργηθεί (ειδικά για την συγκεκριμένη εργασία) μία διεύθυνση στο file storage η οποία είναι προσβάσιμη από το Azure Batch και εκεί θα αποθηκευτούν σε ένα αρχείο τα δεδομένα που θα χρησιμοποιηθούν για την εκπαίδευση. Το αρχείο αυτό αποτελεί ένα αρχείο .pkl με την κωδικοποίηση του pandas.Dataframe που έχει δημιουργηθεί μέσω του πακέτου pickle.

Αφού εκκινηθεί η διαδικασία εκπαίδευσης στο Azure Batch ο Py-Pupil ανανεώνει την κατάσταση του Task με τον κατάλληλο κωδικό. Η παρακολούθησή του και η διαχείρισή της ολοκλήρωσής του αποτελεί αρμοδιότητα της διαχειριστικής εφαρμογής Pupil. Ο Pupil διατηρεί μία υπηρεσία η οποία εκτελεί περιοδικούς ελέγχους για την ολοκλήρωση Task που έχουν σημανθεί με τον συγκεκριμένο κωδικό, που δηλώνει ότι εκτελούνται στο Azure Batch. Ο έλεγχος αυτός πραγματοποιείται μέσω του Java API που προσφέρει το Aazure Batch. Κατά την ολοκλήρωση της εκτέλεσης της εκπαίδευσης

στο Azure Batch, έχει αποθηκευτεί στην διεύθυνση, που δημιουργήθηκε γι αυτό το σκοπό, το .pkl αρχεί του αντικειμένου του μοντέλου και ένα JSON αρχείο με πεδία που πρέπει να ανανεωθούν στην βάση (όπως η καινούρια έκδοση του μοντέλου). Στην περίπτωση που η εκτέλεση δεν ήταν επιτυχής θα βρεθεί στην διεύθυνση αυτή ένα txt αρχείο με τα logs που παράχθηκαν κατά το σφάλμα της εκτέλεσης. Ο Pupil καθώς εντοπίζει κάποιο Task που έχει ολοκληρωθεί θα μεταφέρει το pkl αρχείο του μοντέλου στην σωστή διεύθυνση και διαβάζοντας το JSON αρχείο ανανεώνει την βάση με τα στοιχεία που περιέχονται σε αυτό. Στην περίπτωση σφάλματος θα ανανεώσει την κατάσταση του Task με τον αντίστοιχο κωδικό αντί για τον κωδικό της επιτυχής ολοκλήρωσης.

4.5 Rest API

Στους παρακάτω πίνακες παρουσιάζεται το API της εφαρμογής. Κάθε πίνακας περιέχει τα endpoints καθενός από τους 4 controllers του Pupil.

Πίνακας 4.2: Info Controller endpoints.

Name	Description	Return Value	Path	Parameters	Method
Get Predictors Info	Returns information regarding a Quantity Predictor.	QuantityPredictorDto	/info/predictor	vesselId, outputQid, onlyUsableModels=true	GET
Get Models Info	Returns information regarding a Model of a Quantity Predictor.	ModelDto	/info/model	vesselId, outputQid, modelName	GET
Get Model Versions Info	Returns information regarding a Models versions.	List<ModelVersionDto>	info/model-versions/modelId	–	GET
Predictor's Usable Models	Returns a list of string with a predictor's model name's.	List<String>	/info/predictor/usables	vesselId, outputQid	GET
Vessel's Predictors Info	Returns information about a vessel's available Quantity Predictors.	List<QuantityPredictorDto>	/info/vessel/predictors/vesselId	–	GET
Get Evaluation	Returns an evaluation record of a model.	ModelEvaluation	/info/evaluation	vesselId, outputQname. modelName, from, to	GET
Get Evaluation Records	Returns the complete record of a models evaluations.	List<ModelEvaluation>	/info/evaluation/records	vesselId, outputQname. modelName	GET

Πίνακας 4.3: Quantity Predictor Controller endpoints.

Name	Description	Return Value	Path	Parameters	Method
Create Predictor	Creates a Predictor	String	/predictor/create	(body) predictor	POST
Update Predictor	Updates an existing Predictor	String	/predictor/update	(body) predictor	POST
Delete Predictor	Deletes a Predictor	String	/predictor/delete	vesselId, outputQid	DELETE
Add Model	Adds a model to a Quantity Predictor. Creates a Quantity Predictor if it doesn't exist.	String	/predictor/add$_{m}odel/and_{p}redictor$	vesselId, outputQid, modelName, mlService, (body) modelsQidParams	POST
Delete Model	Deletes a Model	String	/predictor$_{e}dit/delete_{m}odel/modelId$	–	DELETE
Delete Model Versions	Deletes one or more versions of a model.	String	/predictor/delete$_{v}ersions/modelId$	versions	DELETE
Update Model	Updates an existing Model	String	/predictor/update$_{m}odel$	vesselId, outputQid, (body) model	POST
Update Model	Updates an existing Model	String	/predictor/update$_{m}odel/modelId$	(body) model	POST

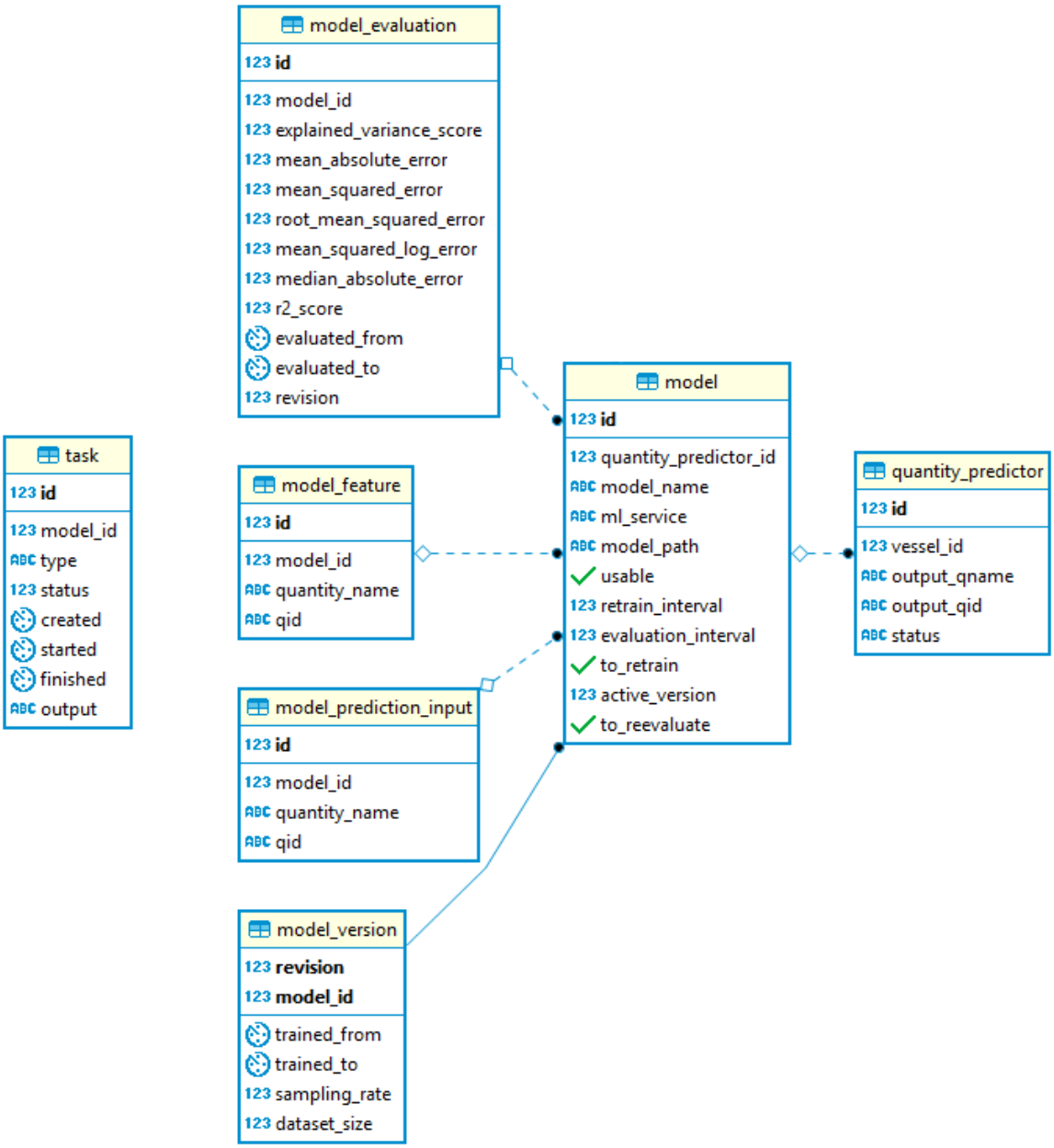
Πίνακας 4.4: Model Controller endpoints.

Train Model	Trains a model, stores the model file, updates the model entry in db, return the taskId.	Integer	/model/train/modelId	from, to, samplingRate or rowsNumber	GET
Predict by Timestamps	Makes Prediction from real data regarding the given timestamps, stores output in db, returns taskId	Integer	/model/predict/points	vesselId, outputQid, modelName, timestamps	GET
Predict by Values	Makes Prediction with the given values as data, stores output in db, returns taskId	Integer	/model/predict	vesselId, outputQid, modelName, parameters	POST
Predict by Period	Makes Prediction from real data regarding the given period, stores output in db, returns taskId	Integer	/model/predict/period	vesselId, outputQid, modelName, from, to, samplingRate or rowsNumber	GET
Evaluate	Performs an evaluation based on data regarding the given period, stores output in db, returns taskId	Integer	/model/evaluate/modelId	from, to, samplingRate or rowsNumber	GET
Model Comparison	Sends a request to start a model comparison task, returns taskId	Integer	/model/comparison	vesselId, outputQid, modelName, size, formulaType, versions, qidToVisualize, (body) featureRanges	POST

Πίνακας 4.5: Task Controller endpoints.

