Predicting NYC Cab Trip Duration

Souvik Roy

Loading the data

- Import pandas and load data

http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml . Subset at https://www.kaggle.com/c/nyc-taxi-trip-duration/data

```
taxi_data=pd.read_csv('trainingDataWithUniqueID.csv')
```

```
taxi_data.head(3)
```

	UniqueID	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	s
0	0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155	40.767937	-73.964630	40.765602	
1	1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415	40.738564	-73.999481	40.731152	
2	2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027	40.763939	-74.005333	40.710087	

Calculating pick up and drop-off hour

- Define functions

```
def calculate_pickup_hour(x):
    pickup_time=datetime.strptime(x[2], '%Y-%m-%d %H:%M:%S')
    return pickup_time.hour

def calculate_dropoff_hour(x):
    dropoff_time=datetime.strptime(x[2], '%Y-%m-%d %H:%M:%S')
    return dropoff_time.hour
```

- Apply functions

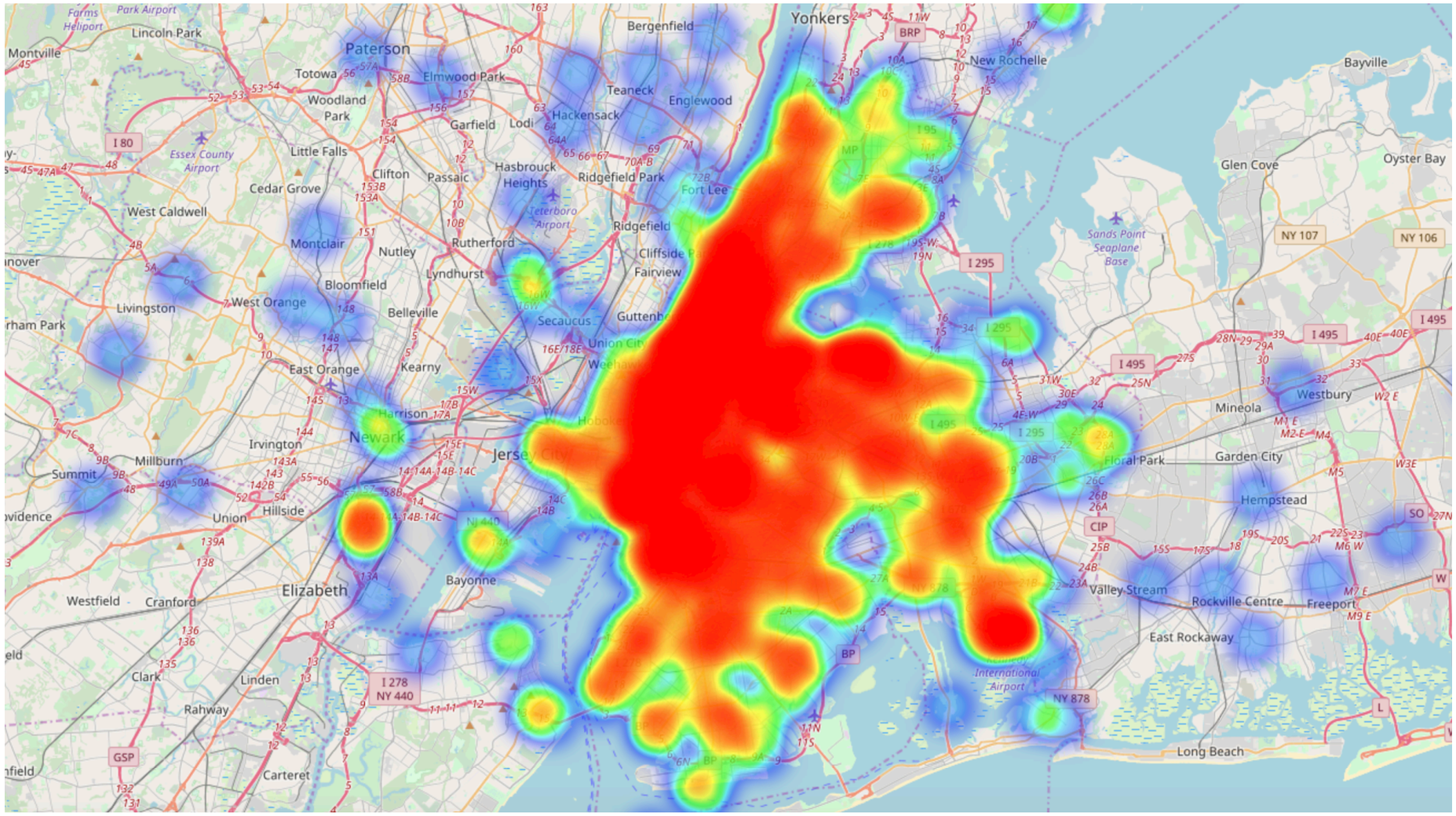
```
taxi_data['pickup_hour']=taxi_data.apply(calculate_pickup_hour,axis=1)
taxi_data['dropoff_hour']=taxi_data.apply(calculate_dropoff_hour,axis=1)
```

