ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόβλεψη των μοτίβων μετακίνησης του πληθυσμού με ταξί, με χρήση Μεγάλων Δεδομένων και Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Φάντι Α. Σεχάντε

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόβλεψη των μοτίβων μετακίνησης του πληθυσμού με ταξί, με χρήση Μεγάλων Δεδομένων και Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Φάντι Α. Σεχάντε

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την _η Μαΐου 2017.

.....	.....	.....
Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου Αναπληρωτής Καθηγητής Ε.Μ.Π.	Γιώργος Στάμου Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2017

.....

Φάντι Α. Σεχάντε

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

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η δημιουργία ενός μοντέλου μηχανικής μάθησης για την περιγραφή και την πρόβλεψη της μετακίνησης του πληθυσμού της Νέας Υόρκης με χρήση των ταξί, βάση των καιρικών συνθηκών και της ημέρας της εβδομάδας.

Αρχικά, για τη δημιουργία των δεδομένων εκπαίδευσης του μοντέλου (συντεταγμένες, καιρικές συνθήκες, z-score) και την ανάλυση των μεγάλων δεδομένων εισόδου, υλοποιείται ο αλγόριθμος Getis Ord GI* σε Scala και Apache Spark. Τα δεδομένα αυτά συνδυάζονται με δεδομένα των καιρικών συνθηκών, και προκύπτει το σύνολο εκπαίδευσης του νευρωνικού δικτύου.

Στη συνέχεια, δημιουργούνται πολλές διαφορετικές αρχιτεκτονικές δικτύων βαθιάς μάθησης με χρήση του Tensorflow, τα οποία εκπαιδεύονται και αξιολογούνται. Επιλέγεται η αρχιτεκτονική με την καλύτερη σύγκλιση, και δοκιμάζονται και αξιολογούνται όλες οι συναρτήσεις βελτιστοποίησης και ενεργοποίησης. Τελικά επιλέγονται οι καταλληλότερες παράμετροι για την εκπαίδευση του δικτύου, και εκπαιδεύεται και αξιολογείται το δίκτυο βαθιάς μάθησης.

Τέλος δημιουργείται μία διαδικτυακή υπηρεσία και μια διεπαφή χρήστη για τον ευκολότερο πειραματισμό με το μοντέλο και τις προβλέψεις.

Λέξεις Κλειδιά: <<Μεγάλα Δεδομένα, Μηχανική Μάθηση, Βαθιά Μάθηση, Hotspot, Getis Ord GI*, Apache Spark, Tensorflow>>

Abstract

This diploma thesis tries to implement a machine learning model for the description and prediction of the transportation of the population of New York using the yellow taxi cabs, based on the weather conditions and the day of the week.

For the creation of the training dataset (coordinates, weather conditions, z-score) and the parsing of the big data input, the Getis Ord GI* algorithm was implemented in Scala and Apache Spark. The dataset was joined with the weather data, and the neural network training set was exported.

Multiple deep learning architectures where implemented and evaluated using the Tensorflow library. The architecture with the best convergence was chosen, and all the supported optimization and activation functions where tested and evaluated. The most suitable parameters where chosen, and the deep learning network was trained and evaluated.

Finally, a web service and a web interface where implemented to enable the easier experimentation with the model and the predictions.

Keywords: << Big Data, Machine Learning, Deep Learning, Hotspot, Getis Ord GI*, Apache Spark, Tensorflow >>

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον διδάκτορα Δημήτρη Σκούτα για την πολύτιμη βοήθεια του και τον χρόνο που δαπάνησε όλο αυτό το διάστημα με τις παροτρύνσεις και την καθοδήγησή του, καθώς και τον καθηγητή Νεκτάριο Κοζύρη που μου έδωσε την ευκαιρία να ασχοληθώ με το παρόν θέμα.

Θα ήθελα να ευχαριστήσω επίσης την οικογένεια μου που με στήριζε όλα αυτά τα χρόνια κατά τη διάρκεια των σπουδών μου, καθώς και τους φίλους μου και τη σύντροφό μου για την πολύτιμη στήριξη και βοήθειά τους.

Περιεχόμενα

Κεφάλαιο 1 – Εισαγωγή 1

1.1 Εισαγωγικά στοιχεία για Διαχείριση Μεγάλων Δεδομένων και Μηχανική Μάθηση 1

1.1.1 Ορισμός και βασικές τεχνικές ανάλυσης Μεγάλων Δεδομένων 1

1.1.2 Ορισμός Μηχανικής Μάθησης, βασικές έννοιες και τεχνικές 3

1.2 Σκοπός της διπλωματικής 4

1.2.1 Περιγραφή του προβλήματος 4

1.2.2 Συνεισφορά της διπλωματικής 5

Κεφάλαιο 2 – Σχετικές εργασίες και τεχνικό υπόβαθρο 6

2.1 Πλατφόρμες, εργαλεία και βιβλιοθήκες για διαχείριση μεγάλων δεδομένων και μηχανική μάθηση 6

2.2 Hot Spot Analysis - Ο αλγόριθμος Getis Ord GI* 8

2.3 Υπάρχουσες εργασίες σχετικές με το πρόβλημα 9

Κεφάλαιο 3 – Ανάλυση του προβλήματος 10

3.1 Η ανάγκη δημιουργίας μοντέλων πρόβλεψης και η χρησιμότητα τους 10

3.2 Η επιρροή των καιρικών συνθηκών στα μοτίβα μετακίνησης του πληθυσμού 10

Κεφάλαιο 4 - Επεξεργασία των δεδομένων και υπολογισμός των hotspots 12

4.1 Περιγραφή των δεδομένων 12

4.1.1 New York City Taxi & Limousine Commission Dataset 12

4.1.2 Global Historical Climate Network Dataset 12

4.2 Εγκατάσταση και παραμετροποίηση Apache Spark για την επεξεργασία των δεδομένων 13

4.3 Υλοποίηση του Getis Ord GI* σε Scala 14

4.4 Επιλογή παραμέτρων και δημιουργία τελικού dataset 15

Κεφάλαιο 5 - Υλοποίηση Μοντέλου Πρόβλεψης 16

5.1 Google Tensorflow 16

5.1.1 Εγκατάσταση και παραμετροποίηση 16

5.1.2 Η βιβλιοθήκη tf.contrib.learn 17

5.2 Αρχιτεκτονική του μοντέλου 18

5.3 Συναρτήσεις ενεργοποίησης 22

5.3.1 Συναρτήσεις ενεργοποίησης που υποστηρίζει το Tensorflow 22

5.3.2 Επιλογή της βέλτιστης συνάρτησης ενεργοποίησης 23

5.4. Αλγόριθμοι βελτιστοποίησης 24

5.4.1 Αλγόριθμοι που υποστηρίζει το Tensorflow 25

5.4.2 Επιλογή του βέλτιστου αλγορίθμου 27
5.5 Το πρόβλημα του Overfitting 28
5.5.1 Ορισμός Overfitting 28
5.5.2 Τρόπος αντιμετώπισης 28
5.6 Εκπαίδευση του νευρωνικού δικτύου και έλεγχος ακρίβειας 30
Κεφάλαιο 6 - Υλοποίηση διεπαφής χρήστη 31
6.1 Εργαλεία που χρησιμοποιήθηκαν 31
6.2 Περιγραφή διεπαφής χρήστη 31
Κεφάλαιο 7 – Παρουσίαση αποτελεσμάτων 33
Κεφάλαιο 8 - Συμπεράσματα - Μελλοντικές επεκτάσεις 36
Βιβλιογραφία 37
Παράρτημα 39
i. Πηγαίος Κώδικας - Επεξεργασία Δεδομένων 39
ii. Πηγαίος Κώδικας - Υλοποίηση Μοντέλου Πρόβλεψης 45
iii. Πηγαίος Κώδικας - Web Service 49
iv. Πηγαίος Κώδικας – Openlayers map 54

Κεφάλαιο 1 – Εισαγωγή

1.1 Εισαγωγικά στοιχεία για Διαχείριση Μεγάλων Δεδομένων και Μηχανική Μάθηση

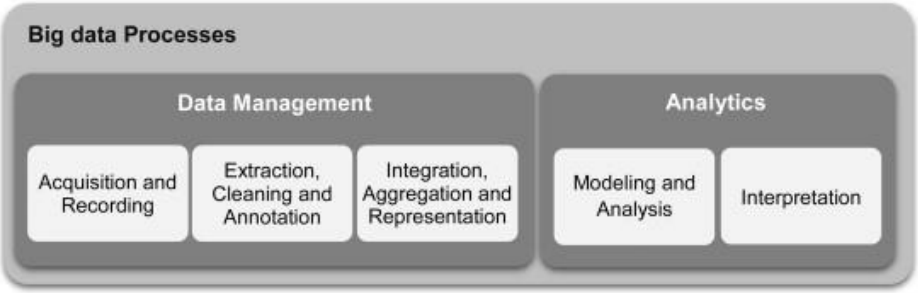
1.1.1 Ορισμός και βασικές τεχνικές ανάλυσης Μεγάλων Δεδομένων

Τα τελευταία χρόνια, ο όρος “Big Data” είναι ένας από τους πιο συχνά εμφανιζόμενους όρους σε επιστημονικά και τεχνολογικά άρθρα παγκοσμίως. Παρότι η χρήση του όρου είναι ευρέως διαδεδομένη, δεν συναντάται συμφωνία στην επιστημονική κοινότητα ως προς τον ακριβή ορισμό του όρου. Παρόλο που έχουν προταθεί διάφοροι ορισμοί, καμία από αυτές τις προτάσεις δεν έχει αποτρέψει μεταγενέστερες εργασίες από το να τροποποιήσουν και να εμπλουτίσουν τους προτεινόμενους ορισμούς [1].

Ο Laney (2011) [2] πρότεινε ότι ο Όγκος, η Ποικιλία και η Ταχύτητα (Volume, Variety, Velocity - 3 V’s) είναι οι τρεις διαστάσεις που χαρακτηρίζουν τα Μεγάλα Δεδομένα. Τα 3 V’s έχουν πλέον αναδειχθεί ως ένα κοινό πλαίσιο για την περιγραφή των Μεγάλων Δεδομένων. Ο Όγκος (Volume) αναφέρεται στο μέγεθος των δεδομένων. Τα μεγέθη των Μεγάλων Δεδομένων είναι συνήθως πολλά terabyte ή petabyte. Η Ποικιλία (Variety) αναφέρεται στη δομική ανομοιογένεια των δεδομένων. Τα δεδομένα μπορεί να είναι δομημένα ή αδόμητα. Παράδειγμα δομημένων δεδομένων είναι τα δεδομένα που αποτελούν μέρος μια σχεσιακής βάσης δεδομένων, ενώ τα αδόμητα μπορεί να είναι κείμενο, εικόνα, βίντεο, χωρική ή χρονική πληροφορία κλπ. Η Ταχύτητα (Velocity) αναφέρεται στο ρυθμό με τον οποίο δημιουργούνται τα δεδομένα, αλλά και την ταχύτητα με την οποία μπορούν να αναλυθούν και να υποστούν επεξεργασία [3].

Ο Dijcks το 2013 πρόσθεσε στον ορισμό και την Αξία (Value). Σύμφωνα με τον Dijcks η οικονομική αξία των διάφορων δεδομένων ποικίλλει σημαντικά. Συνήθως υπάρχει χρήσιμη πληροφορία κρυμμένη μέσα σε τεράστιο όγκο δεδομένων. Η πρόκληση είναι ο προσδιορισμός της πολύτιμης πληροφορίας, και η εξαγωγή και η μετατροπή των δεδομένων σε κατάλληλη μορφή για ανάλυση [4].

Τα Μεγάλα Δεδομένα δεν είναι χρήσιμα από μόνα τους, και αποκτούν αξία όταν μπορούν να αξιοποιηθούν για λήψη αποφάσεων. Για να καταστεί αυτό δυνατό, χρειάζονται αποτελεσματικές διαδικασίες για την μετατροπή μεγάλης ποσότητας ποικιλόμορφων δεδομένων σε ουσιαστική, χρήσιμη πληροφορία. Η διαδικασία εξαγωγής χρήσιμης πληροφορίας από Μεγάλα Δεδομένα μπορεί να χωριστεί σε πέντε στάδια (Εικόνα 1.1). Αυτά τα πέντε στάδια χωρίζονται σε δύο κύριες υποκατηγορίες: την διαχείριση των δεδομένων, και την ανάλυση. Η διαχείριση δεδομένων περιλαμβάνει τις διαδικασίες για την απόκτηση και την αποθήκευση των δεδομένων, την ανάκτησή τους, και την προετοιμασία τους για ανάλυση. Η ανάλυση αναφέρεται σε τεχνικές που χρησιμοποιούνται για την εξαγωγή χρήσιμης πληροφορίας από τα δεδομένα [3].



Σχήμα 1.1 – Διαδικασίες για εξαγωγή πληροφορίας από Big Data [3]

Λόγω της μεγάλης ποικιλομορφίας των Μεγάλων Δεδομένων και των προβλημάτων που επιχειρούν να λύσουν, χρησιμοποιούνται πολλές διαφορετικές τεχνικές διαχείρισης και ανάλυσης των δεδομένων, η αναλυτική περιγραφή των οποίων είναι αδύνατο να γίνει στα πλαίσια αυτής της διπλωματικής εργασίας. Παρακάτω θα αναλυθούν οι τεχνικές που χρησιμοποιήθηκαν για την επίλυση του συγκεκριμένου προβλήματος.

1.1.2 Ορισμός Μηχανικής Μάθησης, βασικές έννοιες και τεχνικές

Ο Arthur Samuel το 1959 όρισε την Μηχανική Μάθηση (Machine Learning) ως "το πεδίο μελέτης που δίνει στους υπολογιστές τη δυνατότητα να μάθουν, χωρίς να έχουν ρητά προγραμματιστεί" [5]. Ο Tom Mitchell το 1997 πρότεινε έναν πιο σαφή ορισμό: "Ένα πρόγραμμα υπολογιστή θεωρείται ότι μαθαίνει από μια εμπειρία E σε σχέση με μια εργασία T και κάποιο μέτρο απόδοσης P, αν η απόδοσή του στο T όπως μετριέται από το P, βελτιώνεται με την εμπειρία E" [6].

Για την εκπαίδευση χρησιμοποιούνται δυο βασικές κατηγορίες δεδομένων. Τα επισημασμένα δεδομένα (labeled data) αποτελούνται από παραδειγματικές εισόδους και επιθυμητά αποτελέσματα. Τα μη επισημασμένα δεδομένα (unlabeled data) αποτελούνται μόνο από τις παραδειγματικές εισόδους, χωρίς τα αποτελέσματα.

Υπάρχουν τρεις κύριες κατηγορίες μηχανικής μάθησης: Η επιβλεπόμενη μάθηση, η μη επιβλεπόμενη μάθηση, και η ημι-επιβλεπόμενη μάθηση.

