MODULE 6: PREDICTIVE MODELING FOR TEMPORARY DATA

CASE STUDY ACTIVITY TUTORIAL

6.1 NEW YORK CITY CASE STUDY

NewYorkCity_taxi_case_study

October 16, 2017

1 New York City Taxi Ride Duration Prediction

In this case study, we will build a predictive model to predict the duration of taxi ride. We will do the following steps: * First install the dependencies * Next load the data as pandas dataframe * Define the outcome variable- the variable we are trying to predict. * Build features using featuretools package - that implements Deep Feature Synthesis. We will start with simple features and incrementally improve the feature definitions and examine the accuracy of the system.

Allocate atleast 2-3 hours to go through this case study end-to-end

2 Install Dependencies

If you have not done so already, download this repository from git. Once you have downloaded this archive, unzip it and cd into the directory from the command line. Next run the command `./install_osx.sh` if you are on a mac or `./install_linux.sh` if you are on linux. This should install all of the dependencies.

If you are on a windows machine, open the requirements.txt folder and make sure to install each of the dependencies listed (featuretools, jupyter, pandas, sklearn, xgboost, numpy)

Once you have installed all of the dependencies, open this notebook. On Mac and Linux, navigate to the directory that you downloaded from git and run `jupyter notebook` to be taken to this notebook in your default web browser. When you open the NewYorkCity_taxi_case_study.ipynb file in the web browser, you can step through the code by clicking the `Run` button at the top of the page. If you have any questions for how to use Jupyter, refer to google or the discussion forum.

3 Running the Code

```
In [1]: import pandas as pd
        import numpy as np
        import featuretools as ft
        import utils
        from utils import load_nyc_taxi_data, compute_features, preview
        from sklearn.metrics import mean_squared_error
        from math import sqrt
        from featuretools.primitives import (Day, Hour, Minute, Month, Weekday,
                                             Week, Weekend, Sum, Mean, Median, Std)
        ft.__version__
```

```
%load_ext autoreload
%autoreload 2
```

4 Step 1: Download and load the raw data as pandas dataframes

If you have not yet downloaded the data it can be downloaded from S3. Once you have downloaded the archive, unzip it and place the nyc-taxi-data folder in the same directory as this script.

In [2]:
```
trips, passenger_cnt, vendors = load_nyc_taxi_data()
preview(trips,10)
```

Out[2]:

	id	vendor_id	pickup_datetime	dropoff_datetime
0	0	2	2016-01-01 00:00:19	2016-01-01 00:06:31
679995	679995	1	2016-04-30 12:57:36	2016-04-30 13:04:36
679996	679996	2	2016-04-30 12:57:40	2016-04-30 13:06:01
679997	679997	2	2016-04-30 12:57:45	2016-04-30 13:07:11
679998	679998	2	2016-04-30 12:57:49	2016-04-30 13:15:28
679999	679999	2	2016-04-30 12:58:04	2016-04-30 13:08:30
680000	680000	2	2016-04-30 12:58:33	2016-04-30 13:08:51
680001	680001	2	2016-04-30 12:58:39	2016-04-30 13:16:19
680002	680002	2	2016-04-30 12:58:47	2016-04-30 13:13:47
680003	680003	1	2016-04-30 12:58:56	2016-04-30 13:24:28

	passenger_count	trip_distance	pickup_longitude	pickup_latitude
0	3	1.32	-73.961258	40.796200
679995	1	1.10	-73.979973	40.770679
679996	5	1.22	-73.940399	40.793880
679997	1	1.58	-73.924728	40.744068
679998	1	4.76	-73.985863	40.746799
679999	1	1.79	-73.959747	40.773682
680000	1	1.32	-73.981300	40.752972
680001	2	1.99	-73.987549	40.756226
680002	2	4.00	-73.951172	40.774220
680003	1	6.10	-74.008163	40.703640

	store_and_fwd_flag	dropoff_longitude	dropoff_latitude	payment_type
0	False	-73.950050	40.787312	2
679995	False	-73.969696	40.785587	1
679996	False	-73.952667	40.804859	1
679997	False	-73.953087	40.749290	1
679998	False	-74.005951	40.711269	2
679999	False	-73.981071	40.778381	2
680000	False	-73.973923	40.764381	1
680001	False	-73.998032	40.765732	2
680002	False	-73.909988	40.801823	1
680003	False	-73.984138	40.758980	2

	trip_duration

```
0              372.0
679995         420.0
679996         501.0
679997         566.0
679998        1059.0
679999         626.0
680000         618.0
680001        1060.0
680002         900.0
680003        1532.0
```

