Αναφορά

Σε αυτή την αναφορά θα περιγράψουμε το πρόβλημα (την άσκηση), την προσέγγισή μας, τις μεθόδους που χρησιμοποιήθηκαν και τα αποτελέσματα που προέκυψαν.

Εκφώνηση:

Εργαστηριακή Άσκηση 1 Έχουμε διαθέσιμα τα δεδομένα μιας αλυσίδας καταστημάτων και είναι σημαντικό να προβλέψουμε τις πωλήσεις στα διάφορα τμήματα των υποκαταστημάτων. Δεδομένου ότι μπορεί να υπάρχουν πολλοί παράγοντες που μπορούν να επηρεάσουν τις πωλήσεις για κάθε τμήμα, καθίσταται επιτακτική ανάγκη να προσδιορίσουμε τους βασικούς παράγοντες που συμβάλλουν στην προώθηση των πωλήσεων και τις χρησιμοποιούν για την ανάπτυξη ενός μοντέλου που μπορεί να βοηθήσει στην πρόβλεψη των πωλήσεων με κάποια ακρίβεια.

Έχουμε εβδομαδιαία δεδομένα πωλήσεων για 45 καταστήματα και 98 τμήματα για μια περίοδο 3 ετών. Επιπλέον, έχουμε συγκεκριμένες πληροφορίες για το κατάστημα και τη γεωγραφία, όπως το μέγεθος του καταστήματος, το ποσοστό ανεργίας, τη θερμοκρασία, κλπ. Χρησιμοποιώντας αυτούς τους παράγοντες, **χρειαζόμασταν να αναπτύξουμε ένα μοντέλο παλινδρόμησης που μπορεί να προβλέψει τις πωλήσεις και είναι επίσης υπολογιστικά αποτελεσματικό και επεκτάσιμο.** Θα πρέπει να ληφθεί υπόψιν η εποχικότητα αλλά άλλες σημαντικές μεταβλητές όπως οι διακοπές και ο τύπος τμήματος για να έχουμε καλύτερη ακρίβεια.

Υπόμνημα Dataset 1

Store: Μοναδικός κωδικός καταστήματος
Dept: Κωδικός Τμήματος
Date: Ημερομηνία
Weekly Sales: Αριθμός εβδομαδιαίων πωλήσεων
IsHoliday: True/False, ανάλογα αν πρόκειται για ημέρα αργίας ή όχι
Temperature: Θερμοκρασία
Unemployment: Ποσοστό ανεργίας

Το κυρίως πρόβλημα, δηλαδή η κατασκευή ενός αποτελεσματικού και επεκτάσιμου μοντέλου παλινδρόμησης μπορεί να χωριστεί στα εξης μικρότερα προβλήματα:

1. Προετοιμασία των δεδομένων που μας δόθηκαν
2. Κατασκευή μοντέλου-πρόβλεψη-αξιολόγηση
3. Βελτίωση μοντέλου ή επιστροφή στο 2

Με βάση αυτά τα υποπροβλήματα, παρακάτω παρουσιάζουμε την προσέγγιση και επίλυσή τους και καταλήγουμε στο τελικό μοντέλο. Όλες οι γραμμές στο *project.py* αρχείο έχουν σχολιαστεί αλλά παραθέτουμε και κάποιες από αυτές όπου κρίνουμε ότι είναι απαραίτητο.

Προετοιμασία Δεδομένων

Για οποιονδήποτε αλγόριθμο παλινδρόμησης αποφασίσουμε να χρησιμοποιήσουμε, χρειάζεται να κάνουμε κάποιες αλλαγές στα αρχεία .csv που μας δόθηκαν. Θα πρέπει σίγουρα να κάνουμε τα εξείς:

Να διαβάσουμε τα αρχεία:
Για την εισαγωγή και την επεξεργασία των αρχείων χρησιμοποιούμε τη βιβλιοθήκη pandas

Να ελέγξουμε για τιμές που λείπουν:
Αν για παράδειγμα κάποιες τιμές του Temperature λείπουν τότε σημαίνει ότι δεν μπορούμε να χρησιμοποιήσουμε εκείνες τις γραμμές.

