Expedia Divvy Challenge

-Ligaj Pradhan (Jan 19, 2018)

1. Goal: To predict if a trip will be SHORT (less than 2 km) or LONG

2. Understanding and Exploring and Pre-Processing the Data

From the Divvy dataset, I had the following information about trips:

trip_id: ID attached to each trip taken
start_time: day and time trip started, in CST
stop_time: day and time trip ended, in CST
bikeid: ID attached to each bike
tripduration: time of trip in seconds
from_station_name: name of station where trip originated
to_station_name: name of station where trip terminated
from_station_id: ID of station where trip originated
to_station_id: ID of station where trip terminated
usertype: "Customer" is a rider who purchased a 24-Hour Pass; "Subscriber" is a rider who purchased an Annual Membership
gender: gender of rider
birthyear: birth year of rider

and I had the following information about each station:

id: ID attached to each station
name: station name
latitude: station latitude
longitude: station longitude
dpcapacity: number of total docks at each station as of 6/30/2017
online_date: date the station was created in the system

My first goal was to **compute the target variable SHORT or LONG** for each trip (later encoded as LONG=0, SHORT=1). **DataPreProcessorAndExplorer** class in my code achieves this by using the **getInterStationDistancesInKm()** function.

For such a problem where we want to predict if a trip will be long or short, I see there are three main key factors:

1. **Person/Human factor**
2. **Time factor**
3. **Station factor including its geographical data**

- Some type of person might favor SHORT trips and some might favor LONG trips. What kind of person favors SHORT trips is of interest to us.
- Sometime of the day or some day of the week might be especially favorable for SHORT trips and some might be especially favorable for LONG trips.
- Similarly, some geographical areas which might be far from other places of interest and would mostly generate LONG trips in contrast to other areas which are close by to other interesting point of interests and mostly generate SHORT trips.

Hence, we want to collect special features describing person, time and place of start (to describe a trip) and train our classification model to predict if a trip will be SHORT or LONG.

Time features:

I also extracted ‘**day_of_week**’ (and represented by 0,1,2,3,4,5,6 and 7. Mon=0, Tue = 1, Wed=3, Thru=4, Fri=5, Sat=6 and Sun=7) and ‘**time_of_day**’ (0-23 for hours) temporal features from the date string in the dataset. Specifying such features separately would help the model understand time far easily than the complex long date string provided in the dataset.

Personal features:

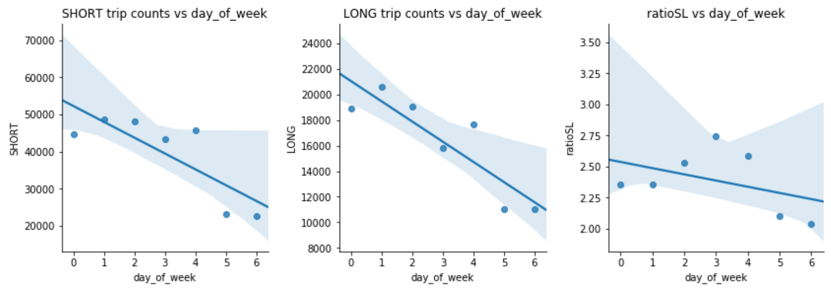
Personal features like age, gender or user type could also help the model learn if a particular user would like SHORT trip over LONG trip or vice versa. Hence, I computed ‘**rider_Age**’ from birthyear, ‘**gender_num**’ from gender (representing Male with 1 and Female with 0) and separate ‘**Customer**’,’ **Dependent**’ and ‘**Subscriber**’ features from usertype (each are binary features and value will be 1 if they are true or else 0).

Station features including its geographical data:

The longitude and latitude of each station is provided. So, feeding such longitudes and latitude would help the classifier learn the geo spatial locations of the stations. Some areas might generate more SHORT trips then LONG trips. However, we are talking about only one city and the values are very close to each other. Hence, we normalize the longitudes and latitudes between 0 and 1 and represent them as **'long_n' and 'lat_n'**. Similarly, I also compute total number of SHORT and LONG trips for each station (from the train data) and represent it as ‘**SHORT_Trips_FromStation**’ and ‘**LONG_Trips_FromStation**’. Their ratio again could describe if any station favors SHORT over LONG trips.