The `trips` table has the following fields * `id` which uniquely identifies the trip * `vendor_id` is the taxi cab company - in our case study we have data from three different cab companies * `pickup_datetime` the time stamp for pickup * `dropoff_datetime` the time stamp for drop-off * `passenger_count` the number of passengers for the trip * `trip_distance` total distance of the trip in miles * `pickup_longitude` the longitude for pickup * `pickup_latitude` the latitude for pickup * `dropoff_longitude`the longitude of dropoff * `dropoff_latitude` the latitude of dropoff * `payment_type` A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided * `trip_duration` this is the duration we would like to predict using other fields

5 Step 2: Prepare the Data

Lets create entities and relationships. The three entities in this data are * trips * vendors (these are the cab companies) * passenger_cnt (a simple entity that has the unique number of passenger counts 1-8)

This data has the following relationships * Vendors --> trips (the same vendor can have multiple trips - vendors is the `parent_entity` and trips it the child entity * passenger_cnt --> trips (the same passenger_cnt can appear in multiple trips. passenger_cnt is the `parent_entity` and trips is the child entity.

In , we specify the list of entities and relationships as follows:

```
In [3]: entities = {
                "trips": (trips, "id", 'pickup_datetime' ),
                "vendors": (vendors, "vendor_id"),
                "passenger_cnt": (passenger_cnt,"passenger_count")
                }

        relationships = [("vendors", "vendor_id","trips", "vendor_id"),
                         ("passenger_cnt", "passenger_count","trips", "passenger_count")]
```

We specify the time for each instance of the target_entity, in this case `trips` to calculate features. The timestamp represents the last time data can be used for calculating features by DFS. This is specified using a dataframe of cutoff time. This cutoff time for each trip is the pickup time.

```
In [4]: cutoff_time = (trips[['id', 'pickup_datetime']])
        preview(cutoff_time,10)
```

```
Out[4]:             id     pickup_datetime
        0            0 2016-01-01 00:00:19
        679995  679995 2016-04-30 12:57:36
        679996  679996 2016-04-30 12:57:40
        679997  679997 2016-04-30 12:57:45
        679998  679998 2016-04-30 12:57:49
        679999  679999 2016-04-30 12:58:04
        680000  680000 2016-04-30 12:58:33
        680001  680001 2016-04-30 12:58:39
        680002  680002 2016-04-30 12:58:47
        680003  680003 2016-04-30 12:58:56
```

6 Step 3: Create baseline features using DFS

Instead of manually creating features, such as month of pickup_datetime, we can let featuretools come up with them.

Featuretools does this by * interpret the types of variables - categorical, numeric and others. We can override this interpretation by specifying the types. In this case study, we wanted passenger_count to be a type of Ordinal, and vendor_id to be of type Categorical. This override occured while loading in the csv files.

- then based on the primitives we specify, it matches up the columns to which those primitives can be applied.

7 Create transform features using transform primitives

As we described in the video, features fall into two major categories, `transform` and `aggregate`. In featureools, we can create transform features by specifying `transform` primitives. Below we specify a `transform` primitive called `weekend` and here is what it does:

- It can be applied to any `datetime` column in the data.
- For each entry in the column, it assess if it is a `weekend` and returns a boolean.

In this specific data, there are two `datetime` columns `pickup_datetime` and `dropoff_datetime`. The tool automatically creates features using the primitive and these two columns as shown below.

```
In [5]: trans_primitives = [Weekend]

        features = ft.dfs(entities=entities,
                          relationships=relationships,
                          target_entity="trips",
                          trans_primitives=trans_primitives,
                          agg_primitives=[],
                          features_only=True)
```

Here are the features created.

```
In [6]: print len(features)
        features
```

12

```
Out[6]: [<Feature: vendor_id>,
         <Feature: passenger_count>,
         <Feature: payment_type>,
         <Feature: dropoff_longitude>,
         <Feature: pickup_latitude>,
         <Feature: trip_duration>,
         <Feature: store_and_fwd_flag>,
         <Feature: trip_distance>,
         <Feature: dropoff_latitude>,
         <Feature: pickup_longitude>,
         <Feature: IS_WEEKEND(dropoff_datetime)>,
         <Feature: IS_WEEKEND(pickup_datetime)>]
```