- Check data columns

```
list(taxi_data)
```

['uniqueid', 'id', 'vendor_id', 'pickup_datetime', 'dropoff_datetime', 'passenger_count', 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag', 'trip_duration', 'pickup_hour', 'dropoff_hour']

Pickup location heat map



Making sense of Location Data

- Reverse Geolocator

In [13]:
```
from geopy.geocoders import Nominatim
geolocator = Nominatim()
location = geolocator.reverse("40.735806, -73.985437")
print location.raw
```

```
{u'display_name': u'157, East 18th Street, Gramercy, Manhattan, Manhattan Community Board 6, New York County, NYC, Ne
w York, 10003, United States of America', u'place_id': u'29350552', u'lon': u'-73.9855157', u'boundingbox': [u'40.735
8788', u'40.7360788', u'-73.9856157', u'-73.9854157'], u'osm_type': u'node', u'licence': u'Data \xa9 OpenStreetMap co
ntributors, ODbL 1.0. https://osm.org/copyright', u'osm_id': u'2704034185', u'lat': u'40.7359788', u'address': {u'cit
y': u'NYC', u'house_number': u'157', u'country': u'United States of America', u'county': u'New York County', u'subur
b': u'Gramercy', u'state': u'New York', u'city_district': u'Manhattan', u'postcode': u'10003', u'country_code': u'u
s', u'road': u'East 18th Street'}}
```

In [15]:
```
x=location.raw
x['address']['county']
```

Out[15]:
```
u'Manhattan'
```

In [27]:
```
def extract_borough(x):
    lat_long_string=str(x['pickup_latitude']) +', ' +str(x['pickup_longitude'])
    location = geolocator.reverse(lat_long_string)
    y=location.raw

    if 'city_district' in y['address']:
        print y['address']['city_district']
        return y['address']['city_district']
    else:
        return 'NA'
```

Too slow due to repeated calls to the API!

Making sense of Location Data

- Export location data and import in Carto www.carto.com

```
taxi_data[['UniqueID','pickup_latitude','pickup_longitude']].to_csv('taxi_data_lat_long.csv',index=False)
```

taxi_data_lat_long

PRIVATE Updated 5 days ago

+ ADD ROW + ADD COLUMN EXPORT

cartodb_id number	the_geom geometry	uniqueid number	longitude number	latitude number	_feature_count number
1	-73.98215485, 40.7679...	0	-73.9821548461914	40.767936706543	1
2	-73.98041534, 40.7385...	1	-73.9804153442383	40.7385635375977	1
3	-73.97902679, 40.7639...	2	-73.9790267944336	40.7639389038086	1
4	-74.01004028, 40.7199...	3	-74.0100402832031	40.719970703125	1
5	-73.97305298, 40.7932...	4	-73.9730529785156	40.7932090759277	1
6	-73.98285675, 40.7421...	5	-73.9828567504883	40.7421951293945	1
7	-73.96901703, 40.7578...	6	-73.9690170288086	40.7578392028809	1
8	-73.96927643, 40.7977...	7	-73.9692764282227	40.797779083252	1
9	-73.9994812, 40.73839...	8	-73.9994812011719	40.7383995056152	1
10	-73.98104858, 40.7443...	9	-73.9810485839844	40.7443389892578	1
11	-73.98265076, 40.7638...	10	-73.9826507568359	40.7638397216797	1

1 TO 40

METADATA SQL PREVIEW CREATE MAP

- Download NYC Shape Files from zillow.com

Making sense of Location Data

- Write SQL Query to map location into a neighborhood using the Zillow ShapeFiles. Gives us City, County, Neighborhood info

zillowneighborhoods_ny SQL

PRIVATE Updated 25 minutes ago

EXPORT

cartodb_id number	the_geom geometry	uniqueid number	latitude number	longitude number	_feature_count number	county string	city string
47182	-73.85697174, 40.5743...	90095	40.5743255615234	-73.8569717407227	1	Queens	New York
47535	-73.85538483, 40.5750...	145844	40.575065612793	-73.8553848266601	1	Queens	New York
48320	-73.85430145, 40.5748...	121294	40.574836730957	-73.8543014526367	1	Queens	New York
51632	-73.85482788, 40.5741...	1311246	40.574146270752	-73.8548278808594	1	Queens	New York
51836	-73.8578949, 40.5725975	731669	40.5725975036621	-73.8578948974609	1	Queens	New York

1 TO 40

```
SELECT a.*,
       b.county,b.city,b.name
FROM taxi_data_dropoff_lat_long as a,
   zillowneighborhoods_ny as b
WHERE ST_Within(a.the_geom,b.the_geom)
```

CMD + S to apply your query. CTRL + Space to autocomplete.

