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.

BEGINNER PYTHON

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

Identification Information

In [1]:

```
# YOUR NAME               = Hiral
# YOUR MITX PRO USERNAME = hiralsatasia
# YOUR MITX PRO E-MAIL    = hiralsatasia@live.com
```

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/ebfd50f1b824355bc8f965f3a755fc30c1d2e172f143a15c6601c
118/featuretools-0.1.19.tar.gz (133kB)
    100% |████████████████████████████████| 143kB 2.9MB/s ta 0:00:01
Requirement already satisfied: numpy>=1.13.3 in /home/nbuser/anaconda3_501/lib/python3.6/site-
packages (from featuretools==0.1.19) (1.15.2)
Collecting scipy>=1.0.0 (from featuretools==0.1.19)
  Downloading
https://files.pythonhosted.org/packages/a8/0b/f163da98d3a01b3e0ef1cab8dd2123c34aee2bafbb1c5bffa354c
730/scipy-1.1.0-cp36-cp36m-manylinux1_x86_64.whl (31.2MB)
    100% |████████████████████████████████| 31.2MB 33kB/s eta 0:00:011  7% |██
| 2.3MB 5.5MB/s eta 0:00:06    19% |██████                          | 6.2MB 5.3MB/s eta 0:00:05
8% |████████                        | 9.0MB 5.4MB/s eta 0:00:05    29% |██████████
| 9.3MB 8.4MB/s eta 0:00:03    45% |██████████████                  | 14.3MB 5.3MB/s eta 0:00:04
50% |████████████████                | 15.7MB 4.2MB/s eta 0:00:04    74% |████████████████████████
| 23.2MB 5.2MB/s eta 0:00:02▒                       | 24.5MB 5.0MB/s eta 0:00:02    81% |
                          | 25.3MB 4.9MB/s eta 0:00:02▒▒          | 26.8MB 6.4MB/s eta
0:00:01    96% |██████████████████████████████  | 30.1MB 2.0MB/s eta 0:00:01
Requirement already satisfied: pandas>=0.20.3 in /home/nbuser/anaconda3_501/lib/python3.6/site-
packages (from featuretools==0.1.19) (0.23.4)
Collecting s3fs>=0.1.2 (from featuretools==0.1.19)
  Downloading
https://files.pythonhosted.org/packages/53/69/7e14a5a883a74a469b5edca792ff0eab168b4a0531e9f49f486c0
bfb/s3fs-0.1.6.tar.gz (45kB)
    100% |████████████████████████████████| 51kB 1.7MB/s ta 0:00:011
Requirement already satisfied: tqdm>=4.19.2 in /home/nbuser/anaconda3_501/lib/python3.6/site-
packages (from featuretools==0.1.19) (4.26.0)
Requirement already satisfied: toolz>=0.8.2 in /home/nbuser/anaconda3_501/lib/python3.6/site-
packages (from featuretools==0.1.19) (0.8.2)
Requirement already satisfied: pyyaml>=3.12 in /home/nbuser/anaconda3_501/lib/python3.6/site-
packages (from featuretools==0.1.19) (3.13)
Requirement already satisfied: cloudpickle>=0.4.0 in
/home/nbuser/anaconda3_501/lib/python3.6/site-packages (from featuretools==0.1.19) (0.4.0)
Requirement already satisfied: future>=0.16.0 in /home/nbuser/anaconda3_501/lib/python3.6/site-
```