Εφόσον λείπουν τιμές, να αποφασίσουμε πως θα τις χειριστούμε:
Παρατηρούμε ότι στο αρχείο *features.csv* που μας δόθηκε λείπουν κάποιες τιμές από τη στήλη *Unemployment*. Προκειμένου να διατηρήσουμε τις γραμμές για να τις χρησιμοποιήσουμε για την εκπαίδευση του αλγορίθμου τις “γεμίσαμε” με τον μέσο όρο των υπολοίπων τιμών εκείνης της στήλης.

```
feat_data['Unemployment'] =
feat_data['Unemployment'].fillna(np.mean(feat_data['Unemployment'] ))
```

Να συγχωνεύσουμε τα δύο αρχεία csv που μας δόθηκαν:
Συγκεκριμένα των *features.csv* και *train.csv*. Χρειαζόμαστε ένα αρχείο με βάση το οποίο θα "εκπαιδεύσουμε" τον αλγόριθμο που θα επιλέξουμε, επομένως χρειάζεται να βάλουμε όλα τα δεδομένα μας στο ίδιο αρχείο.

```
# Merge feature and training data in all_data
all_data = pd.merge(feat_data, train_data, on = ['Store', 'Date',
'IsHoliday'], how = 'inner')
```

Για την συγχώνευση κάνουμε inner join στις κοινές στήλες των δύο αρχείων, *Store-Date-IsHoliday.*

Παρατηρούμε ωστόσο ότι με αυτόν τον τρόπο, το αρχείο στο οποίο καταλλήγουμε έχει μόνο 7 στήλες και καμία γραμμή. Αυτό οφείλεται στον τρόπο με τον οποίο δουλεύει η inner join συγχώνευση, δηλαδή ταιριάζοντας τις κοινές γραμμές στις στήλες τις οποίες έχουμε ορίσει, καθώς και στο γεγονός ότι οι στήλες *Date* των δύο αρχείων έχουν διαφορετική μορφοποίηση.

Features.csv: 5/2/10
Train.csv: 2010-02-05

Αυτό σημαίνει ότι καμία γραμμή στην στήλη Date των δύο αρχείων δεν ταιριάζει επομένως πριν τη συγχώνευση θα πρέπει να

Μετατρέψουμε τις στήλες Date των δύο αρχείων στο ίδιο format:

Αποφασίσαμε ότι θα μετατρέψουμε τις στήλες Date από Objects/Strings σε datetime-objects (datetime64[ns])

```
train_data['Date'] = pd.to_datetime(train_data.Date)

# dayfirst=True γιατί τα θέλουμε και τα δύο στο ευρωπαικό format (η
datetime κάνει default στο αμερικάνικο και μας δημιουργεί προβλήματα)
feat_data['Date'] = pd.to_datetime(feat_data.Date, dayfirst=True)
```

Αυτό λύνει αρχικά το πρόβλημα της συγχώνευσης ωστόσο μας δημιουργεί ένα καινούργιο πρόβλημα το οποίο εξηγούμε παρακάτω.

Να δημιουργήσουμε ένα αρχείο test.csv για τον έλεγχο της επεκτασιμότητας του αλγορίθμου.

Overfitting refers to a model that models the training data too well.

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

Ως "επεκτασιμότητα" του αλγορίθμου ορίζουμε την ικανότητα του να παράγει αρκετά ακριβή αποτελέσματα σε κάποιο dataset πάνω στο οποίο δεν έχει εκπαιδευτεί. Για τον λόγο αυτό, χωρίσαμε το αρχείο train σε δύο διαφορετικά αρχεία. Στο πρόγραμμά μας αναφέρονται ως "train_data" και "test_data". Συγκεκριμένα, το "test_data" περιέχει τις ~100.000 τελευταίες γραμμές (ταξινομημένες με βάση την ημερομηνία) του αρχείου *train.csv* . Ο σκοπός ήταν να χρησιμοποιήσουμε το αρχείο test_data για προβλέψεις ενώ ο αλγόριθμος μας έχει ήδη εκπαιδευτεί με βάση το "train_data".

Να αντικαταστήσουμε τη στήλη Date των συγχωνευμένων αρχείων με μία αντίστοιχη μετρική.