Now let's compute the features.

```
In [7]: feature_matrix = compute_features(features,cutoff_time)
```

8 Step 4: Build the Model

To build a model, * we first seperate the data into a porition for `training` (75% in this case) and a portion for `testing` * We also get the log of the trip duration so that a more linear relationship can be found. * We use `XGBOOST` to train a model.

```
In [8]: # separates the whole feature matrix into train data feature matrix,
        # train data labels, and test data feature matrix
        X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
        y_train = np.log(y_train.values + 1)
```

```
In [9]: model = utils.train_xgb(X_train, y_train)
```

```
[0]        train-rmse:4.98698        valid-rmse:4.98587
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stopping.

Will train until valid-rmse hasn't improved in 50 rounds.
[10]        train-rmse:0.973206        valid-rmse:0.972554
[20]        train-rmse:0.436417        valid-rmse:0.436489
[30]        train-rmse:0.380745        valid-rmse:0.382061
[40]        train-rmse:0.37503        valid-rmse:0.377282
[50]        train-rmse:0.367368        valid-rmse:0.370566
[60]        train-rmse:0.362789        valid-rmse:0.366918
[70]        train-rmse:0.358907        valid-rmse:0.364013
[80]        train-rmse:0.357262        valid-rmse:0.362921
[90]        train-rmse:0.354699        valid-rmse:0.361165
[100]        train-rmse:0.353081        valid-rmse:0.360219
[110]        train-rmse:0.351461        valid-rmse:0.359141
```

```
[120]        train-rmse:0.35009        valid-rmse:0.358254
[130]        train-rmse:0.34822        valid-rmse:0.357092
[140]        train-rmse:0.346831        valid-rmse:0.35624
[150]        train-rmse:0.346074        valid-rmse:0.355775
[160]        train-rmse:0.345375        valid-rmse:0.3554
[170]        train-rmse:0.34477        valid-rmse:0.355074
[180]        train-rmse:0.343869        valid-rmse:0.35461
[190]        train-rmse:0.343394        valid-rmse:0.354408
[200]        train-rmse:0.343124        valid-rmse:0.354356
[210]        train-rmse:0.342747        valid-rmse:0.354204
[220]        train-rmse:0.342269        valid-rmse:0.353976
[226]        train-rmse:0.34179        valid-rmse:0.353823
Modeling RMSE 0.35382
```

9 Step 5: Adding more Transform Primitives

- Adding `Minute` `Hour` `Week` `Month` `Weekday` primitives
- All these transform primitives apply to `datetime` column

```
In [10]: 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=[],
                           features_only=True)

In [11]: print len(features)
         features

36


Out[11]: [<Feature: passenger_count>,
          <Feature: dropoff_longitude>,
          <Feature: payment_type>,
          <Feature: store_and_fwd_flag>,
          <Feature: vendor_id>,
          <Feature: pickup_latitude>,
          <Feature: pickup_longitude>,
          <Feature: trip_duration>,
          <Feature: trip_distance>,
          <Feature: dropoff_latitude>,
          <Feature: WEEKDAY(pickup_datetime)>,
          <Feature: WEEK(dropoff_datetime)>,
          <Feature: HOUR(pickup_datetime)>,
```

```
 <Feature: WEEKDAY(dropoff_datetime)>,
 <Feature: DAY(pickup_datetime)>,
 <Feature: MONTH(pickup_datetime)>,
 <Feature: WEEK(pickup_datetime)>,
 <Feature: DAY(dropoff_datetime)>,
 <Feature: MONTH(dropoff_datetime)>,
 <Feature: HOUR(dropoff_datetime)>,
 <Feature: IS_WEEKEND(pickup_datetime)>,
 <Feature: IS_WEEKEND(dropoff_datetime)>,
 <Feature: MINUTE(pickup_datetime)>,
 <Feature: MINUTE(dropoff_datetime)>,
 <Feature: passenger_cnt.WEEK(first_trips_time)>,
 <Feature: vendors.DAY(first_trips_time)>,
 <Feature: passenger_cnt.WEEKDAY(first_trips_time)>,
 <Feature: vendors.WEEKDAY(first_trips_time)>,
 <Feature: vendors.MONTH(first_trips_time)>,
 <Feature: passenger_cnt.DAY(first_trips_time)>,
 <Feature: passenger_cnt.MINUTE(first_trips_time)>,
 <Feature: passenger_cnt.HOUR(first_trips_time)>,
 <Feature: vendors.HOUR(first_trips_time)>,
 <Feature: passenger_cnt.MONTH(first_trips_time)>,
 <Feature: vendors.MINUTE(first_trips_time)>,
 <Feature: vendors.WEEK(first_trips_time)>]
```

