shireen-sfirstnotebook

September 24, 2018

In [ ]:
```
import numpy as np
import pandas as pd
import feather
import seaborn as sb
import os
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
import warnings
from datetime import datetime
import calendar
warnings.filterwarnings('ignore', category=RuntimeWarning)
print(os.listdir("../input/shireen-sfirstnotebook"))
df_tmp = pd.read_csv('../input/new-york-city-taxi-fare-prediction/train.csv', nrows = 10
holidays_df = pd.read_csv("../input/new-york-city-taxi-fare-prediction/NYC_holidays.csv"
test_df = pd.read_csv('../input/new-york-city-taxi-fare-prediction/test.csv')
path = 'train_data.feather'
df_tmp.to_feather('train_data.feather')
```

In [ ]:
```
df = pd.read_feather('train_data.feather')
df['pickup_latitude'] = df['pickup_latitude'].astype("float32")
df['pickup_longitude'] = df['pickup_longitude'].astype("float32")
df['dropoff_longitude'] = df['dropoff_longitude'].astype("float32")
df['dropoff_latitude'] = df['dropoff_latitude'].astype("float32")
df['pickup_latitude'] = df['pickup_latitude'].astype("float32")
df['passenger_count'] = df['passenger_count'].astype("uint8")
df['fare_amount'] = df['fare_amount'].astype("float32")
```

In [ ]:
```
df = pd.read_feather('train_data.feather')
df['abs_lat_diff'] = (df['dropoff_latitude'] - df['pickup_latitude']).abs()
df['abs_lon_diff'] = (df['dropoff_longitude'] - df['pickup_longitude']).abs()
test_df['abs_lat_diff'] = (test_df['dropoff_latitude'] - test_df['pickup_latitude']).abs
test_df['abs_lon_diff'] = (test_df['dropoff_longitude'] - test_df['pickup_longitude']).a
```

In [ ]:
```
df = df.dropna(axis=0, how='any') #Removed NaN's
print('After removing NaN, New size: %d'  % len(df))
df = df[(df != 0).all(1)] #Removed all zeroes
df = df[(df.fare_amount > 2.5) & (df.fare_amount <= 100)]
```

```
        print('After removing zeroes and fare outliers, New size: %d' % len(df))
        df = df.loc[df['passenger_count'] <= 6]
        print('After removing passenger_counts greater than 6, New size: %d' % len(df))

In [ ]: df['pickup_date'] , df['pickup_hour'] = df['pickup_datetime'].str.split(' ', 1).str
        df['day_of_week'] = df['pickup_date'].apply(lambda x: calendar.day_name[(datetime.strpti
        print("Pickup Date,Pickup Hour and Day of week added to data frame")

In [ ]: df['pickup_hour'] = df.pickup_hour.str[0:2]
        df['pickup_hour'] = df['pickup_hour'].apply(pd.to_numeric)
        df['pickup_hour'] = df['pickup_hour'].astype("uint8")

In [ ]: test_df['pickup_date'] , test_df['pickup_hour'] = test_df['pickup_datetime'].str.split('
        test_df['pickup_hour'] = test_df.pickup_hour.str[0:2]
        test_df['pickup_hour'] = test_df['pickup_hour'].apply(pd.to_numeric)

In [ ]: max_long = -71.7517
        min_long = -79.7624
        max_lat = 45.0153
        min_lat = 40.4772
        df = df[(df.pickup_longitude >= min_long) & (df.pickup_longitude <= max_long) & #Removed
                (df.dropoff_longitude >= min_long) &(df.dropoff_longitude <= max_long) &
                (df.pickup_latitude >= min_lat) & (df.pickup_latitude <= max_lat) &
                (df.dropoff_latitude >= min_lat) &(df.dropoff_latitude <= max_lat)]
        print("After NY outliers, Newest Size : %d" %len(df))

In [ ]: from math import radians, sin, cos, sqrt, asin
        df.info()
        #Referenced logic to compute haversine from https://en.wikipedia.org/wiki/Haversine_form
        def haversine_np(lon1, lat1, lon2, lat2):
            lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])
            a = np.sin((lat2-lat1)/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin((lon2-lon1)/2.
            return 6367 * 2 * np.arcsin(np.sqrt(a)) *0.62137

        df['trip_distance'] = haversine_np(df.pickup_longitude, df.pickup_latitude,df.dropoff_lo
                              df.dropoff_latitude)

        # df['trip_distance'] = haversine_np(df.pickup_latitude, df.pickup_longitude, df.dropoff
        print("After less distance entries removal, Newest Size : %d" %len(df))

In [ ]: df = df[(df['trip_distance']>0.5) & (df['trip_distance']<30)]

In [ ]: df = df.assign(holiday_or_not=df['pickup_date'].apply(lambda x: x in holidays_df["Date"]
        df["holiday_or_not"] = df["holiday_or_not"].astype("unit8")
        df_new = df[df["holiday_or_not"] == 1]
        holiday_data_df_group = holiday_data_df[["date", "vendor_id"]].groupby("date").count()
        len(df_new)

In [ ]: test_df['trip_distance'] =  haversine_np(test_df.pickup_longitude, test_df.pickup_latitu
                              test_df.dropoff_latitude)
```