Στην επιβλεπόμενη μάθηση χρησιμοποιούνται αποκλειστικά επισημασμένα δεδομένα (labeled data) για την εκπαίδευση και το πρόγραμμα κάνει προβλέψεις για δεδομένα που δεν έχει αντιμετωπίσει ξανά. Αυτό είναι το πιο συνηθισμένο σενάριο που σχετίζεται με προβλήματα ταξινόμησης, παλινδρόμησης και κατάταξης.

Στην μη επιβλεπόμενη μάθηση χρησιμοποιούνται αποκλειστικά μη επισημασμένα δεδομένα εκπαίδευσης (unlabeled data) και το πρόγραμμα κάνει προβλέψεις για δεδομένα που δεν έχει αντιμετωπίσει ξανά. Αφού δεν υπάρχουν επισημασμένα δεδομένα είναι δύσκολο να εκτιμηθεί ποσοτικά η απόδοση του προγράμματος. Συνηθισμένα παραδείγματα προβλημάτων μη επιβλεπόμενης μάθησης είναι τα προβλήματα ομαδοποίησης και μείωσης διάστασης.

Στην ημι-επιβλεπόμενη μάθηση χρησιμοποιείται μια μίξη επισημασμένων και μη επισημασμένων δεδομένων για την εκπαίδευση και το πρόγραμμα κάνει προβλέψεις για δεδομένα που δεν έχει αντιμετωπίσει ξανά. Η ημι-επιβλεπόμενη μάθηση χρησιμοποιείται συνήθως σε σενάρια όπου υπάρχουν εύκολα διαθέσιμα μη επισημασμένα δεδομένα, αλλά είναι δαπανηρή η απόκτηση επισημασμένων

δεδομένων. Προβλήματα ημι-επιβλεπόμενης μάθησης είναι συνήθως προβλήματα ταξινόμησης, παλινδρόμησης και κατάταξης όπου λόγω έλλειψης αρκετών διαθέσιμων επισημασμένων δεδομένων, επιχειρείται η βελτίωση της απόδοσης με χρήση μη επισημασμένων δεδομένων εκπαίδευσης [7].

1.2 Σκοπός της διπλωματικής

1.2.1 Περιγραφή του προβλήματος

Σκοπός της παρούσας διπλωματικής ήταν η δημιουργία ενός μοντέλου μηχανικής μάθησης για την περιγραφή και την πρόβλεψη της μετακίνησης του πληθυσμού της Νέας Υόρκης με χρήση των ταξί, βάση των καιρικών συνθηκών και της ημέρας της εβδομάδας.

Χρησιμοποιήθηκαν δεδομένα από κάθε διαδρομή ταξί που έλαβε χώρα στη Νέα Υόρκη το 2015, και συγκεκριμένα η ημερομηνία, οι συντεταγμένες του σημείου άφιξης, ο αριθμός των επιβατών που επέβαιναν, η μέση θερμοκρασία και ο υετός. Με τη χρήση αυτών των δεδομένων υπολογίστηκε η τιμής της χωρικής συγκέντρωσης των ταξί στη Νέα Υόρκη (hotspot) με χρήση του Getis Ord GI* και του εργαλείου μεγάλων δεδομένων Apache Spark, και αναπτύχθηκε και εκπαιδεύτηκε ένα μοντέλο μηχανικής μάθησης με χρήση του Tensorflow.

Το μοντέλο που αναπτύχθηκε, δεχόμενο ως είσοδο τις συντεταγμένες ενός σημείου, τις καιρικές συνθήκες και την ημέρα της βδομάδας επιχειρεί να προβλέψει τη χωρική συγκέντρωση των ταξί στο σημείο αυτό.

Οι λύσεις και οι επεκτάσεις του προβλήματος θα μπορούσαν να χρησιμοποιηθούν για την καλύτερη συγκοινωνιακή μελέτη και οργάνωση των πόλεων, καθώς επίσης και για ανθρωπολογική και συμπεριφοριολογική μελέτη της μετακίνησης του πληθυσμού μιας πόλης, και της επιρροής που έχει ο καιρός στις μετακινήσεις και τις συνήθειες του πληθυσμού.

Η έλλειψη έτοιμων εργαλείων για την χωρική και χρονική ανάλυση και επεξεργασία μεγάλων δεδομένων αποτέλεσε σημαντική δυσκολία και πρόκληση για την εκπόνηση της παρούσας εργασίας.

1.2.2 Συνεισφορά της διπλωματικής

Για την επίλυση του προβλήματος αναπτύχθηκαν εργαλεία ανάλυσης και επεξεργασίας χωροχρονικών δεδομένων, και δημιουργήθηκε μοντέλο μηχανικής μάθησης το οποίο αναγνωρίζει, επεξεργάζεται και προβλέπει χωροχρονική πληροφορία. Τα εργαλεία που αναπτύχθηκαν, καθώς και οι επεκτάσεις τους, θα μπορούσαν να χρησιμοποιηθούν για την μοντελοποίηση και ανάλυση αντίστοιχων χωροχρονικών προβλημάτων.

Η παρούσα διπλωματική εργασία έδειξε επίσης ότι οι καιρικές συνθήκες επηρεάζουν σε μεγάλο βαθμό την μετακίνηση του πληθυσμού με ταξί, και το μοντέλο που αναπτύχθηκε θα μπορούσε να χρησιμοποιηθεί για την περαιτέρω μελέτη του παραπάνω ισχυρισμού ή και την επέκτασή του σε άλλα μέσα μεταφοράς. Με χρήση διαφορετικών χωροχρονικών δεδομένων όπως π.χ. δεδομένων από check-in σε κοινωνικά μέσα δικτύωσης, και των τεχνικών που αναλύονται στην παρούσα διπλωματική θα μπορούσαν να αναπτυχθούν μοντέλα που να προβλέπουν και να αναλύουν την επιρροή των καιρικών συνθηκών όχι μόνο στις μετακινήσεις, αλλά και στον τρόπο διασκέδασης και ψυχαγωγίας του πληθυσμού.

Κεφάλαιο 2 – Σχετικές εργασίες και τεχνικό υπόβαθρο

2.1 Πλατφόρμες, εργαλεία και βιβλιοθήκες για διαχείριση μεγάλων δεδομένων και μηχανική μάθηση

Τα τελευταία χρόνια έχει αναπτυχθεί πληθώρα εργαλείων για την ανάλυση και διαχείριση μεγάλων δεδομένων καθώς και για τη μηχανική μάθηση, η πλειοψηφία των οποίων διατίθεται με άδειες ελεύθερου λογισμικού.

Κάποια από τα κυριότερα εργαλεία μεγάλων δεδομένων είναι:

- Apache Hadoop: Είναι μια βιβλιοθήκη ελεύθερου λογισμικού που επιτρέπει την κατανεμημένη επεξεργασία μεγάλων συνόλων δεδομένων σε συστοιχίες υπολογιστών με χρήση απλών μοντέλων προγραμματισμού. Έχει σχεδιαστεί για να μπορεί να κλιμακώνεται από έναν υπολογιστή σε χιλιάδες μηχανήματα, το κάθε ένα από τα οποία προσφέρει τοπική επεξεργασία και αποθήκευση. Αντί να βασίζεται στο υλικό για την παροχή υψηλής διαθεσιμότητας, η βιβλιοθήκη έχει σχεδιαστεί έτσι ώστε η ίδια να εντοπίζει και να διαχειρίζεται τις αστοχίες στο επίπεδο της εφαρμογής, παρέχοντας έτσι μια υπηρεσία υψηλής διαθεσιμότητας πάνω σε μια συστοιχία υπολογιστών, ο καθένας από τους οποίους μπορεί να είναι επιρρεπής σε αστοχίες [8].
- Apache Spark: Είναι μία πλατφόρμα ελεύθερου λογισμικού που παρέχει μια διεπαφή προγραμματισμού που βασίζεται σε μια δομή δεδομένων που ονομάζεται Resilient Distributed Dataset (RDD). Μπορεί να τρέξει έως 100 φορές πιο γρήγορα τα προγράμματα από το Hadoop MapReduce στη μνήμη, και έως 10 φορές πιο γρήγορα στον δίσκο [9].
- MongoDB: Είναι μια βάση δεδομένων ελεύθερου λογισμικού που παρέχει υψηλή απόδοση, υψηλή διαθεσιμότητα και αυτόματη κλιμάκωση. Κάθε εγγραφή είναι ένα έγγραφο, παρόμοιο με ένα JSON object [10].

Κάποια από τα κυριότερα εργαλεία μηχανικής μάθησης είναι:

- Tensorflow: Είναι μια βιβλιοθήκη ελεύθερου λογισμικού για αριθμητικούς υπολογισμούς χρησιμοποιώντας γράφους ροής δεδομένων. Οι κόμβοι στο γράφο αντιπροσωπεύουν μαθηματικές πράξεις, ενώ οι ακμές του γράφου αντιπροσωπεύουν τους πίνακες πολυδιάστατων δεδομένων (τανυστές – tensors) που μεταβιβάζονται μεταξύ τους. Το Tensorflow αρχικά αναπτύχθηκε από ερευνητές και μηχανικούς που εργάζονται στην Google Brain Team με σκοπό τη διεξαγωγή έρευνας στους τομείς της μηχανικής μάθησης και των νευρωνικών δικτύων [11].
- Theano: Είναι μια βιβλιοθήκη ελεύθερου λογισμικού σε Python που επιτρέπει τον αποδοτικό ορισμό, την βελτιστοποίηση και την αξιολόγηση μαθηματικών εκφράσεων σε πίνακες πολυδιάστατων δεδομένων [12].
- Caffe: Είναι ένα framework μηχανικής μάθησης το οποίο αναπτύχθηκε από το Berkeley Vision and Learning Center (BVLC) και μέλη της κοινότητας ελεύθερου λογισμικού [13].
- MLlib: Είναι η βιβλιοθήκη μηχανικής μάθησης του Apache Spark η οποία λειτουργεί σε συνεργασία με το NumPy στη Python και τις βιβλιοθήκες του R [14].
- Scikit-Learn: Είναι μια Python βιβλιοθήκη μηχανικής μάθησης ελεύθερου λογισμικού βασισμένη στα NumPy, SciPy και matplotlib [15].
- Torch: Είναι ένα framework μηχανικής μάθησης γραμμένο σε C++, το οποίο υλοποιεί μια εύκολη και γρήγορη scripting γλώσσα και είναι σχεδιασμένο για χρήση GPU για τις αριθμητικές πράξεις [16].

Από τα παραπάνω εργαλεία χρησιμοποιήθηκε το Apache Spark για την επεξεργασία των δεδομένων εισόδου και την εξαγωγή των δεδομένων εκπαίδευσης του νευρωνικού δικτύου, και το Tensorflow για τη δημιουργία του μοντέλου μηχανικής μάθησης και την εκπαίδευση και αξιολόγησή του.

2.2 Hot Spot Analysis - Ο αλγόριθμος Getis Ord GI*

Σε πολλά προβλήματα προκύπτει η ανάγκη της εύρεσης των περιοχών του χώρου που έχουν υψηλή ή χαμηλή συγκέντρωση του στοιχείου που μελετάμε (hotspot). Ένα συνηθισμένο πρόβλημα αυτού του είδους είναι η μελέτη της εγκληματικότητας μιας πόλης. Για μια τέτοια μελέτη, θα μπορούσαμε να χωρίσουμε την πόλη σε περιοχές, και να υπολογίσουμε την συγκέντρωση διαιρώντας τον αριθμό των εγκλημάτων με το μέγεθος της κάθε περιοχής.

Αν και αυτή η απλή μέθοδος δίνει πληροφορία για το που υπάρχει μεγάλη συγκέντρωση εγκλήματος, δεν καθορίζει πόσο στατιστικά σημαντικές είναι αυτές οι συγκεντρώσεις. Αυτό το πρόβλημα λύνει το Getis Ord GI* score, το οποίο είναι ένα z-score της συγκέντρωσης των τιμών των χαρακτηριστικών που μελετάμε.

Το Getis Ord GI* score [17] για κάθε χαρακτηριστικό (feature) i δίνεται από τον τύπο:

$$G_i^* = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{X}\sum_{j=1}^{n} w_{i,j}}{S\sqrt{\frac{n\sum_{j=1}^{n} w_{i,j}^2 - \left(\sum_{j=1}^{n} w_{i,j}\right)^2}{n-1}}}$$

όπου το x_j είναι η τιμή του χαρακτηριστικού j, $w_{i,j}$ είναι το χωρικό βάρος μεταξύ των χαρακτηριστικών i και j, n είναι το πλήθος των χαρακτηριστικών και:

$$\bar{X} = \frac{\sum_{j=1}^{n} x_j}{n}$$

$$S = \sqrt{\frac{\sum_{j=1}^{n} x_j^2}{n} - (\bar{X})^2}$$

Τα χωρικά βάρη $w_{i,j}$ υπολογίζονται συνήθως με χρήση της ανάστροφης ευκλείδειας απόστασης, αλλά μπορεί να χρησιμοποιηθεί οποιαδήποτε μέθοδος υπολογισμού της απόστασης κριθεί κατάλληλη για τις ανάγκες του εκάστοτε προβλήματος.

2.3 Υπάρχουσες εργασίες σχετικές με το πρόβλημα

Αν και ο κλάδος των μεγάλων δεδομένων και της μηχανικής μάθησης αναπτύσσεται πολύ γρήγορα τα τελευταία χρόνια, δεν υπάρχουν αρκετές μελέτες και εργαλεία για την ανάλυση χωρικών μεγάλων δεδομένων στο πρόβλημα που εστιάζει η συγκεκριμένη διπλωματική εργασία. Δεν βρέθηκαν υπάρχουσες εργασίες σχετικές με την εκπαίδευση δικτύων βαθιάς μάθησης με χωρικά δεδομένα ενώ πρόκληση αποτέλεσε και η εξαγωγή των hotspot αφού τα προγράμματα που παραδοσιακά χρησιμοποιούνται για τέτοιου είδους αναλύσεις είναι τα προγράμματα GIS, τα οποία όμως δεν έχουν τη δυνατότητα να επεξεργαστούν μεγάλα δεδομένα.

Ένα ενδιαφέρον εργαλείο υπό ανάπτυξη είναι το SparkProject που επιτρέπει τη σύνδεση του ArcGIS με το Apache Spark [18].

Επειδή για την εκπόνηση της παρούσας διπλωματικής εργασίας χρησιμοποιήθηκαν δεδομένα της τάξης των 50GB, υπήρξε η ανάγκη υλοποίησης του αλγορίθμου Getis Ord GI* με τρόπο που να επιτρέπει την εφαρμογή του σε μεγάλα δεδομένα.

