ΕΞΟΡΥΞΗ ΓΝΩΣΗ ΑΠΟ Β.Δ. & ΑΠΟ ΤΟΝ Π. ΙΣΤΟ

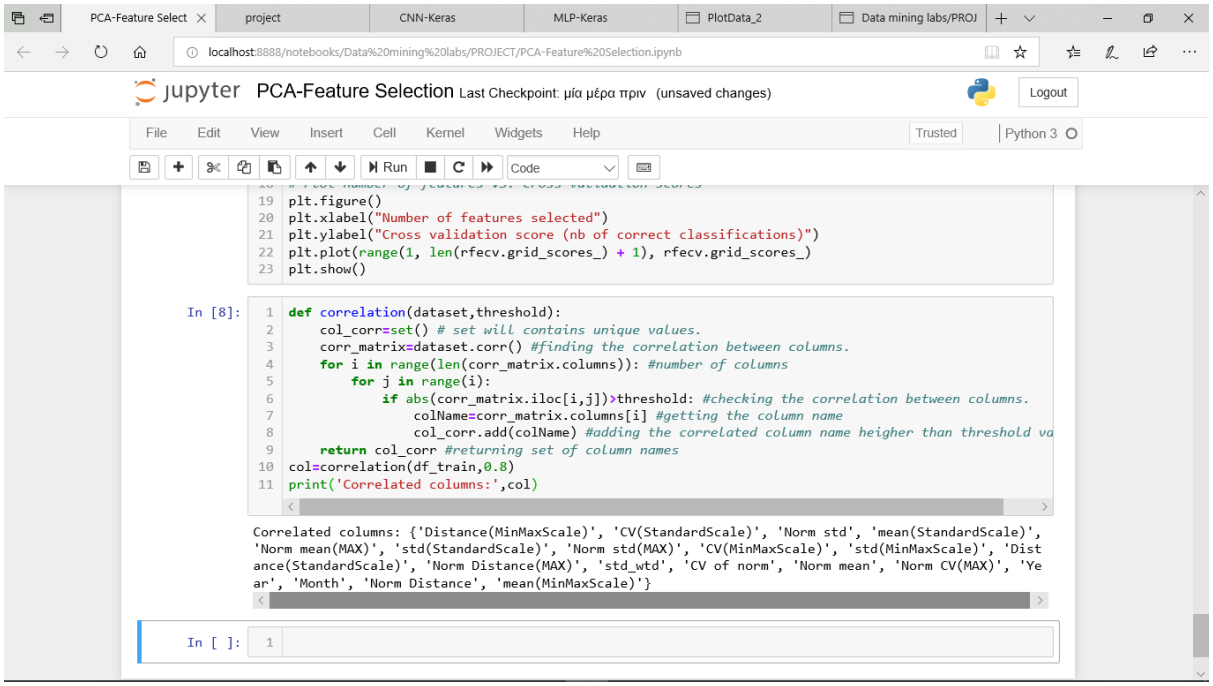
ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ ΑΚΑΔΗΜΑΪΚΟΥ ΕΤΟΥΣ 2018-2019

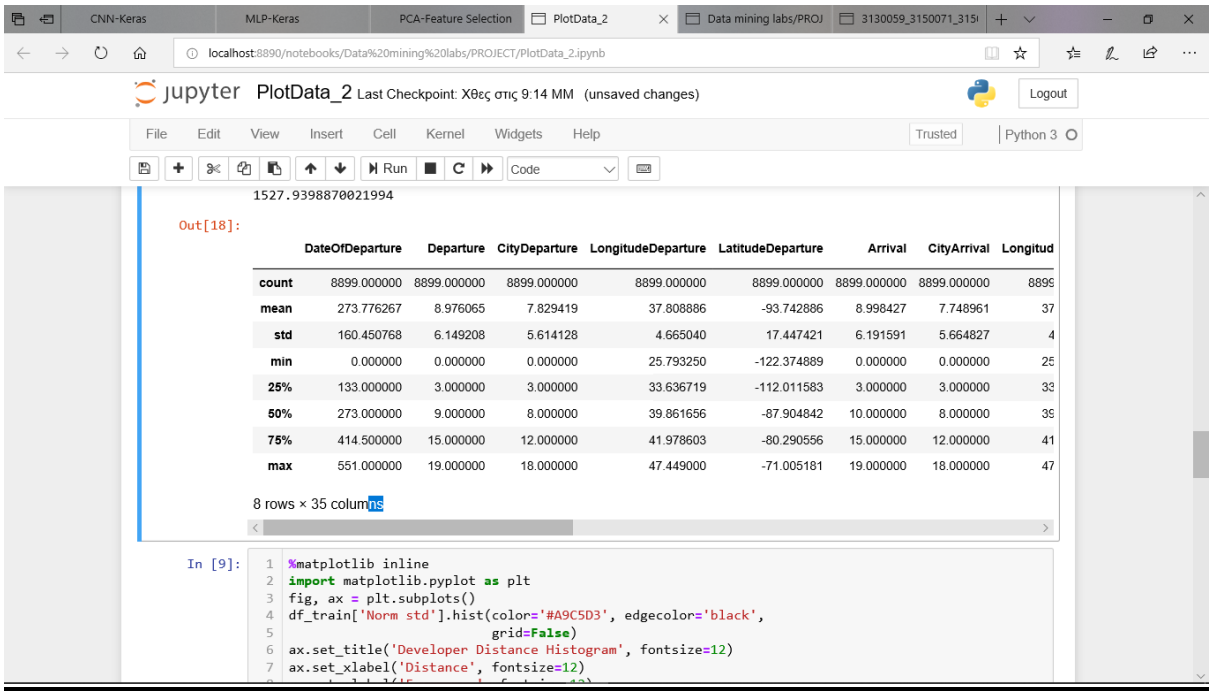
ΜΕΛΗ : 1. ΙΑΚΩΒΟΣ ΕΥΔΑΙΜΩΝ 3130059
2. ΚΩΝΣΤΑΝΤΙΝΟΣ ΚΕΧΡΗΣ 3150071
3. ΑΘΑΝΑΣΙΟΣ ΜΕΜΤΣΑΣ 3150103