```
packages (from featuretools==0.1.19) (0.16.0)
Requirement already satisfied: pympler>=0.5 in /home/nbuser/anaconda3_501/lib/python3.6/site-
packages (from featuretools==0.1.19) (0.6)
Requirement already satisfied: python-dateutil>=2.5.0 in
/home/nbuser/anaconda3_501/lib/python3.6/site-packages (from pandas>=0.20.3->featuretools==0.1.19)
(2.7.3)
Requirement already satisfied: pytz>=2011k in /home/nbuser/anaconda3_501/lib/python3.6/site-
packages (from pandas>=0.20.3->featuretools==0.1.19) (2017.2)
Requirement already satisfied: botocore in /home/nbuser/anaconda3_501/lib/python3.6/site-packages
(from s3fs>=0.1.2->featuretools==0.1.19) (1.8.50)
Requirement already satisfied: boto3 in /home/nbuser/anaconda3_501/lib/python3.6/site-packages
(from s3fs>=0.1.2->featuretools==0.1.19) (1.4.8)
Requirement already satisfied: six>=1.5 in /home/nbuser/anaconda3_501/lib/python3.6/site-packages
(from python-dateutil>=2.5.0->pandas>=0.20.3->featuretools==0.1.19) (1.11.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
/home/nbuser/anaconda3_501/lib/python3.6/site-packages (from botocore->s3fs>=0.1.2-
>featuretools==0.1.19) (0.9.3)
Requirement already satisfied: docutils>=0.10 in /home/nbuser/anaconda3_501/lib/python3.6/site-
packages (from botocore->s3fs>=0.1.2->featuretools==0.1.19) (0.14)
Requirement already satisfied: s3transfer<0.2.0,>=0.1.10 in
/home/nbuser/anaconda3_501/lib/python3.6/site-packages (from boto3->s3fs>=0.1.2-
>featuretools==0.1.19) (0.1.13)
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/99b59608fad48046821e6c3a5585cd8797ee02e2c8e53e9d62
  Running setup.py bdist_wheel for s3fs ... done
  Stored in directory:
/home/nbuser/.cache/pip/wheels/71/5d/ed/77e5b8e4a26dc4f5436f7b729f3de92eb7fa4febef04f2d042
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 via the 'pip install --upgrade pip' command.
```

Import

Import the required tools into the notebook.

In [3]:

```
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!
```

```
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: Dep
recationWarning: numpy.core.umath_tests is an internal NumPy module and should not be imported. It
will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [4]:

```
%matplotlib inline
```

In [5]:

```
assert ft.__version__ == '0.1.19', 'Make sure you run the command above with the correct version.'
```

Data

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

In [6]:

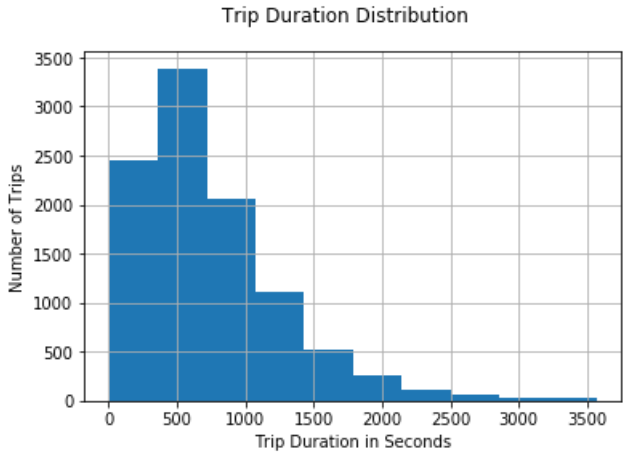
```
trips, pickup_neighborhoods, dropoff_neighborhoods = load_nyc_taxi_data()
preview(trips, 10)
print('Data load successful!')
```

```
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 [7]:

```
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 [8]:

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

Out[8]:

```
10000
```

In [9]:

```
trips
```

Out[9]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_lat
0	514030	2	2016-04-02 00:00:00	2016-04-02 00:17:00	1	2.46	-73.987595	40.754711
1	514031	1	2016-04-02 00:00:00	2016-04-02 00:24:00	2	7.90	-73.924713	40.745567