Now let's compute the features.

```
In [12]: feature_matrix = compute_features(features,cutoff_time)

In [13]: preview(feature_matrix,10)
```

Out[13]:

id	passenger_count	dropoff_longitude	payment_type	store_and_fwd_flag	vendor_id	pickup_latitude	pickup_longitude	trip_duration
0	3	-73.950050	2	False	2	40.796200	-73.961258	372.0
679995	1	-73.969696	1	False	1	40.770679	-73.979973	420.0
679996	5	-73.952667	1	False	2	40.793880	-73.940399	501.0
679997	1	-73.953087	1	False	2	40.744068	-73.924728	566.0
679998	1	-74.005951	2	False				
679999	1	-73.981071	2	False				
680000	1	-73.973923	1	False				
680001	2	-73.998032	2	False				
680002	2	-73.909988	1	False				
680003	1	-73.984138	2	False				

```
679998         2        40.746799        -73.985863         1059.0
679999         2        40.773682        -73.959747          626.0
680000         2        40.752972        -73.981300          618.0
680001         2        40.756226        -73.987549         1060.0
680002         2        40.774220        -73.951172          900.0
680003         1        40.703640        -74.008163         1532.0

        trip_distance  dropoff_latitude               ...                \
id                                                   ...
0                1.32         40.787312               ...
679995           1.10         40.785587               ...
679996           1.22         40.804859               ...
679997           1.58         40.749290               ...
679998           4.76         40.711269               ...
679999           1.79         40.778381               ...
680000           1.32         40.764381               ...
680001           1.99         40.765732               ...
680002           4.00         40.801823               ...
680003           6.10         40.758980               ...

        passenger_cnt.WEEKDAY(first_trips_time)  \
id
0                                             4
679995                                        4
679996                                        4
679997                                        4
679998                                        4
679999                                        4
680000                                        4
680001                                        4
680002                                        4
680003                                        4

        vendors.WEEKDAY(first_trips_time)  vendors.MONTH(first_trips_time)  \
id
0                                       4                                 1
679995                                  4                                 1
679996                                  4                                 1
679997                                  4                                 1
679998                                  4                                 1
679999                                  4                                 1
680000                                  4                                 1
680001                                  4                                 1
680002                                  4                                 1
680003                                  4                                 1

        passenger_cnt.DAY(first_trips_time)  \
id
```

0	1
679995	1
679996	1
679997	1
679998	1
679999	1
680000	1
680001	1
680002	1
680003	1

id	passenger_cnt.MINUTE(first_trips_time)	\
0	0	
679995	45	
679996	7	
679997	45	
679998	45	
679999	45	
680000	45	
680001	47	
680002	47	
680003	45	

id	passenger_cnt.HOUR(first_trips_time)	vendors.HOUR(first_trips_time)	\
0	0	0	
679995	1	0	
679996	0	0	
679997	1	0	
679998	1	0	
679999	1	0	
680000	1	0	
680001	1	0	
680002	1	0	
680003	1	0	

id	passenger_cnt.MONTH(first_trips_time)	\
0	1	
679995	1	
679996	1	
679997	1	
679998	1	
679999	1	
680000	1	
680001	1	
680002	1	

```
680003                                        1

        vendors.MINUTE(first_trips_time)  vendors.WEEK(first_trips_time)
id
0                                        1                             53
679995                                   1                             53
679996                                   1                             53
679997                                   1                             53
679998                                   1                             53
679999                                   1                             53
680000                                   1                             53
680001                                   1                             53
680002                                   1                             53
680003                                   1                             53

[10 rows x 36 columns]
```

10 Step 6: Build the new model

In [14]:
```
# separates the whole feature matrix into train data feature matrix,
# train data labels, and test data feature matrix
X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
y_train = np.log(y_train.values + 1)
```