Ο λόγος που το κάνουμε αυτό είναι ότι στους αλγόριθμους που διαλέξαμε δεν μπορούμε να δώσουμε ως "X" μεταβλητή ένα datetime object ή object γενικότερα.
Επομένως δημιουργήσαμε μια καινούργια στήλη η οποία θα αντικαθιστούσε την "Date" και την ονομάσαμε "Days2Christmas". Περιμένουμε ότι η ημερομηνία θα επηρεάζει τα αποτελέσματα όσο περισσότερο πλησιάζουμε στις γιορτές "IsHoliday" οι οποίες βρίσκονται κοντά στα Χριστούγεννα. Η "Days2Christmas επομένως περιέχει τιμές οι οποίες μετράνε το πόσες μέρες απομένουν για τα Χριστούγεννα.

Να εξαλείψουμε τις αρνητικές τιμές της "Y" μεταβλητής.

Η "Y" μεταβλητή, δηλαδή η μεταβλητή που θέλουμε να προβλέψουμε δεν επιτρέπεται να έχει αρνητικές τιμές. Σε αυτήν την περίπτωση αντικαθιστούμε αυτές τις τιμές με 0. Εναλλακτικά θα μπορούσαμε να διαγράψουμε εκείνες τις γραμμές όπου το "Weekly_Sales" < 0.

Να ελέγξουμε τη συσχέτιση μεταξύ στηλών.

Εάν δύο ή περισσότερες μεταβλητές "X" έχουν μεγάλη συσχέτιση (correlation) μεταξύ τους, τότε αυτό θα επηρεάσει αρνητικά τα αποτελέσματά μας. Για τον λόγο αυτό δημιουργήσαμε ένα "heatmap", το οποίο αποθηκεύσαμε ξεχωριστά στο αρχείο *correlation.csv .*
Ο κανόνας αυτός είναι περισσότερο εμπειρικός, δηλαδή δεν γνωρίζαμε πριν ελέγξουμε τα αποτελέσματα των αλγορίθμων, τι σημαίνει "μεγάλη" συσχέτιση, πέρα από το νούμερο "1" στο heatmap το οποίο σημαίνει 100% συσχέτιση μεταξύ δύο μεταβλητών (απαράδεκτο).

Κατασκευή μοντέλου-Πρόβλεψη-Αξιολόγηση

Η κατασκευή ενός μοντέλου μηχανικής μάθησης χωρίζεται σε τέσσερα βήματα:

1.Ορισμός “X”, “Y” μεταβλητών:

Στη δική μας περίπτωση η μεταβλητή που θέλουμε να προβλέψουμε, η μεταβλητή “Y” είναι το “Weekly_Sales”

Οι υπόλοιπες μεταβλητές είναι αυτές που θα χρησιμοποιήσουμε για την πρόβλεψη του “Y”, οι “X” μεταβλητές, που στη δική μας περίπτωση βρίσκονται στην μεταβλητή “select_columns”

```
select_columns = all_data.columns.difference(['Weekly_Sales', 'Date',
'Unemployment', 'Temperature'])
```

2.Κατασκευή και εκπαίδευση του μοντέλου (linear regression):

Εφόσον έχουμε κάνει όλη την προετοιμασία που αναφέρουμε παραπάνω, η κατασκευή και εκπαίδευση του πρώτου μοντέλου που επιλέξαμε γίνεται με δύο εντολές:

```
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

3. Προβλέψεις

Αφού εκπαιδεύσουμε το μοντέλο μας μπορούμε να του ζητήσουμε να κάνει μια πρόβλεψη με βάση τις “X” μεταβλητές για το “Y”

Linear Regression:

```
y_train_pred=linreg.predict(X_train)
```

Random Forest:

```
predict_rfr = rfr.predict(X_test)
```

Προκειμένου να ελέγξουμε την επεκτασιμότητα του μοντέλου το ίδιο κάνουμε και για το “test_data”, χωρίς να το έχουμε εκπαιδεύσει με βάση αυτό. Οι “X” μεταβλητές αυτού του dataframe βρίσκονται στην τιμή “X2_test” και οι “Y” στην τιμή “y2_test”

```
X2_test = test_data[select_columns]
```

```
y2_test = test_data['Weekly_Sales']
```

Οι αντίστοιχες προβλέψεις είναι οι εξείς:

Linear Regression:

```
y_test_pred = linreg.predict( X2_test )
```

Random Forest Regression:

```
predict_test = rfr.predict(X2_test)
```

4. Αξιολόγηση αποτελεσμάτων:

Ελέγχουμε αν οι προβλέψεις του μοντέλου μας ταιριάζουν με τις πραγματικές τιμές της “Y” μεταβλητής.

