beginner_python

October 28, 2018

1 Case Study 6.1 - NYC Taxi Trips

Note: If you close this notebook at any time, you will have to run all cells again upon re-opening it.

2 BEGINNER PYTHON

As this is a beginner version, we include a lot of code here to help you along the way.

3 Identification Information

```
In [ ]: # YOUR NAME              = Ramesh Neupane
        # YOUR MITX PRO USERNAME = ramesh2018
        # YOUR MITX PRO E-MAIL   = rneupane@dallasisd.org
```

4 Setup

Run these cells to install all the packages you need to complete the remainder of the case study. This may take a few minutes, so please be patient.

```
In [2]: !pip install featuretools==0.1.19
```

```
Collecting featuretools==0.1.19
  Downloading https://files.pythonhosted.org/packages/07/c5/ebfd50f1b824355bc8f965f3a755fc30c1d2
    100% || 143kB 2.0MB/s ta 0:00:01
Requirement already satisfied: numpy>=1.13.3 in /home/nbuser/anaconda3_501/lib/python3.6/site-pa
Collecting scipy>=1.0.0 (from featuretools==0.1.19)
  Downloading https://files.pythonhosted.org/packages/a8/0b/f163da98d3a01b3e0ef1cab8dd2123c34aee
    100% || 31.2MB 36kB/s eta 0:00:011  7% |                                | 2.4MB 6.5MB/s eta 0:0
Requirement already satisfied: pandas>=0.20.3 in /home/nbuser/anaconda3_501/lib/python3.6/site-p
Collecting s3fs>=0.1.2 (from featuretools==0.1.19)
  Downloading https://files.pythonhosted.org/packages/53/69/7e14a5a883a74a469b5edca792ff0eab168b
    100% || 51kB 3.3MB/s ta 0:00:011
Requirement already satisfied: tqdm>=4.19.2 in /home/nbuser/anaconda3_501/lib/python3.6/site-pac
Requirement already satisfied: toolz>=0.8.2 in /home/nbuser/anaconda3_501/lib/python3.6/site-pac
```

```
Requirement already satisfied: pyyaml>=3.12 in /home/nbuser/anaconda3_501/lib/python3.6/site-pac
Requirement already satisfied: cloudpickle>=0.4.0 in /home/nbuser/anaconda3_501/lib/python3.6/si
Requirement already satisfied: future>=0.16.0 in /home/nbuser/anaconda3_501/lib/python3.6/site-p
Requirement already satisfied: pympler>=0.5 in /home/nbuser/anaconda3_501/lib/python3.6/site-pac
Requirement already satisfied: pytz>=2011k in /home/nbuser/anaconda3_501/lib/python3.6/site-pack
Requirement already satisfied: python-dateutil>=2.5.0 in /home/nbuser/anaconda3_501/lib/python3.
Requirement already satisfied: botocore in /home/nbuser/anaconda3_501/lib/python3.6/site-package
Requirement already satisfied: boto3 in /home/nbuser/anaconda3_501/lib/python3.6/site-packages (
Requirement already satisfied: six>=1.5 in /home/nbuser/anaconda3_501/lib/python3.6/site-package
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /home/nbuser/anaconda3_501/lib/python3.
Requirement already satisfied: docutils>=0.10 in /home/nbuser/anaconda3_501/lib/python3.6/site-p
Requirement already satisfied: s3transfer<0.2.0,>=0.1.10 in /home/nbuser/anaconda3_501/lib/pytho
Building wheels for collected packages: featuretools, s3fs
  Running setup.py bdist_wheel for featuretools ... done
  Stored in directory: /home/nbuser/.cache/pip/wheels/2b/71/de/99b59608fad48046821e6c3a5585cd879
  Running setup.py bdist_wheel for s3fs ... done
  Stored in directory: /home/nbuser/.cache/pip/wheels/71/5d/ed/77e5b8e4a26dc4f5436f7b729f3de92eb
Successfully built featuretools s3fs
Installing collected packages: scipy, s3fs, featuretools
  Found existing installation: scipy 0.19.1
    Uninstalling scipy-0.19.1:
      Successfully uninstalled scipy-0.19.1
Successfully installed featuretools-0.1.19 s3fs-0.1.6 scipy-1.1.0
You are using pip version 18.0, however version 18.1 is available.You should consider upgrading
```

5 Import

Import the required tools into the notebook.

```
In [6]: import featuretools as ft
        import matplotlib
        import matplotlib.pyplot as plt
        import numpy as np
        import utils
        from featuretools.primitives import (Count, Day, Hour, Max, Mean, Median, Min,
                                             Minute, Month, Std, Sum, Week, Weekday,
                                             Weekend)
        from sklearn.ensemble import GradientBoostingRegressor
        from utils import (compute_features, feature_importances, load_nyc_taxi_data,
                           preview)

        print('Import successful!')
```

```
Import successful!
```

```
In [5]: %matplotlib inline
```

```
In [7]: assert ft.__version__ == '0.1.19', 'Make sure you run the command above with the correct
```

