Submission of assignment for:-

[https://www.springboard.com/workshops/data-science-career-track/learn#/curriculum/451](https://www.springboard.com/workshops/data-science-career-track/learn" \l "/curriculum/451)

**Project: Capstone Project 1 - Data Wrangling**

Honestly, the data that I picked up was more or less clean and in-shape to perform further analysis.

However, I was able to perform some cleanups as below.

* **Cleaning up trip duration.**

As we noted earlier there are some outliers associated with the trip\_duration variable, specifically a 980 hour maximum trip duration and a minimum of 1 second trip duration. I've decided to exclude data that lies outside 2 standard deviations from the mean. It might be worthwhile looking into what effect excluding up to 4 standard deviations would have on the end-results.

*m = np.mean(train['trip\_duration'])*

*s = np.std(train['trip\_duration'])*

*train = train[train['trip\_duration'] <= m + 2\*s]*

*train = train[train['trip\_duration'] >= m - 2\*s]*

* **Latitude and Longitude cleanup**

Looking into it, the borders of NY City, in coordinates comes out to be:

city\_long\_border = (-74.03, -73.75) city\_lat\_border = (40.63, 40.85)

Comparing this to our train.describe() output we see that there are some coordinate points (pick ups/drop offs) that fall outside these borders. So I cleaned up the data to limit our area of investigation to within the NY City borders.

*train = train[train['pickup\_longitude'] <= -73.75]*

*train = train[train['pickup\_longitude'] >= -74.03]*

*train = train[train['pickup\_latitude'] <= 40.85]*

*train = train[train['pickup\_latitude'] >= 40.63]*

*train = train[train['dropoff\_longitude'] <= -73.75]*

*train = train[train['dropoff\_longitude'] >= -74.03]*

*train = train[train['dropoff\_latitude'] <= 40.85]*

*train = train[train['dropoff\_latitude'] >= 40.63]*

* **Formatting datetime for better and easy extraction of these fields.**

*train['pickup\_datetime'] = pd.to\_datetime(train.pickup\_datetime)*

*train.loc[:, 'pick\_month'] = train['pickup\_datetime'].dt.month*

*train.loc[:, 'hour'] = train['pickup\_datetime'].dt.hour*

*train.loc[:, 'week\_of\_year'] = train['pickup\_datetime'].dt.weekofyear*

*train.loc[:, 'day\_of\_year'] = train['pickup\_datetime'].dt.dayofyear*

*train.loc[:, 'day\_of\_week'] = train['pickup\_datetime'].dt.dayofweek*

Had the date been more messy, I would have used some more data wrangling methods cited as below.

1. DATA CLEANING

df.columns = ['a','b','c'] - Renames columns

pd.isnull() - Checks for null Values, Returns Boolean Array

pd.notnull() - Opposite of s.isnull()

df.dropna() - Drops all rows that contain null values

df.dropna(axis=1) - Drops all columns that contain null values

df.fillna(x) - Replaces all null values with x

df.fillna(s.mean()) - Replaces all null values with the mean (mean can be replaced with almost any function from the statistics section)

df.replace(1,'one') - Replaces all values equal to 1 with 'one'

1. JOIN/COMBINE

df1.append(df2) - Adds the rows in df1 to the end of df2 (columns should be identical) pd.concat([df1, df2],axis=1) - Adds the columns in df1 to the end of df2 (rows should be identical)

1. SELECTION

df[col] - Returns column with label col as Series

df[[col1, col2]] - Returns Columns as a new DataFrame df.iloc[0] - Selection by position

df.loc[0] - Selection by index

df.iloc[0,:] - First row

df.iloc[0,0] - First element of first column

1. STATISTICS

These can all be applied to a series as well.

df.describe() - Summary statistics for numerical columns

df.mean() - Returns the mean of all columns

df.corr() - Returns the correlation between columns in a DataFrame

df.count() - Returns the number of non-null values in each DataFrame column

df.max() - Returns the highest value in each column

df.min() - Returns the lowest value in each column

df.median() - Returns the median of each column

df.std() - Returns the standard deviation of each column