METADATA SQL PREVIEW CLEAR APPLY CREATE MAP

Making sense of Location Data

- Export the new location file and import it using Pandas

```
In [93]: taxi_data_with_info=pd.read_csv('taxi_data_lat_long_with_info.csv')
```

- Inner Join it with the original dataset on UniqueID

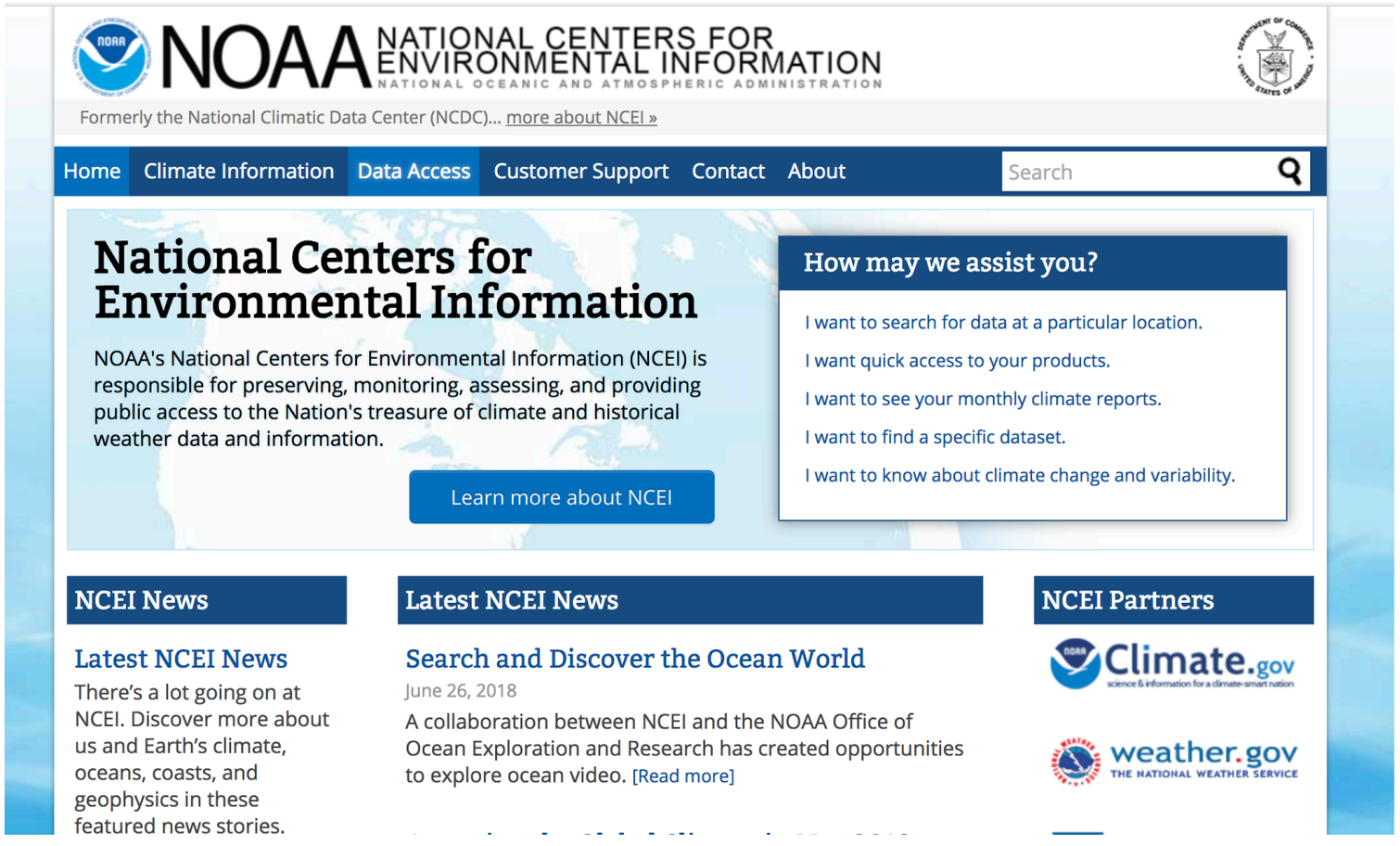
```
taxi_data_full=pd.merge(taxi_data_with_info, taxi_data, how='inner')
```

- Repeat for Drop-off Location Data

uniqueid	longitude	latitude	_feature_count	county	city	name
1041801	-73.849762	40.579079	1	Queens	New York	Belle Harbor
1268052	-73.787239	40.594406	1	Queens	New York	Far Rockaway
901745	-73.786148	40.593884	1	Queens	New York	Far Rockaway

Adding Weather Data

- Request Weather Data from 01/01/16-06/30/16 at ncdc.noaa.gov/



- Create a request and download daily summary data

Adding Weather Data

- Import dataset using Pandas

```
In [125]: weather=pd.read_csv('WeatherData.csv')

In [128]: list(weather)

Out[128]: ['STATION', 'NAME', 'DATE', 'PRCP', 'SNOW', 'SNWD', 'TAVG', 'TMAX', 'TMIN', 'TSUN', 'WT01', 'WT02', 'WT03', 'WT04', 'WT06', 'WT08']

In [157]: weather_refined=weather[['PRCP','SNOW','SNWD','TMAX','TMIN','DATE']]

In [132]: weather_refined
```

Out[132]:

	PRCP	SNOW	SNWD	TMAX	TMIN	DATE
0	0.0	0.0	0.0	5.6	1.1	2016-01-01
1	0.0	0.0	0.0	4.4	0.0	2016-01-02
2	0.0	0.0	0.0	7.2	1.7	2016-01-03
3	0.0	0.0	0.0	2.2	-9.9	2016-01-04
4	0.0	0.0	0.0	-1.6	-11.6	2016-01-05