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_lat
2	514032	1	2016-04-02 00:00:00	2016-04-02 00:19:00	1	1.00	-73.989800	40.732994
3	514033	2	2016-04-02 00:00:00	2016-04-02 00:01:00	1	0.02	-73.987465	40.719822
4	514034	1	2016-04-02 00:01:00	2016-04-02 00:58:00	2	19.00	-73.790131	40.643429
5	514035	1	2016-04-02 00:01:00	2016-04-02 00:09:00	2	1.50	-73.981956	40.765518
6	514036	1	2016-04-02 00:01:00	2016-04-02 00:05:00	1	1.50	-73.980019	40.780594
7	514037	1	2016-04-02 00:01:00	2016-04-02 00:20:00	1	4.20	-73.990501	40.756561
8	514038	1	2016-04-02 00:01:00	2016-04-02 00:10:00	1	0.60	-73.986198	40.730492
9	514039	1	2016-04-02 00:01:00	2016-04-02 00:17:00	1	0.60	-73.990959	40.734890
10	514040	1	2016-04-02 00:01:00	2016-04-02 00:13:00	1	1.90	-74.001495	40.741180
11	514041	2	2016-04-02 00:02:00	2016-04-02 00:11:00	1	1.21	-73.968491	40.754902
12	514042	2	2016-04-02 00:02:00	2016-04-02 00:12:00	1	2.36	-73.962509	40.760345
13	514043	1	2016-04-02 00:02:00	2016-04-02 00:17:00	1	1.80	-73.999115	40.728237
14	514044	2	2016-04-02 00:02:00	2016-04-02 00:15:00	1	1.82	-73.980011	40.780739
15	514045	2	2016-04-02 00:02:00	2016-04-02 00:10:00	6	1.60	-73.985512	40.768261
16	514046	2	2016-04-02 00:03:00	2016-04-02 00:15:00	3	3.03	-73.959267	40.767448
17	514047	2	2016-04-02 00:03:00	2016-04-02 00:10:00	1	1.32	-74.002464	40.751358
18	514048	1	2016-04-02 00:03:00	2016-04-02 00:25:00	2	7.90	-73.863396	40.769886
19	514049	1	2016-04-02 00:03:00	2016-04-02 00:08:00	1	0.40	-74.003174	40.732857
20	514050	2	2016-04-02 00:03:00	2016-04-02 00:08:00	5	0.57	-73.980225	40.783813
21	514051	1	2016-04-02 00:03:00	2016-04-02 00:23:00	1	7.20	-73.987656	40.717880
22	514052	2	2016-04-02 00:03:00	2016-04-02 00:27:00	5	12.06	-73.865921	40.770359
23	514053	2	2016-04-02 00:04:00	2016-04-02 00:10:00	1	1.49	-73.982384	40.772461
24	514054	2	2016-04-02 00:04:00	2016-04-02 00:23:00	1	4.35	-73.990334	40.740292
25	514055	1	2016-04-02 00:04:00	2016-04-02 00:19:00	1	2.80	-73.991188	40.732681
26	514056	2	2016-04-02 00:04:00	2016-04-02 00:11:00	5	1.35	-73.958878	40.678097
27	514057	1	2016-04-02 00:04:00	2016-04-02 00:23:00	1	4.00	-73.988441	40.718571
28	514058	2	2016-04-02	2016-04-02	1	2.30	-74.005409	40.736950