Παρόμοιο πρόβλημα κλήθηκαν να λύσουν ερευνητές στα πλαίσια του διαγωνισμού GISCUP 2016, όπου ζητήθηκε να αναπτύξουν μια αποδοτική χωροχρονική υλοποίηση του αλγορίθμου Getis Ord GI* [19]. Οι λύσεις περιλάμβαναν διάφορες μεθόδους υλοποίησης και βελτιστοποίησης, όπως η χρήση μηχανής πεπερασμένων καταστάσεων (FSM) για την αποδοτική ανάλυση της κάθε γραμμής εισόδου και της εξαγωγής των δεδομένων [20], η χρήση ενός ακεραίου για την αναπαράσταση των τριών χωροχρονικών μεταβλητών (x, y, t) και ο ταυτόχρονος υπολογισμός του αθροίσματος της τιμής του κελιού και των γειτονικών κελιών του [21].

Κεφάλαιο 3 – Ανάλυση του προβλήματος

3.1 Η ανάγκη δημιουργίας μοντέλων πρόβλεψης και η χρησιμότητα τους

Τα μοντέλα πρόβλεψης είναι ένα εργαλείο που χρησιμοποιείται ευρέως στην έρευνα και την επιστήμη αλλά και από εταιρίες, κυβερνήσεις, οργανισμούς, πολιτικά κόμματα κ.α.

Οι μετεωρολόγοι χρησιμοποιούν μοντέλα πρόβλεψης για να προβλέψουν τον καιρό, τα πολιτικά κόμματα για να προβλέψουν και να επηρεάσουν την ψήφο των πολιτών, οι οικονομολόγοι για να προβλέψουν τη κίνηση των δεικτών στο χρηματιστήριο, οι διαφημιστικές εταιρίες για να προβλέψουν ποιες διαφημίσεις ενδιαφέρουν τον λήπτη, οι μηχανές αναζήτησης για να προβλέψουν τον όρο που θέλει να αναζητήσει ο χρήστης κ.ο.κ.

Ιδιαίτερα σημαντική και αναγκαία είναι η χρήση τους και στην ιατρική, αφού πλέον αναπτύσσονται μοντέλα που μπορούν να προβλέπουν την εξέλιξη μιας ασθένειας, ή τη δράση ενός φαρμάκου ή μιας μεθόδου αγωγής.

Παρόλο που τα μοντέλα πρόβλεψης δεν είναι ένα καινούργιο εργαλείο, τα τελευταία χρόνια αναπτύσσονται με γρήγορο ρυθμό λόγω της ραγδαίας αύξησης των διαθέσιμων δεδομένων και της ανάπτυξης αποδοτικών και εύκολων στη χρήση εργαλείων ανάλυσης μεγάλων δεδομένων και μηχανικής μάθησης.

3.2 Η επιρροή των καιρικών συνθηκών στα μοτίβα μετακίνησης του πληθυσμού

Η ανάπτυξη του κλάδου των μεγάλων δεδομένων, και η πληθώρα διαθέσιμης χωρικής πληροφορίας δίνει πλέον τη δυνατότητα στους ερευνητές να μελετούν με μεγαλύτερη ευκολία τα μοτίβα μετακίνησης του πληθυσμού, και πως επηρεάζουν διάφορα φυσικά φαινόμενα και συνθήκες τα μοτίβα αυτά.

Είναι γνωστό ότι οι καιρικές συνθήκες επηρεάζουν σε μεγάλο βαθμό την ανθρώπινη δραστηριότητα και πολλούς τομείς της οικονομίας. Κάποιοι τομείς βασίζονται σε μέσες συνθήκες, άλλοι σε ακραίες συνθήκες, και άλλοι είναι πολύ ευαίσθητοι σε απότομες αλλαγές. Για παράδειγμα, ένας από τους κύριους τομείς της οικονομίας που επηρεάζονται σε μεγάλο βαθμό από τις καιρικές συνθήκες είναι ο αγροτικός τομέας, ενώ είναι εύκολο να παρατηρήσουμε ότι ο καιρός επηρεάζει πολύ έντονα και τον κλάδο των μεταφορών (π.χ. τις μετακινήσεις στη θάλασσα).

Είναι εύκολο να παρατηρηθεί ότι οι καιρικές συνθήκες επηρεάζουν σε κάποιο βαθμό τις καθημερινές δραστηριότητες των ανθρώπων. Το 2013 οι Horanont et al, χρησιμοποιώντας δεδομένα από τα GPS των κινητών 31,855 κατοίκων του Τόκιο, έδειξαν ότι οι καθημερινές δραστηριότητες των κατοίκων του Τόκιο, επηρεάζονταν σημαντικά από την μέση θερμοκρασία και την ταχύτητα του ανέμου [22].

Εφόσον οι καιρικές συνθήκες επηρεάζουν σημαντικά τις καθημερινές δραστηριότητες του πληθυσμού, υποθέτουμε στα πλαίσια αυτής της διπλωματικής εργασίας ότι επηρεάζουν και τη μετακίνηση του με ταξί, και θα προσπαθήσουμε να μοντελοποιήσουμε αυτή την επιρροή με τη χρήση μηχανικής μάθησης. Έτσι θα δημιουργήσουμε ένα μοντέλο που δεχόμενο ως είσοδο τις συντεταγμένες ενός σημείου, την ημέρα της εβδομάδας, τη μέση θερμοκρασία και τον υετό, θα προβλέπει τη χωρική συγκέντρωση των ταξί σε εκείνο το σημείο.

Κεφάλαιο 4 - Επεξεργασία των δεδομένων και υπολογισμός των hotspots

4.1 Περιγραφή των δεδομένων

Για τη δημιουργία του μοντέλου χρησιμοποιήθηκαν δύο πηγές δεδομένων, που διατίθενται ελεύθερα στο διαδίκτυο.

4.1.1 New York City Taxi & Limousine Commission Dataset

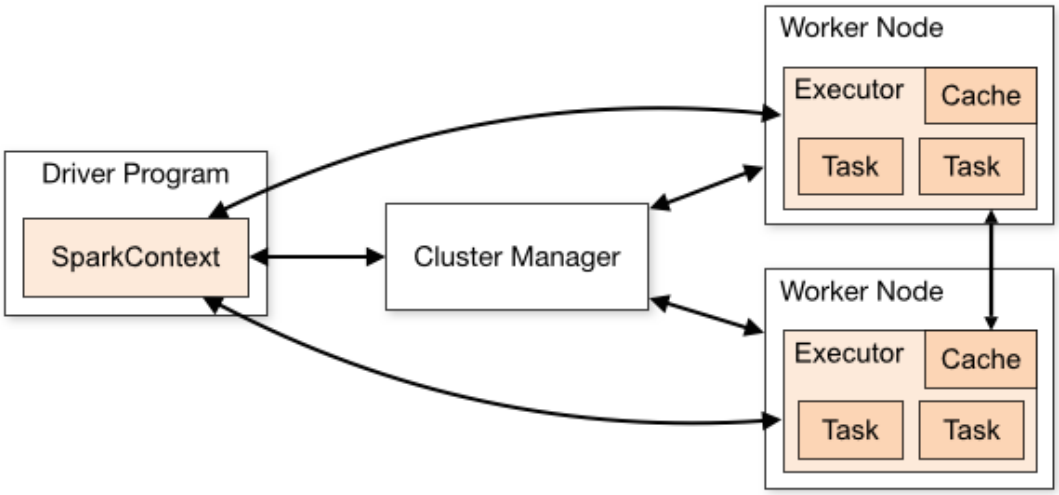
Πρόκειται για ένα σύνολο δεδομένων που διατίθεται ελεύθερα από τον επίσημο ιστότοπο της πόλης της Νέας Υόρκης σε μορφή csv (http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml), και περιέχει ανεπεξέργαστη πληροφορία για όλες τις διαδρομές των κίτρινων ταξί της Νέας Υόρκης, από το 2009 έως σήμερα. Οι πληροφορίες που παρέχονται για κάθε διαδρομή είναι οι συντεταγμένες και ο χρόνος αφετηρίας και προορισμού, ο αριθμός των επιβατών, η απόσταση και το κόστος της διαδρομής, το κόστος διοδίων, ο τρόπος πληρωμής, και τυχών επιπλέον χρεώσεις. Για την παρούσα εργασία χρησιμοποιήθηκαν δεδομένα για το 2015 και το 2016. Τα δεδομένα αποτελούνταν από 169,705,990 εγγραφές και είχαν συνολικό μέγεθος 24GB .

4.1.2 Global Historical Climate Network Dataset

Πρόκειται για ένα σύνολο δεδομένων που διατίθεται ελεύθερα από το αμερικάνικο εθνικό κέντρο για περιβαλλοντικές πληροφορίες (National Centers for Environmental Information) σε μορφή csv (https://www.ncdc.noaa.gov/cdo-web/datasets), και περιέχει ανεπεξέργαστη πληροφορία από πάνω από 25,000 μετεωρολογικούς σταθμούς της Αμερικής. Για την παρούσα εργασία χρησιμοποιήθηκαν πληροφορίες από τους μετεωρολογικούς σταθμούς της Νέας Υόρκης για το 2015 και το 2016 (43,200 εγγραφές).

4.2 Εγκατάσταση και παραμετροποίηση Apache Spark για την επεξεργασία των δεδομένων

Τα προγράμματα του Apache Spark τρέχουν ως ανεξάρτητα σετ διεργασιών πάνω σε ένα cluster, και συντονίζονται από ένα αντικείμενο SparkContext στο κύριο πρόγραμμα (σχήμα 4.1)



Σχήμα 4.1 – Διάγραμμα ροής Apache Spark [23]

Για να τρέξει πάνω στον cluster το SparkContext πρέπει να συνδεθεί πάνω στον Cluster Manager (είτε τον δικό του αυτόνομο cluster manager, είτε μέσω Mesos ή Yarn) ο οποίος κατανέμει τους πόρους στις εφαρμογές.

Για τις ανάγκες της παρούσας διπλωματικής, εγκαταστάθηκε το Spark σε Standalone Mode, δηλαδή χρησιμοποιήθηκε ο αυτόνομος cluster manager του Apache Spark, αφού αυτός ο τρόπος εγκατάστασης ήταν αρκετά γρήγορος και κάλυπτε τις ανάγκες της διπλωματικής.

Για την εγκατάσταση σε Standalone Mode, τοποθετήθηκε το εκτελέσιμο του Apache Spark σε κάθε κόμβο του Cluster. Για την έναρξη του master server εκτελέστηκε η παρακάτω εντολή στον master κόμβο:

```
./sbin/start-master.sh
```

Για την έναρξη του κάθε worker εκτελέστηκε η παρακάτω εντολή σε κάθε slave κόμβο:

```
./sbin/start-slave.sh <master-spark-URL>
```

Για την αποστολή του κώδικα και την εκτέλεσή του στον cluster χρησιμοποιήθηκε η εντολή:

```
./bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  <application-jar>
```

4.3 Υλοποίηση του Getis Ord GI* σε Scala

Για την υλοποίηση του αλγορίθμου Getis Ord GI* έγινε κατάτμηση της γεωγραφικής περιοχής της Νέας Υόρκης σε τετράγωνα κελιά πλάτους 1000m, και για κάθε μέρα υπολογίστηκε το άθροισμα των ανθρώπων που κατέβηκαν με ταξί στο κάθε κελί ($x_{i,j}$) . Για την απλοποίηση του προβλήματος δεν χρησιμοποιήθηκε η ευκλείδεια απόσταση για τον υπολογισμό των χωρικών βαρών $w_{i,j}$, αλλά ελήφθησαν υπόψη μόνο οι άμεσοι γείτονες του κάθε κελιού. Δηλαδή:

$$w_{i,j} = \begin{cases} 1, & \alpha\nu\ \tau\alpha\ \kappa\epsilon\lambda\iota\acute{\alpha}\ \epsilon\acute{\iota}\nu\alpha\iota\ \gamma\epsilon\acute{\iota}\tau o\nu\epsilon\varsigma \\ 0, & \alpha\lambda\lambda o\upsilon \end{cases}$$

Για τον γρήγορο υπολογισμό της τιμής του GI* του κάθε κελιού, και την αποφυγή της δαπανηρής αναζήτησης της τιμής των γειτόνων, ορίστηκε ένα επιπλέον πεδίο σε κάθε κελί, το οποίο κρατούσε την τιμή του αθροίσματος των γειτόνων. Έτσι το κελί είχε τη μορφή:

```
cell= (x, y, t, dayofweek, cell_people, sum_people)
```

Για κάθε γραμμή των δεδομένων εισόδου, δημιουργήθηκαν 9 εγγραφές. Μία για το κελί στο οποίο αναφέρεται η είσοδος, και μία για κάθε γείτονα:

```
val s=Seq((cell_x, cell_y, cell_t, dayofweek, p_count, p_count))
        for (a <- cell_x-1 to cell_x+1) {
          for (b <- cell_y-1 to cell_y+1) {
              if (a != cell_x && b != cell_y) {
                s :+ (a, b, cell_t, dayofweek, 0, p_count)
```

```
            }
          }
        }
```

Οι εγγραφές αθροίστηκαν κατανεμημένα με χρήση της reduceByKey χρησιμοποιώντας ως κλειδί το (cell_x, cell_y, cell_t, dayofweek), δημιουργώντας έτσι μια μοναδική εγγραφή για κάθε κελί που περιείχε το $\sum_{j=1}^{n} w_{i,j}x_j = \sum_{j=1}^{n} x_j$ (αφού τα βάρη των γειτόνων είναι ίσα με 1, και 0 αλλού) , και έπειτα υπολογίστηκε το GI* score για κάθε κελί της Νέας Υόρκης.

4.4 Επιλογή παραμέτρων και δημιουργία τελικού dataset

Για την εκπαίδευση του μοντέλου μηχανικής μάθησης και τη δημιουργία του τελικού dataset, χρησιμοποιήθηκε η μέση θερμοκρασία, και ο υετός από όλους τους μετεωρολογικούς σταθμούς της Νέας Υόρκης για κάθε μέρα, και υπολογίστηκε ο καθημερινός μέσος όρος θερμοκρασίας και υετού για όλη τη Νέα Υόρκη.

Οι παράμετροι που χρησιμοποιήθηκαν είναι οι ακέραιες συντεταγμένες του κάθε κελιού (μετασχηματισμένες χωρικές συντεταγμένες), η μέρα της βδομάδας, η μέση θερμοκρασία, ο μέσος υετός και το GI score.

Η τελική εξαγωγή του dataset έγινε σε μορφή csv με τη χρήση της βιβλιοθήκης com.databricks.spark.csv

Κεφάλαιο 5 - Υλοποίηση Μοντέλου Πρόβλεψης

5.1 Google Tensorflow

5.1.1 Εγκατάσταση και παραμετροποίηση

Η εγκατάσταση του Tensorflow μπορεί να γίνει μέσω virtualenv, pip, Docker, Anaconda ή απευθείας μέσω του πηγαίου κώδικα.

