Before you turn in the homework, make sure everything runs as expected. To do so, select **Kernel** →**Restart & Run All** in the toolbar above. Remember to submit both on **DataHub** and **Gradescope**.

Please fill in your name and include a list of your collaborators below.

In [1]:
```
NAME = "Benjamin Liu"
COLLABORATORS = "Victor Ding"
```

Project 2: NYC Taxi Rides

Extras

Put all of your extra work in here. Feel free to save figures to use when completing Part 4.

In [2]:
```
import os
import pandas as pd
import numpy as np
import sklearn.linear_model as lm
from sklearn.model_selection import cross_val_score, train_test_split, GridSearc
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
from sqlalchemy import create_engine
```

In [3]:
```
# Run this cell to load the data.
data_file = Path("./", "cleaned_data.hdf")
train_df = pd.read_hdf(data_file, "train")
val_df = pd.read_hdf(data_file, "val")
test_df = pd.read_csv("./proj2_test_data.csv")
test_df['tpep_pickup_datetime'] = pd.to_datetime(test_df['tpep_pickup_datetime']
```

In [4]:
```
# get the summary of train df
train_df.describe()
```

Out[4]:

	record_id	VendorID	passenger_count	trip_distance	pickup_longitude	pickup_latitu
count	1.772400e+04	17724.000000	17724.000000	17724.000000	17724.000000	17724.0000
mean	5.320997e+06	1.535150	1.677104	2.791220	-73.973560	40.7509
std	3.158004e+06	0.498777	1.324193	3.407549	0.037279	0.0274
min	6.000000e+02	1.000000	1.000000	0.000000	-74.018150	40.6316
25%	2.604200e+06	1.000000	1.000000	1.000000	-73.991783	40.7375
50%	5.208950e+06	2.000000	1.000000	1.620000	-73.981541	40.7543
75%	8.215850e+06	2.000000	2.000000	3.000000	-73.966925	40.7684
max	1.090610e+07	2.000000	6.000000	35.430000	-73.775398	40.8471

In [5]:
```
# display the summary of test df
test_df.describe()
```

Out[5]:

	record_id	VendorID	passenger_count	trip_distance	pickup_longitude	pickup_latitu
count	1.377400e+04	13774.000000	13774.000000	13774.000000	13774.000000	13774.0000
mean	3.465950e+07	1.536082	1.663642	2.954688	-72.953619	40.1879
std	2.015133e+07	0.498714	1.311739	3.704427	8.628431	4.7531
min	1.000000e+04	1.000000	0.000000	0.000000	-77.039436	0.0000
25%	1.719975e+07	1.000000	1.000000	1.000000	-73.992058	40.7351
50%	3.457400e+07	2.000000	1.000000	1.700000	-73.981846	40.7524
75%	5.216875e+07	2.000000	2.000000	3.157500	-73.967119	40.7672
max	6.940400e+07	2.000000	6.000000	104.800000	0.000000	40.8682

In [ ]:

In [6]:
```
### Try: remove Jan 23 data, 17724 - 17603 is removed
print(train_df.shape)
train_remove = train_df[train_df['tpep_pickup_datetime'].dt.day != 23]
print(train_remove.shape)
```

```
(17724, 21)
(17603, 21)
```

Typesetting math: 100%

In [7]:
```
### Try: replace outliers with the median in training data
train_df_copy = train_remove.copy()
for i in range(len(train_df_copy)):
    if train_df_copy.iloc[i, 19] < 0:
        train_df_copy.iloc[i, 19] = 11.300000
    if train_df_copy.iloc[i, 18] < 0:
        train_df_copy.iloc[i, 18] = 0.3
    if train_df_copy.iloc[i, 15] < 0:
        train_df_copy.iloc[i, 15] = 0.5
    if train_df_copy.iloc[i, 14] < 0:
        train_df_copy.iloc[i, 14] = 0.0
    if train_df_copy.iloc[i, 13] < 0:
        train_df_copy.iloc[i, 13] = 9.0
train_df_copy.describe()
```

