**REPORT**

DATA CLEANING:

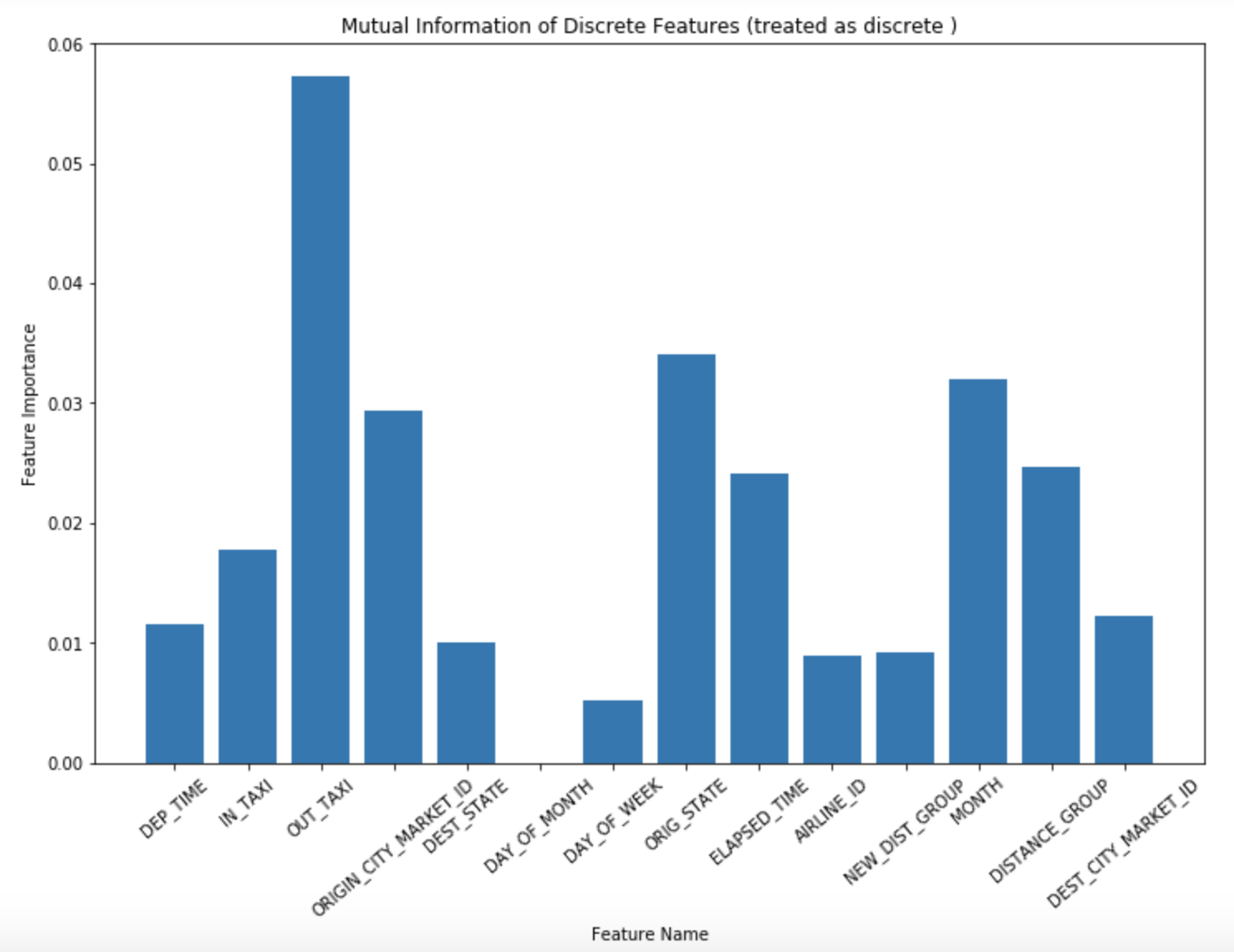
1. Converting strings to numeric values (eg - ‘347’ to 347, ’1,240’ to 1240)
2. ‘FL\_DATE’ – attribute converted from string to date-time and stored as separate features ‘DAY\_OF\_MONTH’ and ‘MONTH’.