In [15]:
```
model = utils.train_xgb(X_train, y_train)
```

```
[0]        train-rmse:4.99672        valid-rmse:4.99546
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stopping.

Will train until valid-rmse hasn't improved in 50 rounds.
[10]        train-rmse:0.926123        valid-rmse:0.925607
[20]        train-rmse:0.398269        valid-rmse:0.398866
[30]        train-rmse:0.336353        valid-rmse:0.338614
[40]        train-rmse:0.319268        valid-rmse:0.322974
[50]        train-rmse:0.29361        valid-rmse:0.299003
[60]        train-rmse:0.281483        valid-rmse:0.288059
[70]        train-rmse:0.257367        valid-rmse:0.265217
[80]        train-rmse:0.242748        valid-rmse:0.251557
[90]        train-rmse:0.236299        valid-rmse:0.246339
[100]        train-rmse:0.221303        valid-rmse:0.232359
[110]        train-rmse:0.21447        valid-rmse:0.226336
[120]        train-rmse:0.205326        valid-rmse:0.217802
[130]        train-rmse:0.203326        valid-rmse:0.21675
[140]        train-rmse:0.195485        valid-rmse:0.209856
[150]        train-rmse:0.194128        valid-rmse:0.209188
[160]        train-rmse:0.187765        valid-rmse:0.203539
[170]        train-rmse:0.178377        valid-rmse:0.19481
[180]        train-rmse:0.175451        valid-rmse:0.19234
```

```
[190]	train-rmse:0.170608	valid-rmse:0.187837
[200]	train-rmse:0.168245	valid-rmse:0.185726
[210]	train-rmse:0.161733	valid-rmse:0.179729
[220]	train-rmse:0.160597	valid-rmse:0.179161
[226]	train-rmse:0.158871	valid-rmse:0.177684
Modeling RMSE 0.17768
```