- Join with original dataset on Pickup Date

```
In [138]: taxi_data_full['pickup_date']=taxi_data_full['pickup_datetime'].apply(lambda x: x.split()[0])

In [139]: taxi_data_dropoff_with_info=pd.read_csv('taxi_data_dropoff_lat_long_with_info.csv')

In [142]: taxi_data_with_all_info=pd.merge(taxi_data_full,taxi_data_dropoff_with_info,how='inner')
```

Adding Additional Features

- Adding Aggregate features

```
trip_time_by_county=taxi_data_with_all_info.groupby(['pickup_county','dropoff_county'],as_index=False).agg({'trip_durat
```

```
trip_time_by_neighborhood=taxi_data_with_all_info.groupby(['pickup_neighborhood','dropoff_neighborhood'],as_index=False
```

```
trip_time_by_city=taxi_data_with_all_info.groupby(['pickup_city','dropoff_city'],as_index=False).agg({'trip_duration':'
```

- Merging it with original dataset

```
taxi_data_with_all_info=pd.merge(taxi_data_with_all_info,trip_time_by_county,on=['pickup_county','dropoff_county'])
```

```
taxi_data_with_all_info=pd.merge(taxi_data_with_all_info,trip_time_by_neighborhood,on=['pickup_neighborhood','dropoff_n
```

```
taxi_data_with_all_info=pd.merge(taxi_data_with_all_info,trip_time_by_city,on=['pickup_city','dropoff_city'])
```

Choosing Features

- Choose features

```
list(taxi_data_with_all_info)
```

['PRCP', 'SNOW', 'SNWD', 'TMAX', 'TMIN', 'pickup_date', 'uniqueid', 'pickup_county', 'pickup_city', 'pickup_neighborhood', 'pickup_datetime', 'dropoff_datetime', 'passenger_count', 'pickup_latitude', 'pickup_longitude', 'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag', 'trip_duration', 'pickup_hour', 'dropoff_hour', 'dropoff_city', 'dropoff_county', 'dropoff_neighborhood', 'trip_duration_by_county', 'trip_duration_by_neighborhood', 'trip_duration_by_city']

```
training_features=['PRCP', 'SNOW', 'SNWD', 'TMAX', 'TMIN', 'passenger_count', 'pickup_hour', 'trip_duration_by_county',
```

```
training_features
```

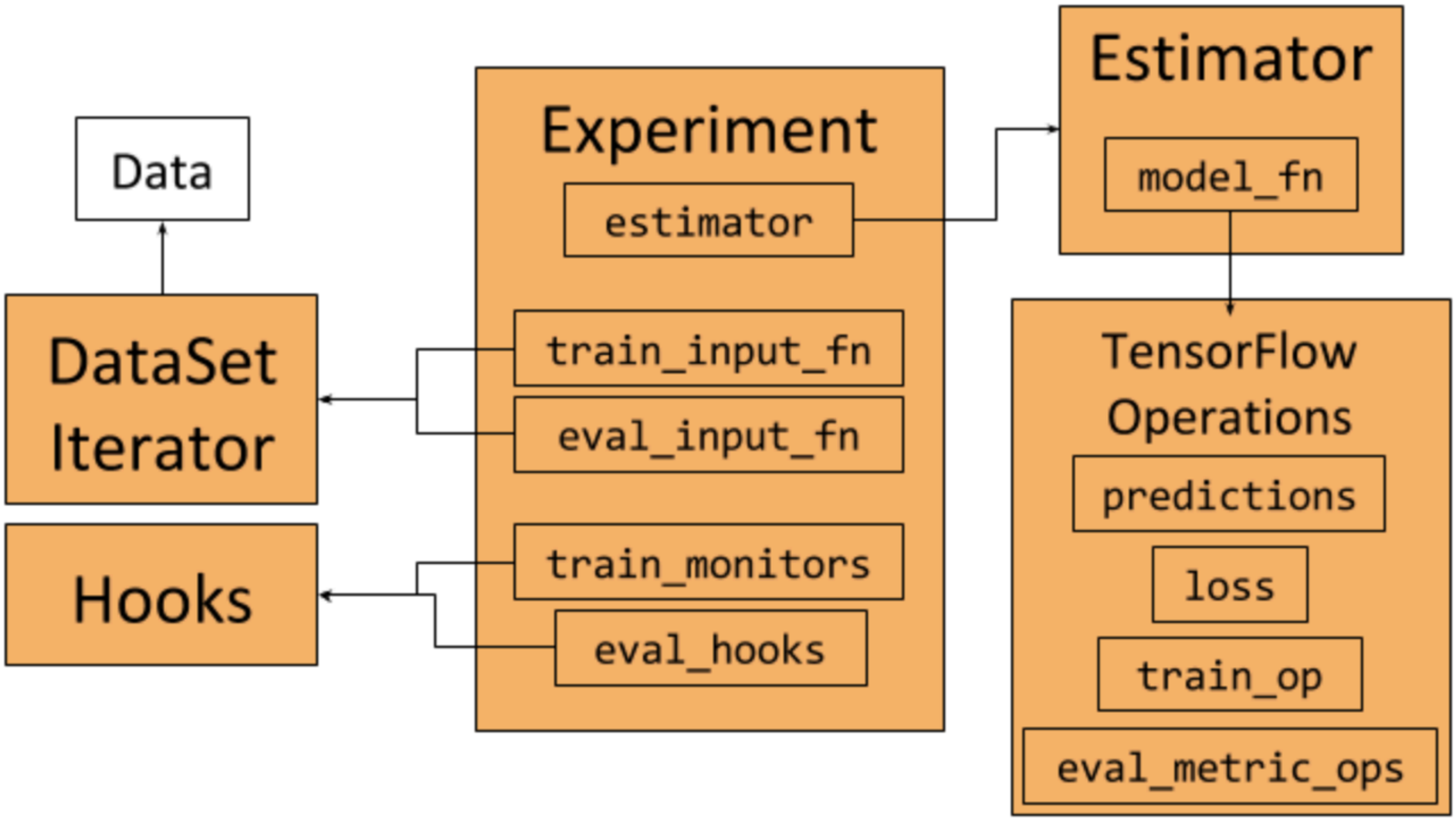
['PRCP', 'SNOW', 'SNWD', 'TMAX', 'TMIN', 'passenger_count', 'pickup_hour', 'trip_duration_by_county', 'trip_duration', 'trip_duration_by_neighborhood', 'trip_duration_by_city']