6 Data

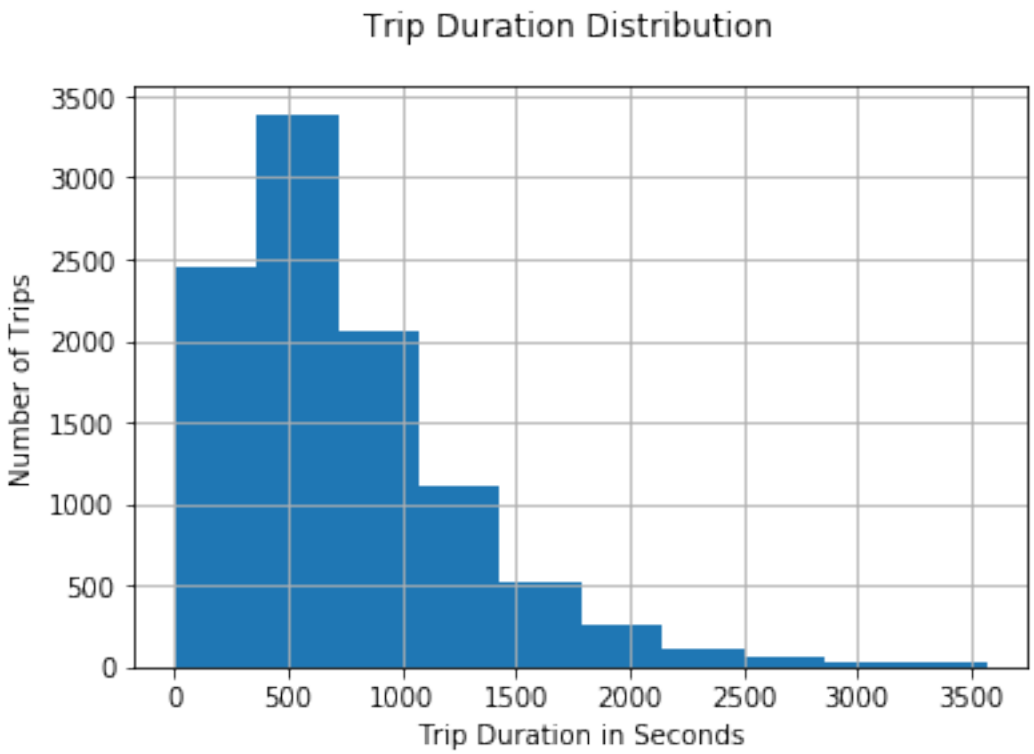
Load the NYC taxi trip data. Note that this may take a minute or two, so please be patient.

```
In [8]: trips, pickup_neighborhoods, dropoff_neighborhoods = load_nyc_taxi_data()
        preview(trips, 10)
        print('Data load successful!')
```

```
Data load successful!
```

We can also plot some aspects of the data to get a better sense of its distributions. For instance, here is the `trip_duration` variable we are going to try to predict.

```
In [9]: trips.trip_duration.hist()
        plt.xlabel('Trip Duration in Seconds')
        plt.ylabel('Number of Trips')
        plt.suptitle('Trip Duration Distribution')
        plt.show()
        print('Histogram generation successful!')
```



```
Histogram generation successful!
```

```
In [10]: trips.shape[0]  # Tells us how many trips are in the dataset
```

```
Out[10]: 10000
```

QUESTION 1: DATA ANALYSIS

Describe the dataset. How many trips are in the dataset? How would you describe the distribution of trip durations? Is there anything else we should observe? Make sure the histogram is visible in the notebook.

There are 10000 data point about car pick up and drop time and in between durations. When observing taxi trip shape, the distribution is right tail.Which means, most of the trip(more than 50% of data) is less than 1000 seconds(Less than 17 mins) in terms of time duration. By looking 'number of trips', we could say that less the drive time, more likely they are to use often.