11 Step 7: Add Aggregation Primitives

Now let's add aggregation primitives. These primitives will generate features for the parent entities in this case both `vendors` and `passenger_cnt` and then add them to the trips entity (which is the entity for which we are trying to make prediction.

```
In [16]: trans_primitives = [Minute, Hour, Day, Week, Month, Weekday, Weekend]
         aggregation_primitives = [Sum, Mean, Median, Std]

         features = ft.dfs(entities=entities,
                           relationships=relationships,
                           target_entity="trips",
                           trans_primitives=trans_primitives,
                           agg_primitives=aggregation_primitives,
                           features_only=True)
```

```
In [17]: print len(features)
         features
```

```
92
```

```
Out[17]: [<Feature: payment_type>,
          <Feature: store_and_fwd_flag>,
          <Feature: dropoff_longitude>,
          <Feature: pickup_longitude>,
          <Feature: trip_duration>,
          <Feature: vendor_id>,
          <Feature: passenger_count>,
          <Feature: pickup_latitude>,
          <Feature: trip_distance>,
          <Feature: dropoff_latitude>,
          <Feature: MONTH(pickup_datetime)>,
          <Feature: HOUR(dropoff_datetime)>,
          <Feature: MINUTE(pickup_datetime)>,
          <Feature: HOUR(pickup_datetime)>,
          <Feature: WEEKDAY(dropoff_datetime)>,
          <Feature: DAY(pickup_datetime)>,
          <Feature: IS_WEEKEND(pickup_datetime)>,
          <Feature: IS_WEEKEND(dropoff_datetime)>,
```

```
<Feature: WEEK(dropoff_datetime)>,
<Feature: WEEK(pickup_datetime)>,
<Feature: MONTH(dropoff_datetime)>,
<Feature: WEEKDAY(pickup_datetime)>,
<Feature: DAY(dropoff_datetime)>,
<Feature: MINUTE(dropoff_datetime)>,
<Feature: passenger_cnt.STD(trips.pickup_longitude)>,
<Feature: passenger_cnt.SUM(trips.pickup_longitude)>,
<Feature: vendors.SUM(trips.dropoff_longitude)>,
<Feature: passenger_cnt.WEEKDAY(first_trips_time)>,
<Feature: passenger_cnt.STD(trips.payment_type)>,
<Feature: vendors.MEDIAN(trips.trip_distance)>,
<Feature: passenger_cnt.MEDIAN(trips.trip_distance)>,
<Feature: vendors.HOUR(first_trips_time)>,
<Feature: passenger_cnt.MEAN(trips.dropoff_longitude)>,
<Feature: vendors.WEEKDAY(first_trips_time)>,
<Feature: passenger_cnt.DAY(first_trips_time)>,
<Feature: vendors.SUM(trips.pickup_longitude)>,
<Feature: vendors.STD(trips.trip_distance)>,
<Feature: passenger_cnt.SUM(trips.pickup_latitude)>,
<Feature: passenger_cnt.STD(trips.trip_duration)>,
<Feature: passenger_cnt.MINUTE(first_trips_time)>,
<Feature: passenger_cnt.HOUR(first_trips_time)>,
<Feature: passenger_cnt.SUM(trips.trip_duration)>,
<Feature: passenger_cnt.MEDIAN(trips.dropoff_longitude)>,
<Feature: vendors.STD(trips.pickup_latitude)>,
<Feature: vendors.STD(trips.trip_duration)>,
<Feature: vendors.MEAN(trips.payment_type)>,
<Feature: vendors.MEAN(trips.dropoff_latitude)>,
<Feature: vendors.MONTH(first_trips_time)>,
<Feature: vendors.SUM(trips.payment_type)>,
<Feature: passenger_cnt.MEAN(trips.payment_type)>,
<Feature: vendors.MEDIAN(trips.dropoff_longitude)>,
<Feature: passenger_cnt.MEDIAN(trips.pickup_latitude)>,
<Feature: passenger_cnt.MONTH(first_trips_time)>,
<Feature: passenger_cnt.MEAN(trips.dropoff_latitude)>,
<Feature: passenger_cnt.MEDIAN(trips.pickup_longitude)>,
<Feature: vendors.WEEK(first_trips_time)>,
<Feature: passenger_cnt.STD(trips.dropoff_longitude)>,
<Feature: vendors.STD(trips.payment_type)>,
<Feature: passenger_cnt.WEEK(first_trips_time)>,
<Feature: vendors.SUM(trips.trip_distance)>,
<Feature: passenger_cnt.MEAN(trips.trip_distance)>,
<Feature: vendors.MEDIAN(trips.trip_duration)>,
<Feature: vendors.STD(trips.dropoff_longitude)>,
<Feature: vendors.DAY(first_trips_time)>,
<Feature: passenger_cnt.STD(trips.pickup_latitude)>,
<Feature: vendors.SUM(trips.pickup_latitude)>,
```

```
 <Feature: passenger_cnt.MEAN(trips.trip_duration)>,
 <Feature: passenger_cnt.SUM(trips.dropoff_latitude)>,
 <Feature: vendors.MEDIAN(trips.pickup_latitude)>,
 <Feature: passenger_cnt.STD(trips.trip_distance)>,
 <Feature: vendors.SUM(trips.trip_duration)>,
 <Feature: passenger_cnt.SUM(trips.dropoff_longitude)>,
 <Feature: passenger_cnt.MEAN(trips.pickup_latitude)>,
 <Feature: vendors.STD(trips.dropoff_latitude)>,
 <Feature: vendors.SUM(trips.dropoff_latitude)>,
 <Feature: vendors.MEAN(trips.pickup_latitude)>,
 <Feature: vendors.MINUTE(first_trips_time)>,
 <Feature: passenger_cnt.SUM(trips.payment_type)>,
 <Feature: vendors.MEAN(trips.trip_distance)>,
 <Feature: vendors.MEAN(trips.trip_duration)>,
 <Feature: vendors.STD(trips.pickup_longitude)>,
 <Feature: vendors.MEDIAN(trips.pickup_longitude)>,
 <Feature: passenger_cnt.MEAN(trips.pickup_longitude)>,
 <Feature: vendors.MEDIAN(trips.dropoff_latitude)>,
 <Feature: passenger_cnt.SUM(trips.trip_distance)>,
 <Feature: passenger_cnt.MEDIAN(trips.payment_type)>,
 <Feature: passenger_cnt.STD(trips.dropoff_latitude)>,
 <Feature: passenger_cnt.MEDIAN(trips.trip_duration)>,
 <Feature: vendors.MEDIAN(trips.payment_type)>,
 <Feature: passenger_cnt.MEDIAN(trips.dropoff_latitude)>,
 <Feature: vendors.MEAN(trips.pickup_longitude)>,
 <Feature: vendors.MEAN(trips.dropoff_longitude)>]
```