- Export training and test data for TensorFlow

```
taxi_data_final_set=taxi_data_with_all_info[training_features]
```

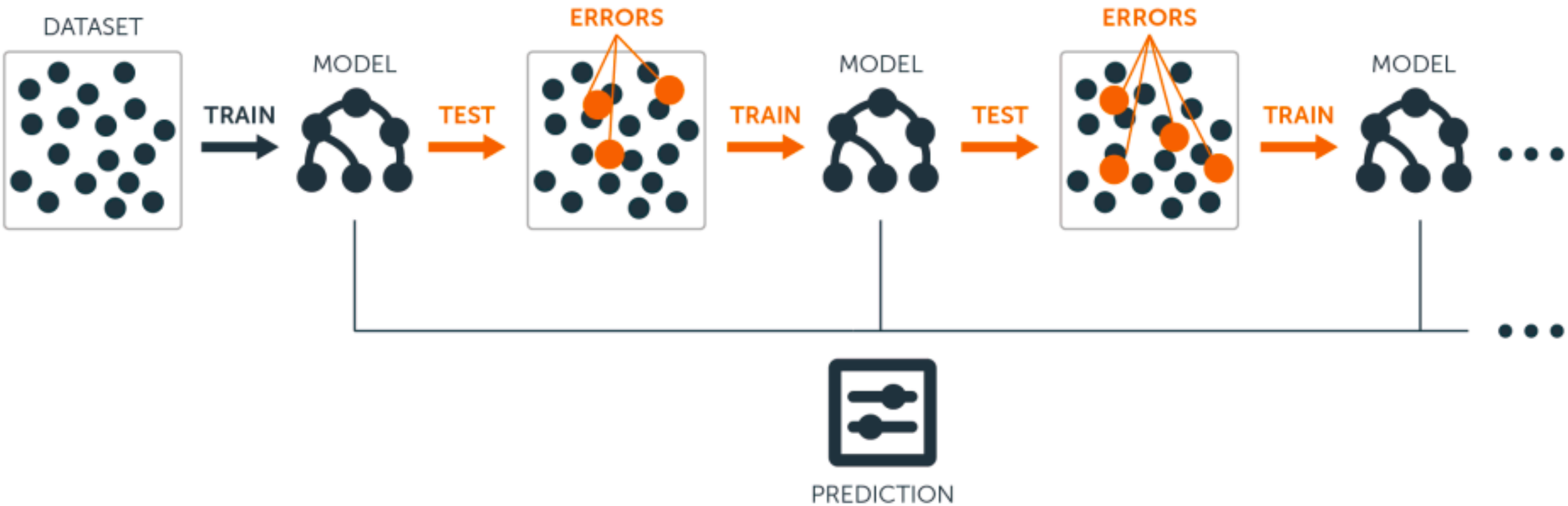
```
taxi_data_final_set.to_csv('taxi_data_final_set.csv',index=False)
```