```
accuracy = np.round(linreg.score( X_test, y_test) * 100, 3))
```

Η τιμή του “accuracy” είναι αυτή που για εμάς κρίνει την ακρίβεια του μοντέλου. Μια τιμή κοντά στο 1 ή στο 100% θα έδειχνε ότι το μοντέλο μας μπορεί να κάνει προβλέψεις με μεγάλη ακρίβεια. Η αντίστοιχη τιμή για το μοντέλο “Random Forest” είναι η

```
forest = ('Acc:', np.round(rfr.score(X_test, y_test) * 100, 2))
```

Για την αξιολόγηση της επεκτασιμότητας του μοντέλου ανανεώνουμε τις τιμές “accuracy” και “forrest” αφού κάνουμε πρόβλεψη με βάση αυτές.

Βελτίωση μοντέλου ή επιστροφή στο 2

Σε αυτό το σημείο θα πρέπει να έχουμε όσα στοιχεία χρειαζόμαστε για να αξιολογήσουμε το μοντέλο και να το βελτιώσουμε.

Η αξιολόγηση χωρίζεται σε δύο σκέλη:

1. Ακρίβεια
2. Επεκτασιμότητα

Η αξιολόγηση της ακρίβειας είναι μια πειραματική διαδικασία καθώς οι έννοιες “υψηλή” και “χαμηλή” ακρίβεια δεν έχουν αντικειμενική σημασία. Αυτό που μπορούμε να κάνουμε είναι να καταγράψουμε τα αποτελέσματα, να προσπαθήσουμε να βελτιώσουμε το μοντέλο μας και να δούμε αν η ακρίβεια των προβλέψεων αυξάνεται ή μειώνεται.

Ξεκινήσαμε με το μοντέλο Linear Regression. Η ακρίβειά του ήταν αρκετά χαμηλή,της τάξεως 3%. Η βελτίωση αυτού του μοντέλου θα ήταν πολύ δύσκολη, καθώς το μόνο που μπορούμε να κάνουμε είναι να χρησιμοποιήσουμε διαφορετικό συνδυασμό “X” μεταβλητών για την πρόβλεψη του “Y”. Παρόλα αυτά, κρίναμε ότι δεν μπορούσαμε να βελτιώσουμε σημαντικά τις προβλέψεις μας, επομένως αποφασίσαμε να κατασκευάσουμε το μοντέλο Random Forest.

Το μοντέλο Random Forest έκανε αμέσως πολύ καλύτερες προβλέψεις, κοντά στο 95%. Προκειμένου να ελέγξουμε αν μπορούμε να πάρουμε καλύτερα αποτελέσματα δοκιμάσαμε να αλλάξουμε τον συνδυασμό των “X” μεταβλητών, καθώς και των estimators και depth του δάσους. Τα αποτελέσματά μας καταγράφονται στο αρχείο *forest_evaluation.txt.*

Βρήκαμε ότι ο καλύτερος συνδυασμός (ακρίβειας) / (χρόνου εκτέλεσης προγράμματος) είναι οι τιμές estimators = 40, depth = 20 και οι στήλες Day-Days2Christmas-Dept-IsHoliday-Store-Year ως “X” μεταβλητές.

Η μεγαλύτερη ακρίβεια που πετύχαμε είναι 96,69%

Τέλος, το μοντέλο μας είναι επεκτάσιμο καθώς η ακρίβεια της πρόβλεψης πάνω στο test_data είναι πολύ κοντά (διαφορά τάξης 1-2%) σε σχέση με την ακρίβεια της πρόβλεψης για το all_data