DATA PRE-PROCESSING:

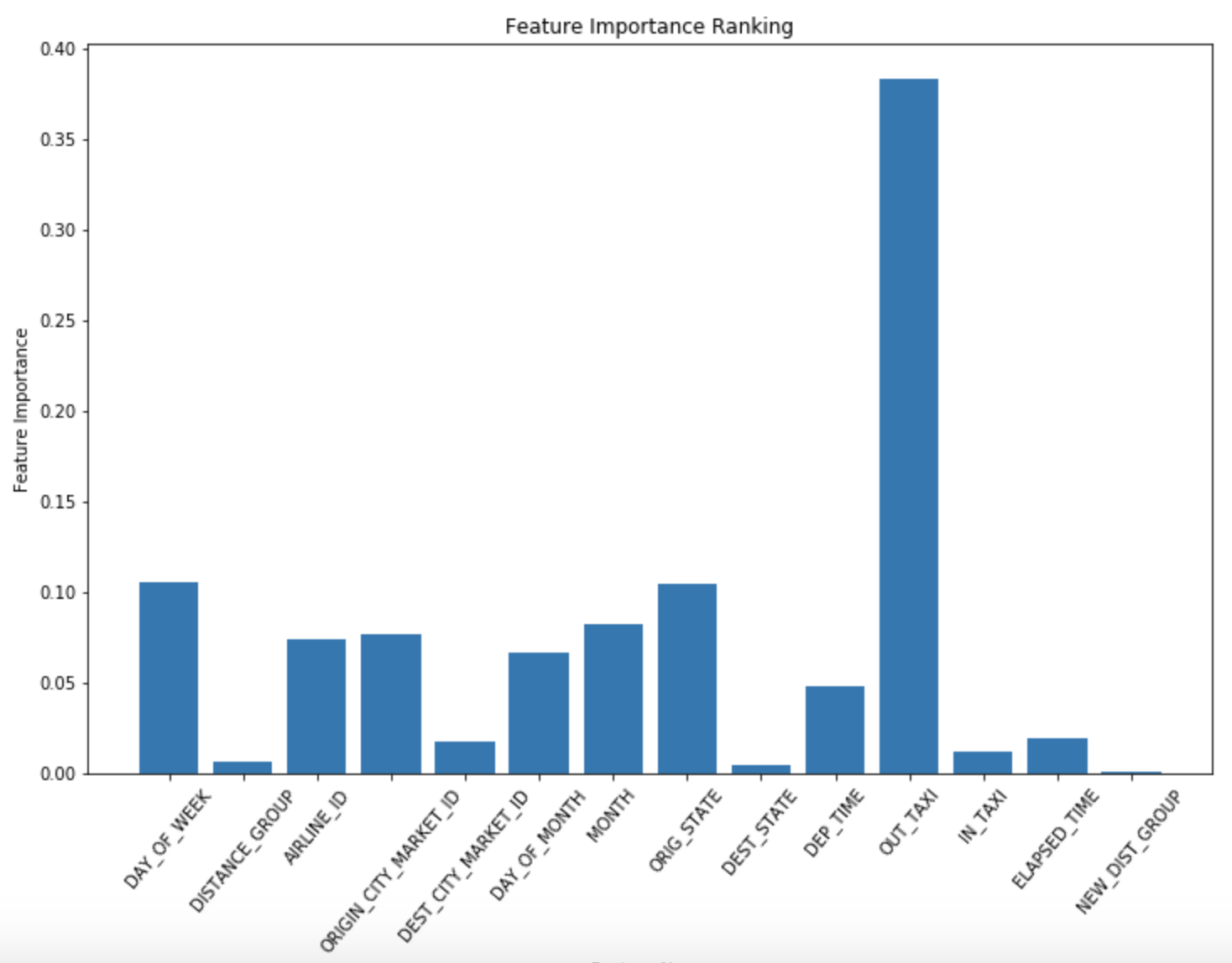
1. Binning – Created bins to convert continuous values to discrete. An interval was generated and it’s mean was taken to represent the range. Following were the bins chosen for each feature:
2. CRS\_DEP\_TIME = intervals [0, 600, 1200, 1800, 2400] => ‘DEP\_TIME’
3. TAXI\_OUT = intervals [0, 20, 40, 60, 80, 150] => ‘OUT\_TAXI’
4. TAXI\_IN = intervals [0, 10, 20, 30, 40, 85] => ‘IN\_TAXI’
5. ACTUAL\_ELAPSED\_TIME = intervals [0, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 650] => ELAPSED\_TIME
6. Removing missing values (NaN) – ‘FIRST\_DEP\_TIME’ feature contained only few values and was a measure of the time that has elapsed since the flight was set off from the first departure gate. This feature seemed irrelevant with missing values and was hence dropped.
7. Converting string values to numeric – Sci-kit Learn’s LabelEncoder( )1 function was used to convert the string values in ‘ORIGIN\_STATE\_ABR’ to numerals. Hash-Map was then used to map the states in ‘DEST\_STATE\_ABR’ attribute so that the states match for both the new numeric features.
8. Following features were removed as they were either useless or redundant –
9. ‘UID’ and ‘FL\_NUM’ features gave no information about the target variable.
10. ‘AIRLINE\_ID’ and ‘UNIQUE\_CARRIER’ represent the same thing so I dropped the ‘UNIQUE\_CARRIER’ feature.
11. ‘ORIGIN’ and ‘ORIGIN\_CITY\_NAME’ were dropped as they both represented the origin city. ‘ORIGIN\_CITY\_MARKET\_ID’ feature was used to represent the Origin City.
12. 'DEST’ and 'DEST\_CITY\_NAME' were also removed for the same reason and ‘DEST\_CITY\_MARKET\_ID’ was used instead.
13. ‘DISTANCE’ was removed as ‘DISTANCE\_GROUP’ seemed a better feature that represented the same thing – distance b/w source and destination.
14. \*\*\*Outlier treatment\*\*\*
15. Scaling – After box-plot analysis, it seemed evident that scaling is required in this case as some attributes like ‘AIRLINE\_ID’, ‘ORIGIN\_CITY\_MARKET\_ID’, ‘DEST\_CITY\_MARKET\_ID’, etc. were numeric but very large values. In addition, since most Machine learning models work better on scaled data, scaling every feature was required. Sci-kit learn’s RobustScaler( )2 function was used to scale the features and make the mean 0. This is called “robust” because it also treats the outliers, which can vary for each feature.