TensorFlow Code Framework



https://medium.com/onfido-tech/higher-level-apis-in-tensorflow-67bfb602e6c0

Boosted Trees



Model : Boosted trees with L2 regularization

https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/

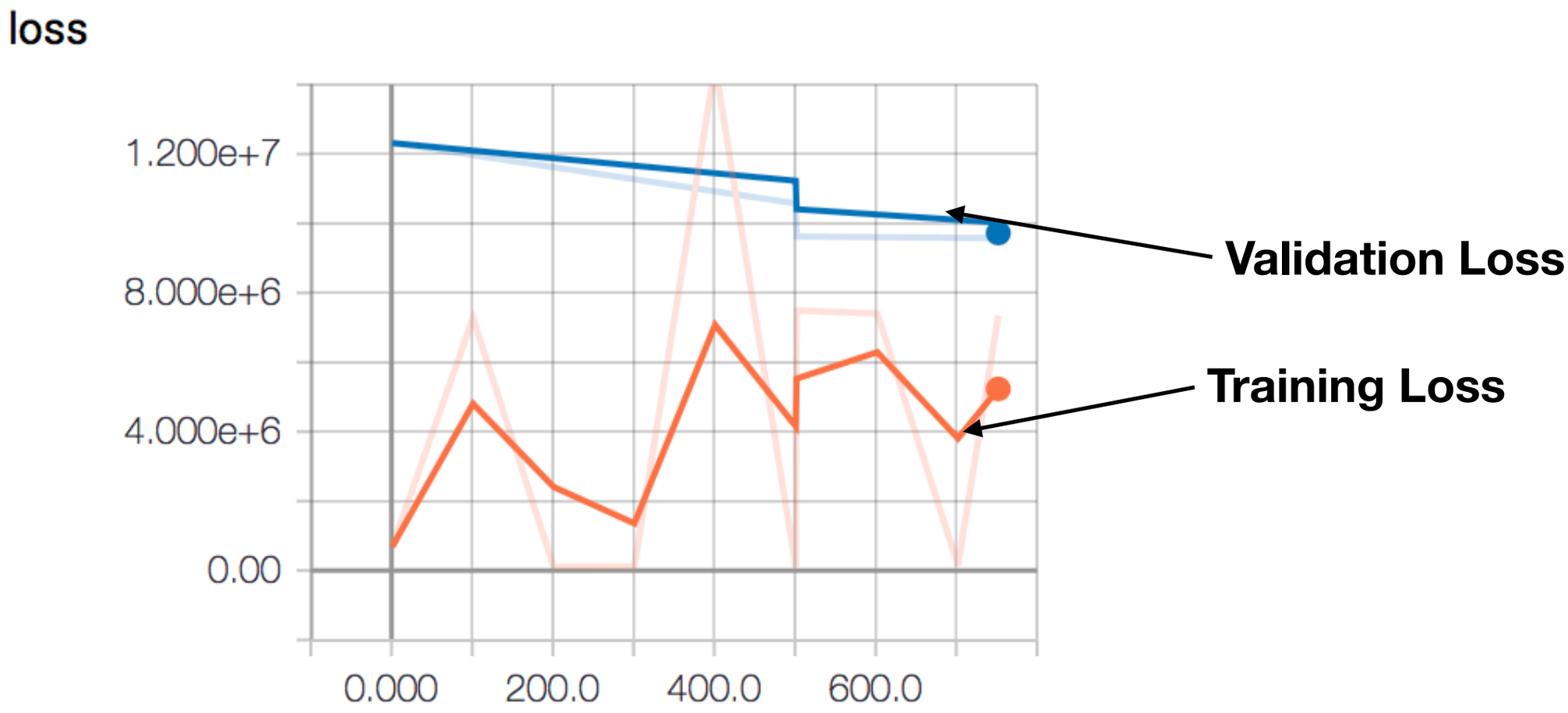
TensorFlow Code

```
def _read_data():

  full_dataset=pd.read_csv('taxi_data_final_set.csv')
  training_features=['PRCP', 'SNOW', 'SNWD', 'TMAX', 'TMIN', 'passenger_co
  target=['trip_duration']
  x_train, x_test, y_train, y_test = train_test_split(full_dataset[trainin
  return (x_train, y_train), (x_test, y_test)
```