7 Entities and Relationships

```
In [11]: entities = {
             "trips": (trips, "id", 'pickup_datetime'),
             "pickup_neighborhoods": (pickup_neighborhoods, "neighborhood_id"),
             "dropoff_neighborhoods": (dropoff_neighborhoods, "neighborhood_id"),
         }

         relationships = [("pickup_neighborhoods", "neighborhood_id", "trips", "pickup_neighborh
                          ("dropoff_neighborhoods", "neighborhood_id", "trips", "dropoff_neighbo

         print('Entities and relationships successful!')
```

```
Entities and relationships successful!
```

8 Transform Primitives

```
In [12]: trans_primitives = [Weekend]

         # This may take some time to compute
         features = ft.dfs(entities=entities,
                           relationships=relationships,
                           target_entity="trips",
                           trans_primitives=trans_primitives,
                           agg_primitives=[],
                           ignore_variables={"trips": ["pickup_latitude", "pickup_longitude",
                                                       "dropoff_latitude", "dropoff_longitude"]}
                           features_only=True)

         print('Transform primitives successful!')
```

```
Transform primitives successful!


/home/nbuser/anaconda3_501/lib/python3.6/site-packages/featuretools/entityset/entity.py:524: Fut
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/featuretools/entityset/entity.py:536: Fut
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)
```

Here are the features that we just created. Note: This list may contain the `trip_duration` variable. But, rest assured that we will not actually use this variable in training. Our code removes that variable in `utils.py`.

```
In [13]: print(f"Number of features: {len(features)}")
         features

Number of features: 13


Out[13]: [<Feature: vendor_id>,
          <Feature: passenger_count>,
          <Feature: trip_distance>,
          <Feature: payment_type>,
          <Feature: trip_duration>,
          <Feature: pickup_neighborhood>,
          <Feature: dropoff_neighborhood>,
          <Feature: IS_WEEKEND(pickup_datetime)>,
          <Feature: IS_WEEKEND(dropoff_datetime)>,
          <Feature: pickup_neighborhoods.latitude>,
          <Feature: pickup_neighborhoods.longitude>,
          <Feature: dropoff_neighborhoods.latitude>,
          <Feature: dropoff_neighborhoods.longitude>]
```

Finally, we compute the feature matrix from these features.

```
In [14]: feature_matrix = compute_features(features, trips[['id', 'pickup_datetime']])
         preview(feature_matrix, 5)

Elapsed: 00:01 | Remaining: 00:00 | Progress: 100%||| Calculated: 1/1 cutoff times
Finishing computing...


Out[14]:         trip_distance  trip_duration  IS_WEEKEND(dropoff_datetime)  \
         id
         514030           2.46           1039                          True
         514031           7.90           1454                          True
         514032           1.00           1168                          True
```

```
514033           0.02             35                          True
514034          19.00           3470                          True

        dropoff_neighborhood = D  dropoff_neighborhood = AA  \
id
514030                         0                          0
514031                         0                          0
514032                         0                          0
514033                         0                          0
514034                         0                          0

        dropoff_neighborhood = H  dropoff_neighborhood = P  \
id
514030                         0                          0
514031                         0                          0
514032                         0                          0
514033                         0                          0
514034                         0                          0

        dropoff_neighborhood = AR  dropoff_neighborhood = AD  \
id
514030                          0                          0
514031                          0                          0
514032                          0                          0
514033                          0                          0
514034                          0                          0

        dropoff_neighborhood = A    ...      pickup_neighborhood = AO  \
id                                  ...
514030                         0    ...                             0
514031                         0    ...                             0
514032                         0    ...                             0
514033                         0    ...                             0
514034                         0    ...                             0

        pickup_neighborhood = AD  pickup_neighborhood = Q  \
id
514030                         0                         0
514031                         0                         0
514032                         0                         0
514033                         0                         0
514034                         0                         0

        pickup_neighborhood = AR  pickup_neighborhood = AP  \
id
514030                         0                          0
514031                         0                          0
514032                         0                          0
```

```
514033                          0                          0
514034                          0                          0

        pickup_neighborhood = H  pickup_neighborhoods.latitude  \
id
514030                        0                     40.757707
514031                        0                     40.744928
514032                        1                     40.729652
514033                        0                     40.720245
514034                        0                     40.646194

        IS_WEEKEND(pickup_datetime)  dropoff_neighborhoods.latitude  vendor_id
id
514030                         True                      40.766809          2
514031                         True                      40.793597          1
514032                         True                      40.740333          1
514033                         True                      40.720245          2
514034                         True                      40.785005          1

[5 rows x 31 columns]
```