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_lat
			00:04:00	00:14:00				
29	514059	1	2016-04-02 00:04:00	2016-04-02 00:22:00	3	2.80	-73.988113	40.754807
...	...	...	...	...	...	...	...	...
9970	524000	1	2016-04-03 14:53:00	2016-04-03 15:10:00	2	4.10	-74.005562	40.707310
9971	524001	2	2016-04-03 14:53:00	2016-04-03 15:36:00	5	19.13	-73.788353	40.641762
9972	524002	2	2016-04-03 14:53:00	2016-04-03 15:19:00	3	3.22	-73.987648	40.759251
9973	524003	2	2016-04-03 14:54:00	2016-04-03 15:01:00	5	0.97	-73.972725	40.761528
9974	524004	1	2016-04-03 14:54:00	2016-04-03 15:02:00	2	0.90	-73.956940	40.777260
9975	524005	1	2016-04-03 14:54:00	2016-04-03 15:05:00	1	1.50	-73.996445	40.752983
9976	524006	2	2016-04-03 14:54:00	2016-04-03 15:17:00	1	1.46	-73.979218	40.740463
9977	524007	1	2016-04-03 14:54:00	2016-04-03 15:00:00	1	0.50	-73.991325	40.755508
9978	524008	1	2016-04-03 14:55:00	2016-04-03 15:02:00	1	0.70	-73.975365	40.765144
9979	524009	1	2016-04-03 14:55:00	2016-04-03 15:18:00	1	3.00	-73.998894	40.744770
9980	524010	1	2016-04-03 14:55:00	2016-04-03 15:08:00	1	1.90	-73.962524	40.778961
9981	524011	1	2016-04-03 14:55:00	2016-04-03 15:07:00	1	0.70	-73.987457	40.750614
9982	524012	2	2016-04-03 14:56:00	2016-04-03 15:06:00	1	2.02	-73.979057	40.750729
9983	524013	2	2016-04-03 14:56:00	2016-04-03 14:58:00	5	0.25	-73.981377	40.780472
9984	524014	1	2016-04-03 14:57:00	2016-04-03 15:19:00	1	0.80	-73.994881	40.760902
9985	524015	2	2016-04-03 14:57:00	2016-04-03 14:57:00	1	0.28	-73.950211	40.775558
9986	524016	2	2016-04-03 14:57:00	2016-04-03 15:19:00	1	9.64	-73.953384	40.775597
9987	524017	2	2016-04-03 14:57:00	2016-04-03 15:11:00	1	1.58	-74.005859	40.736000
9988	524018	1	2016-04-03 14:57:00	2016-04-03 15:09:00	1	1.70	-73.958519	40.777973
9989	524019	1	2016-04-03 14:57:00	2016-04-03 15:02:00	1	0.40	-73.995445	40.764736
9990	524020	2	2016-04-03 14:57:00	2016-04-03 15:32:00	5	6.11	-73.989304	40.741928
9991	524021	1	2016-04-03 14:57:00	2016-04-03 15:04:00	1	1.30	-73.957481	40.774326
9992	524022	2	2016-04-03 14:57:00	2016-04-03 15:02:00	1	0.88	-73.968239	40.751148
9993	524023	2	2016-04-03 14:57:00	2016-04-03 15:02:00	1	0.68	-74.004051	40.713070
9994	524024	2	2016-04-03 14:58:00	2016-04-03 15:06:00	2	1.38	-73.987366	40.760632

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_lat
9995	524025	1	2016-04-03 14:58:00	2016-04-03 15:06:00	2	1.30	-73.974884	40.750610
9996	524026	2	2016-04-03 14:58:00	2016-04-03 15:54:00	2	21.04	-73.788025	40.641590
9997	524027	1	2016-04-03 14:58:00	2016-04-03 15:11:00	1	2.30	-74.001534	40.719940
9998	524028	2	2016-04-03 14:58:00	2016-04-03 15:06:00	1	0.99	-73.981293	40.752968
9999	524029	1	2016-04-03 14:58:00	2016-04-03 15:01:00	1	0.50	-73.998558	40.740112

10000 rows × 14 columns

In [10]:

```
trips.trip_duration.describe()
```

Out[10]:

```
count    10000.000000
mean       746.098900
std        525.049496
min          2.000000
25%        364.000000
50%        616.000000
75%        988.000000
max       3573.000000
Name: trip_duration, dtype: float64
```

In [11]:

```
#busiest neighborhoods
tripcount = trips.groupby(['pickup_neighborhood'])

tripcount =  tripcount.pickup_neighborhood.count()
tripcount
```

Out[11]:

```
pickup_neighborhood
A      337
AA     444
AB     407
AC     407
AD     393
AE       3
AF       7
AG     149
AH     105
AI       3
AJ      28
AK     320
AL      51
AM     301
AN      38
AO     396
AP     373
AQ       1
AR     380
AS      55
AT     343
AU     225
AV     286
AW      15
B        2
C      329
D      434
E       21
F      113
```

```
G     154
H     370
I     291
J      89
K     264
L      46
M      17
N     316
O     272
P     282
Q     383
R     366
S      10
T       5
U     179
V      41
W      40
X     333
Y     295
Z      97
Name: pickup_neighborhood, dtype: int64
```

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.