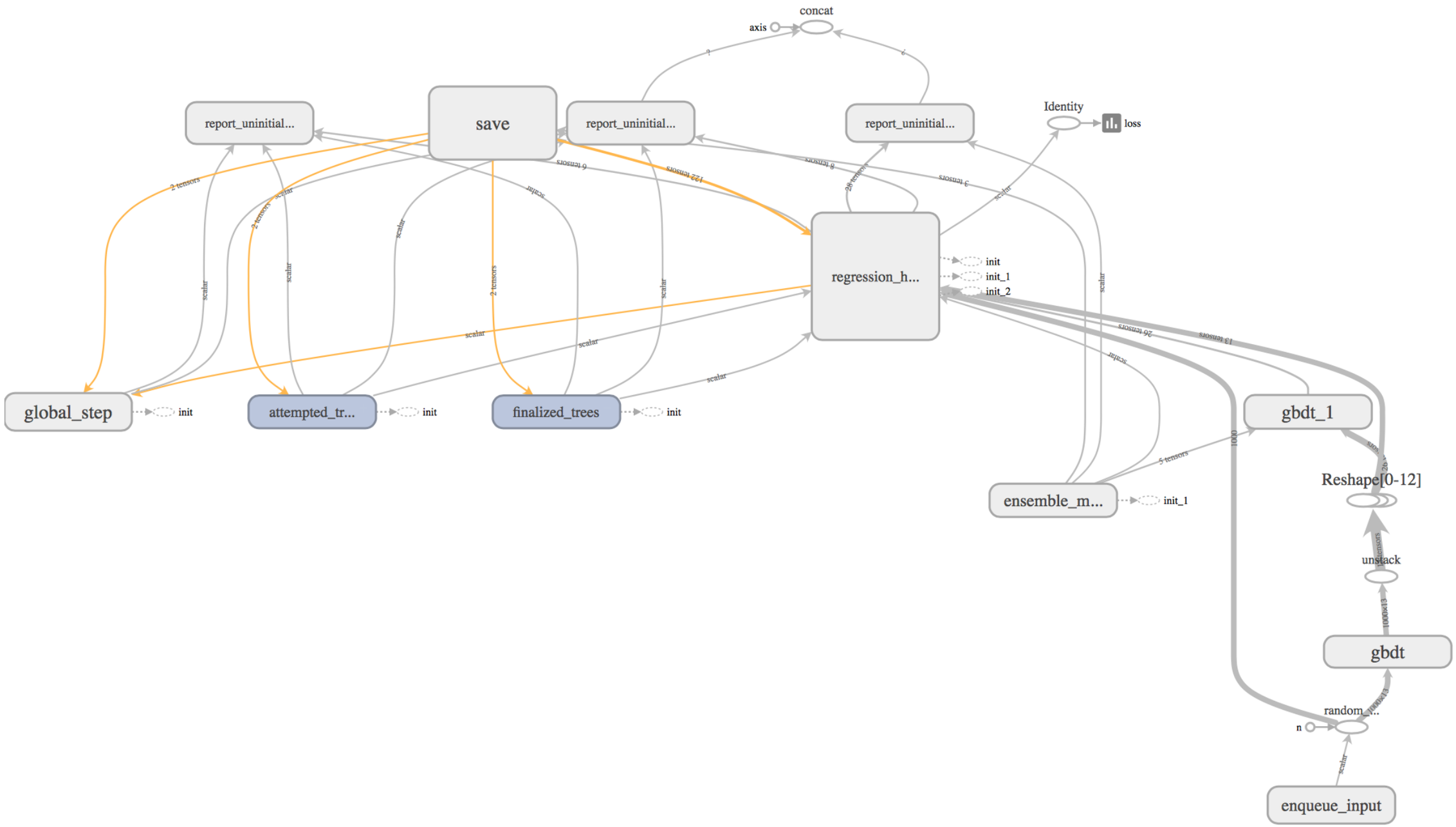
```
# Main config - creates a TF Boosted Trees Estimator based on flags.
def _get_tfbt(output_dir, feature_cols):
  """Configures TF Boosted Trees estimator based on flags."""
  learner_config = learner_pb2.LearnerConfig()
  learner_config.learning_rate_tuner.fixed.learning_rate = FLAGS.learning_rate
  learner_config.regularization.l1 = 0.0
  learner_config.regularization.l2 = FLAGS.l2
  learner_config.constraints.max_tree_depth = FLAGS.depth

  run_config = tf.contrib.learn.RunConfig(save_checkpoints_secs=300)

  # Create a TF Boosted trees regression estimator.
  estimator = GradientBoostedDecisionTreeRegressor(
      learner_config=learner_config,
      examples_per_layer=FLAGS.batch_size,
      feature_columns=feature_cols,
      label_dimension=1,
      model_dir=output_dir,
      num_trees=FLAGS.num_trees,
      center_bias=False,
      config=run_config)
  return estimator
```

```
def _make_experiment_fn(output_dir):
  """Creates experiment for gradient boosted decision trees."""

  (x_train, y_train), (x_test,
                       y_test) = _read_data()

  train_input_fn = tf.estimator.inputs.numpy_input_fn(
      x={"x": x_train},
      y=y_train,
      batch_size=FLAGS.batch_size,
      num_epochs=None,
      shuffle=True)
  eval_input_fn = tf.estimator.inputs.numpy_input_fn(
      x={"x": x_test}, y=y_test, num_epochs=1, shuffle=False)

  feature_columns = [
      feature_column.real_valued_column("x", dimension=_NUM_FEATURES)
  ]
  feature_spec = tf.contrib.layers.create_feature_spec_for_parsing(
      feature_columns)
  serving_input_fn = tf.contrib.learn.utils.build_parsing_serving_input_fn(
      feature_spec)

  export_strategy = custom_export_strategy.make_custom_export_strategy(
      "exports",
      convert_fn=_convert_fn,
      feature_columns=feature_columns,
      export_input_fn=serving_input_fn)
  return tf.contrib.learn.Experiment(
      estimator=_get_tfbt(output_dir, feature_columns),
      train_input_fn=train_input_fn,
      eval_input_fn=eval_input_fn,
      train_steps=None,
      eval_steps=FLAGS.num_eval_steps,
      eval_metrics=None,
      export_strategies=[export_strategy])
```

Loss Plot


loss
1.200e+7
8.000e+6
4.000e+6
0.00
0.000
200.0
400.0
600.0
Validation Loss
Training Loss