Η εγκατάσταση έγινε σε Debian 8 μέσω pip αφού ήταν η πιο γνώριμη και γρήγορη μέθοδος. Για την εγκατάσταση του pip εκτελέστηκε η εντολή:

```
$ apt-get install python-pip python-dev
```

Χρησιμοποιήθηκε η έκδοση του Tensorflow που δεν υποστηρίζει επεξεργασία σε GPU, αφού το σύστημα δεν διέθετε κατάλληλη κάρτα γραφικών. Για την εγκατάσταση του Tensorflow εκτελέστηκε η εντολή:

```
$ pip3 install tensorflow
```

Για λόγους διευκόλυνσης της συγγραφής του κώδικα και των διάφορων πειραματισμών, χρησιμοποιήθηκε το Jupyter Notebook για την συγγραφή και εκτέλεση του Python κώδικα, το οποίο εγκαταστάθηκε με χρήση της εντολής:

```
$ pip3 install jupyter
```

Για την ανάγνωση των δεδομένων εισόδου χρησιμοποιήθηκε η βιβλιοθήκη pandas η οποία εγκαταστάθηκε με χρήση της εντολής:

```
$ pip3 install pandas
```

Χρησιμοποιήθηκαν επίσης οι βιβλιοθήκες NumPy για πράξεις πινάκων και scikit-learn για διαχωρισμό του dataset σε train και test, οι οποίες εγκαταστάθηκαν με χρήση της εντολής:

```
$ pip3 install numpy
$ pip3 install scikit-learn
```

5.1.2 Η βιβλιοθήκη tf.contrib.learn

Η βιβλιοθήκη tf.contrib.learn είναι ένα API μηχανικής μάθησης υψηλού επιπέδου του Tensorflow που επιτρέπει την εύκολη και γρήγορη δημιουργία, παραμετροποίηση, εκπαίδευση και αξιολόγηση μοντέλων μηχανικής μάθησης.

Με χρήση της βιβλιοθήκης tf.contrib.learn ο ορισμός ενός ταξινομητή (classifier) μπορεί να γραφτεί πολύ γρήγορα ως:

```
# Δημιουργία DNN 3 επιπέδων με 10, 20, 10 νευρώνες αντιστοίχως.
  classifier = tf.contrib.learn.DNNClassifier(feature_columns=feature_c,
                                              hidden_units=[10, 20, 10],
                                              n_classes=3,
                                              model_dir="/tmp/ model")
```

Η εκπαίδευση του μοντέλου γίνεται με την εντολή:

```
classifier.fit(input_fn=get_train_inputs, steps=2000)
```

Η αξιολόγηση μπορεί να γραφτεί με την εντολή:

```
evaluation = classifier.evaluate(input_fn=get_test_inputs, steps=1)
```

Ενώ η πρόβλεψη μπορεί να εκτελεστεί εύκολα με χρήση της εντολής:

```
predictions = list(classifier.predict(input_fn=new_samples))
```

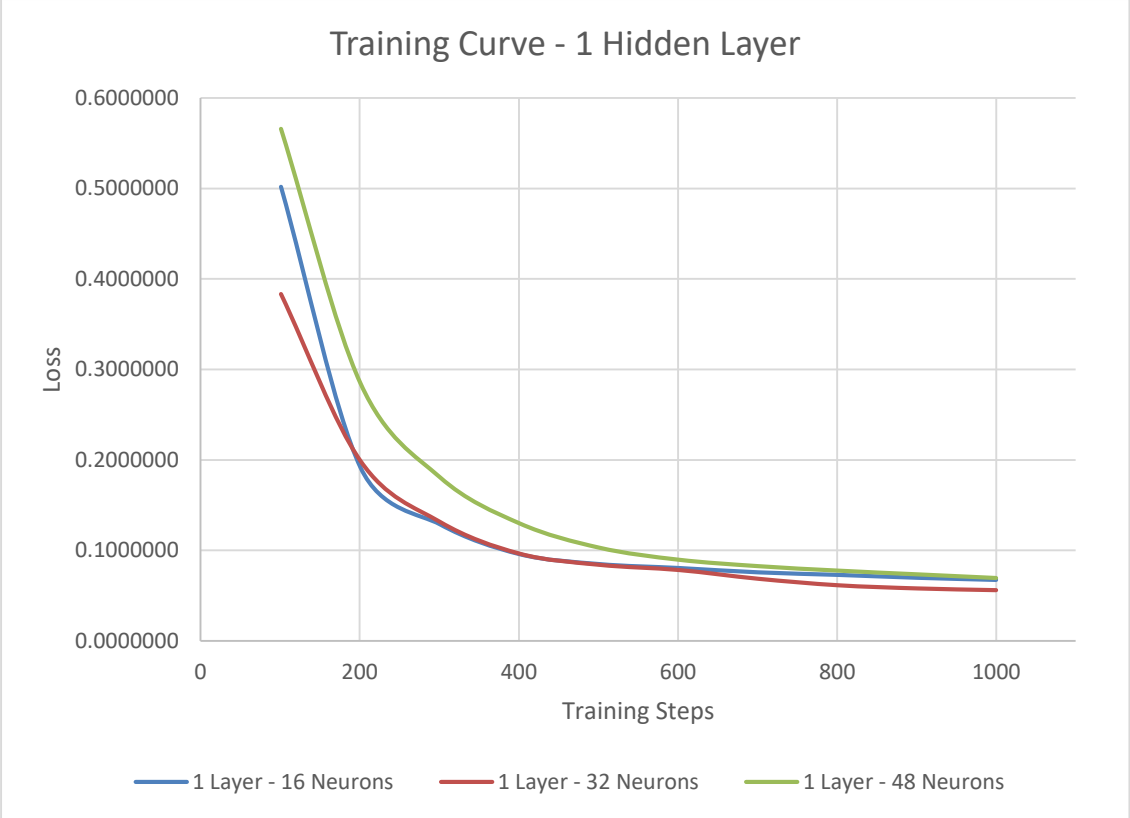
Η χρήση της βιβλιοθήκης αυτής κρίθηκε επιτακτική αφού έδωσε τη δυνατότητα της γρήγορης δημιουργίας, παραμετροποίησης, ελέγχου και σύγκρισης πολλών διαφορετικών μοντέλων μηχανικής μάθησης.

5.2 Αρχιτεκτονική του μοντέλου

Για την ανάλυση και την σωστή πρόβλεψη των δεδομένων του προβλήματος δοκιμάστηκαν διάφορες αρχιτεκτονικές βαθιάς μάθησης.

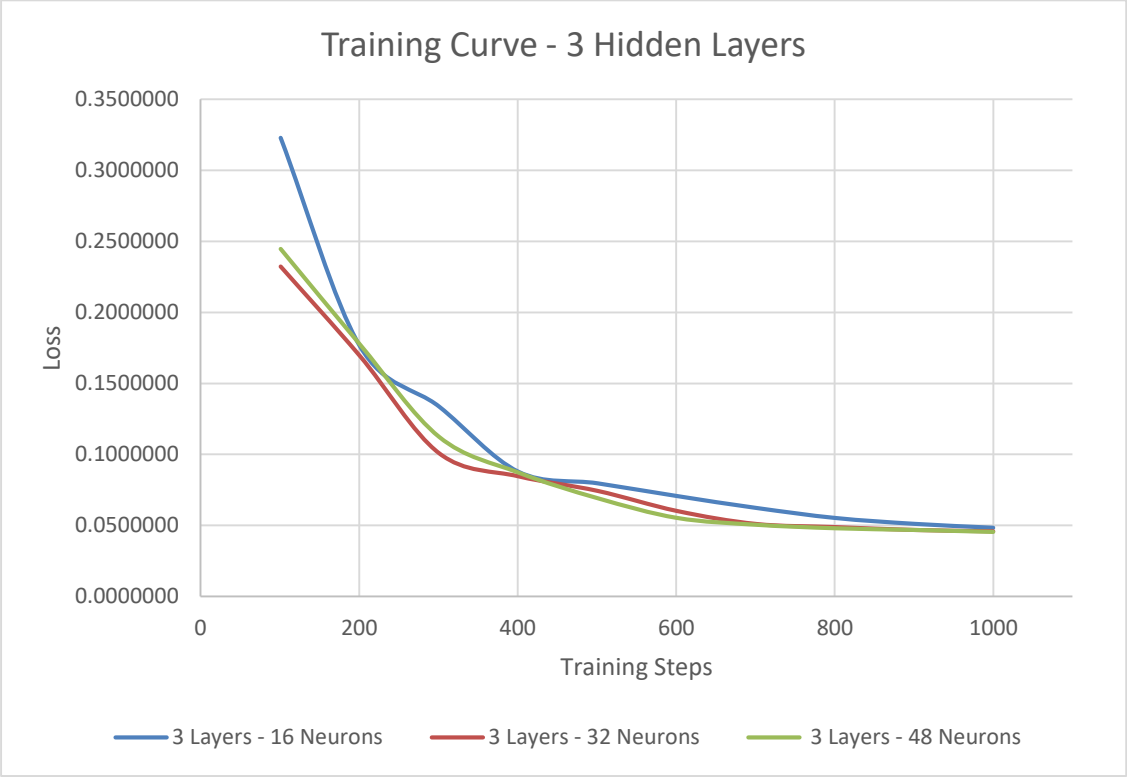
Δοκιμάστηκαν αρχιτεκτονικές με ένα, δύο, τρία και τέσσερα πλήρως συνδεδεμένα κρυφά επίπεδα, με 16, 32, και 48 νευρώνες το κάθε επίπεδο. Συγκρίθηκε το σφάλμα και η σύγκλιση για 1000 επαναλήψεις εκπαίδευσης του νευρωνικού δικτύου, καθώς και ο μέσος χρόνος εκπαίδευσης της κάθε αρχιτεκτονικής.

Όπως φαίνεται από τα παρακάτω σχήματα, οι αρχιτεκτονικές με 3 και 4 κρυφά επίπεδα είχαν την καλύτερη σύγκλιση για τα δεδομένα του προβλήματός μας.



Σχήμα 5.1 – Σύγκλιση αρχιτεκτονικής 1 κρυφού επιπέδου

Σχήμα 5.2 – Σύγκλιση αρχιτεκτονικής 2 κρυφών επιπέδων



Σχήμα 5.3 – Σύγκλιση αρχιτεκτονικής 3 κρυφών επιπέδων

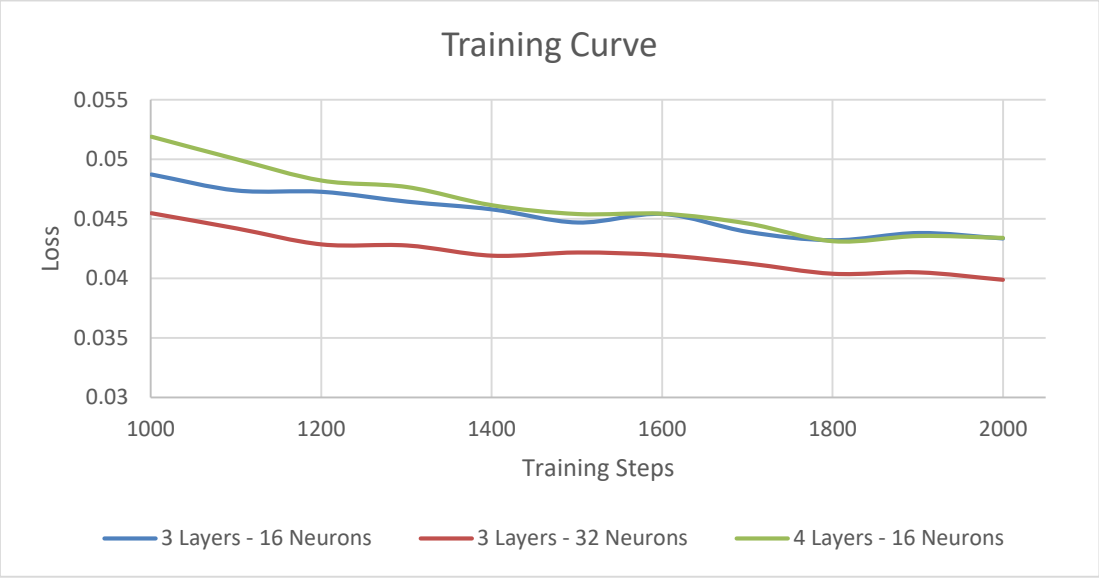
Σχήμα 5.4 – Σύγκλιση αρχιτεκτονικής 4 κρυφών επιπέδων

Με την αύξηση της πολυπλοκότητας της αρχιτεκτονικής του νευρωνικού δικτύου αυξάνει με μεγάλο ρυθμό και ο χρόνος που χρειάζεται για την εκπαίδευση του δικτύου, οπότε για την επιλογή της αρχιτεκτονικής πρέπει να λάβουμε υπόψη και τους χρόνους εκπαίδευσης πέρα από τη σύγκλιση (Σχήμα 5.5).

Σχήμα 5.5 – Ταχύτητα εκπαίδευσης του νευρωνικού δικτύου

Από τα παραπάνω, λαμβάνοντας υπόψη το σφάλμα και την ταχύτητα εκπαίδευσης, οι τρεις καταλληλότερες αρχιτεκτονικές για το νευρωνικό μας δίκτυο είναι με 3 κρυφά επίπεδα και 16 ή 32 νευρώνες και με 4 επίπεδα και 16 νευρώνες. Για την επιλογή της αρχιτεκτονικής που θα χρησιμοποιήσουμε, εκπαιδεύουμε το δίκτυο μας για άλλες 1000 επαναλήψεις.

Όπως φαίνεται στο παρακάτω γράφημα η αρχιτεκτονική με την καλύτερη σύγκλιση είναι με 3 κρυφά επίπεδα και 32 νευρώνες σε κάθε επίπεδο.



Σχήμα 5.6 – Σύγκριση σύγκλισης επικρατέστερων αρχιτεκτονικών

5.3 Συναρτήσεις ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης (a**ctivation functions**) είναι οι συναρτήσεις που καθορίζουν αν θα ενεργοποιηθεί ή όχι ένας νευρώνας και χρησιμοποιούνται για να επιτευχθεί μη-γραμμικότητα σε ένα νευρωνικό δίκτυο. Η ύπαρξη μη-γραμμικότητας στο μοντέλο είναι απαραίτητη αν μας ενδιαφέρει η πρόβλεψη και η μοντελοποίηση μη-γραμμικών σχέσεων.