--Describe the Dataset.

The database consists of 14 variables (id, vendor_id, pickup_datetime, dropoff_datetime, passenger_count, trip_distance, pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude, payment_type, trip_duration, pickup_neighborhood, dropoff_neighborhood). The source of dataset has been provided by NYC Taxi and Limosine Commission: TLC Trip Data" http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

--How many trips are in the dataset? How would you describe the distribution of trip durations?

There are 10,000 trips recorded in NYC taxi dataset. The trip duration data visually appears to have a normal distribution with a median of around 616 seconds (10 minutes). The time of the day is an important factor to determine the duration a trip can take. The same trip during rush hours can take much longer compared to non-rush hours. If we plot the mean Duration/Distance ratio with the pick-up hour of the day.

-- Is there anything else we should observe? Make sure the histogram is visible in the notebook.

The mean trip duration is 746.1 seconds (12.5 minutes), with a standard deviation of +/- 525 seconds. This distribution is right skewed with mean being greater than median.This clearly shows that there are more number of short trip duration with short distances in the city of newyork and and very less number of trips with almost duration of an hr. The mean passenger count was 1.7. The busiest overall pickup neighborhoods were coded AAA and D. There's a lot of granularity in the data, which was combined from three different raw datasets (trips, pickup_neighborhoods, dropoff_neighborhoods). There's considerable opportunity to build more useful features for predictive analysis.

Entities and Relationships

In [12]:

```
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_neighborhood"),
                 ("dropoff_neighborhoods", "neighborhood_id", "trips", "dropoff_neighborhood")]

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

```
Entities and relationships successful!
```

Transform Primitives

In [13]:

```
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!')
```

```
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/featuretools/entityset/entity.py:524:
FutureWarning: 'id' is both an index level and a column label.
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:
FutureWarning: 'neighborhood_id' is both an index level and a column label.
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)
```

```
Transform primitives successful!
```

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 [14]:

```
print(f"Number of features: {len(features)}")
features
```

```
Number of features: 13
```

Out[14]:

```
[<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 [15]:

```
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[15]:

	trip_distance	trip_duration	IS_WEEKEND(dropoff_datetime)	dropoff_neighborhood	dropoff_neighborhood	dropc

id	trip_distance	trip_duration	IS_WEEKEND(dropoff_datetime)	dropoff_neighborhood = D	dropoff_neighborhood = AA	dropo
514030	2.46	1039	True	0	0	0
514031	7.90	1454	True	0	0	0
514032	1.00	1168	True	0	0	0
514033	0.02	35	True	0	0	0
514034	19.00	3470	True	0	0	0

5 rows × 31 columns

First Model

In [16]:

```
# 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 [17]:

```
# 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            5.72s
         2           0.4148            4.97s
         3           0.3661            4.95s
         4           0.3266            4.84s
         5           0.2934            4.72s
         6           0.2665            4.73s
         7           0.2441            4.72s
         8           0.2257            4.69s
         9           0.2103            4.66s
        10           0.1973            4.58s
        20           0.1434            3.85s
        30           0.1312            3.31s
        40           0.1248            2.82s
        50           0.1218            2.22s
        60           0.1191            1.74s
        70           0.1174            1.28s
        80           0.1158            0.84s
        90           0.1147            0.41s
       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?

--Describe the 2 new features that we added to the model.

Two new transform primitives were created: Weekend (pickup_datetime) and WEEKEND (dropoff_datetime). They contain records of

trips that started and/or ended during a weekend (Saturday or Sunday), which is presumably the busiest time for NYC taxi drivers and most certainly reveal a different pattern regarding timing, pickup and dropoff locations, and number of passengers than non-WEEKEND trips.

WEEKEND(pickup_datetime) WEEKEND(dropoff_datetime)

--Do you think these improved the performance from a model that did not have these features? Why?

The day of the week can have a significant impact on the predictions because we expect the weekdays to be more congested than the weekends especially during the day time.We can clearly see that the average speed is higher on the weekend than the weekdays. Given that the pattern of trips varies significantly from the work week to the weekend, the WEEKEND features are likely to be the most heavily referenced features of any prediction of interest.

By pre-processing the dataset to identify this feature, it removed the need for complicated date comparisons during training and testing the predictor model. This definitely will result in computational savings (time, CPU cycles). Just as importantly, the addition of these features will allow for simpler and cleaner code for building the trainer and predictor.

More Transform Primitives

In [18]:

```
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:
FutureWarning: 'id' is both an index level and a column label.
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:
FutureWarning: 'neighborhood_id' is both an index level and a column label.
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)
```

```
Transform primitives successful!
```

In [19]:

```
print(f"Number of features: {len(features)}")
features
```

```
Number of features: 25
```

Out[19]:

```
[<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 [20]:

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

```
Elapsed: 00:04 | Remaining: 00:00 | Progress: 100%|██████████|| Calculated: 1/1 cutoff times
Finishing computing...
```

Out[20]:

	IS_WEEKEND(dropoff_datetime)	dropoff_neighborhood = D	dropoff_neighborhood = AA	dropoff_neighborhood = H	dropoff_n
id					
514030	True	0	0	0	0
514031	True	0	0	0	0
514032	True	0	0	0	0
514033	True	0	0	0	0
514034	True	0	0	0	0

5 rows × 43 columns

In [21]:

```
# 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 [22]:

```
# 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            6.80s
         2           0.4148            7.30s
         3           0.3661            7.01s
         4           0.3264            6.80s
         5           0.2930            6.63s
         6           0.2660            6.59s
         7           0.2432            6.36s
         8           0.2245            6.36s
         9           0.2090            6.22s
        10           0.1960            6.18s
        20           0.1362            5.36s
        30           0.1200            4.54s
        40           0.1126            3.64s
        50           0.1079            2.87s
        60           0.1047            2.20s
        70           0.1016            1.60s
```

```
        70            0.1016            1.60s
        80            0.0986            1.04s
        90            0.0938            0.51s
       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?

--Describe the rest of the new features that we just added to the model.

The new transform features were: Minute, Hour, Day, Week, Month, Weekday. The performance of the model improved by approximately 5% (from .752 to .806), with an improved R-squared value indicating better fit. Training time appears to have increased slightly.

hese features extract the parts of the pickup_datetime and dropoff_datetime that correspond to each particular time period; for example, which hour of the day, which day of the week, which month of the year. These will be useful for identifying time cycles and trends, which is of course the primary component of this dataset as a time series.

MINUTE(pickup_datetime) MINUTE(dropoff_datetime) HOUR(pickup_datetime) HOUR(dropoff_datetime) DAY(pickup_datetime) DAY(dropoff_datetime) WEEK(pickup_datetime) WEEK(dropoff_datetime) MONTH(pickup_datetime) MONTH(dropoff_datetime) WEEKDAY(pickup_datetime) WEEKDAY(dropoff_datetime)

--How did this affect performance? Did we have to sacrifice training time?

R-squared is the percentage of the response variable variation that is explained by a linear model. By comparing the performance of the previous model with the model containing the new features, we see an approximate 5% increase in the ability of the model to explain the behavior of the data. In short, yes, the additional features improved performance of the model.

R-Squared comparison: First Model: 0.754 Second Model:0.806

clear and short answer is "not appreciably". We could have started by creating the second model (instead of the first with a more limited set of primitive features). Technically, the second model took .614 seconds longer to compute.

Time Elapsed For Model Training: First Model: 3.476 Second Model: 4.090

As all these features are categorical in nature, representing them in their absolute form could be a problem of models like linear regression. One hot encoding provides an efficient method to capture patterns corresponding to all these categorical features.