TensorFlow Graph



Improvements

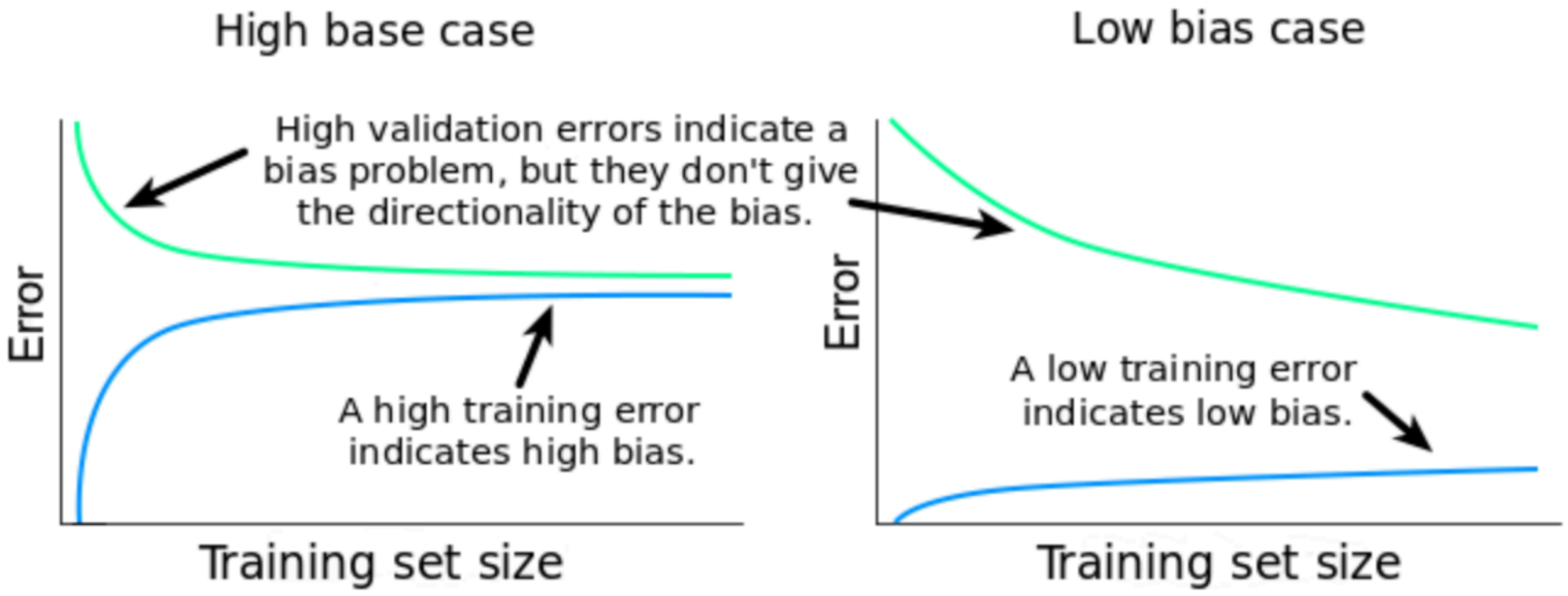
- Adding one-hot vector corresponding to the pickup and drop-off neighborhoods

- Adding more aggregate features corresponding to the hourly pickup and drop-off neighborhoods trip durations

- Trying other regressors. For e.g., a hybrid of Boosted Trees and DNN Based Regressors

Backup

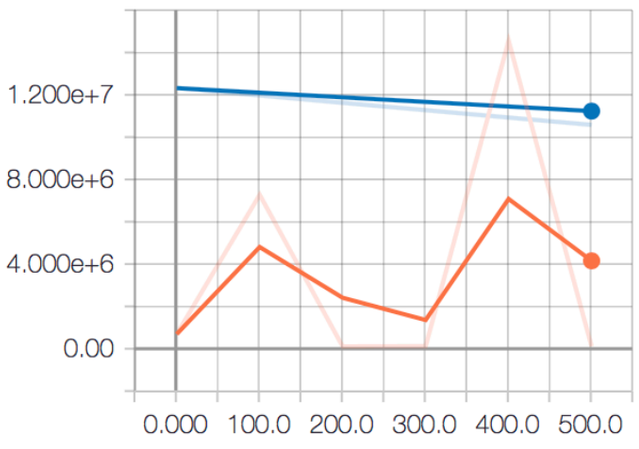
Exploring Bias-Variance TradeOff



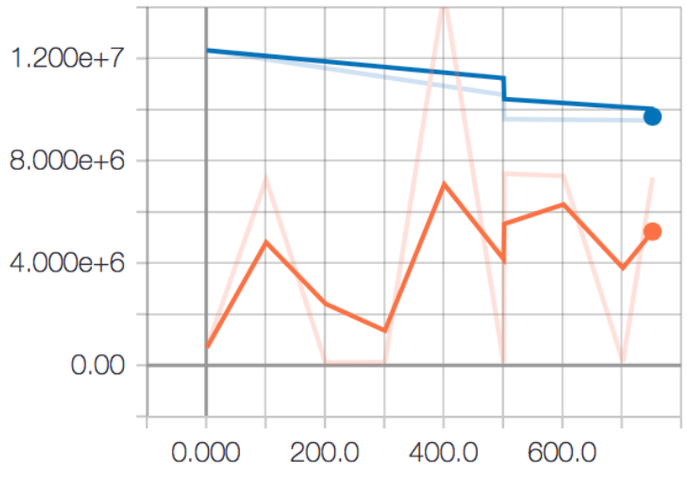
https://www.dataquest.io/blog/learning-curves-machine-learning/

Loss Plot

loss



loss



loss