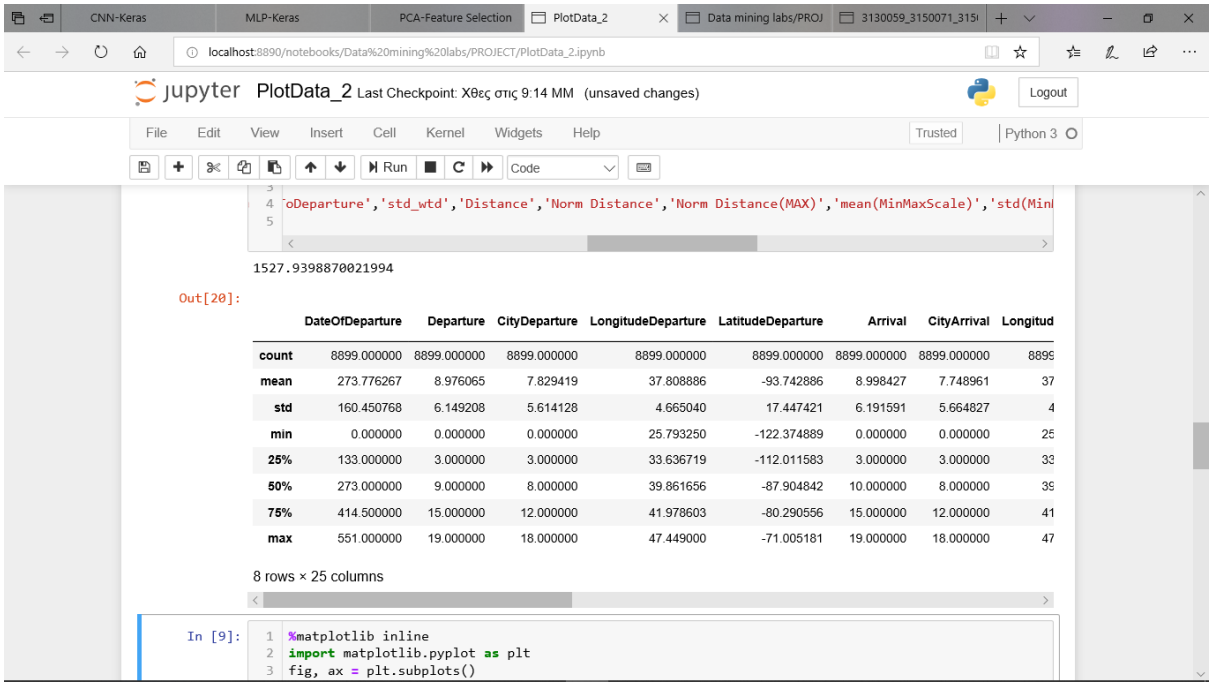
ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΚΑΙ ΠΡΟΕΤΟΙΜΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