FEATURE ENGINEERING:

1. Mutual Information – Sci-kit learn’s mutual\_info\_regression( )3 function was used to get the mutual information of each feature with the target variable. The following plot was obtained:



1. Information Gain – A Decision Tree Regressor was fit with min\_samples\_leaf=20 and Information importance was obtained for each feature



1. There’s another function from Sci-kit learn called SelectKBest( )4 which gives the ‘k’-best features for the target variable using the given scoring function, which in this case was ‘mutual\_info\_regression’. It returned the following features:
2. ‘OUT\_TAXI’
3. ‘AIRLINE\_ID’
4. ‘ELAPSED\_TIME’
5. ‘ORIGIN\_STATE’
6. ‘IN\_TAXI’

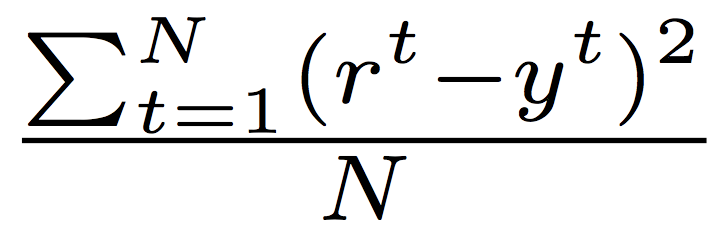
Similar features were deemed important by the mutual information function also but there was some variance in the values when choosing the automatic setting as opposed to the values obtained by setting the ‘discrete\_features’ attribute as ‘True’.

PREDICTIVE MODELLING:

1. TRAIN-TEST SPLIT – Splitting the data into Train and Test set and using the same set for every model and setting did not seem a good idea as the examples might be splitted in an ordered manner which might have produced inaccurate statistics about the model. Therefore, I used KFold( )5 function of Sci-kit learn library to create indices for the given ‘k’ folds. This function was used as opposed to the one created during Assignment 1 because there is an added option of ‘shuffling’ in KFold( ) function which shuffles the indices before splitting them. This seemed to produce more randomness in the splits which would help me make the model generalized.

For each iteration, train and test indices were generated, a model was trained on the generated train and test sets for every setting of hyper-parameter and then the mean of all folds was recorded for each setting of the hyper-parameters. By doing this, I gained information about the effects of different hyper-parameters and also helped in generating generalized and un-biased information about the model.

1. ERROR FUNCTION – Sci-kit learn’s mean\_squared\_error( )6 function was used as the method for calculating the error from this method was the same as given in the assignment PDF:



1. MACHINE LEARNING MODELS:
2. Predicting from the mean – as the first strategy, mean was chosen as the predicted value for each example in the validation set. The values jumped around from 1400 to 2500.
3. Linear Regression –

Linear Regression( )7 package from sci-kit learn was used to implement Linear Regression. PolynomialFeatures( )8 and Pipeline( )9 functions of sci-kit learn were also used in order to implement polynomial regression on the dataset.

1. Linear model –

Mean Squared error for 10 folds –

[[1974.0448932284348], [1034.4327427964172], [2067.1196971651229], [2089.065407004151], [2676.7604944125164], [2129.4889305851475], [1682.0493905344817], [1168.3468966454334], [1618.3248775146167], [1454.8672040730778]]

MSE (Mean of all folds) - 1789.4500533959399

1. Polynomial of Degree 2 –

Mean Squared error for 10 folds –

[[1243.4391455411537], [1443.0348238173394], [2342.5582685687509], [2190.9863691464516], [2265.8563207024058], [1993.3359409083084], [1037.1767544796776], [2000.2431764842552], [1167.4941111897813], [919.40157698724602]]

MSE (Mean of all folds) - 1660.352648782537

1. Polynomial of Degree 2 –

Mean Squared error for 10 folds –

[[1287.3062829449379], [1942.81884369197], [1470.3424789489575], [2522.8061097806772], [3762.2617254363608], [1378.363121082928], [1629.1252418116153], [1724.526694010965], [1343.1156074334299], [2532.3312482267092]]

MSE (Mean of all folds) - 1959.2997353368551

1. Random Forest–

RandomForestRegressor( )10 package from sci-kit learn was used to implement a Random Forest. Best accuracy seemed to have appeared on using Minimum samples in a leaf as 200. Mean-squared error for different number of tress is given. For error rate on each setting of ‘min\_samples\_leaf’, please see the Jupyter Notebook.

Min\_samples\_leaf: 200

Trees: 5

MSE: 1618.74413591

Trees: 10

MSE: 1615.88875839

Trees: 30

MSE: 1615.78709553

Trees: 50

MSE: 1615.3991176

Trees: 80

MSE: 1615.50475472

Trees: 100

MSE: 1613.40026039

Trees: 200

MSE: 1614.13503215

Trees: 500

MSE: 1614.72325741

Trees: 1000

MSE: 1614.58811228

1. Support Vector Machines –

Support Vector Machine for Regression11

FURTHER IMPROVEMENTS:

* ORIGIN\_CITY\_MARKET\_ID represented the origin cities in a numeric way but there were some cities who were far apart but due to similar names, they were close to each other

(eg- 30140 - Albuquerque, NM & 30141 - Aberdeen, SD)

* I believe this dataset requires very detailed analysis to come up with better predictions for ‘ARR\_DELAY’. Turning it into a classification task, where a flight can be termed as ‘delayed’ if it’s delayed by more than 15 minutes, can also be useful to predict when a flight might be late.
* As with my earlier assignments, I started doing it on my own but was late in realizing that this assignment would have been better performed if collaborated with another student. Lot of my initial time went in cleaning and processing the data. After spending much time and effort on feature engineering, results could not improve by a lot.

Feature Selection / Engineering

Methods used to learn

Selected tunable parameter – table or description with settings

How I arrived at the solution

Cross-validation table (compare in a table for each model)

REFERENCES:

1. LabelEncoder( ) - http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
2. RobustScaler( ) - http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler
3. Mutual\_info\_regression( ) - http://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.mutual\_info\_regression.html#sklearn.feature\_selection.mutual\_info\_regression
4. SelectKBest( ) - http://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.SelectKBest.html
5. Kfold( ) - http://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.KFold.html
6. Mean\_Squared\_error( ) - http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean\_squared\_error.html
7. Linear Regression( ) - http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html
8. PolynomialFeatures( ) - http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html
9. Pipeline( ) - http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
10. RandomForestRegressor( ) - http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
11. SupporVectorMachine( ) - http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
12. Nathalie Kuhn∗ and Navaneeth Jamadagni, “Application of Machine Learning Algorithms to Predict Flight Arrival Delays”, CS229: AUTUMN 2017 - http://cs229.stanford.edu/proj2017/final-reports/5243248.pdf
13. Interesting Ideas - https://www.kaggle.com/fabiendaniel/predicting-flight-delays-tutorial/code