In [ ]:
```
import matplotlib.pyplot as plt
pearsoncoeff_tripdistance_amount = df['trip_distance'].corr(df['fare_amount'], method='p
print(pearsoncoeff_tripdistance_amount)

figure, graph = plt.subplots(1, 2, figsize=(16,6))
graph[0].scatter(df.trip_distance, df.fare_amount, alpha=0.3)
graph[0].set_xlabel('Distance Travelled')
graph[0].set_ylabel('Fare Amount')
graph[0].set_title('Distance vs Fare')
```

In [ ]:
```
pearsoncoeff_triphour_distance = df['pickup_hour'].corr(df['trip_distance'], method='pea
print(pearsoncoeff_triphour_distance)

df.groupby('pickup_hour')['trip_distance'].mean().sort_index().plot.bar(color='c');
plt.title('Correlation between Average Fare Amount and Distance Travelled');
plt.ylabel('Trip Distance');
```

In [ ]:
```
pearsoncoeff_hour_fare =  df['pickup_hour'].corr(df['fare_amount'], method='pearson')
print(pearsoncoeff_hour_fare)
```

In [ ]:
```
df.groupby('pickup_hour')['fare_amount'].mean().sort_index().plot.bar(color='g');
plt.title('Correlation between Average Fare Amount and Time of the day');
plt.ylabel('Fare Amount');
```

In [ ]:
```
#Exciting Plot
df.groupby('day_of_week')['trip_distance'].mean().plot.bar(color='c');
plt.title('Correlation between day of the week and distance travelled');
plt.ylabel('Distance Travelled');
```

In [ ]:
```
from sklearn import metrics
X = df.drop(['key','fare_amount','pickup_datetime','day_of_week','pickup_date'],axis=1)
X_train, X_test, y_train, y_test = train_test_split(X,df['fare_amount'], test_size=0.2)
lm = linear_model.LinearRegression()
lm.fit(X_train,y_train)
y_pred = lm.predict(X)
lrmse = np.sqrt(metrics.mean_squared_error(y_pred, df['fare_amount']))
lrmse
```

In [ ]:
```
lr = linear_model.LinearRegression()
lr.fit(df[['trip_distance', 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
            'passenger_count','abs_lat_diff','abs_lon_diff','pickup_hour']], df['fare_a
print('Intercept', round(lr.intercept_, 4))
predicted_values = lr.predict(test_df[['trip_distance', 'pickup_longitude', 'pickup_lati
                                       'dropoff_latitude', 'passenger_count','abs_lat_diff'
print('Intercept', round(lr.intercept_, 4))
print('Trip Distance: ', round(lr.coef_[0], 4),
      '\tPickup Longitude:', round(lr.coef_[1], 4),
      '\tPickup Latitude:', round(lr.coef_[2], 4),
     '\Dropoff Longitude:', round(lr.coef_[3], 4),
```

```
      '\Dropoff Latitude:', round(lr.coef_[4], 4),
      '\Passenger Count:', round(lr.coef_[5], 4),
      '\Absolute Latitude Difference:', round(lr.coef_[6], 4),
      '\Absolute Longitude Difference:', round(lr.coef_[7], 4),
      '\Pickup Hour:', round(lr.coef_[8], 4))
# submission = pd.DataFrame(
#     {'key': test_df.key, 'fare_amount': predicted_values},
#     columns = ['key', 'fare_amount'])
# submission.to_csv('submission.csv', index = False)

print(os.listdir('.'))
```

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor as randomforest

# Create the random forest
random_forest = randomforest(n_estimators = 20, max_depth = 20,
                                        max_features = None, oob_score = True,
                                        bootstrap = True, verbose = 1, n_jobs = -1)

# Train on data
random_forest.fit(df[['trip_distance', 'pickup_longitude','pickup_latitude','dropoff_lon
predicted_values = random_forest.predict(test_df[['trip_distance', 'pickup_longitude', '
                                       'dropoff_latitude','abs_lat_diff', 'abs_lon_diff', '
submission = pd.DataFrame(
    {'key': test_df.key, 'fare_amount': predicted_values},
    columns = ['key', 'fare_amount'])
submission.to_csv('submission.csv', index = False)

print(os.listdir('.'))
```