Out[7]:

	record_id	VendorID	passenger_count	trip_distance	pickup_longitude	pickup_latitu
count	1.760300e+04	17603.000000	17603.000000	17603.000000	17603.000000	17603.0000
mean	5.294103e+06	1.534682	1.676135	2.790611	-73.973551	40.7510
std	3.152077e+06	0.498810	1.323330	3.406663	0.037278	0.0273
min	6.000000e+02	1.000000	1.000000	0.000000	-74.018150	40.6316
25%	2.585650e+06	1.000000	1.000000	1.000000	-73.991768	40.7375
50%	5.175400e+06	2.000000	1.000000	1.610000	-73.981529	40.7543
75%	8.155700e+06	2.000000	2.000000	3.000000	-73.966923	40.7684
max	1.090610e+07	2.000000	6.000000	35.430000	-73.775398	40.8471

Typesetting math: 100%

In [8]:

```
### Copy from part 2, data pre-processing
def haversine(lat1, lng1, lat2, lng2):
    """
    Compute haversine distance

    The haversine formula determines the great-circle distance between two point
    on a sphere given their longitudes and latitudes. Important in navigation, i
    is a special case of a more general formula in spherical trigonometry,
    the law of haversines, that relates the sides and angles of spherical triang
    """
    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
    average_earth_radius = 6371
    lat = lat2 - lat1
    lng = lng2 - lng1
    d = np.sin(lat * 0.5) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(lng * 0.5)
    h = 2 * average_earth_radius * np.arcsin(np.sqrt(d))
    return h

def manhattan_distance(lat1, lng1, lat2, lng2):
    """
    Computes Manhattan distance

    The name alludes to the grid layout of most streets on the island of Manhatta
    which causes the shortest path a car could take between two intersections in
    to have length equal to the intersections' distance in taxicab geometry.
    """
    a = haversine(lat1, lng1, lat1, lng2)
    b = haversine(lat1, lng1, lat2, lng1)
    return a + b

def bearing(lat1, lng1, lat2, lng2):
    """
    Compute the bearing, or angle, from (lat1, lng1) to (lat2, lng2).
    A bearing of 0 refers to a NORTH orientation.
    """
    lng_delta_rad = np.radians(lng2 - lng1)
    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
    y = np.sin(lng_delta_rad) * np.cos(lat2)
    x = np.cos(lat1) * np.sin(lat2) - np.sin(lat1) * np.cos(lat2) * np.cos(lng_d
    return np.degrees(np.arctan2(y, x))

def add_distance_columns(df):
    df.loc[:, 'manhattan'] = manhattan_distance(lat1=df['pickup_latitude'],
                                                lng1=df['pickup_longitude'],
                                                lat2=df['dropoff_latitude'],
                                                lng2=df['dropoff_longitude'])

    df.loc[:, 'bearing'] = bearing(lat1=df['pickup_latitude'],
                                   lng1=df['pickup_longitude'],
                                   lat2=df['dropoff_latitude'],
                                   lng2=df['dropoff_longitude'])
    df.loc[:, 'haversine'] = haversine(lat1=df['pickup_latitude'],
                                   lng1=df['pickup_longitude'],
                                   lat2=df['dropoff_latitude'],
                                   lng2=df['dropoff_longitude'])
```

Typesetting math: 100%

```
    return df

def add_time_columns(df):
    """
    Add temporal features to df
    """
    df.is_copy = False
    df.loc[:, 'month'] = df['tpep_pickup_datetime'].dt.month
    df.loc[:, 'week_of_year'] = df['tpep_pickup_datetime'].dt.weekofyear
    df.loc[:, 'day_of_month'] = df['tpep_pickup_datetime'].dt.day
    df.loc[:, 'day_of_week'] = df['tpep_pickup_datetime'].dt.dayofweek
    df.loc[:, 'hour'] = df['tpep_pickup_datetime'].dt.hour
    df.loc[:, 'week_hour'] = df['tpep_pickup_datetime'].dt.weekday * 24 + df['hou

    # No real need to return here, but we harmonize with remove_outliers for late
    return df

def select_columns(data, *columns):
    return data.loc[:, columns]
```