In [18]:
```
feature_matrix = compute_features(features,cutoff_time)
```

In [19]:
```
preview(feature_matrix,10)
```

Out[19]:

id	payment_type	store_and_fwd_flag	dropoff_longitude	pickup_longitude	\
510001	2	False	-73.998131	-73.982216	
679994	1	False	-74.000252	-74.009727	
679995	1	False	-73.969696	-73.979973	
679996	1	False	-73.952667	-73.940399	
679997	1	False	-73.953087	-73.924728	
679998	2	False	-74.005951	-73.985863	
679999	2	False	-73.981071	-73.959747	
680000	1	False	-73.973923	-73.981300	
680001	2	False	-73.998032	-73.987549	
680002	1	False	-73.909988	-73.951172	

id	trip_duration	vendor_id	passenger_count	pickup_latitude	\
510001	674.0	1	1	40.763084	
679994	612.0	2	1	40.713009	

```
679995          420.0         1                 1        40.770679
679996          501.0         2                 5        40.793880
679997          566.0         2                 1        40.744068
679998         1059.0         2                 1        40.746799
679999          626.0         2                 1        40.773682
680000          618.0         2                 1        40.752972
680001         1060.0         2                 2        40.756226
680002          900.0         2                 2        40.774220

        trip_distance  dropoff_latitude  \
id
510001           1.50         40.765652
679994           1.08         40.726639
679995           1.10         40.785587
679996           1.22         40.804859
679997           1.58         40.749290
679998           4.76         40.711269
679999           1.79         40.778381
680000           1.32         40.764381
680001           1.99         40.765732
680002           4.00         40.801823

                      ...                    \
id                    ...
510001                ...
679994                ...
679995                ...
679996                ...
679997                ...
679998                ...
679999                ...
680000                ...
680001                ...
680002                ...

        passenger_cnt.MEAN(trips.pickup_longitude)  \
id
510001                                  -73.974532
679994                                  -73.974591
679995                                  -73.974591
679996                                  -73.973670
679997                                  -73.974591
679998                                  -73.974591
679999                                  -73.974591
680000                                  -73.974591
680001                                  -73.974236
680002                                  -73.974236
```

```
        vendors.MEDIAN(trips.dropoff_latitude)  \
id
510001                                40.754410
679994                                40.754723
679995                                40.754452
679996                                40.754723
679997                                40.754723
679998                                40.754723
679999                                40.754723
680000                                40.754723
680001                                40.754723
680002                                40.754723

        passenger_cnt.SUM(trips.trip_distance)  \
id
510001                              5549035.11
679994                              5948678.22
679995                              5948678.22
679996                               102501.77
679997                              5948678.22
679998                              5948678.22
679999                              5948678.22
680000                              5948678.22
680001                               277797.69
680002                               277797.69

        passenger_cnt.MEDIAN(trips.payment_type)  \
id
510001                                       1.0
679994                                       1.0
679995                                       1.0
679996                                       1.0
679997                                       1.0
679998                                       1.0
679999                                       1.0
680000                                       1.0
680001                                       1.0
680002                                       1.0

        passenger_cnt.STD(trips.dropoff_latitude)  \
id
510001                                   0.029019
679994                                   0.029112
679995                                   0.029112
679996                                   0.029189
679997                                   0.029112
679998                                   0.029112
679999                                   0.029112
```

```
680000                                    0.029112
680001                                    0.029832
680002                                    0.029832

        passenger_cnt.MEDIAN(trips.trip_duration)  \
id
510001                                      622.0
679994                                      631.0
679995                                      631.0
679996                                      651.0
679997                                      631.0
679998                                      631.0
679999                                      631.0
680000                                      631.0
680001                                      666.0
680002                                      666.0

        vendors.MEDIAN(trips.payment_type)  \
id
510001                                 1.0
679994                                 1.0
679995                                 1.0
679996                                 1.0
679997                                 1.0
679998                                 1.0
679999                                 1.0
680000                                 1.0
680001                                 1.0
680002                                 1.0

        passenger_cnt.MEDIAN(trips.dropoff_latitude)  \
id
510001                                   40.754593
679994                                   40.754616
679995                                   40.754616
679996                                   40.754700
679997                                   40.754616
679998                                   40.754616
679999                                   40.754616
680000                                   40.754616
680001                                   40.754372
680002                                   40.754372

        vendors.MEAN(trips.pickup_longitude)  \
id
510001                           -73.975087
679994                           -73.973981
679995                           -73.975138
```

```
679996                                -73.973981
679997                                -73.973981
679998                                -73.973981
679999                                -73.973981
680000                                -73.973981
680001                                -73.973981
680002                                -73.973981

        vendors.MEAN(trips.dropoff_longitude)
id
510001                                 -73.974294
679994                                 -73.974037
679995                                 -73.974177
679996                                 -73.974037
679997                                 -73.974037
679998                                 -73.974037
679999                                 -73.974037
680000                                 -73.974037
680001                                 -73.974037
680002                                 -73.974037

[10 rows x 92 columns]
```