9 First Model

```
In [15]: # Split data
         X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
         y_train = np.log(y_train + 1)
         y_test = np.log(y_test + 1)

         print('Data split successful!')
```

```
Data split successful!
```

```
In [16]: # This should train within a minute or so
         model = GradientBoostingRegressor(verbose=True)
         model.fit(X_train, y_train)
         print(model.score(X_test, y_test)) # This is the R^2 value of the prediction

         print('Training successful!')
```

```
      Iter       Train Loss   Remaining Time
         1           0.4736           6.02s
         2           0.4148           5.55s
         3           0.3661           5.64s
         4           0.3266           5.25s
         5           0.2934           4.92s
         6           0.2665           5.06s
         7           0.2441           4.88s
```

```
          8           0.2257           4.87s
          9           0.2103           4.82s
         10           0.1973           4.73s
         20           0.1434           3.79s
         30           0.1312           3.28s
         40           0.1248           2.57s
         50           0.1218           2.01s
         60           0.1191           1.57s
         70           0.1174           1.15s
         80           0.1158           0.76s
         90           0.1147           0.37s
        100           0.1137           0.00s
0.7527788676087426
Training successful!
```

QUESTION 2: FIRST MODEL

Describe the 2 new features that we added to the model. Do you think these improved the performance from a model that did not have these features? Why?

The new two features (Primitives); target and aggration, which gave singnificantly higher R^2 value which shows that it has a positve reinforcement in the performance, however does not have a comapirative R^2 to test the result.We decides performance of any model from two measures; one either comparing F-score or second, by checking R^2. Higher the both score, better the performance.

10 More Transform Primitives

```
In [17]: trans_primitives = [Minute, Hour, Day, Week, Month, Weekday, Weekend]

         features = ft.dfs(entities=entities,
                           relationships=relationships,
                           target_entity="trips",
                           trans_primitives=trans_primitives,
                           agg_primitives=[],
                           ignore_variables={"trips": ["pickup_latitude", "pickup_longitude",
                                                       "dropoff_latitude", "dropoff_longitude"]},
                           features_only=True)

         print('Transform primitives successful!')
```

```
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/featuretools/entityset/entity.py:524: Fut
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/featuretools/entityset/entity.py:536: Fut
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)


Transform primitives successful!
```

```
In [18]: print(f"Number of features: {len(features)}")
         features

Number of features: 25


Out[18]: [<Feature: vendor_id>,
          <Feature: passenger_count>,
          <Feature: trip_distance>,
          <Feature: payment_type>,
          <Feature: trip_duration>,
          <Feature: pickup_neighborhood>,
          <Feature: dropoff_neighborhood>,
          <Feature: MINUTE(pickup_datetime)>,
          <Feature: MINUTE(dropoff_datetime)>,
          <Feature: HOUR(pickup_datetime)>,
          <Feature: HOUR(dropoff_datetime)>,
          <Feature: DAY(pickup_datetime)>,
          <Feature: DAY(dropoff_datetime)>,
          <Feature: WEEK(pickup_datetime)>,
          <Feature: WEEK(dropoff_datetime)>,
          <Feature: MONTH(pickup_datetime)>,
          <Feature: MONTH(dropoff_datetime)>,
          <Feature: WEEKDAY(pickup_datetime)>,
          <Feature: WEEKDAY(dropoff_datetime)>,
          <Feature: IS_WEEKEND(pickup_datetime)>,
          <Feature: IS_WEEKEND(dropoff_datetime)>,
          <Feature: pickup_neighborhoods.latitude>,
          <Feature: pickup_neighborhoods.longitude>,
          <Feature: dropoff_neighborhoods.latitude>,
          <Feature: dropoff_neighborhoods.longitude>]

In [19]: feature_matrix = compute_features(features, trips[['id', 'pickup_datetime']])
         preview(feature_matrix, 5)

Elapsed: 00:03 | Remaining: 00:00 | Progress: 100%||| Calculated: 1/1 cutoff times
Finishing computing...


Out[19]:         IS_WEEKEND(dropoff_datetime)  dropoff_neighborhood = D  \
         id
         514030                          True                          0
         514031                          True                          0
         514032                          True                          0
         514033                          True                          0
         514034                          True                          0

                 dropoff_neighborhood = AA  dropoff_neighborhood = H  \
         id
```

```
514030                          0                          0
514031                          0                          0
514032                          0                          0
514033                          0                          0
514034                          0                          0

        dropoff_neighborhood = P  dropoff_neighborhood = AR  \
id
514030                         0                          0
514031                         0                          0
514032                         0                          0
514033                         0                          0
514034                         0                          0

        dropoff_neighborhood = AD  dropoff_neighborhood = A  \
id
514030                          0                          0
514031                          0                          0
514032                          0                          0
514033                          0                          0
514034                          0                          0

        dropoff_neighborhood = AB  dropoff_neighborhood = AV  \
id
514030                          0                           0
514031                          0                           0
514032                          0                           0
514033                          0                           0
514034                          0                           0

                ...           WEEK(pickup_datetime)  trip_distance  \
id              ...
514030          ...                              13           2.46
514031          ...                              13           7.90
514032          ...                              13           1.00
514033          ...                              13           0.02
514034          ...                              13          19.00

        MINUTE(pickup_datetime)  HOUR(pickup_datetime)  \
id
514030                        0                      0
514031                        0                      0
514032                        0                      0
514033                        0                      0
514034                        1                      0

        pickup_neighborhoods.latitude  payment_type  \
id
```

```
514030                      40.757707             1
514031                      40.744928             1
514032                      40.729652             1
514033                      40.720245             2
514034                      40.646194             1

        WEEKDAY(dropoff_datetime)  passenger_count  WEEKDAY(pickup_datetime)  \
id
514030                          5               1                         5
514031                          5               2                         5
514032                          5               1                         5
514033                          5               1                         5
514034                          5               2                         5

        DAY(pickup_datetime)
id
514030                     2
514031                     2
514032                     2
514033                     2
514034                     2

[5 rows x 43 columns]
```

In [20]:
```
# Re-split data
X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
y_train = np.log(y_train + 1)
y_test = np.log(y_test + 1)

print('Data split successful!')
```

```
Data split successful!
```

In [21]:
```
# This should train within a minute or so
model = GradientBoostingRegressor(verbose=True)
model.fit(X_train, y_train)
print(model.score(X_test, y_test)) # This is the R^2 value of the prediction

print('Training successful!')
```

```
      Iter       Train Loss   Remaining Time
         1           0.4736            7.11s
         2           0.4148            7.05s
         3           0.3661            6.16s
         4           0.3264            5.90s
         5           0.2930            5.71s
         6           0.2660            5.49s
         7           0.2432            5.25s
```

```
          8           0.2245           5.26s
          9           0.2090           5.27s
         10           0.1960           5.16s
         20           0.1362           4.26s
         30           0.1200           3.65s
         40           0.1126           3.05s
         50           0.1079           2.47s
         60           0.1047           1.92s
         70           0.1016           1.40s
         80           0.0986           0.91s
         90           0.0938           0.44s
        100           0.0899           0.00s
0.8059573190397493
Training successful!
```

QUESTION 3: SECOND MODEL

Describe the rest of the new features that we just added to the model. How did this affect performance? Did we have to sacrifice training time?

New added features have helped affect the performance in positve way. Because of which R^2 result has improved and is higher. while comparing R^2, we can say that, previouly 75.2% , and now by new model, 80.5% of variations that happended in the data is being explained, which means, new feature that we added to model helped affect the performance better..

11 Aggregation Primitives

```
In [22]: trans_primitives = [Minute, Hour, Day, Week, Month, Weekday, Weekend]
         aggregation_primitives = [Count, Sum, Mean, Median, Std, Max, Min]

         features = ft.dfs(entities=entities,
                           relationships=relationships,
                           target_entity="trips",
                           trans_primitives=trans_primitives,
                           agg_primitives=aggregation_primitives,
                           ignore_variables={"trips": ["pickup_latitude", "pickup_longitude",
                                                       "dropoff_latitude", "dropoff_longitude"]},
                           features_only=True)

         print('Aggregation primitives successful!')
```

```
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/featuretools/entityset/entity.py:524: Fut
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/featuretools/entityset/entity.py:536: Fut
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)
```

```
Aggregation primitives successful!
```

```
In [23]: print(f"Number of features: {len(features)}")
         features
```

```
Number of features: 75
```

```
Out[23]: [<Feature: vendor_id>,
          <Feature: passenger_count>,
          <Feature: trip_distance>,
          <Feature: payment_type>,
          <Feature: trip_duration>,
          <Feature: pickup_neighborhood>,
          <Feature: dropoff_neighborhood>,
          <Feature: MINUTE(pickup_datetime)>,
          <Feature: MINUTE(dropoff_datetime)>,
          <Feature: HOUR(pickup_datetime)>,
          <Feature: HOUR(dropoff_datetime)>,
          <Feature: DAY(pickup_datetime)>,
          <Feature: DAY(dropoff_datetime)>,
          <Feature: WEEK(pickup_datetime)>,
          <Feature: WEEK(dropoff_datetime)>,
          <Feature: MONTH(pickup_datetime)>,
          <Feature: MONTH(dropoff_datetime)>,
          <Feature: WEEKDAY(pickup_datetime)>,
          <Feature: WEEKDAY(dropoff_datetime)>,
          <Feature: IS_WEEKEND(pickup_datetime)>,
          <Feature: IS_WEEKEND(dropoff_datetime)>,
          <Feature: pickup_neighborhoods.latitude>,
          <Feature: pickup_neighborhoods.longitude>,
          <Feature: dropoff_neighborhoods.latitude>,
          <Feature: dropoff_neighborhoods.longitude>,
          <Feature: pickup_neighborhoods.COUNT(trips)>,
          <Feature: pickup_neighborhoods.SUM(trips.vendor_id)>,
          <Feature: pickup_neighborhoods.SUM(trips.passenger_count)>,
          <Feature: pickup_neighborhoods.SUM(trips.trip_distance)>,
          <Feature: pickup_neighborhoods.SUM(trips.trip_duration)>,
          <Feature: pickup_neighborhoods.MEAN(trips.vendor_id)>,
          <Feature: pickup_neighborhoods.MEAN(trips.passenger_count)>,
          <Feature: pickup_neighborhoods.MEAN(trips.trip_distance)>,
          <Feature: pickup_neighborhoods.MEAN(trips.trip_duration)>,
          <Feature: pickup_neighborhoods.MEDIAN(trips.vendor_id)>,
          <Feature: pickup_neighborhoods.MEDIAN(trips.passenger_count)>,
          <Feature: pickup_neighborhoods.MEDIAN(trips.trip_distance)>,
          <Feature: pickup_neighborhoods.MEDIAN(trips.trip_duration)>,
          <Feature: pickup_neighborhoods.STD(trips.vendor_id)>,
```

```
 <Feature: pickup_neighborhoods.STD(trips.passenger_count)>,
 <Feature: pickup_neighborhoods.STD(trips.trip_distance)>,
 <Feature: pickup_neighborhoods.STD(trips.trip_duration)>,
 <Feature: pickup_neighborhoods.MAX(trips.vendor_id)>,
 <Feature: pickup_neighborhoods.MAX(trips.passenger_count)>,
 <Feature: pickup_neighborhoods.MAX(trips.trip_distance)>,
 <Feature: pickup_neighborhoods.MAX(trips.trip_duration)>,
 <Feature: pickup_neighborhoods.MIN(trips.vendor_id)>,
 <Feature: pickup_neighborhoods.MIN(trips.passenger_count)>,
 <Feature: pickup_neighborhoods.MIN(trips.trip_distance)>,
 <Feature: pickup_neighborhoods.MIN(trips.trip_duration)>,
 <Feature: dropoff_neighborhoods.COUNT(trips)>,
 <Feature: dropoff_neighborhoods.SUM(trips.vendor_id)>,
 <Feature: dropoff_neighborhoods.SUM(trips.passenger_count)>,
 <Feature: dropoff_neighborhoods.SUM(trips.trip_distance)>,
 <Feature: dropoff_neighborhoods.SUM(trips.trip_duration)>,
 <Feature: dropoff_neighborhoods.MEAN(trips.vendor_id)>,
 <Feature: dropoff_neighborhoods.MEAN(trips.passenger_count)>,
 <Feature: dropoff_neighborhoods.MEAN(trips.trip_distance)>,
 <Feature: dropoff_neighborhoods.MEAN(trips.trip_duration)>,
 <Feature: dropoff_neighborhoods.MEDIAN(trips.vendor_id)>,
 <Feature: dropoff_neighborhoods.MEDIAN(trips.passenger_count)>,
 <Feature: dropoff_neighborhoods.MEDIAN(trips.trip_distance)>,
 <Feature: dropoff_neighborhoods.MEDIAN(trips.trip_duration)>,
 <Feature: dropoff_neighborhoods.STD(trips.vendor_id)>,
 <Feature: dropoff_neighborhoods.STD(trips.passenger_count)>,
 <Feature: dropoff_neighborhoods.STD(trips.trip_distance)>,
 <Feature: dropoff_neighborhoods.STD(trips.trip_duration)>,
 <Feature: dropoff_neighborhoods.MAX(trips.vendor_id)>,
 <Feature: dropoff_neighborhoods.MAX(trips.passenger_count)>,
 <Feature: dropoff_neighborhoods.MAX(trips.trip_distance)>,
 <Feature: dropoff_neighborhoods.MAX(trips.trip_duration)>,
 <Feature: dropoff_neighborhoods.MIN(trips.vendor_id)>,
 <Feature: dropoff_neighborhoods.MIN(trips.passenger_count)>,
 <Feature: dropoff_neighborhoods.MIN(trips.trip_distance)>,
 <Feature: dropoff_neighborhoods.MIN(trips.trip_duration)>]
```

In [24]:
```
# This may take a bit longer to compute, so please be patient
feature_matrix = compute_features(features, trips[['id', 'pickup_datetime']])
preview(feature_matrix, 5)
```

```
Elapsed: 00:00 | Remaining: ? | Progress:   0%|          || Calculated: 0/1 cutoff times

/home/nbuser/anaconda3_501/lib/python3.6/site-packages/featuretools/computational_backends/calcu
Defaulting to column, but this will raise an ambiguity error in a future version
  on=target_index_var, how='left')[rvar].values


Elapsed: 00:05 | Remaining: 00:00 | Progress: 100%||| Calculated: 1/1 cutoff times
```

```
Finishing computing...
```

Out[24]:

id	IS_WEEKEND(dropoff_datetime)	pickup_neighborhoods.MEDIAN(trips.trip_duration)	pickup_neighborhoods.STD(trips.trip_distance)	MONTH(pickup_datetime)	dropoff_neighborhoods.STD(trips.trip_duration)	WEEK(dropoff_datetime)	pickup_neighborhoods.longitude	pickup_neighborhoods.MIN(trips.passenger_count)
514030	True	NaN	NaN	4	NaN	13	-73.986446	NaN
514031	True	NaN	NaN	4	NaN	13	-73.919159	NaN
514032	True	NaN	NaN	4	NaN	13	-73.991595	NaN
514033	True	NaN	NaN	4	NaN	13	-73.987205	NaN
514034	True	NaN	NaN	4	NaN	13	-73.785073	NaN

```
514033                                                NaN
514034                                                NaN

        HOUR(pickup_datetime)  payment_type  \
id
514030                      0             1
514031                      0             1
514032                      0             1
514033                      0             2
514034                      0             1

                             ...                          \
id                           ...
514030                       ...
514031                       ...
514032                       ...
514033                       ...
514034                       ...

        pickup_neighborhood = AO  pickup_neighborhood = AD  \
id
514030                         0                         0
514031                         0                         0
514032                         0                         0
514033                         0                         0
514034                         0                         0

        pickup_neighborhood = Q  pickup_neighborhood = AR  \
id
514030                        0                         0
514031                        0                         0
514032                        0                         0
514033                        0                         0
514034                        0                         0

        pickup_neighborhood = AP  pickup_neighborhood = H  vendor_id  \
id
514030                         0                        0          2
514031                         0                        0          1
514032                         0                        1          1
514033                         0                        0          2
514034                         0                        0          1

        pickup_neighborhoods.SUM(trips.vendor_id)  DAY(dropoff_datetime)  \
id
514030                                        NaN                      2
514031                                        NaN                      2
514032                                        NaN                      2
```

```
514033                                                NaN                       2
514034                                                NaN                       2

        dropoff_neighborhoods.MAX(trips.trip_duration)
id
514030                                             NaN
514031                                             NaN
514032                                             NaN
514033                                             NaN
514034                                             NaN

[5 rows x 93 columns]
```

In [25]:
```
# Re-split data
X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
y_train = np.log(y_train + 1)
y_test = np.log(y_test + 1)

print('Data split successful!')
```

```
Data split successful!
```

In [26]:
```
# This should train within a minute or so
model = GradientBoostingRegressor(verbose=True)
model.fit(X_train, y_train)
print(model.score(X_test, y_test)) # This is the R^2 value of the prediction

print('Training successful!')
```

```
      Iter       Train Loss   Remaining Time
         1           0.4736            5.19s
         2           0.4148            5.19s
         3           0.3661            5.06s
         4           0.3264            4.77s
         5           0.2930            4.85s
         6           0.2660            5.05s
         7           0.2432            4.99s
         8           0.2245            5.16s
         9           0.2090            4.96s
        10           0.1960            4.95s
        20           0.1362            4.07s
        30           0.1200            3.40s
        40           0.1126            2.82s
        50           0.1079            2.32s
        60           0.1047            1.80s
        70           0.1016            1.31s
        80           0.0986            0.85s
        90           0.0938            0.42s
```

```
       100           0.0899            0.00s
0.8060128240391375
Training successful!
```

12 Evaluate on Test Data

```
In [27]: y_pred = model.predict(X_test)
         y_pred = np.exp(y_pred) - 1 # undo the log we took earlier

         print('y_pred computation successful!')
```

```
y_pred computation successful!
```