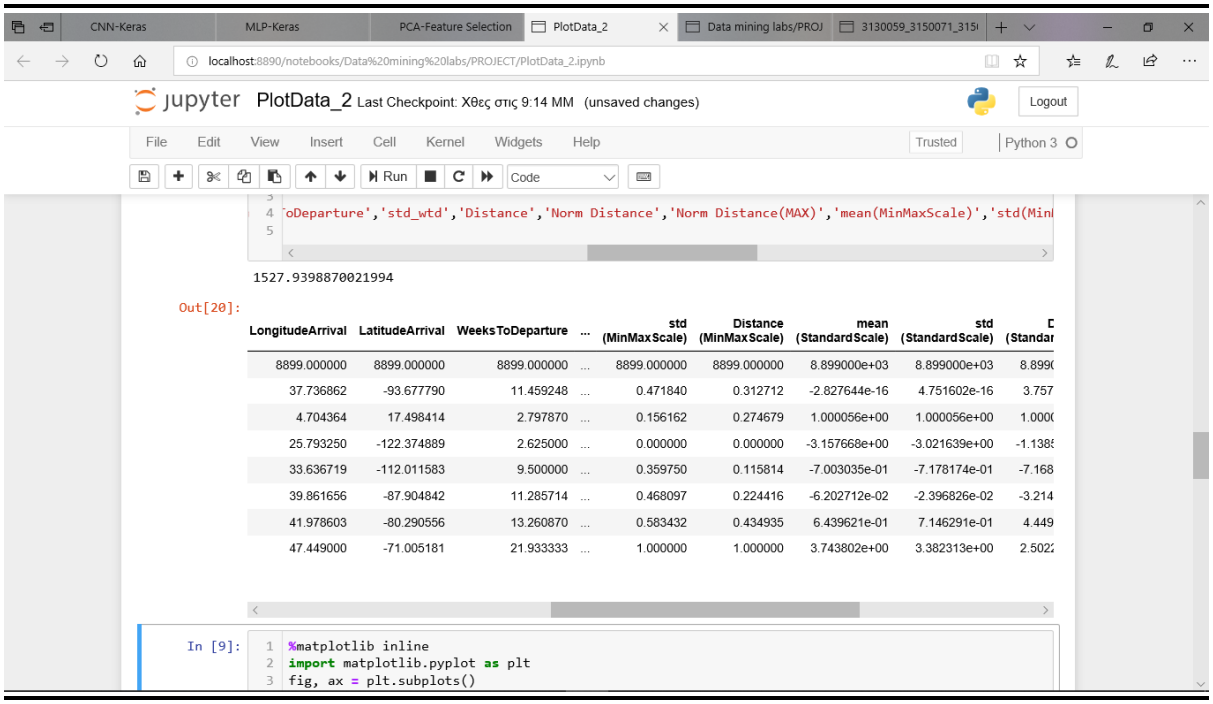
Αρχικά, επικεντρωθήκαμε στην σωστή διαχείριση και προεπεξεργασία των δεδομένων μας καθώς αποτελεί ίσως το βασικότερο κομμάτι ώστε να πετύχουμε να φτιάξουμε έναν κατηγοριοποιητή που θα έχει καλά αποτελέσματα. Πρωτίστως, σκεφτήκαμε να εισάγουμε ορισμένα καινούργια features, από τα οποία ορισμένα ίσως να ήταν επικαλυπτόμενα με κάποια που είχαμε ήδη. Ανάλογα την μέθοδο που θα δοκιμάζουμε θα κρατάμε και τα κατάλληλα δεδομένα που θα μας δίνουν το καλύτερο σκορ έτσι ώστε να μην έχουμε επικαλυπτόμενα χαρακτηριστικά. Τα επιπρόσθετα χαρακτηριστικά που εισάγαμε είναι η απόσταση(Distance), η οποία υπολογίζεται από τα δοσμένα longitude και latitude του Department και του Arrival. Επίσης, προσθέσαμε ένα feature Airports όπου κατηγοριοποιεί κάθε ένα sample ανάλογα με το αν το αεροδρόμιο αναχώρησης ή άφιξης ή και τα δύο ανήκουν στα 10 πιο πολυσύχναστα αεροδρόμια της Αμερικής σύμφωνα με στατιστικά από διάφορες πηγές. Αν ούτε το αεροδρόμιο αναχώρησης, ούτε το αεροδρόμιο άφιξης ανήκει στα 10 πιο πολυσύχναστα τότε τους βάζει την τιμή 0, άμα ένα από τα δύο αεροδρόμια ανήκει στα πιο πολυσύχναστα τότε βάζει την τιμή 1 και στην περίπτωση που και τα 2 αεροδρόμια(αναχώρησης και άφιξης) ανήκουν στα πιο πολυσύχναστα τότε βάζει την τιμή 2. Ένα ακόμα feature που προσθέσαμε ήταν το IsHolidays το οποίο είναι μία μεταβλητή που δείχνει αν η ημερομηνία αναχώρησης αποτελεί ημερομηνία μέσα στις ημερομηνίες με την πιο πολύ ζήτηση συνήθως. Δηλαδή άμα είναι η ημερομηνία αναχώρησης κοντά στην ημερομηνία των Χριστουγέννων ή της Πρωτοχρονιάς, κοντά στην ημερομηνία της 4ης Ιουλίου, άμα είναι Παρασκευή είτε Σάββατο είτε Κυριακή μέσα στους καλοκαιρινούς μήνες(Ιούλιο,Αύγουστο) όπου παρατηρείται να ταξιδεύει πιο πολύς κόσμος. Επίσης, ελέγχεται άμα η ημερομηνία είναι λίγες μέρες πριν την γιορτή των Ευχαριστιών, την γιορτή