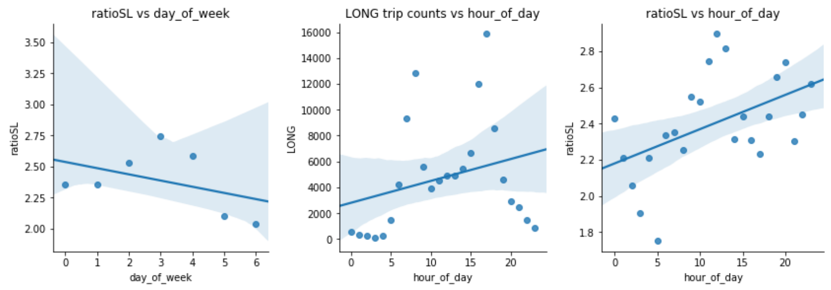
To explore if these features favor SHORT over LONG trips I computed ‘ratioSL’ i.e. ratio of SHORT trips over LONG trips for each feature and plotted them.

<u>SHORT, LONG and ratioSL vs day_of_week:</u>



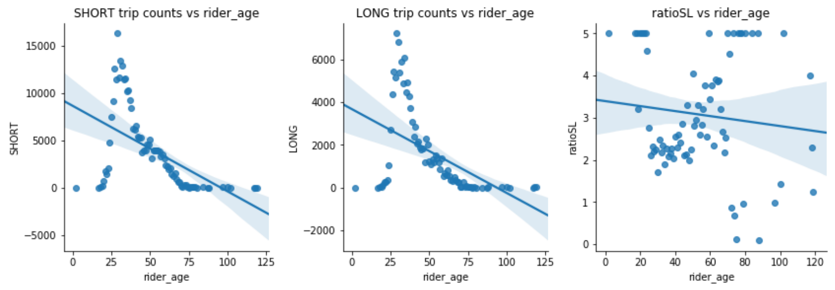
Note: If we look at the third plot, ratioSL decreases dramatically in the weekend. Hence suggesting the fall of SHORT trips. SHORT trips look more frequent in the week days.

<u>SHORT, LONG and ratioSL vs time_of_day:</u>



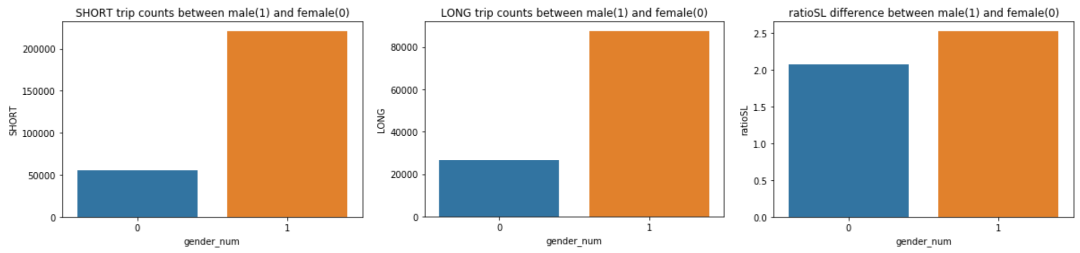
Note: If we look at the third plot, ratioSL increases later in the day. Hence suggesting the rise of SHORT trips as night falls. SHORT trips look more frequent in the afternoon and evening.