12 Step 8: Build the new model

```
In [20]: # separates the whole feature matrix into train data feature matrix,
         # train data labels, and test data feature matrix
         X_train, y_train, X_test, y_test = utils.get_train_test_fm(feature_matrix,.75)
         y_train = np.log(y_train.values + 1)

In [21]: model = utils.train_xgb(X_train, y_train)

[0]        train-rmse:4.9967        valid-rmse:4.99551
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stopping.

Will train until valid-rmse hasn't improved in 50 rounds.
[10]        train-rmse:0.906906        valid-rmse:0.9066
[20]        train-rmse:0.366739        valid-rmse:0.367707
[30]        train-rmse:0.321117        valid-rmse:0.323958
[40]        train-rmse:0.302714        valid-rmse:0.307198
[50]        train-rmse:0.268837        valid-rmse:0.275106
[60]        train-rmse:0.245893        valid-rmse:0.253568
[70]        train-rmse:0.239083        valid-rmse:0.24787
[80]        train-rmse:0.226624        valid-rmse:0.236732
[90]        train-rmse:0.216111        valid-rmse:0.22746
[100]        train-rmse:0.214139        valid-rmse:0.226118
[110]        train-rmse:0.206802        valid-rmse:0.219553
[120]        train-rmse:0.205206        valid-rmse:0.218796
```

```
[130]        train-rmse:0.198584        valid-rmse:0.212981
[140]        train-rmse:0.194905        valid-rmse:0.209702
[150]        train-rmse:0.19152        valid-rmse:0.206743
[160]        train-rmse:0.190528        valid-rmse:0.206151
[170]        train-rmse:0.184665        valid-rmse:0.201195
[180]        train-rmse:0.178866        valid-rmse:0.195849
[190]        train-rmse:0.172794        valid-rmse:0.190609
[200]        train-rmse:0.1664        valid-rmse:0.18483
[210]        train-rmse:0.161816        valid-rmse:0.180809
[220]        train-rmse:0.160038        valid-rmse:0.17947
[226]        train-rmse:0.157971        valid-rmse:0.17762
Modeling RMSE 0.17762
```

13 Step 9: Evalute on test data

```
In [22]: y_pred = utils.predict_xgb(model, X_test)
         y_pred.head(5)
```

Out[22]:

id	trip_duration
765003	684.261780
765004	519.715454
765005	1334.568848
765006	1080.803589
765007	2010.385010

```
In [23]: mean_squared_error(y_test, y_pred['trip_duration'])**0.5
```

Out[23]: 172.11322567227265

14 Additional Analysis

Let's look at how important each feature was for the model.

```
In [24]: feature_names = X_train.columns.values
         ft_importances = utils.feature_importances(model, feature_names)
         ft_importances[:20]
```

Out[24]:

	feature_name	importance
8	dropoff_latitude	3910.0
3	pickup_longitude	3665.0
6	pickup_latitude	3232.0
56	MINUTE(pickup_datetime)	3084.0
2	dropoff_longitude	2960.0
30	MINUTE(dropoff_datetime)	2896.0
7	trip_distance	2732.0
72	HOUR(dropoff_datetime)	1821.0

45	HOUR(pickup_datetime)	1663.0
0	payment_type	750.0
71	WEEKDAY(dropoff_datetime)	724.0
31	DAY(dropoff_datetime)	585.0
32	WEEKDAY(pickup_datetime)	566.0
75	WEEK(dropoff_datetime)	525.0
67	DAY(pickup_datetime)	507.0
64	WEEK(pickup_datetime)	289.0
77	IS_WEEKEND(pickup_datetime)	243.0
9	MONTH(pickup_datetime)	151.0
74	IS_WEEKEND(dropoff_datetime)	150.0
69	passenger_cnt.STD(trips.trip_duration)	111.0