Aggregation Primitives

In [23]:

```
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:
FutureWarning: 'id' is both an index level and a column label.
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:
FutureWarning: 'neighborhood_id' is both an index level and a column label.
Defaulting to column, but this will raise an ambiguity error in a future version
  inplace=True)
```

Aggregation primitives successful!

In [24]:

```
print(f"Number of features: {len(features)}")
features
```

Number of features: 75

Out[24]:

```
[<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 [25]:

```
# 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/calculate_feature_matrix.py:410: FutureWarning: 'id'
is both an index level and a column label.
Defaulting to column, but this will raise an ambiguity error in a future version
  on=target_index_var, how='left')[rvar].values
```

```
Elapsed: 00:07 | Remaining: 00:00 | Progress: 100%|██████████|| Calculated: 1/1 cutoff times
Finishing computing...
```

Out[25]:

	IS_WEEKEND(dropoff_datetime)	pickup_neighborhoods.MEDIAN(trips.trip_duration)	pickup_neighborhoods.STD(t
id			
514030	True	NaN	NaN
514031	True	NaN	NaN
514032	True	NaN	NaN
514033	True	NaN	NaN
514034	True	NaN	NaN

5 rows × 93 columns

In [26]:

```
# 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 [27]:

```
# 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            7.46s
         2           0.4148            7.05s
```

```
          3            0.3661            6.96s
          4            0.3264            7.37s
          5            0.2930            7.29s
          6            0.2660            7.12s
          7            0.2432            6.89s
          8            0.2245            6.89s
          9            0.2090            6.68s
         10            0.1960            6.51s
         20            0.1362            5.62s
         30            0.1200            4.75s
         40            0.1126            3.87s
         50            0.1079            3.09s
         60            0.1047            2.36s
         70            0.1016            1.71s
         80            0.0986            1.10s
         90            0.0938            0.54s
        100            0.0899            0.00s
0.8060128240391375
Training successful!
```

Evaluate on Test Data

In [28]:

```
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 [29]:

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

Out[29]:

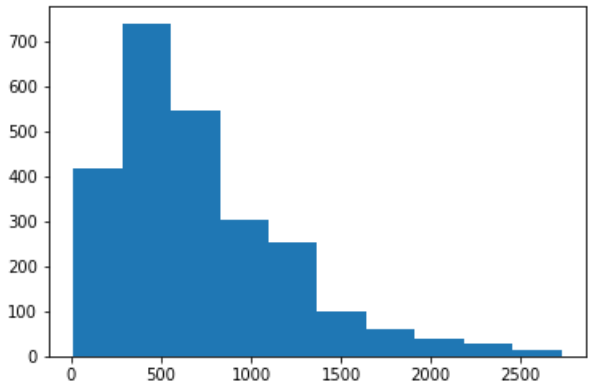
```
array([453.4415368 , 666.8066337 , 634.00494622, 617.34019506,
       407.96332751])
```

In [30]:

```
# Create a histogram of all of them
matplotlib.pyplot.hist(y_pred)

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

```
Histogram generation successful!
```



In [33]:

```
#median duration
y_pred.std()
```

Out[33]:

```
483.1724095255709
```

In [32]:

```
#average duration
y_pred.mean()
```

Out[32]:

```
718.2028273559516
```

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?

--Analyze the model predictions. The output distribution is close, but shifted slightly to the left, with the center of the distribution oriented closer to around 350 seconds and a smaller range with the tail extending not as far out (original histogram extended beyond 3,500 seconds).

The model predicted approximately 80.5% of the variation in the data, which is good (but not phenomenal). Chances are that better feature identification or model selection would improve the fitness of the model. The first five predictions of the model were as follows (in trip_duration seconds):

453.4415368

666.8066337

634.00494622

617.34019506

407.96332751

If we improved the R-squared for our model through better feature and model selection, it would expect that the key indicator statistics would match better between the predictions and the original data. For example, the predictions' statistical mean and standard deviation are close to, but not precisely the same as those of the original data for trip_duration. However, the predictions do appear to match our expectations.