5.3.1 Συναρτήσεις ενεργοποίησης που υποστηρίζει το Tensorflow

tf.nn.relu

Η συνάρτηση ReLU (Rectified Linear Unit) υπολογίζει το

$$f(x) = max(0, x)$$

tf.nn.relu6

Η συνάρτηση ReLU6 (Rectified Linear Unit 6) υπολογίζει το

$$f(x) = min(max(0, x), 6)$$

tf.nn.crelu

Η συνάρτηση cReLU (Concatenated ReLU) ενώνει την έξοδο μιας ReLU που επιλέγει το θετικό μέρος μιας ενεργοποίησης, με την έξοδο μιας ReLU που επιλέγει το αρνητικό μέρος της ενεργοποίησης. Έτσι ως αποτέλεσμα δίνει μια μη-γραμμικότητα με διπλάσιο εύρος ενεργοποιήσεων [24].

tf.nn.elu

Η συνάρτηση ELU (Exponential Linear Unit) υπολογίζει το

$f(x) = \begin{cases} x & ,x > 0 \\ a(\exp(x) - 1) & ,x \leq 0 \end{cases}$ $,a > 0$ [25]

tf.nn.softplus

Η συνάρτηση Softplus υπολογίζει το

$$f(x) = log(exp(x) + 1)$$

tf.nn.softsign

Η συνάρτηση Softsign υπολογίζει το

$$f(x) = \frac{x}{abs(x) + 1}$$

tf.sigmoid

Η συνάρτηση Sigmoid υπολογίζει το

$$f(x) = \frac{1}{1 + exp(-x)}$$

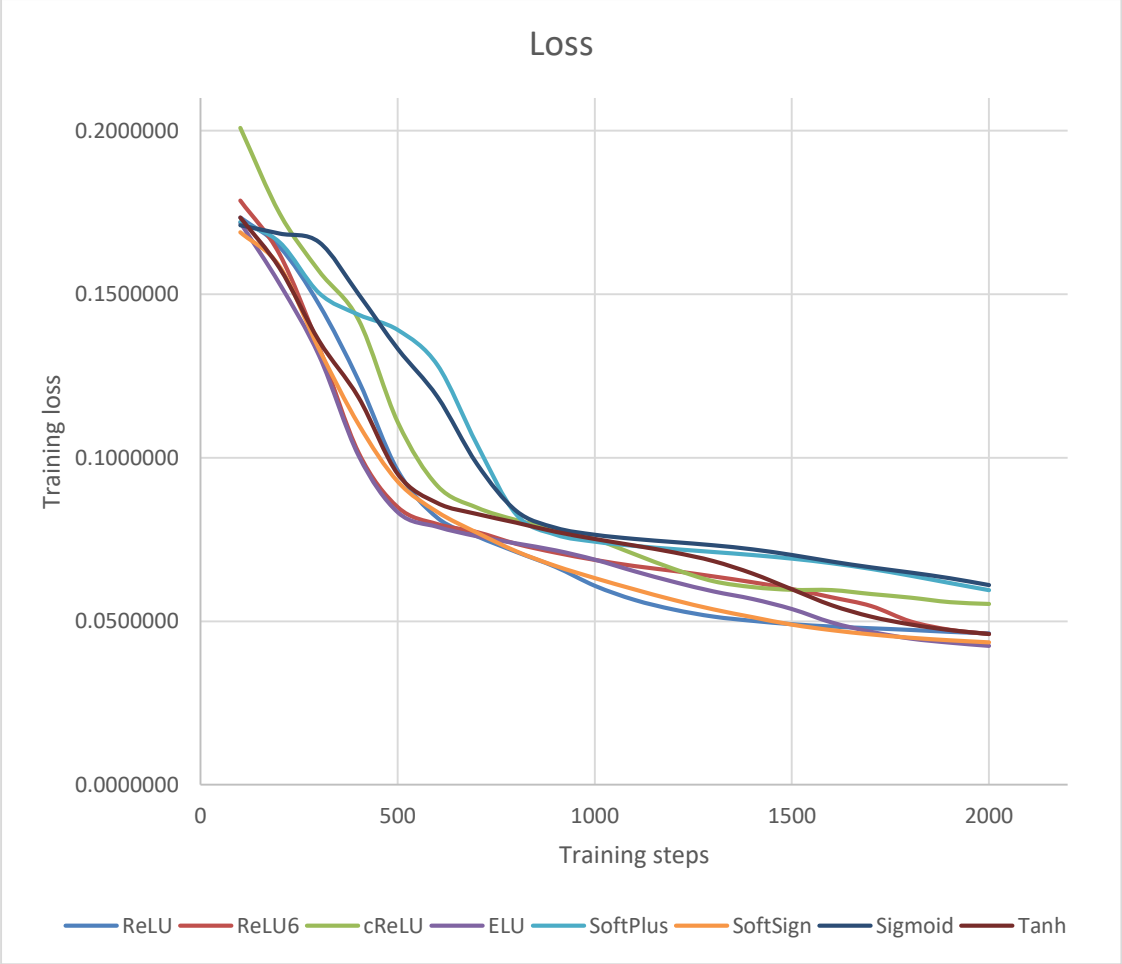
tf.tanh

Η συνάρτηση αυτή υπολογίζει την υπερβολική εφαπτομένη της εισόδου.

5.3.2 Επιλογή της βέλτιστης συνάρτησης ενεργοποίησης

Για την επιλογή της βέλτιστης συνάρτησης ενεργοποίησης για τα δεδομένα του προβλήματος δημιουργήθηκε ένα μοντέλο βαθιάς μάθησης με 3 κρυφά επίπεδα και 16 νευρώνες το κάθε επίπεδο. Δεν χρησιμοποιήθηκε η αρχιτεκτονική που επιλέχτηκε για το τελικό μοντέλο, αλλά ένα μικρότερο και απλούστερο για λόγους χρονικής πολυπλοκότητας.

Το μοντέλο εκπαιδεύτηκε με τα πραγματικά δεδομένα του προβλήματος, και συγκρίθηκε το σφάλμα και η σύγκλιση για 2000 επαναλήψεις εκπαίδευσης.



Σχήμα 5.6 – Σύγκριση σύγκλισης συναρτήσεων ενεργοποίησης

Από το παραπάνω διάγραμμα παρατηρούμε ότι οι καταλληλότερες συναρτήσεις ενεργοποίησης για τα δεδομένα του προβλήματός μας είναι οι SoftSign και η ELU.

5.4. Αλγόριθμοι βελτιστοποίησης

Για τον υπολογισμό των βαρών και των σφαλμάτων (εκπαίδευση του νευρωνικού δικτύου) στα προβλήματα μηχανικής μάθησης, χρησιμοποιείται ο αλγόριθμος της απότομης καθόδου (gradient descent algorithm). Υπάρχουν πολλές διαφορετικές υλοποιήσεις αυτού του αλγορίθμου, με διαφορετικά αποτελέσματα σύγκλισης για κάθε αλγόριθμο. Παρακάτω αναλύουμε και προσπαθούμε να συγκρίνουμε τις πέντε πιο δημοφιλείς υλοποιήσεις που υποστηρίζει το Tensorflow.

5.4.1 Αλγόριθμοι που υποστηρίζει το Tensorflow

5.4.1.1 Stochastic Gradient Descent

Ο αλγόριθμος Stochastic gradient descent (SGD) υπολογίζει την κλίση της συνάρτησης κόστους σε σχέση με τις παραμέτρους θ για κάθε παράδειγμα εκπαίδευσης $x^{(i)}$ και έξοδο $y^{(i)}$ ως εξής:

$$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right)$$

όπου η ο ρυθμός μάθησης (learning rate).

5.4.1.2 Momentum

Ο αλγόριθμος Momentum βελτιώνει τον SGD σε προβλήματα με περιοχές που η επιφάνεια καμπυλώνεται πιο απότομα στη μια διάσταση σε σύγκριση με την άλλη. Σε αυτές τις περιπτώσεις ο SGD ταλαντώνεται σε αυτές τις περιοχές και έχει πολύ αργή σύγκλιση. Ο Momentum επιτυγχάνει αυτή τη βελτίωση προσθέτοντας ένα κλάσμα γ του διανύσματος του προηγούμενου βήματος:

$$u_t = \gamma u_{t-1} + \eta \cdot \nabla_\theta J(\theta)$$

$$\theta = \theta - u_t$$

5.4.1.3 Adagrad

Ο Adagrad προσαρμόζει το ρυθμό μάθησης στις παραμέτρους κάνοντας μεγαλύτερες ανανεώσεις σε πιο σπάνιες παραμέτρους, και μικρότερες στις πιο συχνές.

$$g_{t,i} = \nabla_\theta J(\theta_i)$$

$$\theta_{t+1,i} = \theta_t - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}$$

όπου $G_t \in \mathbb{R}^{dxd}$ένας διαγώνιος πίνακας όπου το κάθε διαγώνιο στοιχείο i,i είναι το άθροισμα των τετραγώνων των κλίσεων των παραμέτρων θ_i μέχρι τη χρονική στιγμή t.

5.4.1.4 RMSprop

Ο αλγόριθμος RMSprop είναι ένας μη δημοσιευμένος αλγόριθμος ο οποίος προτάθηκε από τον Geoff Hinton. Ο RMSprop προσαρμόζει το ρυθμό μάθησης διαιρώντας τον με έναν εκθετικά φθίνοντα μέσο όρο των τετραγώνων των κλίσεων.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} \cdot g_t$$

5.4.1.5 Adam

Ο αλγόριθμος Adam είναι και αυτός αλγόριθμος που προσαρμόζει τον ρυθμό μάθησης για κάθε παράμετρο. Εκτός από το να αποθηκεύει έναν εκθετικά φθίνοντα μέσο όρο των τετραγώνων των προηγούμενων κλίσεων όπως ο RMSprop, αποθηκεύει και έναν εκθετικά φθίνοντα μέσο όρο των προηγούμενων κλίσεων όπως ο Momentum.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$u_t = \beta_2 u_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

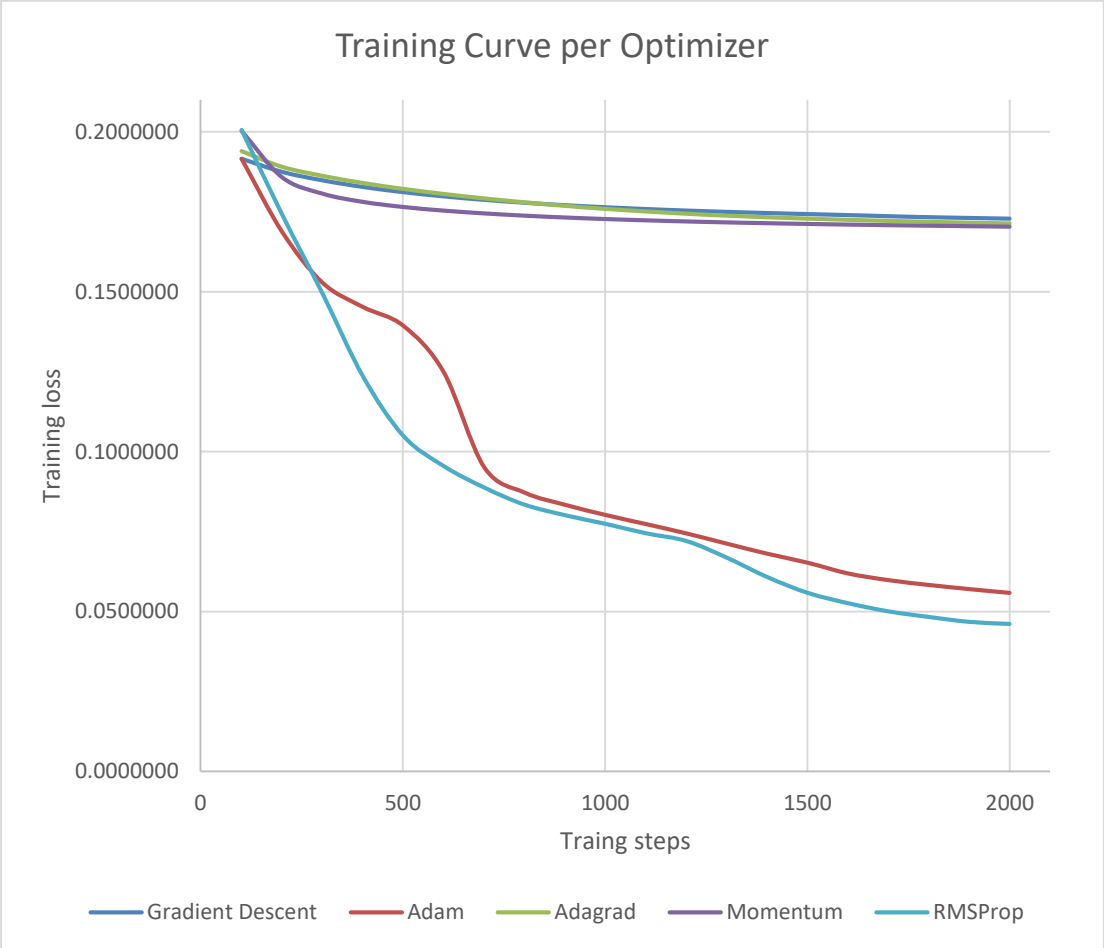
$$\hat{u}_t = \frac{u_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{u}_t} + \varepsilon} \cdot \hat{m}_t$$

5.4.2 Επιλογή του βέλτιστου αλγορίθμου

Για την επιλογή του κατάλληλου αλγορίθμου βελτιστοποίησης για τα δεδομένα του προβλήματος δημιουργήθηκε ένα μοντέλο βαθιάς μάθησης με 3 κρυμμένα επίπεδα και 16 νευρώνες το κάθε επίπεδο, παρόμοιο με το μοντέλο που χρησιμοποιήθηκε για την επιλογή της συνάρτησης ενεργοποίησης.

Χρησιμοποιήθηκε η cReLU ως συνάρτηση ενεργοποίησης και το μοντέλο εκπαιδεύτηκε με τα πραγματικά δεδομένα του προβλήματος, και συγκρίθηκε το σφάλμα και η σύγκλιση για 2000 επαναλήψεις εκπαίδευσης, για κάθε αλγόριθμο από τους παραπάνω αλγορίθμους βελτιστοποίησης. Ως βήμα εκπαίδευσης χρησιμοποιήθηκε το βήμα 0.001.



Σχήμα 5.7 – Σύγκριση σύγκλισης αλγορίθμων βελτιστοποίησης

Από το παραπάνω διάγραμμα παρατηρούμε ότι ο καταλληλότερος αλγόριθμος βελτιστοποίησης για τα δεδομένα του προβλήματός μας είναι ο RMSProp.