<u>SHORT, LONG and ratioSL vs rider_Age:</u>



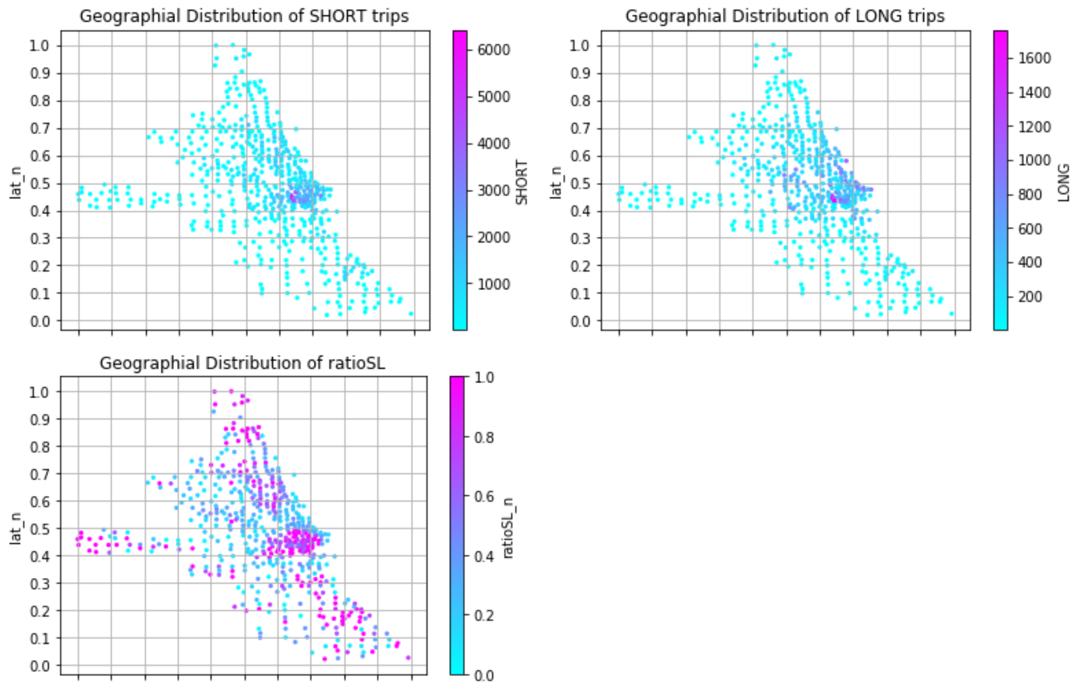
Note: If we look at the third plot, ratioSL decreases with age. SHORT trips look less frequent specially after around 60.

SHORT, LONG and ratioSL vs gender:



Note: Males seems to favor SHORT trips a little bit more, though both favors SHORT trips quite more than LONG trips.

Regional distribution of SHORT LONG and ratioSL:



Note: Though distribution of LONG and SHORT trips looks similar from the two top plots, we can see that there are some pockets that favors SHORT trips more than LONG trips from the third plot.

NOTE: All these plots can be generated using the exploreFeatures() function of the DataPreProcessorAndExplorer class

These plots gave us an idea of the data and looks like SHORT and LONG trips are dependent on time place and personal traits. Hence, we came up with a set of features as follows after the initial data preparation phase:

['lat_n', 'long_n', 'LONG_Trips_FromStation', 'SHORT_Trips_FromStation', 'ratioSL_FromStation', 'rider_age', 'gender_num', 'Customer', 'Dependent', 'Subscriber', 'day_of_week', 'hour_of_day', 'GroundTruth']

Finally, DataPreProcessorAndExplorer. finalizeDataset() divide the dataset into train(80%) validation(10%) and test(10%) datasets and saves them in the data folder

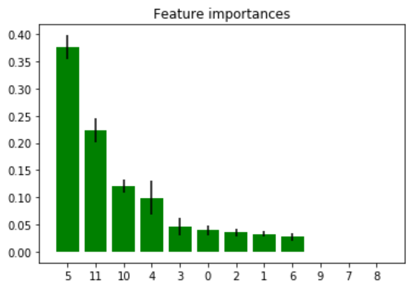
3.Training and Validation

Now since my feature set contains both numerical and categorical features, I decided to train a **RamdomForest** classifier with 200 trees. All training and validation is handled by the **TrainingAndValidationHandler** class in my code.

An optional **feature selection** step with **Univariate_Statistical_Tests** is also implemented inside this class to select top **9 features**. **TrainingAndValidationHandler .train_N_Validate_Classifier(doFeatureSelection, persistModel)** takes 2 boolean parameters to either perform feature selection or not and to save the trained model or not.