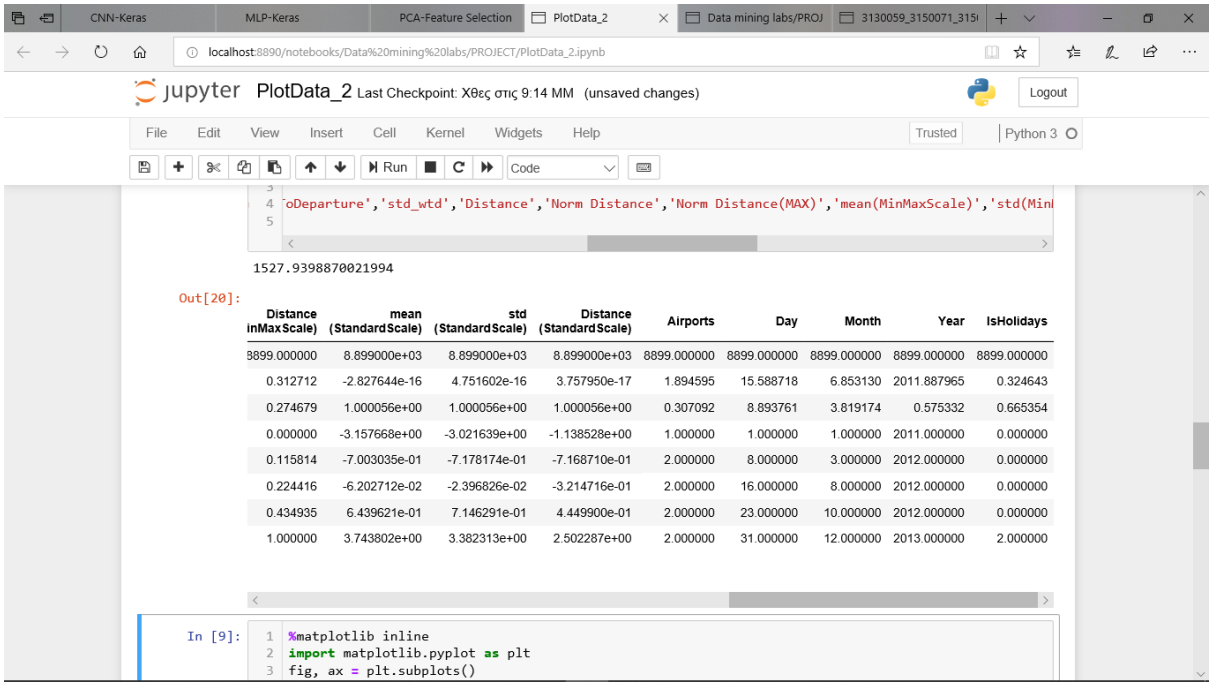
της Labor Day και την γιορτή της Memorial Day. Άμα η ημερομηνία αναχώρησης ανήκει μέσα σε μία από αυτές τις ημερομηνίες που αναφέρθηκαν παραπάνω τότε το χαρακτηριστικό IsHolidays για το συγκεκριμένο παράδειγμα παίρνει την τιμή 1 αλλιώς την τιμή 0. Το χαρακτηριστικό IsHolidays παίρνει την τιμή 2 στις περιπτώσεις όπου η ημερομηνία αναχώρησης είναι κοντά στην ημερομηνία της γιορτής των Ευχαριστιών ή κοντά στις ημερομηνίες των διακοπών των Χριστουγέννων και Πρωτοχρονιάς, όπου παρατηρείται να έχουν την πιο μεγάλη ζήτηση σαν ημερομηνίες. Μία ακόμα σκέψη μας ήταν να σπάσουμε το χαρακτηριστικό της ημερομηνίας αναχώρησης σε Ημέρα, Μήνα και Έτος, βέβαια δοκιμάζοντας αυτή την προσέγγιση σε όσους κατηγοριοποιητές υλοποιήσαμε μας έδωσε χειρότερα αποτελέσματα από ότι όταν χρησιμοποιούσαμε την μεταβλητή DateOfDeparture. Οι επικαλυπτόμενες μεταβλητές που παρατηρήσαμε είναι η μεταβλητή CityDeparture με τη μεταβλητή Departure, η μεταβλητή CityArrival με τη μεταβλητή Arrival, οι μεταβλητές Longitude,Latitude Arrival και Destination με τη μεταβλητή Distance που δημιουργήσαμε καθώς και η μεταβλητή WeeksToDeparture με τη μεταβλητή std_wtd. Ανάλογα με το μοντέλο μας διαλέγουμε και την μεταβλητή που μας δίνει τα καλύτερα αποτελέσματα. Τις ποσοτικές μας μεταβλητές τις κανονικοποιήσαμε με τον Standard Scaler του Sklearn είτε με τον MinMaxScaler. Η πρώτη μέθοδος ουσιαστικά μετατρέπει την τυπική απόκλιση των τιμών ενός χαρακτηριστικού σε 1 και την μέση τιμή σε 0, ενώ η δεύτερη μέθοδος ουσιαστικά κανονικοποιεί τις τιμές των χαρακτηριστικών μεταξύ του διαστήματος [0,1]. Παρατηρήσαμε μια ανισορροπία στις κλάσεις μας γεγονός που προσπαθήσαμε να το προσπεράσουμε με την μέθοδο του oversampling που ουσιαστικά αντιγράφουμε τα ήδη samples των μειονεκτικών κλάσεων ώστε να έχουν όλες οι κλάσεις παρόμοιο αριθμό παραδειγμάτων. Αυτή η μέθοδος όμως μας δημιούργησε προβλήματα καθώς ουσιαστικά πέσαμε στην παγίδα της, η οποία είναι το overfitting και έτσι μειώθηκε το f1_score μας. Γενικά, το f1_score είναι μια καλή μετρική μέθοδος για να μετρήσουμε το σκορ μας σε datasets με ανισορροπία κλάσεων.

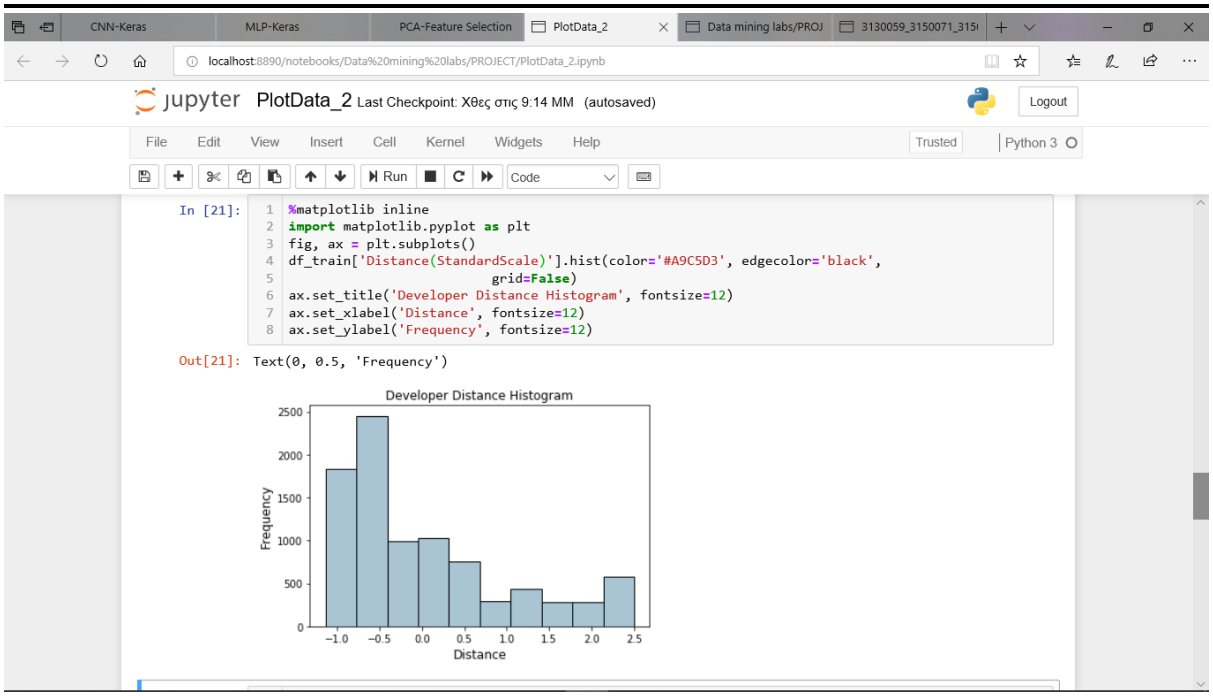

PCA-Feature Select
project
CNN-Keras
MLP-Keras
PlotData_2
Data mining labs/PROJ
localhost:8888/notebooks/Data%20mining%20labs/PROJECT/PCA-Feature%20Selection.ipynb
jupyter PCA-Feature Selection Last Checkpoint: μία μέρα πριν (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
In [8]:
def correlation(dataset,threshold):
col_corr=set() # set will contains unique values.
corr_matrix=dataset.corr() #finding the correlation between columns.
for i in range(len(corr_matrix.columns)): #number of columns
for j in range(i):
if abs(corr_matrix.iloc[i,j])>threshold: #checking the correlation between columns.
colName=corr_matrix.columns[i] #getting the column name
col_corr.add(colName) #adding the correlated column name heigher than threshold va
return col_corr #returning set of column names
col=correlation(df_train,0.8)
print('Correlated columns:',col)
Correlated columns: {'Distance(MinMaxScale)', 'CV(StandardScale)', 'Norm std', 'mean(StandardScale)', 'Norm mean(MAX)', 'std(StandardScale)', 'Norm std(MAX)', 'CV(MinMaxScale)', 'std(MinMaxScale)', 'Distance(StandardScale)', 'Norm Distance(MAX)', 'std_wtd', 'CV of norm', 'Norm mean', 'Norm CV(MAX)', 'Year', 'Month', 'Norm Distance', 'mean(MinMaxScale)'}
In [ ]:


CNN-Keras
MLP-Keras
PCA-Feature Selection
PlotData_2
Data mining labs/PROJ
localhost:8890/notebooks/Data%20mining%20labs/PROJECT/PlotData_2.ipynb
jupyter PlotData_2 Last Checkpoint: Χθες στις 9:14 ΜΜ (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
1527.9398870021994
Out[18]:
DateOfDeparture Departure CityDeparture LongitudeDeparture LatitudeDeparture Arrival CityArrival Longitud
count 8899.000000 8899.000000 8899.000000 8899.000000 8899.000000 8899.000000 8899.000000 8899
mean 273.776267 8.976065 7.829419 37.808886 -93.742886 8.998427 7.748961 37
std 160.450768 6.149208 5.614128 4.665040 17.447421 6.191591 5.664827 4
min 0.000000 0.000000 0.000000 25.793250 -122.374889 0.000000 0.000000 25
25% 133.000000 3.000000 3.000000 33.636719 -112.011583 3.000000 3.000000 33
50% 273.000000 9.000000 8.000000 39.861656 -87.904842 10.000000 8.000000 39
75% 414.500000 15.000000 12.000000 41.978603 -80.290556 15.000000 12.000000 41
max 551.000000 19.000000 18.000000 47.449000 -71.005181 19.000000 18.000000 47
8 rows × 35 columns
In [9]:
%matplotlib inline
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
df_train['Norm std'].hist(color='#A9C5D3', edgecolor='black',
grid=False)
ax.set_title('Developer Distance Histogram', fontsize=12)
ax.set_xlabel('Distance', fontsize=12)


CNN-Keras
MLP-Keras
PCA-Feature Selection
PlotData_2
Data mining labs/PROJ
3130059_3150071_315
localhost:8890/notebooks/Data%20mining%20labs/PROJECT/PlotData_2.ipynb
jupyter PlotData_2 Last Checkpoint: Χθες στις 9:14 ΜΜ (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
oDeparture','std_wtd','Distance','Norm Distance','Norm Distance(MAX)','mean(MinMaxScale)','std(Mini
1527.9398870021994
Out[20]:
DateOfDeparture Departure CityDeparture LongitudeDeparture LatitudeDeparture Arrival CityArrival Longitud
count 8899.000000 8899.000000 8899.000000 8899.000000 8899.000000 8899.000000 8899.000000 8899
mean 273.776267 8.976065 7.829419 37.808886 -93.742886 8.998427 7.748961 37
std 160.450768 6.149208 5.614128 4.665040 17.447421 6.191591 5.664827 4
min 0.000000 0.000000 0.000000 25.793250 -122.374889 0.000000 0.000000 25
25% 133.000000 3.000000 3.000000 33.636719 -112.011583 3.000000 3.000000 33
50% 273.000000 9.000000 8.000000 39.861656 -87.904842 10.000000 8.000000 39
75% 414.500000 15.000000 12.000000 41.978603 -80.290556 15.000000 12.000000 41
max 551.000000 19.000000 18.000000 47.449000 -71.005181 19.000000 18.000000 47
8 rows × 25 columns
In [9]:
%matplotlib inline
import matplotlib.pyplot as plt
fig, ax = plt.subplots()


CNN-Keras
MLP-Keras
PCA-Feature Selection
PlotData_2
Data mining labs/PROJ
3130059_3150071_315
localhost:8890/notebooks/Data%20mining%20labs/PROJECT/PlotData_2.ipynb
jupyter PlotData_2 Last Checkpoint: Χθες στις 9:14 ΜΜ (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
oDeparture','std_wtd','Distance','Norm Distance','Norm Distance(MAX)','mean(MinMaxScale)','std(Mini
1527.9398870021994
Out[20]:
LongitudeArrival LatitudeArrival WeeksToDeparture ... std (MinMaxScale) Distance (MinMaxScale) mean (StandardScale) std (StandardScale) D (Standar
8899.000000 8899.000000 8899.000000 ... 8899.000000 8899.000000 8.899000e+03 8.899000e+03 8.899
37.736862 -93.677790 11.459248 ... 0.471840 0.312712 -2.827644e-16 4.751602e-16 3.757
4.704364 17.498414 2.797870 ... 0.156162 0.274679 1.000056e+00 1.000056e+00 1.000
25.793250 -122.374889 2.625000 ... 0.000000 0.000000 -3.157668e+00 -3.021639e+00 -1.138
33.636719 -112.011583 9.500000 ... 0.359750 0.115814 -7.003035e-01 -7.178174e-01 -7.168
39.861656 -87.904842 11.285714 ... 0.468097 0.224416 -6.202712e-02 -2.396826e-02 -3.214
41.978603 -80.290556 13.260870 ... 0.583432 0.434935 6.439621e-01 7.146291e-01 4.449
47.449000 -71.005181 21.933333 ... 1.000000 1.000000 3.743802e+00 3.382313e+00 2.502
In [9]:
%matplotlib inline
import matplotlib.pyplot as plt
fig, ax = plt.subplots()