5.5 Το πρόβλημα του Overfitting

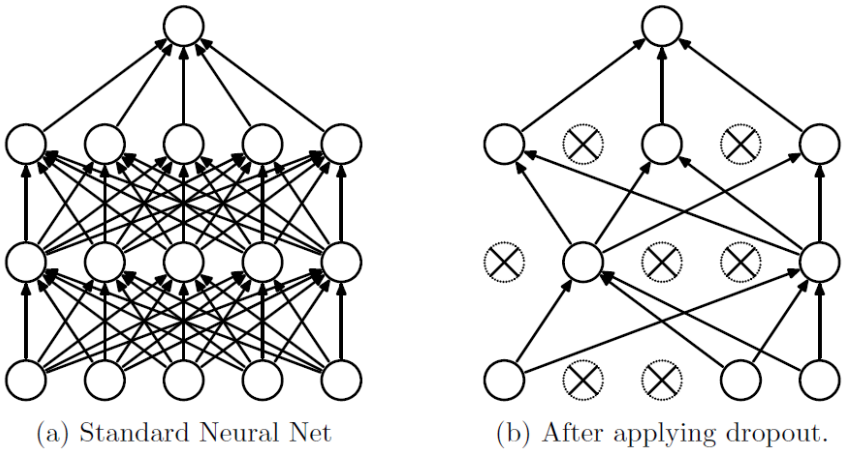
5.5.1 Ορισμός Overfitting

Overfitting έχουμε όταν το μοντέλο αποδίδει καλά για τα δεδομένα εκπαίδευσης, αλλά δεν αποδίδει καλά στα δεδομένα αξιολόγησης. Αυτό οφείλεται στο γεγονός ότι το μοντέλο απομνημονεύει τα δεδομένα που έχει συναντήσει, αλλά δεν είναι σε θέση να γενικεύσει για δεδομένα που δεν έχει συναντήσει [26].

5.5.2 Τρόπος αντιμετώπισης

Για την αντιμετώπιση του προβλήματος του overfitting χρησιμοποιήθηκε η τεχνική dropout (απόρριψη) με ποσοστό απόρριψης 10% σε κάθε επανάληψη.

Η βασική ιδέα αυτής της τεχνικής είναι η τυχαία απόρριψη μονάδων του νευρωνικού δικτύου κατά τη διάρκεια της εκπαίδευσης. Αυτό εμποδίζει τις μονάδες από το να προσαρμόζονται υπερβολικά στα δεδομένα. Ο όρος “απόρριψη” αναφέρεται στην προσωρινή αφαίρεση ενός κόμβου από το δίκτυο, μαζί με όλες τις εισερχόμενες και εξερχόμενες ακμές του. Οι μονάδες που θα απορριφθούν επιλέγονται τυχαία σε κάθε βήμα εκπαίδευσης με αποτέλεσμα τη δημιουργία ενός τυχαίου “αραιωμένου” δικτύου σε κάθε βήμα. Έτσι για ένα νευρωνικό δίκτυο με n μονάδες, η εκπαίδευση του με dropout είναι σαν να εκπαιδεύουμε μια συλλογή από 2^n “αραιωμένα” δίκτυα [27].



Σχήμα 5.8 – Η μέθοδος του dropout [27]

Η χρήση της συγκεκριμένης τεχνικής αύξησε λίγο το τελικό σφάλμα κατά τη διάρκεια της εκπαίδευσης, αλλά μείωσε πολύ το σφάλμα κατά τη διάρκεια της αξιολόγησης με νέα δεδομένα και βελτίωσε πολύ την ικανότητα πρόβλεψης του δικτύου.

5.6 Εκπαίδευση του νευρωνικού δικτύου και έλεγχος ακρίβειας

Για την εκπαίδευση του νευρωνικού δικτύου επιλέχτηκε τελικά αρχιτεκτονική 3 κρυφών επιπέδων, με 32 νευρώνες το κάθε επίπεδο. Η εισαγωγή μη-γραμμικότητας στο δίκτυο έγινε με χρήση της συνάρτησης ELU, και για την βελτιστοποίηση χρησιμοποιήθηκε ο αλγόριθμος RMSProp. Για την αντιμετώπιση του overfitting χρησιμοποιήθηκε η μέθοδος dropout με ποσοστό απόρριψης 10% και το βήμα εκπαίδευσης που επιλέχτηκε ήταν 0.001. Για την εκπαίδευση του μοντέλου χρησιμοποιήθηκαν δεδομένα για το 2015.

Η υλοποίηση του μοντέλου έγινε με τη χρήση της βιβλιοθήκης tf.contrib.learn:

```
rmsprop = tf.train.RMSPropOptimizer(learning_rate=0.001)
model  =  tf.contrib.learn.DNNRegressor(feature_columns=[dayofweek_emb,x_emb,
y_emb, avgtemp, precipitation],
                                        hidden_units=[32,32,32],
                                        dropout=0.1,
                                        activation_fn=tf.nn.elu,
                                        model_dir=model_dir,
                                        optimizer=rmsprop)
```

Η εκπαίδευση του δικτύου έγινε σταδιακά και για 25,000 βήματα στο σύνολο:

```
model.fit(input_fn=lambda: input_fn(df_train),steps=2000)
```

Η αξιολόγηση έγινε με χρήση του 1% των δεδομένων εκπαίδευσης αλλά και με δεδομένα από τον Φεβρουάριο και τον Μάρτιο του 2016:

```
# Evaluating our model:
results = model.evaluate(input_fn=lambda: input_fn(df_test), steps=1)
for key in sorted(results):
     print("%s: %s" % (key, results[key]))
```

Κεφάλαιο 6 - Υλοποίηση διεπαφής χρήστη

6.1 Εργαλεία που χρησιμοποιήθηκαν

Για τις ανάγκες της διπλωματικής εργασίας υλοποιήθηκε ένα web service με χρήση Python και της βιβλιοθήκης web.py. Με τη συγκεκριμένη βιβλιοθήκη υλοποιήθηκε ένα web service το οποίο καλεί το μοντέλο μηχανικής μάθησης με τα δεδομένα εισόδου, και επιστρέφει την έξοδο του μοντέλου σε μορφή geoJSON. Το geoJSON είναι ένας τύπος JSON με υποστήριξη χωρικής πληροφορίας.

Για την διεπαφή χρήστη, και την απεικόνιση των hotspot σε χάρτη χρησιμοποιήθηκε η βιβλιοθήκη OpenLayers. Η βιβλιοθήκη αυτή είναι μια βιβλιοθήκη ελεύθερου λογισμικού γραμμένη σε JavaScript η οποία επιτρέπει την εύκολη δημιουργία δυναμικών χαρτών. Η βιβλιοθήκη OpenLayers υποστηρίζει και επιτρέπει την ανάγνωση και την προβολή της πληροφορίας που περιέχεται σε geoJSON μορφή, και υποστηρίζει ασύγχρονη επικοινωνία με web service και δυναμική φόρτωση των δεδομένων σε χάρτη.

Η επικοινωνία της διεπαφής χρήστη με το web service πραγματοποιείται με χρήση http get και html φόρμας με είσοδο τα επίπεδα θερμοκρασίας και υετού και την ημέρα της βδομάδας.

6.2 Περιγραφή διεπαφής χρήστη

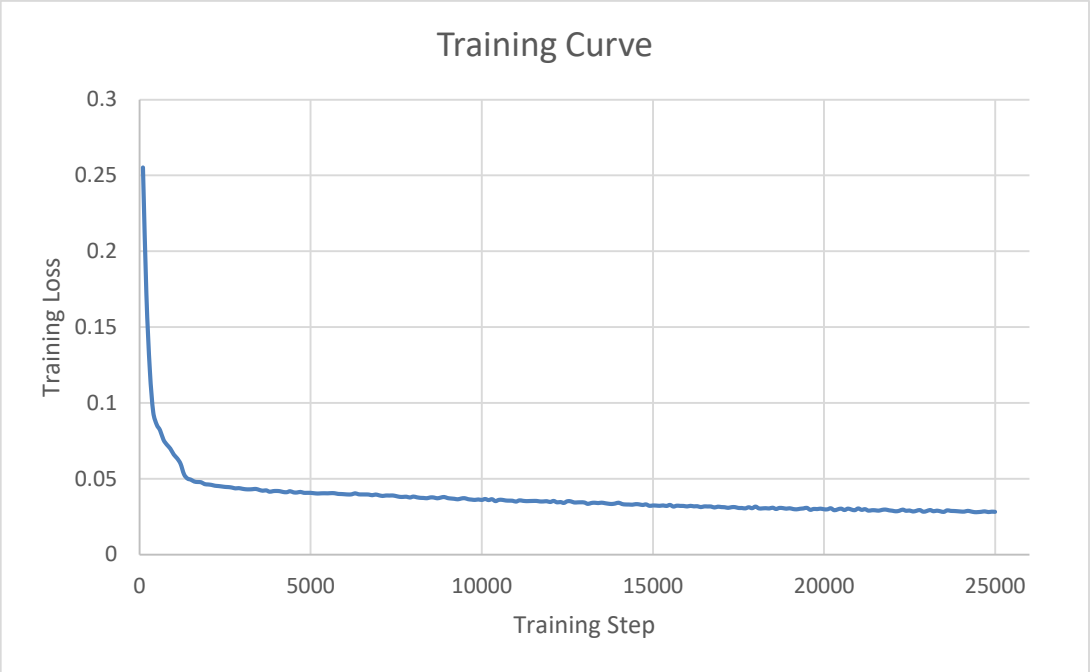
Η διεπαφή που υλοποιήθηκε είναι πολύ λιτή, με μόνο στόχο την επίδειξη και την προβολή των δυνατοτήτων του μοντέλου πρόβλεψης, καθώς και τον ευκολότερο οπτικό έλεγχο ορθότητας των αποτελεσμάτων και δεν είναι κατάλληλη για χρήση στην παραγωγή.

Ο χρήστης επιλέγει τα επίπεδα θερμοκρασίας και υετού, καθώς και την ημέρα για την οποία επιθυμεί να δει την πρόβλεψη, και το αποτέλεσμα προβάλλεται δυναμικά στον χάρτη.

Σχήμα 6.1 – Στιγμιότυπα χρήσης της διεπαφής χρήστη

Κεφάλαιο 7 – Παρουσίαση αποτελεσμάτων

Το νευρωνικό δίκτυο που δημιουργήθηκε εκπαιδεύτηκε για 25,000 επαναλήψεις και επιτεύχθηκε σφάλμα 0.0281434 στην τελευταία επανάληψη. Το σφάλμα εκπαίδευσης και η σύγκλιση φαίνονται στο παρακάτω σχήμα.

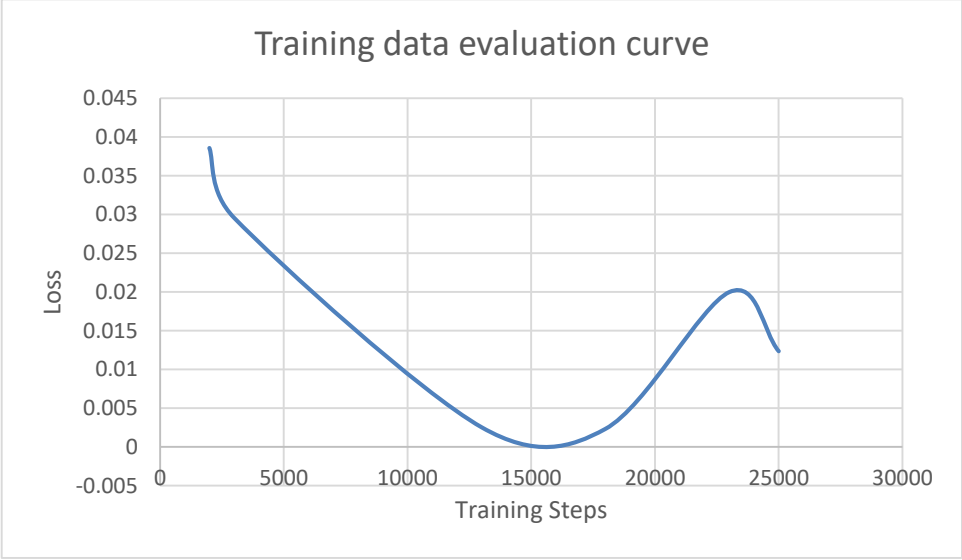


Σχήμα 7.1 – Καμπύλη σύγκλισης κατά τη διάρκεια της εκπαίδευσης

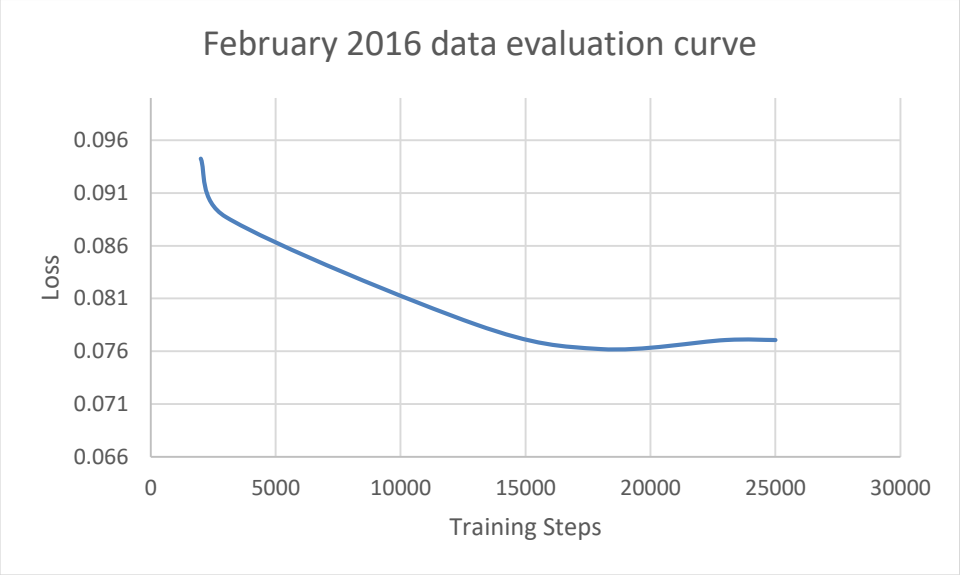
Η εκπαίδευση του δικτύου διήρκησε συνολικά 14 ώρες και η μέση ταχύτητα εκπαίδευσης ήταν 0.5 Steps/sec.