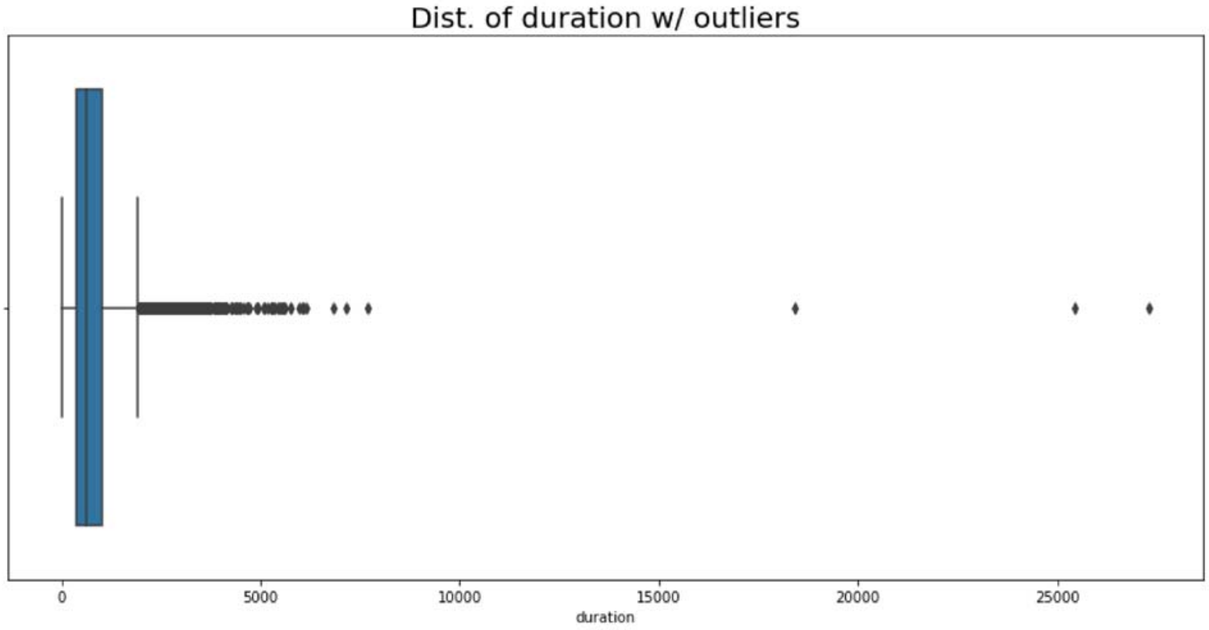
In [9]:

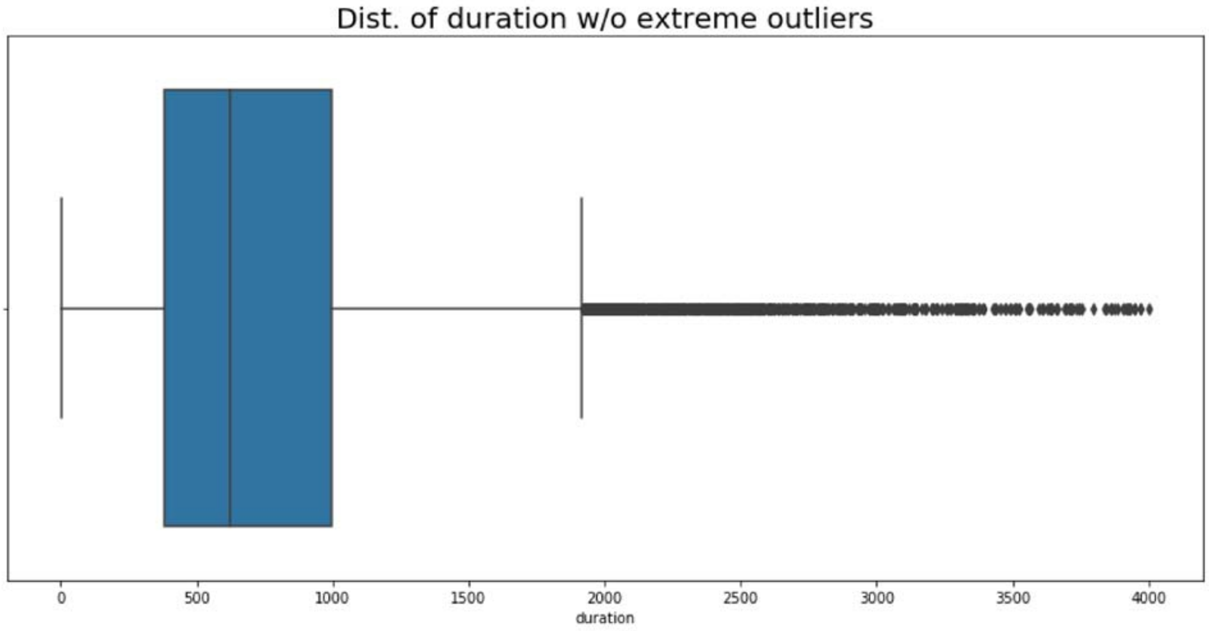
```
### Try: LASSO and Ridge Regression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
def process_data_fm(data, test=False):

    X = (
        data
        # Transform data
        .pipe(add_time_columns)
        .pipe(add_distance_columns)
        .pipe(select_columns,
        'pickup_longitude',
        'pickup_latitude',
        'dropoff_longitude',
        'dropoff_latitude',
        'manhattan',
        'haversine',
        'hour',
        'trip_distance',
        'day_of_week',
        'total_amount',
        'tolls_amount',
        'tip_amount',
        'extra',
        'fare_amount'
        )
    )
    if test:
        y = None
    else:
        y = data['duration']
    return X, y

def mae(actual, predicted):
    """
    mean abs error
    """
    return np.mean(np.abs(actual - predicted))
```

Typesetting math: 100%

In [10]:

```
# draw a plot to view the outliers in the duration of training data
plt.figure(figsize=(15, 7))
sns.boxplot(train_df['duration'])
plt.title('Dist. of duration w/ outliers', fontsize=20)
plt.show()
plt.figure(figsize=(15, 7))
eva_train = train_df.loc[train_df['duration'] <= 4000]
plt.title('Dist. of duration w/o extreme outliers', fontsize=20)
sns.boxplot(eva_train['duration'])
plt.show()
```





Typesetting math: 100%

In [11]:

```
train_df_clean = train_df_copy[(train_df_copy['duration'] <= 4000) & (train_df_c
X_train_new, y_train_new = process_data_fm(train_df_clean)
X_val_new, y_val_new = process_data_fm(val_df)
```

```
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:438
8: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a fu
ture version.
  object.__getattribute__(self, name)
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:438
9: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a fu
ture version.
  return object.__setattr__(self, name, value)
```

In [12]:

```
### Try: compare Lasso and Ridge
model = Lasso(alpha=1)
model.fit(X_train_new, y_train_new)
y_train_pred_new = model.predict(X_train_new)
y_val_pred_new = model.predict(X_val_new)
print(mae(y_train_pred_new, y_train_new))
print(mae(y_val_pred_new, y_val_new))
```

```
91.6189609903
110.372183238
```

```
/srv/conda/envs/data100/lib/python3.6/site-packages/sklearn/linear_model/coordi
nate_descent.py:491: ConvergenceWarning: Objective did not converge. You might
want to increase the number of iterations. Fitting data with very small alpha m
ay cause precision problems.
  ConvergenceWarning)
```