# Training without feature selection:

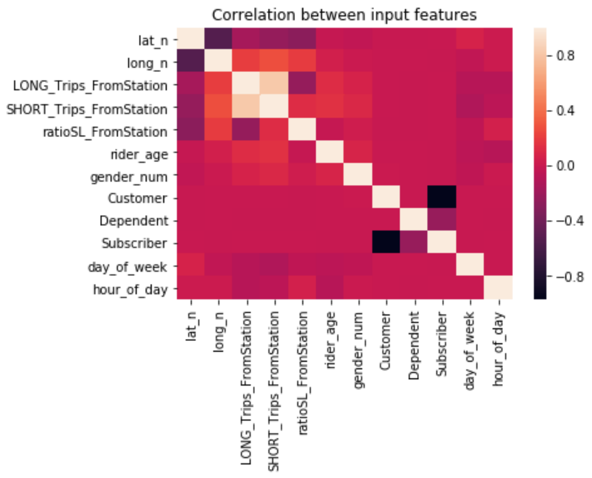
TrainingAndValidationHandler. reportFeatureImportances(classifier) plots importance of each features while training the random forest classifier.



Feature ranking:

1. rider_age (feature 5) (0.375649)
2. hour_of_day (feature 11) (0.223701)
3. day_of_week (feature 10) (0.120782)
4. ratioSL_FromStation (feature 4) (0.098540)
5. SHORT_Trips_FromStation (feature 3) (0.046263)
6. lat_n (feature 0) (0.039303)
7. LONG_Trips_FromStation (feature 2) (0.035532)
8. long_n (feature 1) (0.032861)
9. gender_num (feature 6) (0.027166)
10. Subscriber (feature 9) (0.000103)
11. Customer (feature 7) (0.000090)
12. Dependent (feature 8) (0.000009)

TrainingAndValidationHandler.showFeatureCorrelations() function also displays the correlation between each feature:



TrainingAndValidationHandler.reportPerformance () displays the accuracy in terms of percentage, confusion matrix and precision, recall and f1 score for the **validation dataset**.

Validation Test Accuracy: 80.48218083062181 %

Confusion matrix :

```
[[ 6785  4687]
 [ 2931 24628]]
```

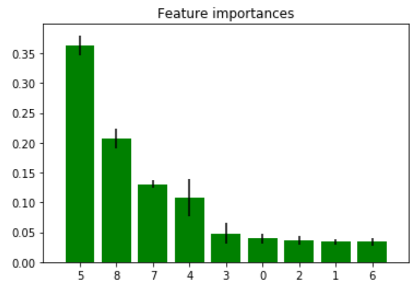
Meaning: Out of total (6785 + 4687) LONG trips 6785 were predicted correctly. Out of total (2931 + 24628) SHORT trips 24628 were predicted correctly.

	Precision	recall	f1-score	support
0	0.70	0.59	0.64	11472
1	0.84	0.89	0.87	27559
avg / total	0.80	0.80	0.80	39031

Note: SHORT Trips are encoded as 1 and LONG trips are encoded as 0, so for SHORT trips a precision of 0.84, recall or 0.89 and f1-score of 0.87 was achieved in the validation dataset. In average f1 –score was 0.80.

# Training with feature selection:

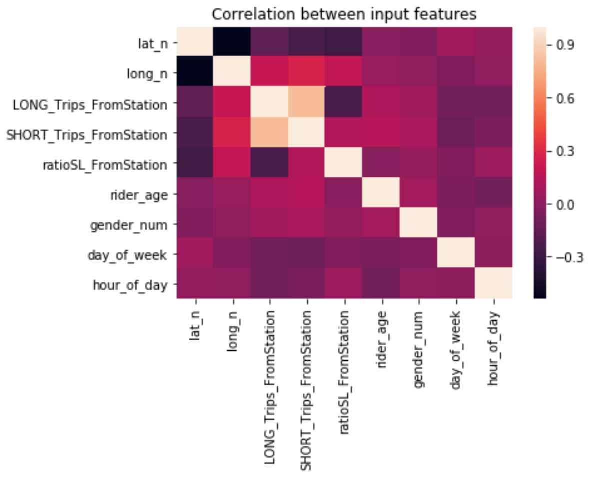
TrainingAndValidationHandler. reportFeatureImportances(classifier) plots importance of each features while training the random forest classifier.