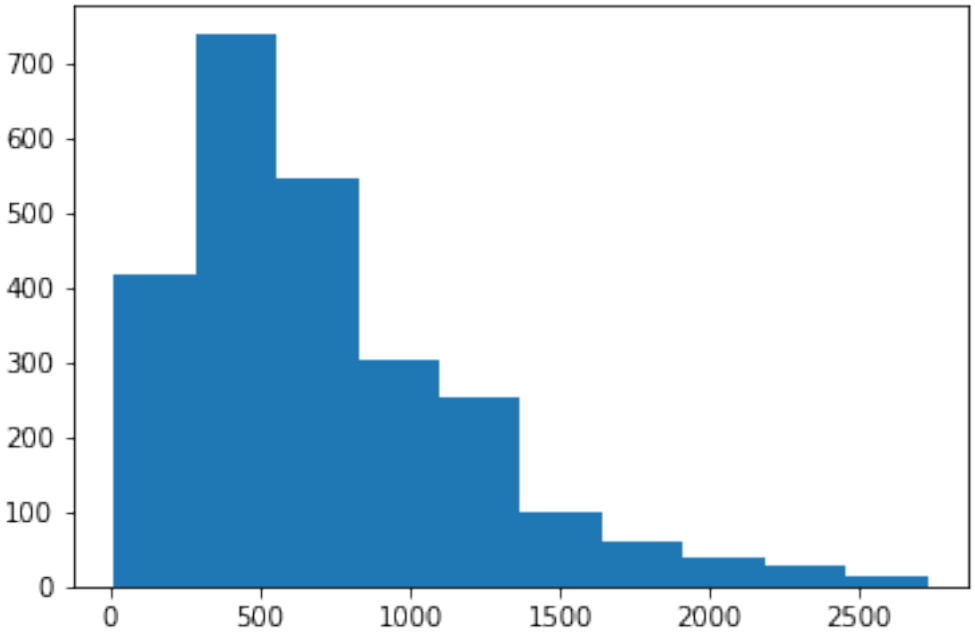
```
In [28]: # Print the first 5 predictions
         y_pred[:5]
```

```
Out[28]: array([453.4415368 , 666.8066337 , 634.00494622, 617.34019506,
                407.96332751])
```

```
In [29]: # Create a histogram of all of them
         matplotlib.pyplot.hist(y_pred)

         print('Histogram generation successful!')
```

```
Histogram generation successful!
```



QUESTION 4: MODEL PREDICTIONS

Analyze the model predictions. Does the output distribution match the one you made earlier in the case study? What other features/strategies could we use to make our model even better, if we had more time?

This model is better then before as it has higher R^2. This new model is being able to explain 80.6% of the veriation that is being observed in the data. and yes, still the new model prediction is right tail. This distribution also suggest that more than 50% of the travel duration is less than 1000 seconds.

13 Feature Importance

```
In [30]: feature_importances(model, feature_matrix.columns, n=25)
```

```
1: Feature: pickup_neighborhoods.MAX(trips.passenger_count), 0.325
2: Feature: dropoff_neighborhoods.STD(trips.trip_duration), 0.107
3: Feature: HOUR(pickup_datetime), 0.105
4: Feature: MINUTE(dropoff_datetime), 0.085
5: Feature: dropoff_neighborhoods.MEAN(trips.passenger_count), 0.070
6: Feature: pickup_neighborhoods.longitude, 0.069
7: Feature: MONTH(pickup_datetime), 0.062
8: Feature: payment_type, 0.060
9: Feature: dropoff_neighborhoods.MEDIAN(trips.vendor_id), 0.029
10: Feature: WEEK(dropoff_datetime), 0.012
11: Feature: pickup_neighborhoods.SUM(trips.trip_distance), 0.012
12: Feature: IS_WEEKEND(pickup_datetime), 0.009
13: Feature: pickup_neighborhoods.MIN(trips.trip_distance), 0.009
14: Feature: dropoff_neighborhoods.STD(trips.passenger_count), 0.007
15: Feature: dropoff_neighborhoods.MEAN(trips.trip_distance), 0.006
16: Feature: dropoff_neighborhoods.MEDIAN(trips.trip_distance), 0.005
17: Feature: dropoff_neighborhood = AA, 0.004
18: Feature: dropoff_neighborhoods.MAX(trips.vendor_id), 0.003
19: Feature: dropoff_neighborhoods.MEDIAN(trips.trip_duration), 0.003
20: Feature: pickup_neighborhoods.SUM(trips.passenger_count), 0.003
21: Feature: dropoff_neighborhoods.MEAN(trips.trip_duration), 0.002
22: Feature: pickup_neighborhoods.latitude, 0.002
23: Feature: dropoff_neighborhood = H, 0.002
24: Feature: pickup_neighborhoods.MAX(trips.trip_duration), 0.002
25: Feature: dropoff_neighborhoods.COUNT(trips), 0.002
```

QUESTION 5: FEATURE IMPORTANCE

Analyze the feature importance values you just computed above. Do they make sense? Are there any values you are surprised by? Give some brief explanations as to why these features are relevant in computing the `trip_duration` target variable.

The result is typical situation of cab in a perticular area. If we compair with other neighborhoods and thier features, we might have some intereting pattrens and behaviour. I did not observed any surprising

result except one important observation from the distribution of the data, which is most of the people who travel for short period of time (less than 1000 second), use taxi most often, which is also feels logical too.

Great job! Now, make sure you check out the **Conclusion** section of the instruction manual to wrap up this case study properly.