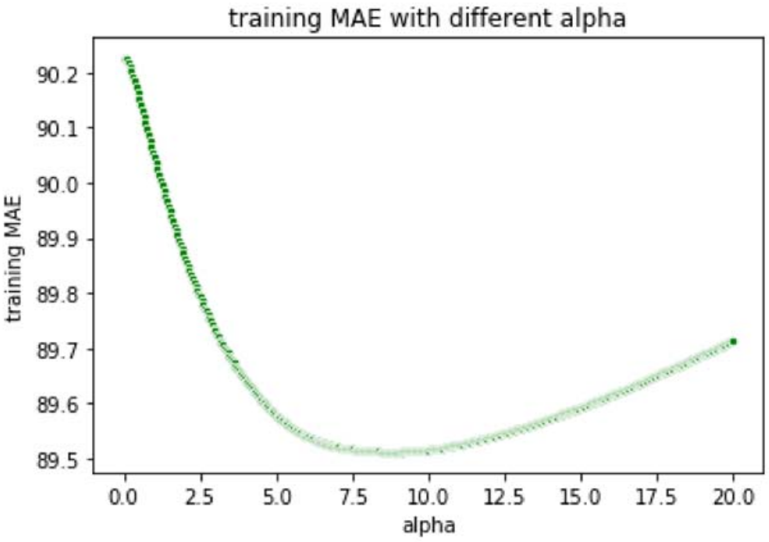
In [13]:

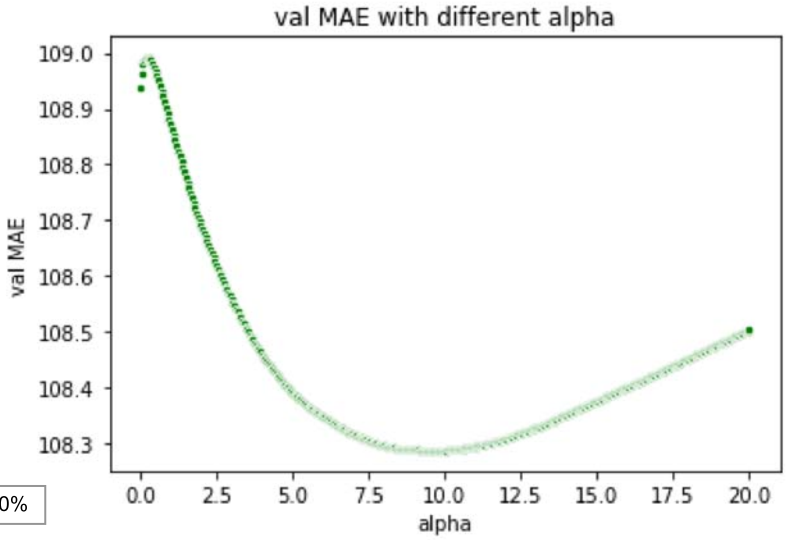
```
### Try: compare Lasso and Ridge
model = Ridge(alpha=1)
model.fit(X_train_new, y_train_new)
y_train_pred_new = model.predict(X_train_new)
y_val_pred_new = model.predict(X_val_new)
print(mae(y_train_pred_new, y_train_new))
print(mae(y_val_pred_new, y_val_new))
```

```
90.0460977754
108.872421536
```

In [14]:

```
# Find the best hyperparam regualrization alpha for Ridge Regression
penalty = np.linspace(0, 20, 400)
mae_train = []
mae_val = []
for i in penalty:
    model = Ridge(alpha=i)
    model.fit(X_train_new, y_train_new)
    y_train_pred = model.predict(X_train_new)
    y_val_pred = model.predict(X_val_new)
    mae_train.append(mae(y_train_pred, y_train_new))
    mae_val.append(mae(y_val_pred, y_val_new))

sns.scatterplot(x=penalty, y=mae_train, s=20, color='g')
plt.xlabel('alpha')
plt.ylabel('training MAE')
plt.title('training MAE with different alpha')
plt.show()
sns.scatterplot(x=penalty, y=mae_val, s=20, color='g')
plt.xlabel('alpha')
plt.ylabel('val MAE')
plt.title('val MAE with different alpha')
plt.show()
```