CNN-Keras
MLP-Keras
PCA-Feature Selection
PlotData_2
Data mining labs/PROJ
3130059_3150071_315
localhost:8890/notebooks/Data%20mining%20labs/PROJECT/PlotData_2.ipynb
jupyter PlotData_2 Last Checkpoint: Χθες στις 9:14 ΜΜ (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
oDeparture','std_wtd','Distance','Norm Distance','Norm Distance(MAX)','mean(MinMaxScale)','std(Mini
1527.9398870021994
Out[20]:
Distance inMaxScale) mean (StandardScale) std (StandardScale) Distance (StandardScale) Airports Day Month Year IsHolidays
8899.000000 8.899000e+03 8.899000e+03 8.899000e+03 8899.000000 8899.000000 8899.000000 8899.000000 8899.000000
0.312712 -2.827644e-16 4.751602e-16 3.757950e-17 1.894595 15.588718 6.853130 2011.887965 0.324643
0.274679 1.000056e+00 1.000056e+00 1.000056e+00 0.307092 8.893761 3.819174 0.575332 0.665354
0.000000 -3.157668e+00 -3.021639e+00 -1.138528e+00 1.000000 1.000000 1.000000 2011.000000 0.000000
0.115814 -7.003035e-01 -7.178174e-01 -7.168710e-01 2.000000 8.000000 3.000000 2012.000000 0.000000
0.224416 -6.202712e-02 -2.396826e-02 -3.214716e-01 2.000000 16.000000 8.000000 2012.000000 0.000000
0.434935 6.439621e-01 7.146291e-01 4.449900e-01 2.000000 23.000000 10.000000 2012.000000 0.000000
1.000000 3.743802e+00 3.382313e+00 2.502287e+00 2.000000 31.000000 12.000000 2013.000000 2.000000
In [9]:
%matplotlib inline
import matplotlib.pyplot as plt
fig, ax = plt.subplots()


CNN-Keras
MLP-Keras
PCA-Feature Selection
PlotData_2
Data mining labs/PROJ
3130059_3150071_315
localhost:8890/notebooks/Data%20mining%20labs/PROJECT/PlotData_2.ipynb
jupyter PlotData_2 Last Checkpoint: Χθες στις 9:14 ΜΜ (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [21]:
%matplotlib inline
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
df_train['Distance(StandardScale)'].hist(color='#A9C5D3', edgecolor='black',
grid=False)
ax.set_title('Developer Distance Histogram', fontsize=12)
ax.set_xlabel('Distance', fontsize=12)
ax.set_ylabel('Frequency', fontsize=12)
Out[21]: Text(0, 0.5, 'Frequency')
Developer Distance Histogram
Frequency
Distance
0
500
1000
1500
2000
2500
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

CNN-Keras | MLP-Keras | PCA-Feature Selection | PlotData_2 | Data mining labs/PROJ | 3130059_3150071_315

localhost:8890/notebooks/Data%20mining%20labs/PROJECT/PlotData_2.ipynb

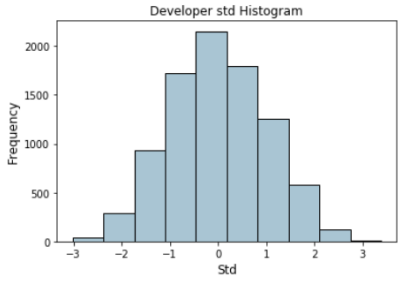
jupyter PlotData_2 Last Checkpoint: Χθες στις 9:14 ΜΜ (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [22]:
```
%matplotlib inline
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
df_train['std(StandardScale)'].hist(color='#A9C5D3', edgecolor='black',
                                    grid=False)
ax.set_title('Developer std Histogram', fontsize=12)
ax.set_xlabel('Std', fontsize=12)
ax.set_ylabel('Frequency', fontsize=12)
```

Out[22]: Text(0, 0.5, 'Frequency')



CNN-Keras | MLP-Keras | PCA-Feature Selection | PlotData_2 | Data mining labs/PROJ | 3130059_3150071_315

localhost:8890/notebooks/Data%20mining%20labs/PROJECT/PlotData_2.ipynb

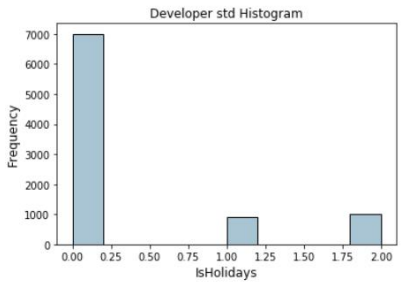
jupyter PlotData_2 Last Checkpoint: Χθες στις 9:14 ΜΜ (unsaved changes) Logout

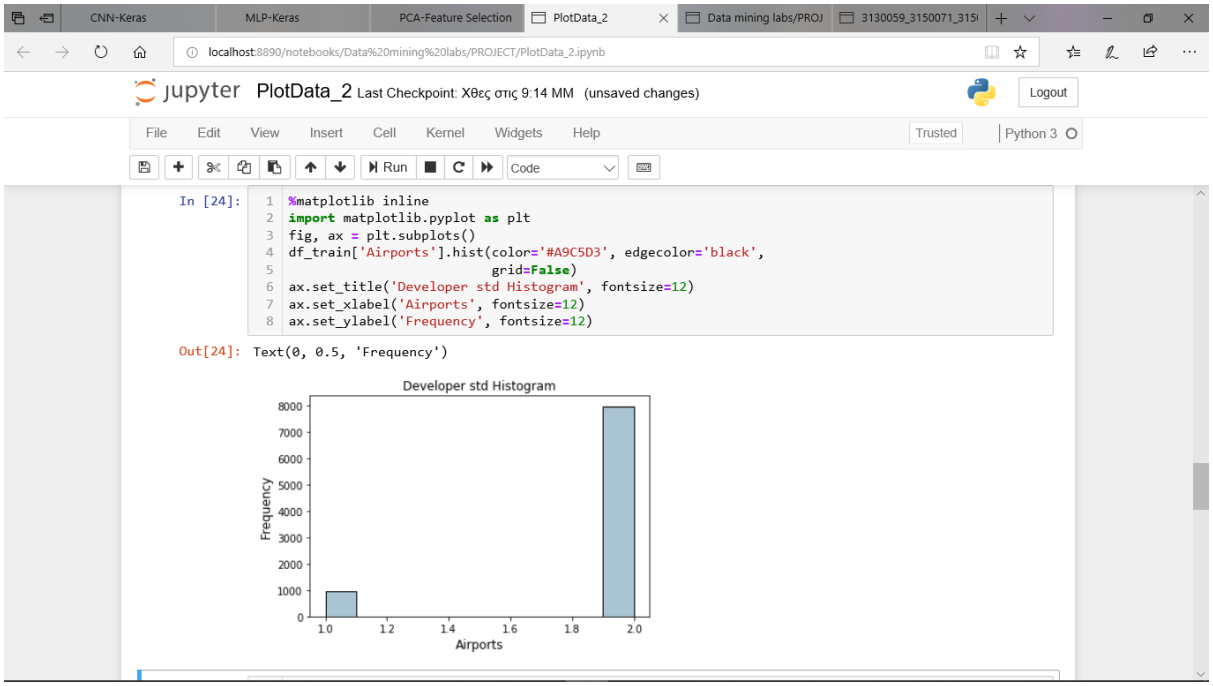
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [23]:
```
%matplotlib inline
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
df_train['IsHolidays'].hist(color='#A9C5D3', edgecolor='black',
                            grid=False)
ax.set_title('Developer std Histogram', fontsize=12)
ax.set_xlabel('IsHolidays', fontsize=12)
ax.set_ylabel('Frequency', fontsize=12)
```

Out[23]: Text(0, 0.5, 'Frequency')





Support Vector Machines

Για τη δημιουργία προβλέψεων δοκιμάστηκε και SVM Classifier. Δοκιμάσαμε τρόπους για να βρούμε τις καλύτερες παραμέτρους και το καλύτερο kernel για τον classifier. Από kernels δοκιμάσαμε linear, rbf, sigmoid και polynomial. Στις δοκιμές μας καλύτερο δείχνει να είναι το rbf (Radial basis function). Το rbf χρησιμοποιεί τις παραμέτρους C, Gamma. Για να βρεθούν οι πιο κατάλληλες τιμές των δύο παραμέτρων χρησιμοποιήθηκε cross validation με kfold = 5 και λογαριθμικό grid για τις δύο παραμέτρους.

Για το input χρησιμοποιήσαμε τα: DateOfDeparture, LongitudeDeparture, LatitudeDeparture, LongitudeArrival, LatitudeArrival, WeeksToDeparture, std_wtdDistance και Distance. Όλες οι συνεχείς μεταβλητές έγιναν normalize πρώτα ώστε να έχουν μέση τιμή 0 και διασπορά 1. Στη συνέχεια έγιναν rescale ώστε να παίρνουν τιμές στο εύρος [0,1]. Categorical μεταβλητές δεν χρησιμοποιήθηκαν.

Η μέθοδος αυτή απέδωσε score 0.37 χωρίς πάρα πολλά features (μόνο συνεχή features) και έχει περιθώρια για βελτίωση. Για να μπορέσουν να χρησιμοποιηθούν categorical features μπορεί κάποιος να χρησιμοποιήσει την παρακάτω τεχνική (δεν δοκιμάστηκε):

Έστω ότι μια categorical μεταβλητή μετασχηματίζεται ώστε να παίρνει ακέραιες τιμές από το 0 έως το 255. Αυτό σημαίνει ότι χρειάζονται 8-bits για να αναπαρασταθεί. Τότε αντί για αυτό το feature, 8 νέα features μπορούν να εισέλθουν όπου το κάθε νεο feature θα έχει τιμή 0 ή 1 ανάλογα με τη τιμή του δυαδικού αριθμού.

Random Forest Classifier

Για το input χρησιμοποιήσαμε τα: Departure, Arrival, Norm, mean Norm Distance, Airports, Day, Month, Year and IsHolidays. Το IsHolidays είναι μία δικιά μας μεταβλητή που δείχνει αν η ημερομηνία είναι ημερομηνία διακοπών ή όχι.

Η παραμετροποίηση του classifier:

- Θα πρέπει να υπάρχουν minimum 2 samples σε κάθε leaf.
- Ο minimum αριθμός από samples που απαιτούνται για να γίνει split ένας κόμβος ενός decision tree ήταν 2.
- Ο αριθμός των trees του forest ήταν 5000.
- Το μέγιστο επιτρεπτό depth ήταν 60.
- Και για κάθε tree επιλέγονται τυχαία sqrt(#features) features. όπου #features είναι το πλήθος των διαφορετικών features (max_features).

Η μέθοδος αυτή απέδωσε score 0.43.

Neural Network (MLP)

Χρησιμοποιήσαμε για inputdata τα εξής: DateOfDeparture, Departure, Arrival, Airports, IsHolidays, Επιπλέον χρησιμοποιήσαμε τα std, Distance αφού πρώτα τα κανονικοποιούμε με standard scale. Δηλαδή, όλες οι τιμές των std και Distance μετατρέπονται έτσι ώστε να έχουν μέση τιμή ίση με μηδέν και τυπική απόκλιση ίση με ένα. Χρησιμοποιήσαμε τα παραπάνω features διότι με αυτά λαμβάνετε το μεγαλύτερο σκόρ. Επίσης, “περάσαμε” όλα τα labels μας από τον onehotencoder ώστε να τα μετατρέψει σε 0-1 το κάθε ένα από τα 8 labels. Στη συνέχεια

χρησιμοποιούμε το sequential του keras ώστε να φτιάξουμε το νευρωνικό μας δίκτυο. Το νευρωνικό δίκτυο αποτελείται απο το input layer δύο hidden layers και το output layer. μεταξύ κάθε level του νευρωνικού δικτύου μας βάζουμε dropout προκειμένου να αποφύγουμε το overfitting. Το dropout αναγκάζει το νευρικό δίκτυο να μάθει περισσότερα δυνατά χαρακτηριστικά που είναι χρήσιμα σε συνδυασμό με πολλά διαφορετικά τυχαία υποσύνολα των άλλων νευρώνων. Επιλέγουμε ένα dropout ίσο με 0.5. Πειραματιστήκαμε με διάφορες τιμές για τις παραμέτρους του νευρωνικού δικτύου και καταλήξαμε στις εξής τιμές όπου λάβαμε το καλύτερο σκορ. Συγκεκριμένα για 200 εποχές, batch size = 200, learning rate = 0.007, momentum = 0.9 συνάρτηση ενεργοποίησης relu για το input layer, συνάρτηση ενεργοποίησης μεταξύ των hidden layers relu και συνάρτηση που δίνει output την softmax λάβαμε σκόρ 0.57591. Επιλέξαμε για output function την softmax διότι κατόπιν αναζήτησης που κάναμε καταλήξαμε στο ότι αυτή ταιριάζει καλύτερα για τα προβλήματα τύπου multiclass classification, σαν το πρόβλημα που είχαμε να αντιμετωπίσουμε. Επιπλέον, στο output layer έχουμε 8 νευρώνες όπου ο καθένας βγάζει μία πιθανότητα για κάθε νέο πρόβλημα που πρέπει να κατατάξει. Από αυτές τις 8 πιθανότητες κρατάμε αυτή με τη μεγαλύτερη τιμή, την θέτουμε ως 1 και τις υπόλοιπες τις θέτουμε 0 και κάνουμε inverse transfromation στα labels που προβλέψαμε ώστε να βρούμε σε ποιο από τα 0-7 labels ανήκει το κάθε νέο sample του test dataset.

Logistic Regression

Για τον logistic regression πειραματιστήκαμε με διάφορα attributes προκειμένου να δούμε ποια δίνουν το καλύτερο αποτέλεσμα. Καταλήξαμε στα εξής. DateOfDeparture, CityDeparture, CityArrival, Airports, IsHolidays, Επιπλέον χρησιμοποιούμε τα Distance και WeeksToDeparture (το οποίο είναι κανονικοποιημένο με standard scale δηλαδή όλες οι τιμές μετατρέπονται έτσι ώστε η μέση τιμή να είναι μηδέν και η τυπική απόκλιση να είναι ένα.Το σκόρ που λάβαμε ήταν 0.41. Παραδόξως ο logistic regression δεν δούλευε καλά με όλες τις τιμές να είναι κανονικοποιημένες

KNN

Εφαρμόζοντας τον knn πειραματιστήκαμε με το ποια features να χρησιμοποιήσουμε και ποιο k. Συγκεκριμένα για k = 77 και τα εξής features DateOfDeparture, Departure, Arrival, Airports, IsHolidays και τα std_wtd και Distance κανονικοποιημένα με την MinMaxScale δηλαδή μετατρέπονται ως εξής: Xnew = (Xold-Xmin)/(Xmax-Xmin)), λάβαμε ως αποτέλεσμα το 0.39718. Γενικά το k το αποφασίσαμε δοκιμάζοντας διάφορα k και έχοντας στο νου μας τον εμπειρικό κανόνα που λέει ότι ένα κατάλληλο k = sqrt(samples), όπου για μας αυτός ο αριθμός ήταν περίπου ίσος με 94 οπότε πήραμε το 93 ως k ώστε να είναι περιττός για να μην έχουμε ισοπαλίες. Οπότε έχοντας υπόψη τον κανόνα αυτό περιορίσαμε στην αναζήτηση του k μεταξύ του διαστήματος [60,100]. Για k=93 ο KNN απλά μας έδωσε λίγο χειρότερα αποτελέσματα(0.39525) από ότι για k=77.

PCA

Σε όλες τις υλοποιήσεις που κάναμε λόγω των features που διαλέξαμε και των encoders που χρησιμοποιούσαμε τα features μας ήταν κάπου στα 599 στις περισσότερες υλοποιήσεις μας. Έτσι, δοκιμάσαμε PCA ώστε να πετύχουμε μείωση διαστάσεων με σκοπό να έχουμε καλύτερα αποτελέσματα. Επιλέγοντας n_components =[4,10] (ώστε να έχουμε λίγες διαστάσεις αλλά ταυτόχρονα και μεγάλη πληροφορία) διαπιστώσαμε ότι δεν πήραμε κανένα καλύτερα αποτελέσματα σε κανένα κατηγοριοποιητή μας αλλά αντιθέτως πήραμε χειρότερα αποτελέσματα.

Ενδεικτικά για τον Random Forest με PCA είχαμε 0.35874, για τον KNN με PCA είχαμε 0.36643, για τον Logistic regression με PCA είχαμε περίπου 0.29 και για το νευρωνικό σύστημα που κατασκευάσαμε είχαμε περίπου 0.40. Με αποτέλεσμα να “εγκαταλείψουμε” τον αλγόριθμο PCA.