Get Task Status	Returns the tasks status	PupilTaskStatus	/task/status	taskId	GET
Get Output	Returns the output stored after the completion of a task as a JSON String.	String	/task/output	taskId	GET

4.6 Σχήμα βάσης δεδομένων



Σχήμα 4.5: Σχήμα βάσης δεδομένων

Κεφάλαιο 5

Ανάπτυξη

5.1 Ανάπτυξη Πλατφόρμας

5.1.1 Διαχείριση HTTP Αιτημάτων

Η διαχείριση των HTTP αιτημάτων πραγματοποιείται με την συμβολή του Spring Framework και συγκεκριμένα του Spring Boot. Για την διαχείριση τους αρχικά δημιουργείται μία κλάση την οποία γίνεται annotate με το @Controller. Πρόκειται για μία εξειδίκευση του annotation @Component η οποία δηλώνει μία κλάση ως ελεγκτή και επιτρέπει τον αυτόματο εντοπισμό της μέσω της σάρωσης του classpath. Για την διαχείριση του κάθε entpoint, δηλαδή κάθε διαφορετικού url του API, ορίζεται μία μέθοδος με το annotation @RequestMapping. Παρακάτω παρουσιάζεται μία μέθοδος με την οποία πραγματοποιείται η διαχείριση των αιτημάτων για την εισαγωγή κάποιου μοντέλου στην βάση.

```
/**
 * Adds a model to a Quantity Predictor. Creates a Quantity Predictor
    if it
 * doesn't exist.
 *
 *
 * @param vesselId
 *          The id of the vessel which is related to the Quantity
    Predictor.
 * @param outputQid
 *          The id of the target quantity.
 * @param modelName
 *          The name of the model.
 * @param mlService
 *          The mlService of the model.
 * @param modelsQidParams
 *          Model's sets of qids to construct features and prediction
    input.
 * @return Response message.
```

```
 */
@PostMapping(value = { "/add_model/and_predictor" })
@ResponseBody
public ResponseEntity<?> addModel(@RequestParam("vesselId") int
    vesselId,
     @RequestParam("outputQid") String outputQid, @RequestParam("
        modelName") String modelName,
     @RequestParam("mlService") MlService mlService,
     @RequestBody ModelsQidParams modelsQidParams) {

  if (validationService.modelExists(vesselId, outputQid, modelName)) {
    String message = "A model with name \"" + modelName
        + "\" already exists for this Quantity Predictor."
        + "Please give another name for this model or update the
           existing one.";
    return new ResponseEntity<>(message, HttpStatus.BAD_REQUEST);
  }
  Long modelId = quantityPredictorEditingService.addModel(vesselId,
     outputQid, modelName,
      modelsQidParams.getModelFeatureQids(), modelsQidParams.
         getModelPredictionInputQids(),
      mlService);
  if (modelId != null) {
    return new ResponseEntity<>(modelId, HttpStatus.CREATED);
  } else {
    return new ResponseEntity<>("Failed to add this model.", HttpStatus
       .INTERNAL_SERVER_ERROR);
  }
}
```

Η μέθοδος addModel έχει γίνει annotate με τα @PostMapping και @ResponseBody. Το @PostMapping αποτελεί μία εξειδίκευση του annotation @RequestMapping και για την διαχείριση POST αιτημάτων. Το @ResponseBody ορίζει ότι η δομή που θα επιστραφεί αυτόματα κωδικοποιείται σε JSON μορφή και τοποθετείται στο αντικείμενο HttpResponse που θα λάβει ο client. Τα ορίσματα της μεθόδου είναι τα vesselId, outputQid, modelName και mlService τα οποία με την χρήση του annotation @RequestParam αντιστοιχίζονται στα ορίσματα που αναγράφονται στο URL και το modelsQidParams το οποίο με το annotation @RequestBody δηλώνεται ότι θα αρχικοποιηθεί από το body του αιτήματος. Το vesselId και outputQid αποτελούν τα αναγνωριστικά για το πλοίο και την ποσότητα πρόβλεψης, με το modelName ορίζεται το επιθυμητό όνομα του μοντέλου και το mlService ορίζει το περιβάλλον με το οποίο θα είναι συμβατό. Τέλος η δομή modelsQidParams περιέχει τα αναγνωριστικά των ποσοτήτων για την είσοδο της πρόβλεψης και της εκπαίδευσης του μοντέλου.

Κατά την κλήση της μεθόδου, πρώτο βήμα αποτελεί η επικύρωση των ορισμάτων. Αυτό πραγματοποιείται από την υπηρεσία validationService, η οποία ελέγχει αν το όνομα που έχει αιτηθεί για το μοντέλο χρησιμοποιείται ήδη από κάποιο άλλο μοντέλο του ίδιου Quantity Predictor. Στην περί-

πτωση που το όνομα χρησιμοποιείται η μέθοδος θα τερματίσει επιστρέφοντας ένα ResponseEntity αντικείμενο με τον κωδικό σφάλματος 400 (Bad Reqtuest) και ένα μήνυμα που περιγράφει το σφάλμα. Σε διαφορετική περίπτωση η ροή εκτέλεσης συνεχίζεται με την κλήση της μεθόδου addModel της υπηρεσίας quantityPredictorEditingService, η οποία θα αναλάβει να προσθέσει το μοντέλο στην λίστα του Quantity Predictor για το συγκεκριμένο πλοίο και ποσότητα πρόβλεψης, αν αυτός υπάρχει. Αν δεν υπάρχει Quantity Predictor η υπηρεσία θα τον δημιουργήσει και θα του προσθέσει το μοντέλο. Στην συνέχει ελέγχεται αν η προσθήκη ήταν επιτυχής εξετάζοντας την τιμή του modelId, στην οπόια θα πρέπει να υπάρχει το αναγνωριστικό του μοντέλου που δημιουργήθηκε. Στην περίπτωση που υπάρχει τιμή επιστρέφεται από την μέθοδο ένα αντικείμενο ResponseEntity με το αναγνωριστικό του μοντέλου και την κατάσταση 200 (Created), αλλιώς το αντικείμενο θα περιέχει την κατάσταση γενικού σφάλματος 500 (Internal Server Error) μαζί με ένα μήνυμα.

5.1.2 Διασύνδεση Βάσης Δεδομένων

Διασύνδεση με την βάση δεδομένων πραγματοποιείται και από την διαχειριστική εφαρμογή Pupil μέσω Java, αλλά και από τις slave εφαρμογές με χρήση της Python. Για την γλώσσα Java έχει θεσπιστεί η προδιαγραφή Java Persistence API (JPA), η οποία αποτελεί μία διεπαφή για την διασύνδεση Java εφαρμογών με βάσεις δεδομένων. Η υλοποίηση της διεπαφής JPA που χρησιμοποιήθηκε για την εφαρμογή είναι το Hibernate. Το Hibernate είναι ένα εργαλείο Object Relational Mapping (ORM) για τη Java. Παρέχει ένα framework για την αντιστοίχηση ενός αντικειμενοστραφούς μοντέλου αναπαράστασης δεδομένων με μια σχεσιακή βάση δεδομένων. Το Spring Framework κάνοντας χρήση του Hibernate (κατ' επιλογή του χρήστη) παρέχει το module Spring Data JPA το οποίο απλοποιεί σε μεγαλύτερο βαθμό την διασύνδεση με την βάση δεδομένων. Δημοφιλές χαρακτηριστικό του Spring Data JPA αποτελεί η εφαρμογή χωρίς την συγγραφή κώδικα του σχεδιαστικού προτύπου repository και τη δημιουργία ερωτημάτων για την βάση δεδομένων από ονόματα μεθόδων.

Παρακάτω παρουσιάζεται η κλάση με την οποία αντιστοιχίζονται καταχωρήσεις του πίνακα model με Java αντικείμενα στο περιβάλον εκτέλεσης της εφαρμογής.

```
/**
 * Pupil model entity. Mapped with pupil.model table in pupil schema.
 *
 * @author Isaias.Vrakidis
 *
 */
@Entity
public class Model implements Serializable {

  private static final long serialVersionUID = 1L;

  /**
   * A unique id for the model.
   */
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
private Long id;

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "quantity_predictor_id")
private QuantityPredictor quantityPredictor;

/**
 * The type of the model.
 */
@Column(nullable = false, unique = false)
private String modelName;

/**
 * The path where the models file is stored.
 */
@Column(nullable = false, unique = true)
private String modelPath;

/**
 * The ML Service under this model is or will be created.
 */
@Enumerated(EnumType.STRING)
@Column(nullable = false, unique = false)
private MlService mlService;

/**
 * The fixed rate with which model will be retrained.
 */
@Column(nullable = false, unique = false)
private Long retrainInterval = 2592000L;

/**
 * The fixed rate with which the model will be evaluated.
 */
@Column(nullable = false, unique = false)
private Long evaluationInterval = 2592000L;

/**
 * A flag to enable or disable the automatic scheduled retrain.
 */
@Column(nullable = false, unique = false)
private boolean toRetrain;

/**
 * A flag to enable or disable the automatic scheduled reevaluation.
```

```
     */
    @Column(nullable = false, unique = false)
    private boolean toReevaluate;

    @Column(nullable = true, unique = false)
    private Integer activeVersion;

    /**
     * List of all the evaluation records.
     */
    @OneToMany(orphanRemoval = true, cascade = CascadeType.ALL)
    @LazyCollection(LazyCollectionOption.FALSE)
    @JoinColumn(name = "model_id", referencedColumnName = "id", insertable
       = true, updatable = true)
    private List<ModelEvaluation> evaluationRecords;

    /**
     * The qids of the the models input quantities/parameters.
     */
    @OneToMany(orphanRemoval = true, cascade = CascadeType.ALL, fetch =
       FetchType.EAGER)
    @JoinColumn(name = "model_id", referencedColumnName = "id", insertable
       = true, updatable = true)
    @OrderBy(value = "id")
    private List<ModelFeature> features;

    /**
     * The qids of the the models input quantities/parameters.
     */
    @OneToMany(orphanRemoval = true, cascade = CascadeType.ALL)
    @LazyCollection(LazyCollectionOption.FALSE)
    @JoinColumn(name = "model_id", referencedColumnName = "id", insertable
       = true, updatable = true)
    @OrderBy(value = "id")
    private List<ModelPredictionInput> predictionInput;

    /**
     * The model Versions.
     */
    @OneToMany(orphanRemoval = true, cascade = CascadeType.ALL)
    @LazyCollection(LazyCollectionOption.FALSE)
    @JoinColumn(name = "modelId", referencedColumnName = "id", insertable =
        false, updatable = false)
    @OrderBy(value = "revision")
    private List<ModelVersion> modelVersions;
```

```
    /**
     * Indicates if the model is safe to use.
     */
    @Column(nullable = false, unique = false)
    private boolean usable;

    /*  Constructors,
        getters, setters
        and other methods... */

}
```

Πάνω από την δήλωση της κλάσης υπάρχει το annotation @Entity με το οποίο απλά δηλώνεται η αντιστοίχηση με κάποιον πίνακα της βάσης δεδομένων. Στην περίπτωση αυτή όπου το όνομα της κλάσης ταυτίζεται με το όνομα του πίνακα, το Spring θα κάνει την αντιστοίχηση μόνο του. Το Spring έχει την δυνατότητα να αντιστοιχίζει λέξεις μεταξύ τους, παραβλέποντας το στυλ γραφής τους. Μπορεί να κάνει δηλαδή μετατροπές από CamelCase σε snake_case και αντίστροφα. Τα πεδία της κλάσης αφορούν είτε κάποια κολόνα του πίνακα είτε κάποια σχέση του με κάποιον άλλο πίνακα. Το πρώτο πεδίο που αντιστοιχίζεται είναι το αναγνωριστικό των μοντέλων, το οποίο δηλώνεται με το annotation @Id. Στο συγκεκριμένο πεδίο ανατίθεται τιμή αυτόματα κατά την καταχώρηση κάποιας καινούριας εγγραφής στον πίνακα, το οποίο δηλώνεται με το @GeneratedValue(strategy=GenerationType.IDENTITY). Στην συνέχεια ορίζεται ένα πεδίο για να αντιστοιχιστεί η σχέση του πίνακα model με τον πίνακα quantity_predictor. Για την επίτευξη της αντιστοίχησης χρησιμοποιείται το annotation @ManyToOne(fetch=FetchType.LAZY) που δηλώνει το είδος της σχέσης καθώς και τον τρόπο με τον οποίο θα υπολογίζεται μέσω του ορίσματος fetch. Η τιμή FetchType.LAZY δηλώνει ότι το πεδίο θα υπολογίζεται και η εφαρμογή θα ανατρέχει στην βάση για να φέρει τα δεδομένα, μόνο όταν πρόκειται να χρησιμοποιηθεί και όχι κατά την δημιουργία του αντικειμένου. Από κάτω, με το @JoinColumn(name="quantity_predictor_id"), δηλώνεται το όνομα της κολόνας που αποτελεί το foreign key και αναφέρεται στο αναγνωριστικό του πίνακα quantity_predictor. Τέλος, για την αντιστοίχηση των απλών κολόνων του πίνακα με πεδία χρησιμοποιείται το annotation @Column() στο οποίο μπορούν να περαστούν ορίσματα για την επίτρεψη ή όχι της ανάθεσης null τιμής σε αυτά (nullable), την μοναδικότητά τους (unique) κ.λ.π.

Παρακάτω παρουσιάζεται η δομή η οποία χρησιμοποιείται για την εκτέλεση ερωτημάτων στην βάση σχετικά με τον πίνακα quantity_predictor.

```
/**
 * Data access service for Quantity Predictor related data.
 *
 * @author Isaias.Vrakidis
 *
 */
@RepositoryRestResource(exported = false)
public interface QuantityPredictorSDao extends CrudRepository<
    QuantityPredictor, Long> {
```

```
    /**
     * Finds the Quantity Predictor.
     *
     * @param vesselId
     *          The id of the vessel which is related to the Quantity
     *          Predictor.
     * @param outputQid
     * @return A QuantityPredictor object.
     */
    List<QuantityPredictor> findByVesselIdAndOutputQid(int vesselId, String
        outputQid);

    /**
     * Finds the Quantity Predictors.
     *
     * @param vesselId
     *          The id of the vessel for which the related Quantity
     *          Predictors will be found.
     * @return A List of QuantityPredictor.
     */
    List<QuantityPredictor> findAllByVesselId(@Param(value = "vesselId")
       int vesselId);

}
```

Πρόκειται για ένα Java Interface, που κληρονομεί από το Generic Interface CrudRepository<QuantityPredictor, Long>, στο οποίο παρέχει υλοποίηση το Spring αυτόματα με την χρήση του annotation @RepositoryRestResource(exported=false). Δίνοντας στο όρισμα exported τιμή true ενεργοποιείται η λειτουργία του Spring, με την οποία θα παράξει αυτόματα μία προκαθορισμένη λογική για να διαχειρίζεται HTTP αιτήματα που αφορούν CRUD λειτουργίες του συγκεκριμένου πίνακα. Στην συγκεκριμένη περίπτωση αυτό δεν είναι επιθυμητό καθώς έχει αναπτυχθεί η λογική για την διαχείριση των HTTP αιτημάτων με τρόπο που εξυπηρετεί την εφαρμογή καλύτερα. Στο Generic Interface CrudRepository δίνονται ως παράμετροι ο τύπος της δομής που αντιστοιχίζεται στον πίνακα και ο τύπος του πεδίου που αντιστοιχίζεται στο αναγνωριστικό κλειδί του πίνακα. Στο κομμάτι κώδικα ορίζονται δύο μέθοδοι με τις οποίες μπορούν να παραχθούν πιο εξειδικευμένα ερωτήματα. Αυτά αφορούν την εύρεση όλων των καταχωρήσεων με συγκεκριμένο αναγνωριστικό πλοίου και ποσότητας πρόβλεψης, και μόνο με συγκεκριμένο αναγνωριστικό πλοίου. Σημειώνεται ότι για συγκεκριμένο πλοίο και ποσότητα πρόβλεψης θα υπάρχει μόνο μία καταχώρηση στον πίνακα quantity_predictor, οπότε η μέθοδος θα επιστρέφει μια λίστα με το πολύ ένα στοιχείο. Το Spring χρησιμοποιώντας τα ονόματα των μεθόδων παράγει και το αντίστοιχο ερώτημα.

Για την διασύνδεση με την βάση δεδομένων μέσω της γλώσσας Python χρησιμοποιήθηκε το SQLAlchemy. Το SQLAlchemy είναι ένα SQL και Object Relational Mapping (ORM) toolkit για την Python. Παρέχει μια πλήρη σουίτα δημοφιλών, σε επιχειρησιακό επίπεδο, προτύπων διατήρησης

δεδομένων, σχεδιασμένα για αποτελεσματική και υψηλής απόδοσης πρόσβαση σε βάσεις δεδομένων, προσαρμοσμένα σε μια απλή, εναρμονισμένη με τον χαρακτήρα της Python, διεπαφής.

Παρακάτω παρουσιάζεται ένα κομμάτι κώδικα για τον ορισμό μίας δομής με την οποία θα παράγονται αντικείμενα για να αντιστοιχιστούν με καταχωρήσεις στην βάσης δεδομένων.

```
class Model(Base):
    """Represent a Model in the database."""
    __tablename__ = "model"
    __table_args__ = {'schema': 'pupil'}

    id = Column(Integer, primary_key=True)
    quantity_predictor_id = Column(Integer, ForeignKey('pupil.
       quantity_predictor.id'))
    quantity_predictor = relationship("QuantityPredictor", back_populates
       ="models")
    model_name = Column(String)
    ml_service = Column(String)
    model_path = Column(String)
    usable = Column(Boolean)
    retrain_interval = Column(BIGINT)
    evaluation_interval = Column(BIGINT)
    to_retrain = Column(Boolean)
    active_version = Column(Integer)
    features = relationship("ModelFeature")
    prediction_input = relationship("ModelPredictionInput")
    pipeline = relationship("PipelineStep")
    evaluation_record = relationship("ModelEvaluation")
    model_versions = relationship("ModelVersion")
```

Η παραπάνω κλάση κληρονομεί από την κλάση Base η οποία είναι μία κλάση που παρέχει το SQLAlchemy μέσω της μεθόδου declarative_base() και απ αυτήν θα πρέπει να κληρονομούν όλες οι κλάσης που αντιστοιχίζονται με πίνακες της βάσης δεδομένων. Το όνομα του πίνακα με το οποίο θα αντιστοιχίζεται η κλάση δηλώνεται στο πεδίο __tablename__, ενώ στην περίπτωση μας όπου ο πίνακας ανήκει σε ένα σχήμα της βάσης, αυτό ορίζεται στο πεδίο __table_args__. Ακολουθούν τα υπόλοιπα πεδία της κλάσης τα οποία αντιστοιχίζονται είτε με κολόνες του πίνακα είτε με σχέσεις του. Στα πεδία που αντιστοιχίζονται με κολόνες ανατίθεται η τιμή τους μέσω της συνάρτησης Column() η οποία παίρνει ως όρισμα τον τύπο δεδομένων του πεδίου και καθώς αυτά φέρουν το ίδιο όνομα με την κολόνα. Για τα πεδία που αντιπροσωπεύουν σχέσεις του πίνακα χρησιμοποιείται η συνάρτηση relationship() η οποία πέρνει σαν όρισμα το όνομα του πίνακα που αφορά αυτή η σχέση.

Για την εκτέλεση ερωτημάτων μέσω του SQLAlchemy δημιουργούνται το αντικείμενο engine το οποίο κρατά τα στοιχεία σύνδεσης με την βάση και με χρήση αυτού η κλάση Session, όπως φαίνεται παρακάτω.

```
engine = create_engine(make_connection_string())
Session = sessionmaker(bind=engine)
```

Με την κλάση Session δημιουργούνται αντικείμενα τα οποία φέρουν μεθόδους για την εκτέλεση ερωτημάτων στην βάση. Παρακάτω παρουσιάζεται ένα παράδειγμα χρήσης ενός τέτοιου αντικειμένου για την ανάκτηση ενός μοντέλου από την βάση.

```
    def get_model(self, id):
        """Retrieve a Model from the database.

        Args:
            id (int): The id of the Model.

        Returns:
            :obj:'entities.Model': The Model that was retrieved.

        """
        self.maintain_session()
        result = self.session.query(Model).get(id)
        self.session.commit()
```

Η παραπάνω μέθοδος ανήκει σε ένα αντικείμενο το οποίο δημιουργεί και κρατάει το αντικείμενο session το οποίο συντηρεί με την βοήθεια της μεθόδου maintain_session() σε περίπτωση που έχει καταστραφεί.

5.1.3 Αποστολή και Λήψη Μηνυμάτων

Η αποστολή και η λήψη μηνυμάτων μέσω Apache Kafka συμβαίνει και στις δύο εφαρμογές. Η εφαρμογή Pupil χρησιμοποιεί μία επέκταση που προσφέρει το Spring Framework η οποία παρέχει εργαλεία για να αποστέλλει τα μηνύματα με τα οποία αιτούνται οι εκτελέσεις των Task. Αντίστοιχα οι slave εφαρμογές χρησιμοποιούν το πακέτο confluent-kafka που προσφέρει την λειτουργικότητα για να καταναλώσουν μηνύματα.

Παρακάτω θα παρουσιαστεί ο τρόπος που γίνεται χρήση του πακέτου confluent-kafka από τις slave εφαρμογές. Καθώς η κατανάλωση μηνυμάτων αποτελεί την βάση των εφαρμογών αυτών η λειτουργικότητα αυτή βρίσκεται στην main() συνάρτηση που τρέχει με την εκτέλεση της εφαρμογής.

```
# creating consumer and subsribing to channel
c = Consumer(config.kafka['consumer_settings'])
c.subscribe(config.kafka['channel'])

def terminate():
    logger = LOGGER
    logger.warning('Exiting, closing kafka connection.')
    try:
        c.close()
    except RuntimeError as e:
        logger.warning(str(e))

def main():
```

```
    """Polls continuously for any new messages in the subscribed channels

    When a new message is detected, it is delivered for handling after it
    has been decoded from binary to json and from json to dictonary. When
    the end of partition is reached or an error occures, propper messages
    are logged.

    Raises:
        KeyboardInterrupt: If interruption is forced externally, app will
            exit.

    """
    # Get a logger instance.
    logger = LOGGER
    logger.info('Py-pupil started. Use Ctrl+c to stop!')
    signal_handler = SignalHandler(c)

    try:
        while True:
            msg = c.poll(config.app['polling_seconds'])
            if msg is None:
                continue
            elif not msg.error():
                try:
                    decoded_message = msg.value().decode()
                    # log only the first 250 chars of message to avoid
                       long logs.
                    logger.info('New message detected: {}'.format(
                       decoded_message[:250]))
                    # pass signal_handler to the flow of task execution.
                    message_body = json.loads(decoded_message)
                    message_body['signal_handler'] = signal_handler
                    message_handling.message_delivery(message_body)
                    if signal_handler.terminate:
                        break
                except:
                    logger.error('An error occurred.')
                    traceback.print_exc(file=sys.stdout)
            elif msg.error().code() == KafkaError._PARTITION_EOF:
                signal_handler.on_polling()
                logger.info('End of partition reached {0}/{1}'
                      .format(msg.topic(), msg.partition()))
            else:
                logger.error('Error occured: {0}'.format(msg.error().str
                   ()))
```

```
except KeyboardInterrupt:
    logger.info('Exiting...')
    exit()
finally:
    terminate()
```

Αρχικά δημιουργείται το αντικείμενο c το οποίο αντιπροσωπεύει έναν kafka consumer, κρατά την σύνδεση με τον cluster του kafka και παρέχει μεθόδους για τον χειρισμό του. Η πρώτη μέθοδος που καλείται είναι για την εγγραφή του consumer στο αντίστοιχο topic. Τα στοιχεία για την σύνδεση και την εγγραφή στο topic προέρχονται από ένα config αρχείο το οποίο μετατρέπεται σε ένα python dictionary. Στην συνέχεια όπως παρατηρείται ορίζεται μία συνάρτηση για τον τερματισμό της εφαρμογής στην οποία καλείται η μέθοδος close() του Consumer για τον τερματισμό της σύνδεσης.

Μέσα στην συνάρτηση main() εκτελείται μία ”ατέρμονη” επανάληψη. Στο ξεκίνημα κάθε βρόχου καλείται η μέθοδος poll του Consumer με όρισμα έναν αριθμό που προκύπτει από το config αρχείο. Η μέθοδος poll() μπλοκάρει έως ότου ένα αντικείμενο μηνύματος είναι έτοιμο για κατανάλωση ή μέχρι να παρέλθει η περίοδος λήξης (που έχει δοθεί ως όρισμα σε δευτερόλεπτα), οπότε η τιμή επιστροφής είναι None. Το αντικείμενο msg που επιστρέφεται από την μέθοδο poll() αν έχει τιμή, αυτή θα περιέχει το μήνυμα που καταναλώθηκε ή κάποιο σφάλμα το οποίο συναντήθηκε. Με την μέθοδο error() του αντικειμένου msg εξετάζεται η ύπαρξη κάποιου σφάλματος. Μία συνηθισμένη κατάσταση η οποία συναντάται σε μορφή σφάλματος είναι η συνάντηση του τέλους του καναλιού μετά από την κατανάλωση κάποιου μηνύματος. Καθώς δεν συναντάται κάποιο σφάλμα η ροή εκτέλεσης συνεχίζεται με την παράδοση του μηνύματος στο module message_handling όπως έχει προαναφερθεί. Στην περίπτωση που η εφαρμογή ανιχνεύσει κάποιο σήμα για τον τερματισμό της, τότε η ροή εκτέλεσης θα βγει από τον ατέρμον βρόχο και θα καλεστούν οι συναρτήσεις για την έξοδο και τον τερματισμό της εφαρμογής.

5.2 Παρουσίαση ενός μοντέλου και της ανάπτυξη του.

Σε αυτή την ενότητα θα παρουσιαστεί η ανάπτυξη ενός μοντέλου για την πρόβλεψη του ρυθμού κατανάλωσης της κύριας μηχανής ενός πλοίου. Θα παίρνει ως είσοδο τα εξής σήματα:

- Ταχύτητα του πλοίου σχετικά με το έδαφος
- Ταχύτητα του αέρα. (Η ταχύτητα του αέρα μπορεί να δοθεί είτε σχετικά με το πλοίο είτε σχετικά με το έδαφος)
- Η κατεύθυνση του αέρα. (Η κατεύθυνση του αέρα μπορεί να δοθεί είτε σχετικά με το πλοίο είτε σχετικά με το έδαφος)
- Το ύψος των κυμάτων και των φουσκωμάτων της θάλασσας.
- Τα βυθίσματα του πλοίου.

Παρακάτω περιγράφεται ο τρόπος με τον οποίο η λειτουργικότητά του έχει αναπτυχθεί υπό το απαιτούμενο από την εφαρμογή interface.

```
"""
Created on 2018-10-23T09:29:30.924Z

@author: Isaias.Vrakidis
"""

import numpy as np
import eif
from math import sqrt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    explained_variance_score, mean_absolute_error,
    mean_squared_error, mean_squared_log_error,
    median_absolute_error, r2_score
)
from sklearn.linear_model import Lasso
from pandas import DataFrame, Series
```

Αρχικά γίνονται import όλα οι συναρτήσεις και κλάσεις των βιβλιοθηκών που θα χρησιμοποιηθούν. Βασικές βιβλιοθήκες που χρησιμοποιούνται είναι η scikit-learn για τις υλοποιήσεις των αλγορίθμων μηχανικής μάθησης κ.α., η numpy για αποδοτικές πράξεις σε πίνακες δεδομένων και η pandas για τις αποδοτικές δομές δεδομένων που προσφέρει.

```
class Demo_Model_414:

    def __init__(self):
        self.y_qid = '3467780526351061192'
        self.train_input_qids = {
            'relative_wind_speed': '2314854621547480456',
            'relative_wind_angle': '2314854621547656584',
            'wave_height_MB': '4647719228531150856',
            'swell_height_MB': '4647719228531154952',
            'over_ground_speed_GPS': '2360172082060464136',
            'draft_aft': '3467776115418210312',
            'draft_fore': '3467776115418206216',
            'draft_midP': '3467776115418214408',
            'draft_midS': '3467776115418218504'
        }

        self.x_feature_ordering = ('relative_wind_speed',
                                   'wave_height_MB',
                                   'swell_height_MB',
                                   'over_ground_speed_GPS',
```

```
                    'over_ground_speed_2_GPS',
                    'draft_midS',
                    'wind_speed_relative_x',
                    'wind_speed_relative_y',
                    'wind_speed_true_x',
                    'wind_speed_true',
                    'trim',
                    'trim_ver')
```

Ακολουθεί ο constructor της κλάσης που ορίζεται, στον οποίο αρχικοποιούνται τα πεδία y_qid, train_input_qids και x_feature_ordering. Το δύο πρώτα περιέχουν τα αναγνωριστικά της ποσότητας πρόβλεψης και των ποσοτήτων εισόδου της εκπαίδευσης ως μία αντιστοίχηση με κάποια κλειδιά που περιγράφουν ονομαστικά τις ποσότητες. Το τελευταίο πεδίο διατηρεί μία σειρά με τα ονομαστικά κλειδιά των τελικών features που θα προκύψουν και θα χρησιμοποιηθούν, ώστε να επιβάλετε κάθε φορά στις κολόνες των πινάκων πριν την είσοδο τους στα μοντέλα.

```
def get_grid_search(self):
    pipe = Pipeline(steps=[('scaler', StandardScaler()), ('rgr',
       Lasso(verbose=1))])
    param_grid = {
        'rgr__alpha': [0.05, 0.3, 0.5, 0.7, 0.9],
    }
    return GridSearchCV(pipe, param_grid, scoring='
       neg_mean_squared_error', n_jobs=-1,
                        verbose =1, iid=False, cv=10,
                           return_train_score=False)
```

Η παραπάνω συνάρτηση επιστρέφει ένα αντικείμενο GridSearchCV της βιβλιοθήκης scikit-learn στο οποίο ορίζεται ένα πλέγμα υποψήφιων αλγορίθμων και παραμέτρων αυτών. Στην συνέχει μέσω του αντικειμένου αυτού μπορεί να πραγματοποιηθεί μία αναζήτηση στο πλέγμα, ώστε να βρεθεί ο συνδυασμός που αποδίδει καλύτερα. Στην δικιά μας περίπτωση έχουν οριστεί δύο βήματα, ένα βήμα για την κοινωνικοποίηση των δεδομένων και ένα βήμα με τον αλγόριθμο Lasso Regression για την παραγωγή προβλέψεων. Η παράμετρος για την οποία γίνεται αναζήτηση της τιμής της που καταλήγει σε καλύτερα αποτελέσματα είναι η παράμετρος alpha του αλγορίθμου Lasso Regression.

```
def extract_features_t(self, df):
    print('Extracting features from df: {}'.format(df.shape))
    draft_aft = self.train_input_qids['draft_aft']
    draft_fore = self.train_input_qids['draft_fore']
    draft_midS = self.train_input_qids['draft_midS']
    draft_midP = self.train_input_qids['draft_midP']
    relative_wind_angle = self.train_input_qids['relative_wind_angle'
       ]
    relative_wind_speed = self.train_input_qids['relative_wind_speed'
       ]
```

```
        over_ground_speed_GPS = self.train_input_qids['
            over_ground_speed_GPS']

        df['over_ground_speed_2_GPS'] = np.square(df[
            over_ground_speed_GPS])
        df['trim'] = (df[draft_aft]-df[draft_fore])
        df['trim_ver'] = (df[draft_midS]-df[draft_midP])
        df['wind_speed_relative_x'] = df[relative_wind_speed] * \
            np.cos(np.radians(df[relative_wind_angle]))
        df['wind_speed_relative_y'] = df[relative_wind_speed] * \
            np.sin(np.radians(df[relative_wind_angle]))
        df['wind_speed_true_x'] = df['wind_speed_relative_x']+df[
            over_ground_speed_GPS]
        df['wind_speed_true'] = np.sqrt(df['wind_speed_true_x']**2 + df['
            wind_speed_relative_y']**2)

        print('Extraction ended. df: {}'.format(df.shape))
        return df
```

Η παραπάνω συνάρτηση παίρνει ως όρισμα ένα pandas.DataFrame το οποίο θα πρέπει να περιέχει μέσα δεδομένα για την εκπαίδευσή του μοντέλου. Εκτελεί σε αυτό διάφορους μετασχηματισμούς στα δεδομένα αυτά παράγοντας νέες στήλες στο DataFrame. Οι νέες στήλες που παράγονται είναι οι συνιστώσες της ταχύτητας του αέρα στους άξονες του πλοίου, το τετράγωνο της ταχύτητας του πλοίου και η διαφορές στα βυθίσματά του μεταξύ των ακρών του. Σημειώνεται ότι αυτή η συνάρτηση διαμορφώνει μόνο τα δεδομένα που προορίζονται για εκπαίδευση καθώς αυτά είναι συγκεκριμένα. Τα δεδομένα τα οποία θα χρησιμοποιηθούν για να γίνει πρόβλεψη μπορεί να διαφέρουν σε κάποια σημεία αλλά θα μετασχηματιστούν και αυτά στην ίδια μορφή. Για παράδειγμα ενώ η ταχύτατα του αέρα που θα χρησιμοποιηθεί για την εκπαίδευση είναι σχετική με το πλοίο, η ταχύτητα του αέρα που δίνεται για πρόβλεψη μπορεί να είναι και σχετική με το έδαφος.

```
    def extract_features(self, df):
        draft_aft = self.train_input_qids['draft_aft']
        draft_fore = self.train_input_qids['draft_fore']
        draft_midS = self.train_input_qids['draft_midS']
        draft_midP = self.train_input_qids['draft_midP']
        relative_wind_angle = self.train_input_qids['relative_wind_angle'
            ]
        relative_wind_speed = self.train_input_qids['relative_wind_speed'
            ]
        over_ground_speed_GPS = self.train_input_qids['
            over_ground_speed_GPS']
        true_wind_speed_MB = '4647719228525397320'
        true_wind_speed_A = '2314854621547480392'

        draft_qids = [draft_midP, draft_midS, draft_fore, draft_aft]
```

```
cont = list()
rest = list()
for d in draft_qids:
    if d in df:
        cont.append(d)
    else:
        rest.append(d)
for r in rest:
    df[r] = df[cont[0]]

if true_wind_speed_MB in df or true_wind_speed_A in df:
    df['over_ground_speed_2_GPS'] = np.square(df[
       over_ground_speed_GPS])
    if true_wind_speed_MB in df:
        df['wind_speed_true'] = df[true_wind_speed_MB]
    elif true_wind_speed_A in df:
        df['wind_speed_true'] = df[true_wind_speed_A]
    df['trim'] = (df[draft_aft]-df[draft_fore])
    df['trim_ver'] = (df[draft_midS]-df[draft_midP])

    sin_a = (df[over_ground_speed_GPS]/df['wind_speed_true']) * \
        np.sin(np.radians(df[relative_wind_angle]))
    # sin_a must be in range of [-1, 1], if thats not the case
       that means
    # that this instance of data is invalid couse a triangle
       between true
    # wind speed, relative wind speed and vessel speed cant be
       formed.
    df = df.loc[(sin_a <= 1) & (sin_a >= -1)]
    sin_a = sin_a.loc[(sin_a <= 1) & (sin_a >= -1)]
    c = 180 - df[relative_wind_angle] - np.degrees(np.arcsin(
       sin_a))
    sin_c = np.sin(np.radians(c))
    df[relative_wind_speed] = (sin_c/sin_a)*df[
       over_ground_speed_GPS]
    self.handle_special_cases(df, Vwr=relative_wind_speed, Vg=
       over_ground_speed_GPS,
        Vwt='wind_speed_true', Ywr=relative_wind_angle)

    df['wind_speed_relative_x'] = df[relative_wind_speed] * \
        np.cos(np.radians(df[relative_wind_angle]))
    df['wind_speed_relative_y'] = df[relative_wind_speed] * \
        np.sin(np.radians(df[relative_wind_angle]))
    df['wind_speed_true_x'] = df['wind_speed_relative_x']+df[
       over_ground_speed_GPS]
```

```
    elif relative_wind_angle in df:
        df = self.extract_features_t(df)

    for name in self.x_feature_ordering:
        if name not in df:
            df[name]=df[self.train_input_qids[name]]
    return df
```

Η παραπάνω συνάρτηση διαχειρίζεται και μετασχηματίζει τον πίνακα των δεδομένων που προορίζεται για είσοδο κάποιας πρόβλεψης. Αυτό σημαίνει ότι πρώτα πραγματοποιεί διάφορους ελέγχους για να εντοπίσει το είδος των δεδομένων και ανάλογα θα εφαρμόσει τον αντίστοιχο μετασχηματισμό.

Ακολουθεί η μέθοδος για την εκπαίδευση του μοντέλου.

```
def train(self, df):
    # Filtering dataframe from NaN values.
    df.dropna(how='any', inplace=True)
    if len(df) <= 1000:
        raise NotEnoughData('No Data')

    # Split dataframe into test and train set.
    train, test = train_test_split(df, test_size=0.2)
    test = self.extract_features_t(test)
    X_test = test.drop(self.y_qid, axis=1)
    y_test = test[self.y_qid]
    # Applying correct collumn ordering
    X_test = X_test.loc[:, self.get_train_keys(list(self.
        x_feature_ordering))]

    # Filter outliers
    thr=0.55
    ifr = eif.iForest(train.values, ntrees=100, sample_size=500,
        ExtensionLevel=train.shape[1]-1)
    scores = Series(data=ifr.compute_paths(X_in=train.values), index=
        train.index.values)
    train = train.loc[scores<thr]
    train = self.extract_features_t(train)
    X_train = train.drop(self.y_qid, axis=1)
    y_train = train[self.y_qid]
    # Applying correct collumn ordering and keep the selected
        features.
    X_train = X_train.loc[:, self.get_train_keys(list(self.
        x_feature_ordering))]
    print('Starting the GS with train set as {}'.format(X_train.shape
        ))
    gs = self.get_grid_search()
```

```
        gs.fit(X=X_train, y=y_train)
        print('Starting the training of best_estimator_: {}'.format(gs.
            best_params_))
        self.model = gs.best_estimator_
        self.model.fit(X=X_train, y=y_train)
        # Evaluate the model.
        y_pred = self.model.predict(X=X_test)
        # assign None to meanSquaredLogError if thera are negative values
            in vectors.
        if (y_pred <= 0).any(axis=0) or (y_test <= 0).any(axis=0):
            _mean_squared_log_error = None
        else:
            _mean_squared_log_error = mean_squared_log_error(y_test,
                y_pred)

        _mean_squared_error = mean_squared_error(y_test, y_pred)
        return {
            'explainedVarianceScore': explained_variance_score(y_test,
                y_pred),
            'meanAbsoluteError': mean_absolute_error(y_test, y_pred),
            'meanSquaredError': _mean_squared_error,
            'rootMeanSquaredError': sqrt(_mean_squared_error),
            'meanSquaredLogError': _mean_squared_log_error,
            'medianAbsoluteError': median_absolute_error(y_test, y_pred),
            'r2Score': r2_score(y_test, y_pred)
        }
```

Αρχικά φιλτράρονται όλες οι γραμμές που έχουν κενές τιμές και ελέγχεται αν ο αριθμός αυτών που έμειναν επαρκεί για να γίνει η εκπαίδευση. Στην συνέχεια, χωρίζουμε το dataset σε δύο σύνολα. Το ένα θα χρησιμοποιηθεί για τις εκπαιδεύσεις των υποψήφιων μοντέλων και θα περιέχει το 80% των δεδομένων και το άλλο για την αξιολόγηση τους. Για το δεύτερο σύνολο ακολουθείται η εξής διαδικασία, η οποία θα ακολουθηθεί και για το σύνολο εκπαίδευσης αμέσως μετά. Αρχικά υπολογίζονται όλες οι νέες στήλες με την συνάρτηση extract_features_t(). Στην συνέχεια χωρίζεται η στήλη της ποσότητας πρόβλεψης από το υπόλοιπο σύνολο και κρατούνται τα features που έχουν επιλεχθεί με την επιθυμητή σειρά των στηλών τους στον πίνακα. Για το σύνολο εκπαίδευσης πριν εφαρμοστεί η παραπάνω διαδικασία θα καθαριστεί πρώτα από outliers με τον αλγόριθμο Extended Isolation Forest.

Αφού έχουν ετοιμαστεί τα δύο αυτά σύνολα δεδομένων θα δημιουργηθεί το αντικείμενο για την εφαρμογή του grid search όπως περιγράφηκε παραπάνω. Μόλις ολοκληρωθεί η αναζήτηση του καλύτερου μοντέλου αυτό θα ανατεθεί στο πεδίο της κλάσης self.model, ώστε να μείνει καρφιτσωμένο στο αντίστοιχο αντικείμενο. Εκτελείται η εκπαίδευσή του πάνω στο σύνολο των δεδομένων εκπαίδευσης και αξιολογείται με την χρήση του συνόλου αξιολόγησης. Τα αποτελέσματα της αξιολόγησης συγκεντρώνονται σε ένα python dictionary και επιστρέφονται από την μέθοδο.

```
    def predict(self, df):
        # Filtering dataframe from NaN values.
```

```
        df.dropna(how='any', inplace=True)
        df = df[~df.isin(['NaN', 'NaT']).any(axis=1)]
        if len(df) <= 0:
            raise NotEnoughData('No Data')

        df = self.extract_features(df)
        # Applying correct collumn ordering
        df = df.loc[:, self.get_train_keys(list(self.x_feature_ordering))
           ]
        return Series(data=self.model.predict(X=df.values), index=df.
           index)
```

Παραπάνω παρουσιάζεται η μέθοδος για την πρόβλεψη. Τα βήματα που ακολουθεί είναι, όπως και στην εκπαίδευση, το φιλτράρισμα από τις κενές τιμές, ο έλεγχος της επάρκειας των δεδομένων και ο υπολογισμός των features. Για τον υπολογισμό των features χρησιμοποιείται η μέθοδος extract_features η οποία είναι σχεδιασμένη για να διαχειρίζεται όλους τους πιθανούς συνδυασμούς στηλών στο αρχικό σύνολο δεδομένων. Στην συνέχεια, κρατούνται τα features που έχουν επιλεχθεί με την επιθυμητή σειρά των στηλών τους στον πίνακα και χρησιμοποιείται το πεδίο self.model για την εκτέλεση της πρόβλεψης.

```
    def evaluate(self, df):
        # Filtering dataframe from NaN values.
        df.dropna(how='any', inplace=True)
        df = df[~df.isin(['NaN', 'NaT']).any(axis=1)]
        if len(df) <= 0:
            raise NotEnoughData('No Data')
        df = self.extract_features(df)

        y_test = df[self.y_qid]
        x_test = df.drop(self.y_qid, axis=1)
        # Applying correct collumn ordering
        x_test = x_test.loc[:, list(self.x_feature_ordering)]
        # Evaluate the model.
        y_pred = self.model.predict(X=x_test)
        # assign None to meanSquaredLogError if thera are negative values
            in vectors.
        if (y_pred <= 0).any(axis=0) or (y_test <= 0).any(axis=0):
            _mean_squared_log_error = None
        else:
            _mean_squared_log_error = mean_squared_log_error(y_test,
               y_pred)

        _mean_squared_error = mean_squared_error(y_test, y_pred)
        return {
            'explainedVarianceScore': explained_variance_score(y_test,
               y_pred),
```

```
        'meanAbsoluteError': mean_absolute_error(y_test, y_pred),
        'meanSquaredError': _mean_squared_error,
        'rootMeanSquaredError': sqrt(_mean_squared_error),
        'meanSquaredLogError': _mean_squared_log_error,
        'medianAbsoluteError': median_absolute_error(y_test, y_pred),
        'r2Score': r2_score(y_test, y_pred)
    }
```

Σε απόλυτη αντιστοιχία με τις μεθόδους της πρόβλεψης και της εκπαίδευσης αναπτύσσεται και η μέθοδος για την αξιολόγηση του μοντέλου η οποία παρουσιάζεται παραπάνω.

Κεφάλαιο 6

Προσομοίωση Πλατφόρμας

6.1 Περιγραφή Προσομοίωσης

Σε αυτή την ενότητα θα περιγραφεί η διαδικασία με την οποία πραγματοποιήθηκε μία προσομοίωση της λειτουργίας της πλατφόρμας. Η προσομοίωση έχει σκοπό την παρουσίαση της λειτουργίας της πλατφόρμας και της λειτουργικότητας. Θα αναδειχτεί πως μέσα από τις περιοδικές επανεκπαιδεύσεις των μοντέλων αποφεύγεται η απόκλιση της απόδοσής τους. Με αυτόν τον τρόπο εκθειάζονται τα πλεονεκτήματα που προσφέρει η χρήση της πλατφόρμας σε σχέση με την λειτουργία κάποιου μοντέλου χωρίς τις επανεκπαιδεύσεις.

Για τις ανάγκες της προσομοίωσης Θα χρησιμοποιηθούν δεδομένα που έχουν συλλεγεί από ένα πλοίο σε διάρκεια 10 περίπου μηνών. Το πλοίο από το οποίο θα συλλεγούν τα δεδομένα είναι ένα πετρελαιοφόρο (Crude Oil Tanker) ηλικίας 14 ετών. Μέσω της προσομοίωσης θα παραχθούν τα αποτελέσματα που θα είχε επιφέρει η χρήσης της πλατφόρμας και θα συγκριθούν με τα αντίστοιχα αποτελέσματα μοντέλων μηχανικής μάθησης που θα λειτουργούσαν εκτός της πλατφόρμας. Το μοντέλο που θα χρησιμοποιηθεί θα είναι αυτό που περιγράφηκε στην προηγούμενη ενότητα.

Τα σύνολα δεδομένων που θα χρησιμοποιηθούν για τις εκπαιδεύσεις θα περιέχουν δεδομένα από διαστήματα τεσσάρων μηνών. Θα πραγματοποιηθούν 4 εκπαιδεύσεις του μοντέλου με την κάθε μία να απέχει από την προηγούμενη 2 μήνες. Δηλαδή ξεκινώντας την πρώτη εκπαίδευση με δεδομένα από το διάστημα 1/12/18 - 31/3/19 ή επόμενες θα είναι στα διαστήματα 1/2/19 - 31/5/19, 1/4/19 - 31/7/19, 1/6/19 - 30/9/19 αντίστοιχα.

Στην συνέχεια θα συγκριθούν τα αποτελέσματα που φέρνει το μοντέλο σε κάθε νέα εκπαίδευση με το μοντέλο της πρώτης εκπαίδευσης. Για την σύγκριση των αποτελεσμάτων θα πραγματοποιηθούν αξιολογήσεις για το κάθε μοντέλο. Γι αυτές θα χρησιμοποιηθούν δεδομένα του διαστήματος των 10 επόμενων ημερών από το τέλος της κάθε περιόδου εκπαίδευσης. Το αρχικό μοντέλο θα αξιολογηθεί τρεις φορές πάνω στα διαστήματα των δέκα ημερών μετά τις περιόδους εκπαίδευσης των τριών τελευταίων εκδόσεων, οι οποίες θα αξιολογηθούν μία φορά με δεδομένων των διαστημάτων αμέσως μετά την εκπαίδευσή τους. Με αυτόν τον τρόπο θα συγκρίνεται κάθε φορά η απόδοση που επιτυγχάνεται εντός της Πλατφόρμας αμέσως μετά από κάποια επανεκπαίδευση με την απόδοση του αρχικού μοντέλου πάνω στα ίδια δεδομένα.

Τέλος, κάνοντας χρήση της υπηρεσία της σύγκρισης μοντέλων θα παρακολουθηθεί η εξέλιξη της σχέσης της κατανάλωσης του πλοίου, με την ταχύτητά του ως προς το έδαφος. Δίνοντας σταθερές

τιμές για την ταχύτητα και κατεύθυνση του αέρα, το ύψος των κυμάτων και των φουσκωμάτων της θάλασσας και το βάθος του πλοίου και για τιμές εντός ενός εύρος τιμών για την ταχύτητα του πλοίου σε σχέση με το έδαφος θα εξεταστεί ο ρυθμός κατανάλωσης που προκύπτει από το κάθε μοντέλο. Η σύγκριση θα πραγματοποιηθεί οπτικά, δημιουργώντας γραφήματα με τις καμπύλες που προκύπτουν από τα σημεία εντός του εύρους τιμών για την ταχύτητα του πλοίου και τα αντίστοιχα της κατανάλωσης. Έτσι θα αναδειχτεί η μορφή της σχέσης των δύο αυτών παραμέτρων του πλοίου και πως αυτή μεταβάλλεται.

6.2 Παρουσίαση αποτελεσμάτων

Αρχικά, συγκρίνεται η απόδοση της πρώτης έκδοσης του μοντέλου με κάθε μία από τις υπόλοιπες τρεις εκδόσεις του. Για τις αξιολογήσεις, προκειμένου να γίνει η σύγκριση, χρησιμοποιούνται δεδομένα δέκα ημερών από το τέλος κάθε μίας περιόδου εκπαίδευσης των 3 τελευταίων εκδόσεων του μοντέλου. Έτσι, η κάθε έκδοση συγκρίνεται με την αρχική, με δεδομένα από τις επόμενες δέκα ημέρες από το τέλος της περιόδου εκπαίδευσής της.

Στον Πίνακα 6.1 παρουσιάζονται τα σφάλματα που μετρήθηκαν για την κάθε έκδοση, καθώς και οι διαφορά τους από αυτό της αρχικής έκδοσης. Η κάθε μία σύγκριση πραγματοποιείται με αξιολογήσεις για τις οποίες γίνεται χρήση των ίδιων δεδομένων. Επίσης στον πίνακα φαίνονται και οι περίοδοι με των οποίων τα δεδομένα μετρήθηκαν τα σφάλματα. Το σφάλμα που χρησιμοποιείται για την παρουσίαση είναι το Root Mean Squared Error.

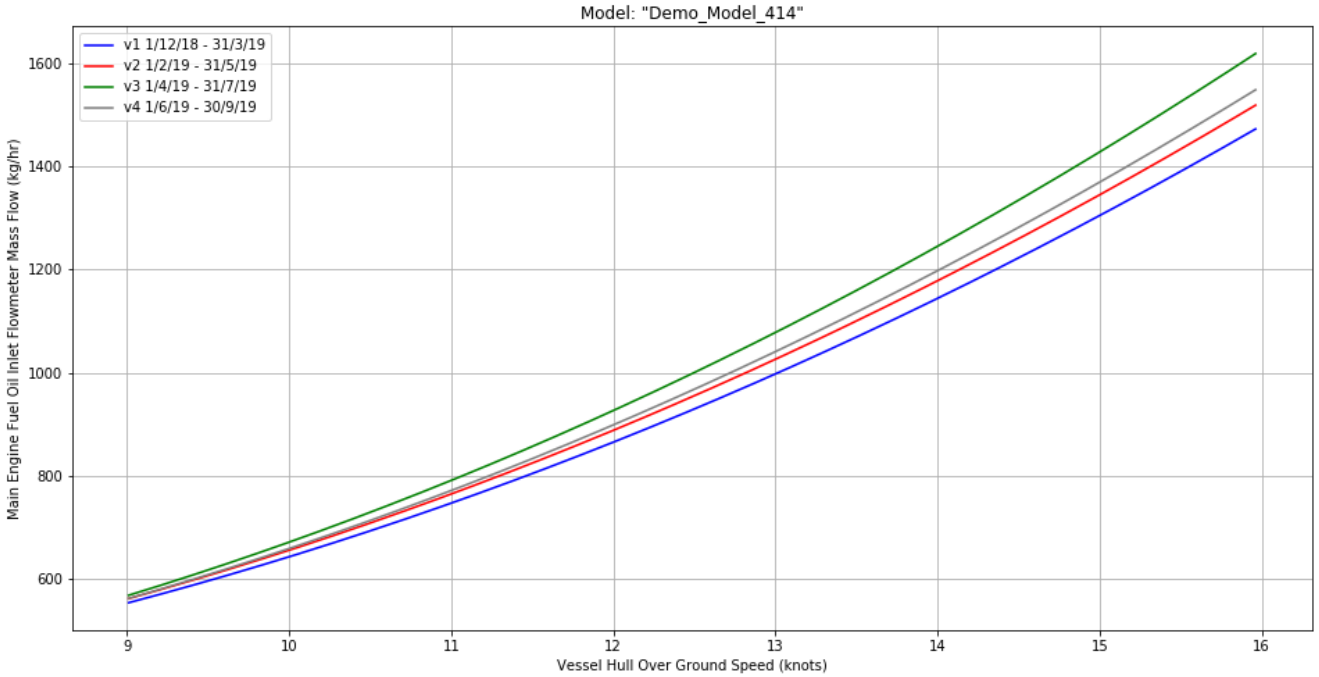
Πίνακας 6.1: Σύγκριση αποδοτικότητας των εκδόσεων του μοντέλου.

Έκδοση Μοντέλου	RMSE Αρχικής Έκδοσης	RMSE Έκδοσης	Διαφορά	Περίοδος Αξιολόγησης
2	343.72	220.58	123.14	1/6/19 - 10/6/19
3	459.03	198.68	260.35	1/8/19 - 10/8/19
4	621.66	319.12	302.54	1/10/19 - 10/10/19

Παρατηρείται ότι με το πέρας του χρόνου το αρχικό μοντέλο εμφανίζει όλο και μεγαλύτερα σφάλματα. Αντιθέτως οι εκδόσεις των μοντέλων που έχουν εκπαιδευτεί με τα πιο πρόσφατα δεδομένα που είναι διαθέσιμα στην κάθε σύγκριση εμφανίζουν σταθερά μικρότερο σφάλμα. Αξιοσημείωτη είναι και συνεχής αύξηση της διαφοράς στο σφάλμα μεταξύ της αρχικής έκδοσης και των υπολοίπων. Τα παραπάνω αποτελέσματα είναι αναμενόμενα, καθώς η κατάσταση του πλοίου μεταβάλλεται με τον καιρό. Αυτό σημαίνει ότι και οι συσχετίσεις μεταξύ των παραμέτρων του θα αλλάζουν, γεγονός το οποίο καθιστά τις μοντελοποιήσεις τους να γίνονται πιο ανακριβείς με το πέρασμα του χρόνου.

Καθώς οι διαφορετικές εκδόσεις του μοντέλου παρουσιάζουν και από μία παρελθοντική κατάσταση του πλοίου αυτές μπορούν να συγκριθούν μεταξύ τους, ώστε να εκθειάσουν τις μεταβολές στην κατάσταση του πλοίου. Στο Σχήμα 6.1 φαίνεται η μεταβολή της σχέσης, όπως αυτή έχει μοντελοποιηθεί, μεταξύ της ταχύτητας του πλοίου και του ρυθμού κατανάλωσης. Για να προκύψει μία δισδιάστατη γραμμή, καθώς η μοντελοποίηση υφίσταται σε έναν πολυδιάστατο χώρο, δίνονται σταθερές τιμές στην ταχύτητα και στην φορά του ανέμου, στο ύψος των κυμάτων και στο βύθισμα του

πλοίου. Οι τιμές αυτές έχουν διαλεχτεί, ώστε να αντιπροσωπεύουν ήπιες συνθήκες ταξιδιού. Συγκεκριμένα η ταχύτητα του άνεμου είναι στους 10 κόμβους με κατεύθυνσή αντίθετα στην κατεύθυνσή του πλοίου, το σημαντικό ύψος των κυμάτων και των φουσκωμάτων της θάλασσας είναι στα 1.5 μέτρα και το βύθισμα του πλοίου είναι στα 14 μέτρα. Τροφοδοτώντας το μοντέλο με αυτές τις τιμές συνεχώς και με τιμές για την ταχύτητα του πλοίου από 9 μέχρι 16 κόμβους προκύπτει μία καμπύλη και καθώς υπάρχουν 4 εκδόσεις το μοντέλου προκύπτουν 4 καμπύλες.



Σχήμα 6.1: Η μεταβολή της σχέσης ταχύτητας - κατανάλωσης.

Η κάθε γραμμή του σχήματος αντιστοιχεί σε μία έκδοση του μοντέλου, το οποίο φαίνεται και πάνω δεξιά στο Σχήμα όπου αναγράφεται και η περίοδος με της οποίας τα δεδομένα έχει εκπαιδευτεί το κάθε μοντέλο. Η κάθε έκδοση έχει εκπαιδευτεί με δεδομένα τεσσάρων μηνών και είναι μπροστά από την προηγούμενη κατά δύο μήνες. Αυτό σημαίνει ότι μέσω αυτών μπορεί να παρατηρηθεί η τάση που έχει η μεταβολή της κατάστασης του πλοίου από δίμηνο σε δίμηνο. Από τις καμπύλες που αντιστοιχούν στις 3 πρώτες εκδόσεις του μοντέλου, παρατηρείται ότι όσο περνάει ο καιρός, το πλοίο για να διατηρήσει την ίδια ταχύτητα χρειάζεται να καταναλώνει παραπάνω ποσότητα καυσίμου. Επίσης η διαφορά αυτή στον ρυθμό κατανάλωσης καυσίμου αυξάνεται αναλογικά με την ταχύτητα που έχει το πλοίο. Αυτό δικαιολογείται από το γεγονός της ανάπτυξης θαλάσσιων φυτών και οργανισμών στο κύτος του πλοίου με το πέρασμα του χρόνου, επιβαρύνοντας έτσι την απόδοσή του. Όσον αφορά την καμπύλη της τέταρτης έκδοσης, παρατηρείται ότι η αποδοτικότητα του πλοίου έχει αυξηθεί. Αυτό αποδίδεται στο γεγονός ότι στο συγκεκριμένο πλοίο καθαρίστηκε το κύτος του στις αρχές του μήνα Αυγούστου. Ως εκ τούτου το μοντέλο της τέταρτης έκδοσης έχει εκπαιδευτεί με δεδομένα από τα οποία τα μισά έχουν προκύψει εν όσο το πλοίο ήταν καθαρισμένο. Αυτό από την μία βλάπτει την ακρίβειά του σε έναν βαθμό, όπως αναφέρθηκε και παραπάνω, αλλά από την άλλη παρουσιάζει την τάση της μεταβολής της κατάστασης του πλοίου.

Κεφάλαιο 7

Συμπεράσματα - Επεκτάσεις

7.1 Συμπεράσματα

Όπως αναφέρθηκε και στα προηγούμενα κεφάλαια, όταν υπάρχει η ανάγκη δημιουργίας, λειτουργίας και συντήρησης πολλαπλών μοντέλων μηχανικής μάθησης, η εκτέλεση αυτών με ''χειρωνακτικό'' τρόπο είναι χρονοβόρα και καθώς ο αριθμός των μοντέλων αυξάνεται μπορεί να καταστεί και αδύνατη. Η πλατφόρμα μπορεί να επιλύσει το παραπάνω ζήτημα προσφέροντας την υποδομή για εύκολη δημιουργία και λειτουργία των μοντέλων σε περιβάλλον ''παραγωγής'', παρέχοντας τις κατάλληλες διεπαφές για την χρήση τους. Παράλληλα αυτοματοποιεί διαδικασίες συντήρησης των μοντέλων αφαιρώντας επιπλέον φόρτο εργασίας που θα χρειαζόταν να πραγματοποιηθεί χειροκίνητα. Λύσεις, όπως αυτή της πλατφόρμας που περιγράφηκε σε αυτή την εργασία, αποτελούν απαραίτητο στοιχείο σε έργα πληροφορικής που κάνουν ευρεία χρήση της μηχανικής μάθησης. Σε μια εποχή οπού τα δεδομένα αυξάνονται με τόσο μεγάλους ρυθμούς, η χειροκίνητη διαχείριση μοντέλων μηχανικής μάθησης θα οδηγήσει σε σημαντικές παραγωγικές δυσκολίες τον εκάστοτε οργανισμό που τα χειρίζεται.

Από τις λειτουργίες αυτές για την αυτόματη συντήρηση των μοντέλων μηχανικής μάθησης, μπορούν να αποβούν εξαιρετικά χρήσιμα αποτελέσματα. Όπως φάνηκε και από το παράδειγμα στην προσομοίωση της προηγούμενης υποενότητας, οι περιοδικές επανεκπαιδεύσεις των μοντέλων με τις οποίες επιτυγχάνεται η συντήρησή του, είναι απαραίτητες για την σωστή λειτουργία της Πλατφόρμας. Το περιβάλλον το οποίο μοντελοποιείται μπορεί να μεταβάλλεται, όπως στην παραπάνω περίπτωση η κατάσταση του πλοίου. Οι συσχετίσεις των παραμέτρων αλλάζουν με τρόπο που δεν μπορεί να προβλεφθεί από τα δεδομένα που έχουν συλλεχθεί και έχουν χρησιμοποιηθεί για την εκπαίδευση του μοντέλου. Αν παραμείνει σε λειτουργία το μοντέλο με την αρχική του εκπαίδευση η αποδοτικότητά του θα φθίνει συνεχώς. Παράλληλα, μπορεί να λάβουν χώρα γεγονότα, που δεν είναι ικανό να προβλεφθούν, τα οποία να προκαλέσουν απότομες αλλαγές στο περιβάλλον που δρα ένα μοντέλο. Στο παραπάνω παράδειγμα πραγματοποιήθηκε ένας καθαρισμός στο κύτος του πλοίου. Αυτό είχε ως αποτέλεσμα την απότομη αύξηση της απόδοσης του πλοίου και ως εκ τούτου την απότομη μεταβολή της σχέσης μεταξύ ταχύτητας και κατανάλωσης. Χωρίς συνεχείς επανεκπαιδεύσεις με τα πιο πρόσφατα δεδομένα, τέτοιου είδους μεταβολές θα καθιστούσαν το μοντέλο ανακριβές.

7.2 Επεκτάσεις

Με αφετηρία την πλατφόρμα όπως περιγράφηκε παραπάνω μπορεί να προκύψει πληθώρα επεκτάσεων προς διάφορες κατευθύνσεις. Αρχικά, ο τρόπος με τον οποίο έχει σχεδιαστεί η πλατφόρμα έχει ως βασικό πυλώνα την υποστήριξη πολλών γλωσσών προγραμματισμού και υλοποιήσεων αλγορίθμων στατιστικής επεξεργασίας και μηχανικής μάθησης. Δημιουργώντας εφαρμογές που μπορούν να υποστηρίξουν διασύνδεση με PostgreSQL και κατανάλωση μηνυμάτων από το Apache Kafka, μπορούν να χρησιμοποιηθούν ως μέρος της πλατφόρμας ανεξάρτητα από την γλώσσα προγραμματισμού με την οποία έχουν γραφτεί. Ενδιαφέρον παράδειγμα σε αυτήν την κατεύθυνση αποτελεί η χρήση της γλώσσας Scala και του framework Apache Spark. Το Apache Spark είναι ένα ανοικτού κώδικα γενικής χρήσεως σύστημα, για ταχεία εκτέλεση κατανεμημένων υπολογισμών σε συστάδες υπολογιστών [14]. Παρέχει μια διεπαφή για τον προγραμματισμό ολόκληρων των συστάδων υπολογιστών, με παράλληλη επεξεργασία δεδομένων και ανοχή σφαλμάτων. Με την χρήση ενός τέτοιου εργαλείου θα μπορούσαν να επιταχυνθούν σημαντικά οι διαδικασίες για τις εκπαιδεύσεις των μοντέλων, αλλά και οι υπόλοιπες καθώς η μεταφορά δεδομένων θα επιταχυνόταν και αυτή σημαντικά.

Ενδιαφέρον έχει και η χρήση που θα μπορούσε να έχει το πεδίο της Αυτόματης Μηχανικής Μάθησης. Πρόκειται για ένα αρκετά καινούριο πεδίο της μηχανικής μάθησης που απασχολεί αρκετά την επιστημονική και ερευνητική κοινότητα. Το πεδίο αυτό καταπιάνεται με το πρόβλημα της αυτόματης (χωρίς την μεσολάβηση ανθρώπων) παραγωγής μοντέλων μηχανικής μάθησης πάνω σε καινούρια σύνολα δεδομένων [15]. Η ανάπτυξη τέτοιων τεχνικών θα αποδέσμευε ολοκληρωτικά το ανθρώπινο δυναμικό ενός οργανισμού από την ανάπτυξη, συντήρησή κ.λ.π πολλαπλών μοντέλων μηχανικής μάθησης. Σε αυτήν την περίπτωση το μόνο που θα χρειαζόταν από τους χρήστες της πλατφόρμας θα ήταν να επιλέξουν μία παράμετρο του πλοίου ως ποσότητα πρόβλεψης και ένα σύνολο παραμέτρων για την είσοδο. Αυτό σημαίνει ότι οι χρήστες της πλατφόρμας, προκείμενου να δημιουργήσουν κάποιο καινούριο μοντέλο, δεν θα ήταν απαραίτητο να είναι καταρτισμένοι στο πεδίο της μηχανικής μάθησης. Αντίθετα η πλατφόρμα θα αναλάμβανε γι αυτούς να συγκεντρώσει τα δεδομένα και να επιλέξει τους αλγορίθμους που θα φέρουν τα καλύτερα αποτελέσματα.

Βιβλιογραφία

[1] ”METIS Cybertechnology Home page”. METIS Cybertechnology `https://metis.tech/` Retrieved 2019-10-05.

[2] Μιχάλης Κασσωτάκης, Γεώργιος Φλουρής (2013). Μάθηση και Διδασκαλία. Εκδόσεις Γρηγόρη.

[3] John R. KozaForrest H. BennettIIIDavid AndreMartin A. Keane, ”Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming”, 1996

[4] Gosling James, Joy Bill, Steele Guy, Bracha Gilad, Buckley Alex, ”The Java Language Specification”, March 2015.

[5] Rod Johnson , Juergen Hoeller , Keith Donald , Colin Sampaleanu , Rob Harrop , Thomas Risberg , Alef Arendsen , Darren Davison , Dmitriy Kopylenko , Mark Pollack , Thierry Templier , Erwin Vervaet , Portia Tung , Ben Hale , Adrian Colyer , John Lewis , Costin Leau , Mark Fisher , Sam Brannen , Ramnivas Laddad , Arjen Poutsma , Chris Beams , Tareq Abedrabbo , Andy Clement , Dave Syer , Oliver Gierke , Rossen Stoyanchev , Phillip Webb , Rob Winch , Brian Clozel , Stephane Nicoll , Sebastien Deleuze, ”Spring Framework Reference Documentation”, 2019.

[6] John V. Guttag, ”Introduction to Computation and Programming Using Python” The MIT Press, September 2012.

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay, ”Scikit-learn: Machine Learning in Python” Journal of Machine Learning Research, January 2012.

[8] Wes McKinney, ”pandas: a Foundational Python Library for Data Analysis and Statistics”, 2011.

[9] PostgreSQL Global Development Group, ”PostgreSQL: History”, March 2017.

[10] PostgreSQL Global Development Group, ”PostgreSQL 10.8 Documentation”, October 2017.

[11] David Bernstein, ”Containers and cloud: From lxc to docker to kubernetes” IEEE Cloud Computing, September 2014.

[12] ”kubernetes/kubernetes”. GitHub. `https://github.com/kubernetes/kubernetes/` Retrieved 2019-06-01.

[13] ”Directory of Azure Cloud Services”. Microsoft `https://azure.microsoft.com/en-us/services/` Retrieved 2019-06-05.

[14] ”Spark Overview”. Apache Software Foundation `https://spark.apache.org/docs/2.4.4/` Retrieved 2019-10-14.

[15] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, Frank Hutter (2016). Efficient and Robust Automated Machine Learning. Advances in Neural Information Processing Systems 28.