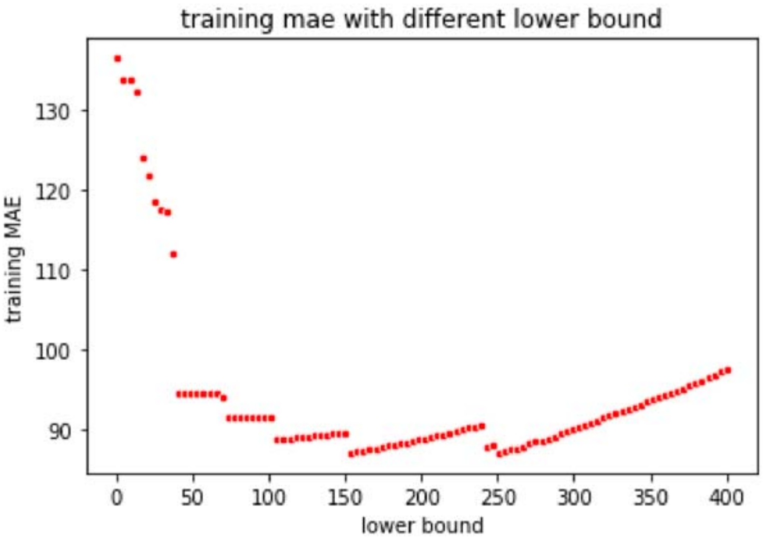




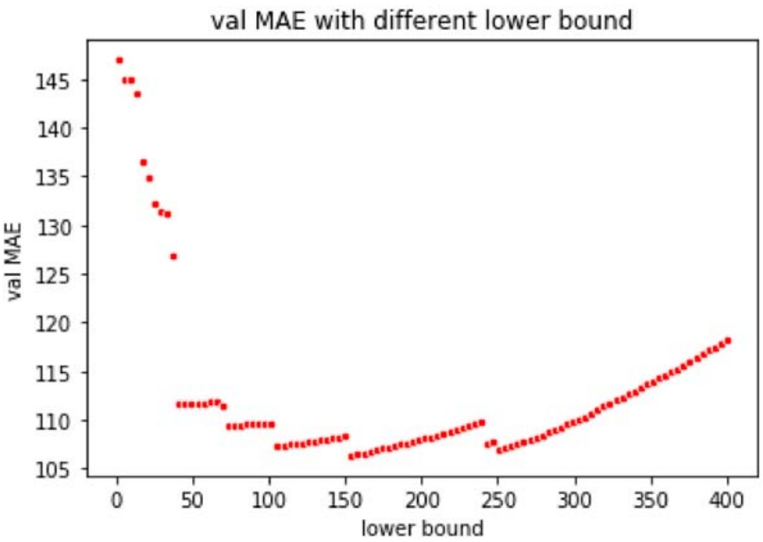
Typesetting math: 100%

In [15]:
```
### Remove outliers in training set
lower = np.linspace(1, 400, 100)
mae_train_1 = []
mae_val_1 = []
for i in lower:
    train_df_clean = train_df[(train_df['duration'] <= 4000) & (train_df['durati
    X_train_new, y_train_new = process_data_fm(train_df_clean)
    X_val_new, y_val_new = process_data_fm(val_df)
    model = Ridge(alpha=9.82)
    model.fit(X_train_new, y_train_new)
    y_train_pred_new = model.predict(X_train_new)
    y_val_pred_new = model.predict(X_val_new)
    mae_train_1.append(mae(y_train_pred_new, y_train_new))
    mae_val_1.append(mae(y_val_pred_new, y_val_new))

sns.scatterplot(x=lower, y=mae_train_1, s=20, color='r')
plt.xlabel('lower bound')
plt.ylabel('training MAE')
plt.title('training mae with different lower bound')
plt.show()
sns.scatterplot(x=lower, y=mae_val_1, s=20, color='r')
plt.xlabel('lower bound')
plt.ylabel('val MAE')
plt.title('val MAE with different lower bound')
plt.show()
```

```
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:438
8: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a fu
ture version.
  object.__getattribute__(self, name)
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:438
9: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a fu
ture version.
  return object.__setattr__(self, name, value)
```