Σχήμα 7.2 – Καμπύλη ταχύτητας εκπαίδευσης

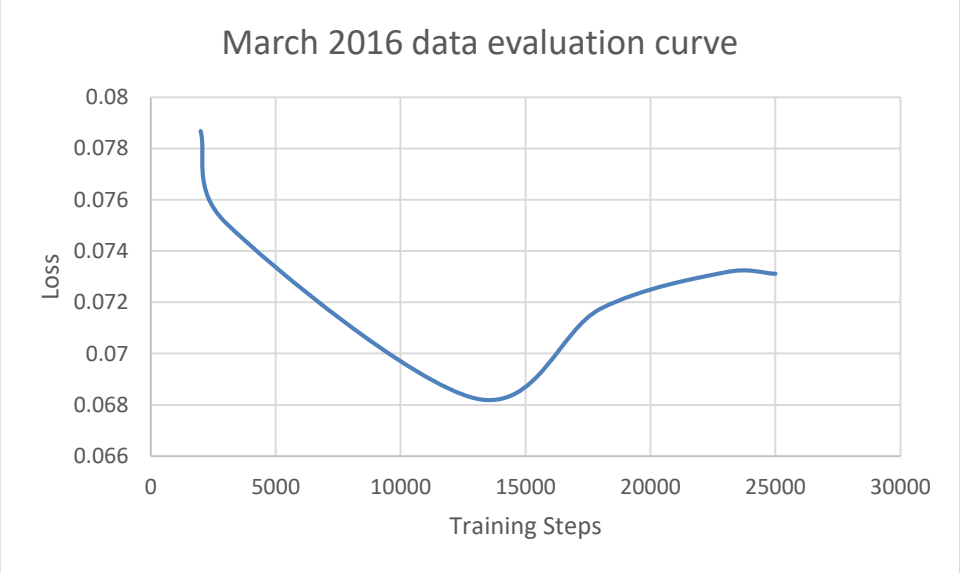
Το νευρωνικό δίκτυο αξιολογήθηκε με 3 διαφορετικά σετ δεδομένων. Το πρώτο σετ περιείχε κάποια δεδομένα του dataset εκπαίδευσης, τα οποία αφήσαμε εκτός για λόγους αξιολόγησης. Τα άλλα δύο σετ περιείχαν καινούργια δεδομένα (Φεβρουάριος και Μάρτης 2016) τα οποία δεν είχε ξανασυναντήσει το νευρωνικό δίκτυο.



Σχήμα 7.3 – Καμπύλη αξιολόγησης με τα δεδομένα του dataset εκπαίδευσης



Σχήμα 7.4 – Καμπύλη αξιολόγησης με τα δεδομένα του Φεβρουαρίου 2016



Σχήμα 7.5 – Καμπύλη αξιολόγησης με τα δεδομένα του Μαρτίου 2016

Από τα παραπάνω διαγράμματα παρατηρούμε ότι το σφάλμα στην αξιολόγηση αρχίζει να αυξάνεται μετά από κάποιο σημείο, οπότε σταματάμε την εκπαίδευση στις 16,000 επαναλήψεις.

Κεφάλαιο 8 - Συμπεράσματα - Μελλοντικές επεκτάσεις

Η παρούσα διπλωματική εργασία έδειξε ότι με τα διαθέσιμα εργαλεία ελεύθερου λογισμικού μπορεί να δημιουργηθεί μοντέλο μηχανικής μάθησης που να προβλέπει χωροχρονικά δεδομένα με ικανοποιητική ακρίβεια. Με χρήση του Apache Spark μπορούν να επεξεργαστούν με ευκολία τεράστιες ποσότητες δεδομένων, και με χρήση του Tensorflow μπορούν να δημιουργηθούν και να δοκιμαστούν με χρήση πολύ μικρού μεγέθους κώδικα πολλές διαφορετικές αρχιτεκτονικές μοντέλων.

Έδειξε επίσης ότι οι καιρικές συνθήκες αποτελούν μια χρήσιμη μεταβλητή για τέτοιου είδους μοντέλα, και συγκεκριμένα ότι η θερμοκρασία και ο υετός επηρεάζουν τις συνήθειες μετακίνησης του πληθυσμού με ταξί.

Ιδιαίτερο ενδιαφέρον θα είχε η μελλοντική επέκταση του μοντέλου με την προσθήκη δημογραφικών μεταβλητών στις μεταβλητές εισόδου του μοντέλου, όπως της οικονομικής κατάστασης και της μέσης ηλικίας του πληθυσμού στην περιοχή εκκίνησης ή λήξης της διαδρομής. Η ακρίβεια των αποτελεσμάτων θα μπορούσε να βελτιωθεί με τη χρήση χωροχρονικών hotspot που να λαμβάνουν υπόψη και την ώρα της ημέρας, όπως επίσης και με κατάτμηση της Νέας Υόρκης σε τετραγωνικά κελιά μικρότερου μεγέθους.

Το συγκεκριμένο μοντέλο θα μπορούσε να επεκταθεί για την πρόβλεψη της μετακίνησης του πληθυσμού ανεξαρτήτως μέσου, αν αντί για το New York City Taxi & Limousine Commission Dataset χρησιμοποιηθεί κάποιο dataset με χωρική πληροφορία από τα GPS των κινητών ή με πληροφορία check-in σε κοινωνικά μέσα δικτύωσης.

Βιβλιογραφία

[1] M. G. M. G. Andrea De Mauro, "A Formal Definition of Big Data Based on its Essential Features," *Library Review,* vol. 65, no. 3, pp. 122-135, 2016.

[2] D. Laney, "3-D data management: Controlling data volume, velocity and variety.," META Group Inc, 2001.

[3] M. H. Amir Gandomi, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management,* vol. 35, no. 2, pp. 137-144, 2015.

[4] J.-P. Dijcks, «Oracle: Big Data for the Enterprise,» Oracle Corporation, 2013.

[5] A. L. Samuel, «Some studies in machine learning using the game of Checkers,» McGraw-Hill, 1959.

[6] T. Mitchell, «Machine Learning,» McGraw Hill, 1997, p. 2.

[7] A. R. A. T. Mehryar Mohri, Foundations of Machine Learning, The MIT Press, 2012.

[8] "Apache Hadoop," The Apache Software Foundation, [Online]. Available: http://hadoop.apache.org/#What+Is+Apache+Hadoop%3F. [Accessed 8 April 2017].

[9] "Apache Spark," The Apache Software Foundation, [Online]. Available: http://spark.apache.org/. [Accessed 8 April 2017].

[10] "Introduction to MongoDB," MongoDB, Inc, [Online]. Available: https://docs.mongodb.com/manual/introduction/. [Accessed 8 April 2017].

[11] "TensorFlow," Google Inc., [Online]. Available: https://www.tensorflow.org/. [Accessed 9 April 2017].

[12] The Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," May 2016.

[13] Berkeley Vision and Learning Center, "Caffe - Deep learning framework," [Online]. Available: http://caffe.berkeleyvision.org/. [Accessed 9 April 2017].

[14] Apache, "MLlib - Apache Spark," [Online]. Available: http://spark.apache.org/mllib/. [Accessed 9 April 2017].

[15] scikit-learn developers, "scikit-learn: machine learning in Python," [Online]. Available: http://scikit-learn.org/stable/. [Accessed 9 April 2017].

[16] Ronan, Clément, Koray and Soumith, "torch - A scientific computing framework for

LuaJIT," [Online]. Available: http://torch.ch/. [Accessed 9 April 2017].

[17] J. K. O. Arthur Getis, "The Analysis of Spatial Association by Use of Distance Statistics," *Geographical Analysis,* vol. 24, no. 3, pp. 189-206, 1992.

[18] "SparkProject," [Online]. Available: https://github.com/mraad/SparkProject. [Accessed 19 April 2017].

[19] A. SIGSPATIAL, "ACM SIGSPATIAL GISCUP 2016," [Online]. Available: http://sigspatial2016.sigspatial.org/giscup2016/.

[20] G. Makrai, "Efficient method for large-scale spatio-temporal hotspot," in *SIGSPATIAL GIS*, 2016.

[21] S. V. G. d. M. W. L. M. V. A. A. W Randolph Franklin, "An efficient map-reduce algorithm for spatio-temporal analysis using Spark," in *SIGSPATIAL GIS*, 2016.

[22] S. P. T. W. L. Y. S. R. S. Teerayut Horanont, "Weather Effects on the Patterns of People's Everyday Activities: A Study Using GPS Traces of Mobile Phone Users," *PLoS ONE,* vol. 8, no. 12, 2013.

[23] Apache, "Cluster Mode Overview," [Online]. Available: http://spark.apache.org/docs/latest/cluster-overview.html. [Accessed 16 April 2017].

[24] K. S. D. A. H. L. Wenling Shang, "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units," [Online]. Available: https://arxiv.org/abs/1603.05201.

[25] T. U. S. H. Djork-Arné Clevert, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," in *ICLR 2016*, 2016.

[26] Amazon Web Services, Inc., "Model Fit: Underfitting vs. Overfitting," [Online]. Available: http://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html. [Accessed 20 April 2017].

[27] G. H. A. K. I. S. R. S. Nitish Srivastava, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[28] S. Ruder, "An Overview of gradient descent optimization algorithms," [Online]. Available: http://sebastianruder.com/optimizing-gradient-descent/. [Accessed 20 April 2017].

Παράρτημα

i. Πηγαίος Κώδικας - Επεξεργασία Δεδομένων

```
package org.fadi.taxiny


/**
  * Created by Fadi Shehadeh
  * Using Dataframes and computing sums on parse
  */


import com.github.nscala_time.time.Imports._
import org.apache.log4j.{Level, Logger}
import org.apache.spark.sql._
import org.apache.spark.sql.functions._
import org.apache.spark.{SparkConf, SparkContext}


object taxiNYDatasetsGID {


  val ERROR: Int = -2147483648
  val OUT_OF_BOUNDS: Int = 2147483647
  val min_date_string = "2015-01-01 00:00:00"


  /**
    * Alter variables below to control window size, spatial
    * and temporal step, and configure the csv path
    */
  val x_min: Double = -74.25
  val x_max: Double = -73.7
  val y_min: Double = 40.5
  val y_max: Double = 40.9
```

```
  val step: Double = 0.009

  val time_step: Int = 1

  val csv_path = "hdfs://hadoop:port/path/to/taxi/csv"

  //val csv_path = "/path/to/taxi/csv"

  val weather_path = "/path/to/weather/csv"


  val rootLogger = Logger.getRootLogger()

  rootLogger.setLevel(Level.ERROR)


  case class taxiCell(x: Int, y: Int, t: Int, dayofweek: Int, people: Long,
sumpeople: Long)

  case class weatherCell(t: Int, dayofweek: Int, avgtemp: Double, precipitat
ion: Double)


  //Parse function that creates a cell and its neighboor cells for each entr
y

  def parse(line: String): Seq[(Int,Int,Int,Int,Long,Long)] = {

    try {


      val min_date = DateTime.parse(min_date_string, DateTimeFormat.forPatte
rn("YYYY-MM-dd HH:mm:ss"))

      val fields = line.split(',')

      val x1 = fields(9).toDouble

      val y1 = fields(10).toDouble

      val p_count = fields(3).toLong

      val date = DateTime.parse(fields(2), DateTimeFormat.forPattern("YYYY-M
M-dd HH:mm:ss"))

      val dayofweek=date.dayOfWeek().get()

      val date_diff = min_date to date

      val cell_t = (date_diff.toDuration.getStandardDays / time_step).toInt

      var dayofweekn=dayofweek

      if (x1 >= x_min && x1 <= x_max && y1 >= y_min && y1 <= y_max && p_coun
t > 0 && (cell_t * time_step) < 367) {
```

```
        val cell_x = (x1 / step).toInt
        val cell_y = (y1 / step).toInt
        val s=Seq((cell_x,cell_y,cell_t,dayofweek,p_count,p_count))
        for (a <- cell_x-1 to cell_x+1) {
          for (b <- cell_y-1 to cell_y+1) {
              if (a != cell_x && b != cell_y) {
                s :+ (a, b, cell_t, dayofweek, 0, p_count)
              }
          }
        }
        s
      }
      else Seq((OUT_OF_BOUNDS, OUT_OF_BOUNDS, 0, 0, 0, 0))
    } catch {
      case e: Exception => Seq((ERROR, ERROR, 0, 0, 0, 0))
    }
  }


  def parseWeather(line: String): Seq[(Int,Int,Double,Double)] = {
    try {
      val min_date = DateTime.parse(min_date_string, DateTimeFormat.forPatte
rn("YYYY-MM-dd HH:mm:ss"))
      val fields = line.split(',')
      val date = DateTime.parse(fields(5), DateTimeFormat.forPattern("YYYYMM
dd"))
      val date_diff = min_date to date
      val cell_t = (date_diff.toDuration.getStandardDays / time_step).toInt
      val avgtemp = fields(7).toDouble
      val prcp=fields(6).toDouble
      val day= date.dayOfMonth().get()
      val month= date.monthOfYear().get()
      val year=date.year().get()
```

```
    val dayofweek=date.dayOfWeek().get()
    val s=Seq((cell_t,dayofweek,avgtemp,prcp))
    s

  } catch {
    case e: Exception => Seq((ERROR, ERROR, 0, 0))
  }
}

//Filter function to filter out errors and out of bounds entries
def fltr(tup: (Int, Int, Int, Int, Long, Long)): Boolean = {

  if (tup._1 > ERROR && tup._1 < OUT_OF_BOUNDS) {
    true
  }
  else false

}

def fltrWeather(tup: (Int, Int, Double, Double)): Boolean = {

  if (tup._3 != -9999 &&tup._4 != -9999 && tup._1 != ERROR) {
    true
  }
  else false
}

def main(args: Array[String]) = {

  val sc = new SparkContext(new SparkConf().setAppName("NYTaxi"))
  val sqlContext = new SQLContext(sc)
```

```
    import sqlContext.implicits._


    val taxiDF : DataFrame = sc.textFile(csv_path).flatMap(parse) //Create t
he RDD using the parse function
      .filter(fltr)
      .map(cell => (cell._1, cell._2, cell._3, cell._4) -> (cell._5, cell._6
) ) //Using (x,y,t) as key
      .reduceByKey((x,y)=>(x._1+y._1,x._2 + y._2)).map(cell => taxiCell(cell
._1._1, cell._1._2, cell._1._3, cell._1._4, cell._2._1, cell._2._2)) //addin
g up the values of each record of each cell
      .toDF()
    taxiDF.registerTempTable("taxi")
    taxiDF.cache()


    //Computing all the statistics needed for the gi*


    val stats= taxiDF.agg(min(taxiDF("x")),max(taxiDF("x")),max(taxiDF("y"))
,min(taxiDF("y"))).first()


    val x_maximum = stats.getInt(0)
    val x_minimum = stats.getInt(1)
    val x = math.abs(x_maximum - x_minimum) + 1


    val y_maximum = stats.getInt(2)
    val y_minimum = stats.getInt(3)
    val y = math.abs(y_maximum - y_minimum) + 1
    val countnum: Double = x.toDouble * y.toDouble
    val wij=9
    val sums= taxiDF.groupBy("t").agg(sum(taxiDF("people")).alias("sum1"),su
m(taxiDF("people")*taxiDF("people")).alias("sum2"),count("*").alias("count")
)
    val sums2=sums.select(sums("t"),(sums("sum1")/countnum).alias("x_mean"),
sums("sum2"))
```

```
    val sums3=sums2.select(sums2("t"),sqrt(sums2("sum2")/countnum.toDouble-sums2("x_mean")*sums2("x_mean")).alias("s"),(sums2("x_mean")*wij).alias("x_mean_wij"))

    val den=math.sqrt((countnum*wij-wij*wij)/(countnum-1))

    val sums4=sums3.select(sums3("t"),sums3("s"),sums3("x_mean_wij"),(sums3("s")*den).alias("denom"))


    val gi=taxiDF.join(sums4,usingColumn="t").select(taxiDF("x"),taxiDF("y"), taxiDF("t"), taxiDF("dayofweek"),

      ((taxiDF("sumpeople") -  sums4("x_mean_wij") ) / sums4("denom")).alias("gi"))


    val weatherDF=sc.textFile(weather_path).flatMap(parseWeather) //Create the RDD using the parse function

      .filter(fltrWeather).map(cell => (cell._1, cell._2) -> (cell._3,cell._4,1)).reduceByKey((x,y)=>(x._1+y._1,x._2 + y._2,x._3 + y._3))

      .map(cell => (cell._1._1, cell._1._2,cell._2._1/cell._2._3,cell._2._2/cell._2._3))

      .map(cell => weatherCell(cell._1, cell._2,cell._3,cell._4))

      .toDF()

    weatherDF.registerTempTable("weather")


    gi.join(weatherDF, usingColumns = Seq("t","dayofweek"), joinType = "inner")

      .select("dayofweek","x","y","avgtemp","precipitation","gi")

      .repartition(1)

      .write

      .format("com.databricks.spark.csv")

      .option("header", "true")

      .save("/path/to/result/dataset")

  }

}
```