Feature ranking:

1. rider_age (feature 5) (0.362919)
2. hour_of_day (feature 8) (0.206968)
3. day_of_week (feature 7) (0.130643)
4. ratioSL_FromStation (feature 4) (0.107555)
5. SHORT_Trips_FromStation (feature 3) (0.048052)
6. lat_n (feature 0) (0.039365)
7. LONG_Trips_FromStation (feature 2) (0.036554)
8. long_n (feature 1) (0.034211)
9. gender_num (feature 6) (0.033734)

TrainingAndValidationHandler.showFeatureCorrelations() function also displays the correlation between each feature:



TrainingAndValidationHandler.reportPerformance () displays the accuracy in terms of percentage, confusion matrix and precision, recall and f1 score for the **validation dataset**.

Validation Test Accuracy: 80. 63846685967565%

Confusion matrix :

```
[[ 6814  4658]
 [ 2899 24660]]
```

Meaning: Out of total (6814 + 4658) LONG trips 6814 were predicted correctly. Out of total (2899 + 24660) SHORT trips 24660 were predicted correctly.

```
              Precision    recall    f1-score    support
          0     0.70        0.59       0.64       11472
          1     0.84        0.89       0.87       27559
avg / total     0.80        0.81       0.80       39031
```

Note: SHORT Trips are encoded as 1 and LONG trips are encoded as 0, so for SHORT trips a precision of 0.84, recall or 0.89 and f1-score of 0.87 was achieved in the validation dataset. In average f1 –score was 0.80.

Hence, after feature selection accuracy in validation dataset improved a little from 80.48 to 80.63%.

4.Testing

Finally, the TestingHandler class handles the testing. It loads the trained saved model and performs testing on the test dataset. TestingHandler. test_Classifier(featureSelected) take a Boolean parameter to test either with trained model (saved as trained_model_with_feature_selection.sav) or with feature selection or trained model without feature selection (saved as trained_model.sav).

#Testing results without feature selected model:

Validation Test Accuracy: 80. 5159868825579%

Confusion matrix :

```
[[ 6892  4682]
 [ 2923 24535]]
```

Meaning: Out of total (6892 + 4682) LONG trips 6892 were predicted correctly. Out of total (2923 + 24535) SHORT trips 24535predicted were correctly.

	Precision	recall	f1-score	support
0	0.70	0.60	0.64	11574
1	0.84	0.89	0.87	27458
avg / total	0.80	0.81	0.80	39032

Note: SHORT Trips are encoded as 1 and LONG trips are encoded as 0, so for SHORT trips a precision of 0.84, recall or 0.89 and f1-score of 0.87 was achieved in the validation dataset. In average f1 –score was 0.80.

#Testing results with feature selected model:

Validation Test Accuracy: 80. 564664890346338%

Confusion matrix :

```
[[ 6900  4674]
 [ 2912 24546]]
```

Meaning: Out of total (6900 + 4674) LONG trips 6900 were predicted correctly. Out of total (2912 + 24546) SHORT trips 24546 predicted were correctly.

	Precision	recall	f1-score	support
0	0.70	0.60	0.65	11574
1	0.84	0.89	0.87	27458
avg / total	0.80	0.81	0.80	39032

Note: SHORT Trips are encoded as 1 and LONG trips are encoded as 0, so for SHORT trips a precision of 0.84, recall or 0.89 and f1-score of 0.87 was achieved in the validation dataset. In average f1 –score was 0.80.

Hence, we obtained a slightly better test results with feature selection model.

5.Future Work

There are more short trips than long trips in 2017_Q1 dataset. So, the classifier tries to be better at correctly classifying short trips compared to classifying long trips. Hence, provided more time, I would use trip data from 2017_Q2 with 2017_Q1 and filter trips to have almost equal number of short and long trips.