Typesetting math: 100%



In [16]:

```
### Find the best solver for Ridge Regression
### auto solver, alpha = 9.82
train_df_clean = train_df_copy[(train_df_copy['duration'] <= 4000) & (train_df_c
X_train_new, y_train_new = process_data_fm(train_df_clean)
X_val_new, y_val_new = process_data_fm(val_df)

grid_params = {'solver':['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag',
'saga']}
model = Ridge(alpha=9.82)
final_model = GridSearchCV(estimator=model, param_grid=grid_params,cv=5)
final_model.fit(X_train_new, y_train_new)

best_param = final_model.best_params_
print(best_param)

y_train_pred_new = final_model.predict(X_train_new)
y_val_pred_new = final_model.predict(X_val_new)
print(mae(y_train_pred_new, y_train_new))
print(mae(y_val_pred_new, y_val_new))
```

```
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:438
8: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a fu
ture version.
  object.__getattribute__(self, name)
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:438
9: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a fu
ture version.
  return object.__setattr__(self, name, value)

{'solver': 'auto'}
89.5152021478
108.286184627
```

Typesetting math: 100%

In [17]:

```
### define a predict function
def predict(model, test_df):
    clean_index = (test_df['pickup_latitude'] <= 40.85) & (test_df['pickup_latitu
                    (test_df['dropoff_latitude'] <= 40.85) & (test_df['dropoff_la
                    (test_df['pickup_longitude'] <= -73.65) & (test_df['pickup_lo
                    (test_df['dropoff_longitude'] <= -73.65) & (test_df['dropoff_

    dirty_index = - clean_index
    if sum(dirty_index) == 0:
        return model.predict(test_df)

    clean_pred = model.predict(test_df.loc[clean_index])
    avg_duration = np.mean(clean_pred)

    pred = pd.DataFrame({
        "id": test_df.index.values,
        "duration": model.predict(test_df)
            },
                columns=["id", "duration"])

    pred.loc[dirty_index, "duration"] = avg_duration
    assert sum(clean_index) + sum(dirty_index) == len(test_df)

    return np.array(pred["duration"])
```

In [18]:

```
from datetime import datetime
def generate_submission(test, predictions, force=False):
    if force:
        if not os.path.isdir("submissions"):
            os.mkdir("submissions")
        submission_df = pd.DataFrame({
            "id": test_df.index.values,
            "duration": predictions,
        },
            columns=['id', 'duration'])

        timestamp = datetime.isoformat(datetime.now()).split(".")[0]

        submission_df.to_csv(f'submissions/submission_{timestamp}.csv', index=Fa

        print(f'Created a CSV file: submission_{timestamp}.csv')
        print('You may now upload this CSV file to Kaggle for scoring.')
```

Typesetting math: 100%

In [19]:

```
test_df.head()
```

Out[19]:

	record_id	VendorID	tpep_pickup_datetime	passenger_count	trip_distance	pickup_longitude	pi
0	10000	1	2016-01-02 01:45:37	1	1.20	-73.982224	
1	19000	2	2016-01-02 03:05:16	1	10.90	-73.999977	
2	21000	1	2016-01-02 03:24:36	1	1.80	-73.986618	
3	23000	2	2016-01-02 03:47:38	1	5.95	-74.002922	
4	27000	1	2016-01-02 04:36:44	1	1.60	-73.986366	

In [20]:

```
_, _ = process_data_fm(test_df, True)
test_df = test_df[['pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'manhattan', 'haversine', 'hour', 'trip_distance',
       'day_of_week', 'total_amount', 'tolls_amount', 'tip_amount', 'extra',
       'fare_amount']]
final_predictions = predict(final_model, test_df)
final_predictions = final_predictions.astype(int)
generate_submission(test_df, final_predictions, True)
```

```
Created a CSV file: submission_2018-12-05T19:00:22.csv
You may now upload this CSV file to Kaggle for scoring.
```

```
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:438
8: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a fu
ture version.
  object.__getattribute__(self, name)
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generic.py:438
9: FutureWarning: Attribute 'is_copy' is deprecated and will be removed in a fu
ture version.
  return object.__setattr__(self, name, value)
```

Submission

You're almost done!

Before submitting this assignment, ensure that you have:

1. Restarted the Kernel (in the menubar, select Kernel$\rightarrow$Restart & Run All)
2. Validated the notebook by clicking the "Validate" button.

Then,

1. **Submit** the assignment via the Assignments tab in **Datahub**
2. **Upload and tag** the manually reviewed portions of the assignment on **Gradescope**

In [ ]:

```

```

Typesetting math: 100%

In [ ]: