Machine Learning

Spring 2018 - Assignment 5

Harshit Srivastava

hs3500@nyu.edu

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:

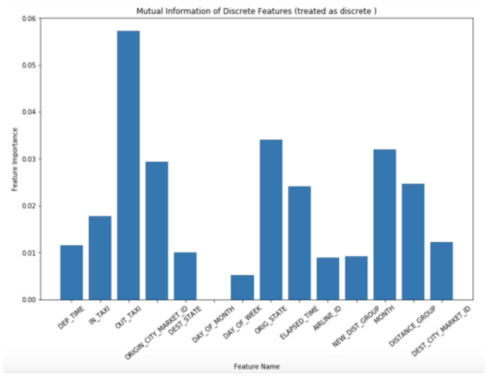
1) Binning – Created bins to convert continuous values to discrete. An interval was generated and it’s mean was taken to represent the range. Following were the bins chosen for each feature:
   (a) CRS_DEP_TIME = intervals [0, 600, 1200, 1800, 2400] => ‘DEP_TIME’
   (b) TAXI_OUT = intervals [0, 20, 40, 60, 80, 150] => ‘OUT_TAXI’
   (c) TAXI_IN = intervals [0, 10, 20, 30, 40, 85] => ‘IN_TAXI’
   (d) ACTUAL_ELAPSED_TIME = intervals [0, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 650] => ‘ELAPSED_TIME’
2) 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.
3) 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.
4) Following features were removed as they were either useless or redundant –
   (a) ‘UID’ and ‘FL_NUM’ features gave no information about the target variable.
   (b) ‘AIRLINE_ID’ and ‘UNIQUE_CARRIER’ represent the same thing so I dropped the ‘UNIQUE_CARRIER’ feature.
   (c) ‘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.
   (d) 'DEST’ and 'DEST_CITY_NAME' were also removed for the same reason and ‘DEST_CITY_MARKET_ID’ was used instead.
   (e) ‘DISTANCE’ was removed as ‘DISTANCE_GROUP’ seemed a better feature that represented the same thing – distance b/w source and destination.

5) 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.

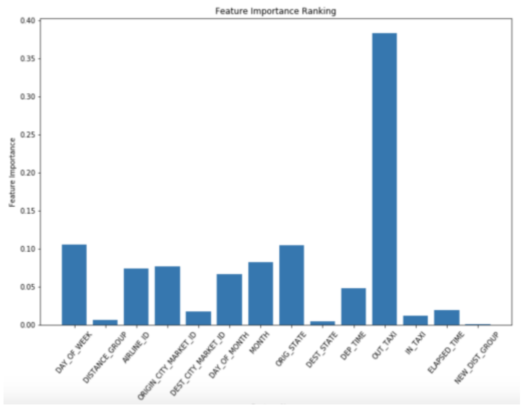
6) Outlier Removal – One of the most impact-creating transformation was the removal of outliers from the target variable. There were many instances when every attribute seemed to be alright but still the flight was delayed. This may be due weather, airline or flight crew issues which cannot be learned from the data. So, it is best to eliminate this noise altogether from the model and train the model on the distribution which it is likely to see.
   Alternatively, I could have trained the model on the subset in which outliers were removed and then test it on an un-treated subset but sadly due to time-constraint this could not be implemented as the indices of cross-validation were getting messed up.

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:



2) Information Gain – A Decision Tree Regressor was fit with min_samples_leaf=20 and Information importance was obtained for each feature



3) 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:
   (1) 'OUT_TAXI'

(2) 'DEP_TIME
(3) 'ELAPSED_TIME'
(4) 'ORIGIN_STATE'
(5) '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'. Therefore, I chose the top 4 features to train the model.

PREDICTIVE MODELLING:

(a) 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.

(b) 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:

$$\frac{\sum_{t=1}^{N}(r^t - y^t)^2}{N}$$

(c) MACHINE LEARNING MODELS:

1) 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 450 to 600.
2) 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.
   (a) Linear model –
   Mean Squared error for 10 folds – 441.6325331211234
   (b) Polynomial of Degree 2 –
   Mean Squared error for 10 folds – 436.85771468651603
   (c) Polynomial of Degree 2 –

Mean Squared error for 10 folds – 435.12397621993148

3) 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.

```
[Fold 1 - 450.28502838102662,
 Fold 2 - 384.44053406568821,
 Fold 3 - 462.49535114255241,
 Fold 4 - 396.1994543262187,
 Fold 5 - 349.4335976856492]

Mean Error = 421.954123436
```

4) Support Vector Machines –

   Support Vector Machine for Regression[11] was used with 3 different kernels: linear, rbf and sigmoid.

```
Kernel: linear
   MSE: 426.388074362
Kernel: rbf
   MSE: 434.25874652
Kernel: sigmoid
   MSE: 3954.17331934
```

Finally, Random Forest Regressor was used due to its ability to generalize well. 80 trees were grown and minimum_samples per leaf was kept at 50. The csv which contains details of the results from cross-validation of this model has been attached with the submitted files.

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.

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