ii. Πηγαίος Κώδικας - Υλοποίηση Μοντέλου Πρόβλεψης

```
# # New York Cab Dataset Deap Learning Model
#


# First we import he modules we are going to use
import tempfile
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.contrib import learn
from sklearn.model_selection import train_test_split


# Define Variables
train_file="/path/to/dataset2015gi_1km.csv"
cross_file="/path/to/dataset2016gi_1km_02.csv"
cross_file2="/path/to/dataset2016gi_1km_03.csv"
model_dir = "/path/to/model_layers_3x32_elu_rmsprop"
pred_file="/path/to/dataset2015gi_1km_test.csv"


# Import the csv
COLUMNS = ["dayofweek", "x", "y", "avgtemp", "precipitation", "gi"]
LABEL_COLUMN = "gi"


df = pd.read_csv(train_file, names=COLUMNS, skipinitialspace=False, skiprows
=1)
df_cross=pd.read_csv(cross_file, names=COLUMNS, skipinitialspace=False, skip
rows=1)
df_cross2=pd.read_csv(cross_file2, names=COLUMNS, skipinitialspace=False, sk
iprows=1)
df_pred = pd.read_csv(pred_file, names=COLUMNS, skipinitialspace=False, skip
rows=1)
```

```
df_train, df_test=train_test_split(df, test_size=0.01)


# Let's define the feature columns that we are going to use:


CATEGORICAL_COLUMNS = ["dayofweek","x", "y"]

CONTINUOUS_COLUMNS = ["avgtemp", "precipitation"]


# Continuous columns.

avgtemp = tf.contrib.layers.real_valued_column("avgtemp")

precipitation =  tf.contrib.layers.real_valued_column("precipitation")


# Categorical columns

dayofweek = tf.contrib.layers.sparse_column_with_integerized_feature("dayofw
eek", bucket_size=8)

dayofweek_emb=tf.contrib.layers.embedding_column(dayofweek, dimension=3)


x = tf.contrib.layers.sparse_column_with_integerized_feature("x", bucket_siz
e=62)

x_emb=tf.contrib.layers.embedding_column(x, dimension=7)


y = tf.contrib.layers.sparse_column_with_integerized_feature("y", bucket_siz
e=45)

y_emb=tf.contrib.layers.embedding_column(y, dimension=7)


# Define input function:

def input_fn(df):

  """Input builder function."""

  # Creates a dictionary mapping from each continuous feature column name (k
) to

  # the values of that column stored in a constant Tensor.

  continuous_cols = {k: tf.constant(df[k].values, shape=[df[k].size, 1]) for
k in CONTINUOUS_COLUMNS}
```

```
  # Creates a dictionary mapping from each categorical feature column name (
k)
  # to the values of that column stored in a tf.SparseTensor.
  categorical_cols = {
      k: tf.SparseTensor(
          indices=[[i, 0] for i in range(df[k].size)],
          values=df[k].values,
          dense_shape=[df[k].size, 1])
      for k in CATEGORICAL_COLUMNS}

  # Merges the two dictionaries into one.
  feature_cols = dict(continuous_cols)
  feature_cols.update(categorical_cols)
  # Converts the label column into a constant Tensor.
  label = tf.constant(df[LABEL_COLUMN].values)
  # Returns the feature columns and the label.
  return feature_cols, label


# Defining the optimizer
rmsprop = tf.train.RMSPropOptimizer(learning_rate=0.001)


# Defining the model
model = tf.contrib.learn.DNNRegressor(feature_columns=[dayofweek_emb,x_emb,
y_emb, avgtemp, precipitation],
                                      hidden_units=[32,32,32],
                                      dropout=0.1,
                                      activation_fn=tf.nn.elu,
                                      model_dir=model_dir,
                                      optimizer=rmsprop)


# Training our model (2015):
model.fit(input_fn=lambda: input_fn(df_train),steps=2000)
```

```
# Evaluating our model:
results = model.evaluate(input_fn=lambda: input_fn(df_test), steps=1)
for key in sorted(results):
     print("%s: %s" % (key, results[key]))


# Evaluating with totally new data (2016/02):
cross_results = model.evaluate(input_fn=lambda: input_fn(df_cross), steps=1)
for key in sorted(cross_results):
     print("%s: %s" % (key, cross_results[key]))


# Evaluating with totally new data (2016/03):
cross_results2 = model.evaluate(input_fn=lambda: input_fn(df_cross2), steps=
1)
for key in sorted(cross_results2):
     print("%s: %s" % (key, cross_results2[key]))



# Print some predictions
results2 = list(classifier.predict(input_fn=lambda: input_fn(df_positive),as
_iterable=True))
print('Predictions: {}'.format(str(results2)))
```

iii. Πηγαίος Κώδικας - Web Service

```
# # New York Cab Dataset Deap Learning Model Web Service


# First we import he modules we are going to use
import tempfile
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.contrib import learn
import web
import json
from sklearn.model_selection import train_test_split
from geojson import Point, Feature, FeatureCollection


# Define Variables
model_dir = "/path/to/model_directory"
step=0.009
x_min=int(-74.25/step)
x_max=int(-73.7/step)
y_min=int(40.5/step)
y_max=int(40.9/step)


# Import the csv
COLUMNS = ["dayofweek", "x", "y", "avgtemp", "precipitation", "gi"]
LABEL_COLUMN = "gi"


#df = pd.read_csv(train_file, names=COLUMNS, skipinitialspace=False, skiprow
s=1)


# Let's define the feature columns that we are going to use:
```

```
CATEGORICAL_COLUMNS = ["dayofweek","x", "y"]

CONTINUOUS_COLUMNS = ["avgtemp", "precipitation"]


# Continuous columns.

avgtemp = tf.contrib.layers.real_valued_column("avgtemp")

precipitation =  tf.contrib.layers.real_valued_column("precipitation")


# Categorical columns

dayofweek = tf.contrib.layers.sparse_column_with_integerized_feature("dayofw
eek", bucket_size=8)

dayofweek_emb=tf.contrib.layers.embedding_column(dayofweek, dimension=3)


x = tf.contrib.layers.sparse_column_with_integerized_feature("x", bucket_siz
e=62)

x_emb=tf.contrib.layers.embedding_column(x, dimension=7)


y = tf.contrib.layers.sparse_column_with_integerized_feature("y", bucket_siz
e=45)

y_emb=tf.contrib.layers.embedding_column(y, dimension=7)


# Define input function:

def input_fn(df):

  """Input builder function."""

  # Creates a dictionary mapping from each continuous feature column name (k
) to

  # the values of that column stored in a constant Tensor.

  continuous_cols = {k: tf.constant(df[k].values, shape=[df[k].size, 1]) for
k in CONTINUOUS_COLUMNS}

  # Creates a dictionary mapping from each categorical feature column name (
k)

  # to the values of that column stored in a tf.SparseTensor.

  categorical_cols = {

      k: tf.SparseTensor(
```

```
          indices=[[i, 0] for i in range(df[k].size)],
          values=df[k].values,
          dense_shape=[df[k].size, 1])
      for k in CATEGORICAL_COLUMNS}

  # Merges the two dictionaries into one.
  feature_cols = dict(continuous_cols)
  feature_cols.update(categorical_cols)
  # Converts the label column into a constant Tensor.
  label = tf.constant(df[LABEL_COLUMN].values)
  # Returns the feature columns and the label.
  return feature_cols, label


# Defining our model


# Defining the optimizer
rmsprop = tf.train.RMSPropOptimizer(learning_rate=0.001)
# Defining the classifier - Using 3 hidden units, 32 neurons each
model = tf.contrib.learn.DNNRegressor(feature_columns=[dayofweek_emb,x_emb,
y_emb, avgtemp, precipitation],
                                      hidden_units=[32,32,32],
                                       dropout=0.1,
                                      activation_fn=tf.nn.elu,
                                      model_dir=model_dir,
                                      optimizer= rmsprop)


# Predict:
inputL=list()
def predict(temp,prec,dayofwe):
    for x in range(x_min,x_max):
        for y in range(y_min,y_max):
```

```
            #temp=np.array([dayofwe,k,y,temp,prec])
            #np.append(input, [temp], axis=0)
            inputL.append([dayofwe,x,y,temp,prec,0])
    df_pred = pd.DataFrame.from_records(inputL,columns=COLUMNS)
    results2 = model.predict(input_fn=lambda: input_fn(df_pred),as_iterable=
False)

    return results2.tolist()

urls = (
  '/', 'index'
)

class index:
    def GET(self):
        global inputL,step
        user_data = web.input()
        temp_get=float(user_data.temp)
        precip_get=float(user_data.precip)
        day_get=int(user_data.day)
        results=predict(temp_get,precip_get,day_get)
        data_all = []
        for i in range(0,len(results)):
            value=results[i]+0.2
            if value>0:
                feature = {}
                feature['type'] = 'Feature'
                feature['geometry'] = {'type': 'Point','coordinates': [input
L[i][1]*step-step/2,inputL[i][2]*step+step/2]}
                feature['properties'] =  {'weight': value}
                data_all.append(feature)
        inputL=list()
```

```
        #pyDict = [{"lat": 40.743, "lng":-73.989, "count": 40}]
        web.header('Content-Type', 'application/json')
        web.header("Access-Control-Allow-Origin", "*")
        web.header("Access-Control-Expose-Headers", "Access-Control-Allow-Or
igin")
        web.header("Access-Control-Allow-Headers", "Origin, X-Requested-With
, Content-Type, Accept")
        return json.dumps(FeatureCollection(data_all))


class MyApplication(web.application):
    def run(self, port=9999, *middleware):
        func = self.wsgifunc(*middleware)
        return web.httpserver.runsimple(func, ('localhost', port))


if __name__ == "__main__":
    app = MyApplication(urls,globals())
    app.run()
```

iv. Πηγαίος Κώδικας – Openlayers map

```
<!doctype html>

<html lang="en">

  <head>

    <link rel="stylesheet" href="https://openlayers.org/en/v4.0.1/css/ol.css
" type="text/css">

    <style>

      .map {

        height: 600px;

        width: 100%;

      }

    </style>

    <script src="https://openlayers.org/en/v4.0.1/build/ol.js" type="text/ja
vascript"></script>

    <!-- The line below is only needed for old environments like Internet Ex
plorer and Android 4.x -->

    <script src="https://cdn.polyfill.io/v2/polyfill.min.js?features=request
AnimationFrame,Element.prototype.classList,URL"></script>

        <script src="scripts/jquery-3.2.0.min.js"></script>

    <title>TaxiModel</title>

  </head>

  <body>

    <h2>New York city yellow cabs GI*</h2>

    <div id="map" class="map"></div>

        <form>

      <label>Temperature (Celcius)</label>

      <input id="temp" type="range" min="-13" max="31" step="1" value="10"/>

      <label>Precipitation (mm)</label>

      <input id="precip" type="range" min="0" max="40" step="0.1" value="1"/
>

          <select name="Day of Month" id="day">

                <option value="1" selected>Monday</option>

                <option value="2">Tuesday</option>
```

```
                <option value="3" >Wednesday</option>
                <option value="4">Thursday</option>
                <option value="5">Friday</option>
                <option value="6">Saturday</option>
                <option value="7">Sunday</option>
          </select><br>
          <input type="text" id="temp_text" disabled>
          <input type="text" id="precip_text" disabled>
    </form>
    <script type="text/javascript">
        var styleCache = {};
      var styleFunction = function(feature) {
        // 2012_Earthquakes_Mag5.kml stores the magnitude of each earthquake
in a
        // standards-violating <magnitude> tag in each Placemark.  We extrac
t it from
        // the Placemark's name instead.
        //var name = feature.get('name');
        var weight = parseFloat(feature.get('weight'));
                //alert(weight);
        var radius = weight*10;
        var style = styleCache[radius];
        if (!style) {
          style = new ol.style.Style({
            image: new ol.style.Circle({
              radius: radius,
              fill: new ol.style.Fill({
                color: 'rgba(255, 0, 0, 0.5)'
              }),
              stroke: new ol.style.Stroke({
                color: 'rgba(255, 20, 0, 0.3)',
                width: 1
```

```
            })
          })
        });
        styleCache[radius] = style;
      }
      return style;
    };


              var raster = new ol.layer.Tile({
      source: new ol.source.Stamen({
        layer: 'toner'
      })
    });
       var temp = document.getElementById('temp');
   var precip = document.getElementById('precip');
       var temp_text = document.getElementById('temp_text');
   var precip_text = document.getElementById('precip_text');
       precip_text.value=precip.value;
       temp_text.value=temp.value;
       var day = document.getElementById('day');
       var url='http://localhost:9999/?temp='+temp.value+'&precip='+precip.
value+'&day='+day.value
       var vectorSource = new ol.source.Vector({
       format: new ol.format.GeoJSON({
              extractStyles: false
              }),
       url: function(extent, resolution, projection){return 'http://localho
st:9999/?temp='+temp.value+'&precip='+precip.value+'&day='+day.value}
              });
       var vector = new ol.layer.Vector({
              source: vectorSource,
              style: styleFunction
```

```
        });
        temp.addEventListener('input', function() {
                temp_text.value=temp.value;
        vectorSource.clear();
      });

     precip.addEventListener('input', function() {
                precip_text.value=precip.value;
                vectorSource.clear();
      });

        day.addEventListener('input', function() {
                vectorSource.clear();
      });
      var map = new ol.Map({
        target: 'map',
        layers: [raster,vector],
        view: new ol.View({
          center: ol.proj.fromLonLat([-73.9767403,40.7800209]),
          zoom: 13
        })
      });
    </script>
  </body>
</html>
```