-- 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?

the distribution of this model matches the distribution of the original data almost exactly. The distribution appears to be normal, in this case with an apparent median around 350 (somewhat lower than the original). The population is also smaller (around 700) which likely affected the distribution. To improve this, we could add more aggregation or transform primitives, using seed features, or specifying maximum depth of features with max_depth.

The best features will capture intuitive elements of typical taxicab trips, especially if they can result in useful insights that can be leveraged for action (e.g. advertising, staffing, logistics, etc.). The key elements or utility, and expected cyclic patterns in the time data. Some examples could include the following:

Lunch_Rush (pickup_datetime), Lunch_Rush (dropoff_datetime): Did the trip occur between 11:30am and 1:30pm? Often co-workers will share cabs for travel to nearby lunch destinations.

-Work_Rush (pickup_datetime), Work_Rush (dropoff_datetime): Did the trip occur between 6:30am and 8:30am, or between 4:00pm and 6pm? These are key times for commuter travel, and will likely focus on major business areas.

-Airport (pickup_destination), Airport (dropoff_destination): Was the trip to or from a key transportation hub? These are likely to not be local residents, and have atypical trip patterns that may otherwise confound the model.

-Short_Trip: Was the trip for a minor distance? Is there a way to maximize profit and minimize lost fares from too many "short trips"?

-Is_Downtown, Is_Suburb, Is_Bronx, Is_Brooklyn, etc: By grouping the Areas into the major NYC buroughs, it may be possible to extract useful travel patterns.

An important aspect of this problem was to use the pickup and drop-off coordinates effectively.

Feature Importance

In [31]:

```
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.

--Analyze the feature importance values you just computed above. Do they make sense? Are there any values you are surprised by?

The data was processed to extract separate features for year,month,day,weekday,hour and minute from the date and time of each ride, as well as trip duration as the difference between dropoff and pickup time. Furthermore, with the objective to model and account for traffic in the predictions, two more features were calculated from the data; rides in an hour and average speed during the hour. The three most influential features in our prediction model are:

pickup_neighborhoods.MAX(trips.passenger_count) HOUR(pickup_datetime) MINUTE(dropoff_datetime)

The highest importance feature is max passenger count for pickup (0.325). This makes sense, as the number of passengers to pickup could have a significant impact on the duration of the trip. Within that, a major determining factor is the number of passengers getting into the cab from which pickup neighborhood. Combining the pickup neighborhood with the hour (time of day) of the pickup, these two variables constitute 43.% of the variability in the prediction model - the remaining 23 features make up the remaining 57%. In other words, where you are picked up and what time of day your are picked up are most determinative of your trip duration. Further, and as I indicated previously, the importance of these features indicate the need for further exploration of transform primitives and feature creation that capture time-of-day cycle patterns, and neighborhood-to-neighborhood or area-to-area (e.g., downtown to suburb) commuting patterns.The importance of minute for dropoff time (0.085) compared to other features was suprising.

These values makes a logical, if naive, sense. What this essentially tells us is that the length of a taxi trip is most determined by the number of passengers that get into the cab (usually, multiple passengers mean multiple end destinations and less direct routes), the time of day when the passengers were picked up, and how long it took before the driver dropped off the last passenger (most cab rides are less than an hour, mean=12.4 minutes). We shouldn't dismiss the confirmatory value that our model is able to extract the most salient patterns which we might automatically infer as humans; it does provide support that we have built our models and handled our data correctly.

However, the feature importance reveals that we might not be illuminating any deeper insights from the data beyond the immediately obvious, and more work should be done to explore and extract such deeper patterns.

--Give some brief explanations as to why these features are relevant in computing the trip_duration target variable.

While time-of-day clearly influences taxi travel patterns, the HOUR feature should have been removed and replaced with more intuitive transform primitives like RUSH_HOUR, LUNCH_HOUR, LATE_NIGHT, etc. Unfortunately, HOUR(pickup_datetime) and MINUTE(dropoff_datetime) appear to be just recreating the formula used to calculate trip_duration (e.g., (time_start - time_end)=trip_duration) , so these features should have been filtered from the training model entirely to let us focus on more meaningful features.

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