The dataset has following columns for analysis:

- **lpep_pickup_datetime -** pick up time
- **passenger_count** - number of passengers
- **trip_distance** - distance covered during the trip (in miles)
- **trip_duration** - time taken for the trip (in seconds)
- **fare_amount** - fare for the ride (in $)
- **tip_amount** - tip given by the rider for the trip (in $)
- **congestion_surcharge -** surcharge added by the taxi agency (in $)

Project ToDos

3.1.1 Dataset Cleaning

- Delete the rows which have *trip_distance* equal to 0.
  Note: once this step is done, all the values in the *trip_distance* column should be non-null and greater than 0. Please make sure this is the case, else there will be problems in the linear interpolation step.
- Check all the columns for negative values, and check the *trip_duration* and *fare_amount* columns for zero values. Remove these values from the dataset (Please note that you only need to delete the cell values, not the entire row. One simple way to achieve this can be to mark these cells as NaN). We will replace them with more reasonable values in the interpolation task.

3.1.2 Outlier Detection

The first task is to perform outlier detection for these columns in your dataset:

- *passenger_count*
- *trip_duration*
- *fare_amount*

Use Tukey's rule from class with α = 1.5 for this task. The rule should be applied across all values for each column separately. For eg., to find outliers for the *trip_duration* column, consider all values of *trip_duration* across the two months to find the outliers. Note the number of outliers detected for each column in your submission. Then, delete these outlier values from your dataset (Again, only delete the cell values, not the entire row. One simple way to achieve this can be to mark these cells as NaN). We will replace them with more reasonable values in the next task.

3.1.3 Perform linear interpolation of missing data-points

For this task, you will use linear interpolation to fill the missing data points. Linear interpolation is a method of estimating values that are missing in a dataset. It involves using the known data points above and below the missing value to estimate what the value would be if it were present. We will do this across the entire 2 months data. The process of linear interpolation involves the following steps:

1. Identify the missing data points in the dataset.

2. Find the known data points that are immediately above and below the missing data point, with respect to the pickup time of the trip. In other words, sort the data by *lpep_pickup_datetime* and for each missing datapoint, find the first known datapoint above and below it.

3. Use linear interpolation to estimate the missing data point, scaled by the corresponding *trip_distance*. **The procedure will be the same for all the columns for which we are linearly interpolating the missing values.**
   For example, let us consider the following scenario:
   Suppose we are interpolating a point in the *fare_amount* column. After sorting the data by *lpep_pickup_datetime*, we will first find the value of *fare_amount* per unit *trip_distance* for the first known points above and below the missing point, as shown below.

lpep_pickup_datetime	*trip_distance (tr_dst)*	*fare_amount (fr_amt)*	*fr_amt / tr_dst*
6/7/21 0:26	5.24	21.5	4.10
6/7/21 0:30	1.42	**NaN**	**–**
6/7/21 0:48	2.56	11	4.29

Table 1: Missing values dataset

4. Calculate the slope of the line that connects the two known data points. This can be done using the formula:

   slope = (y2 − y1) / (x2 − x1)

   where x1 and y1 are the coordinates of the first known data point, and x2 and y2 are the coordinates of the second known data point. Note that in your case y1 and y2 are the '*fr_amt / tr_dst*' values above and below the range of missing data and (x2 − x1) gives the number of seconds between the two pickup times.

5. Use the slope to estimate the *fare_amount* per unit *trip_distance* of the missing data point. This can be done using the formula:

   ‘fr_amt / tr_dist’ = y1 + (slope ∗ (missing value x − x1))

   where (missing value x - x1) is the number of seconds between the missing value datapoint and the point above it.

6. Scale up the obtained *‘fr_amt / tr_dist’* value by taking its product with the *trip_distance* for the missing value to get the *fare_amount* for this row.

   fare_amount = ‘fr_amt / tr_dist’ * trip_distance

7. Replace the missing *fare_amount* with the estimated value.

lpep_pickup_datetime	*trip_distance (tr_dst)*	*fare_amount (fr_amt)*	*fr_amt / tr_dst*
6/7/21 0:26	5.24	21.5	4.10
6/7/21 0:30	1.42	**5.87**	**4.13**
6/7/21 0:48	2.56	11	4.29

Table 2: Completed dataset

3.2 Add *total_amount* Column

Now that we have replaced the missing values in all the columns, we will add a new column to our dataset called *total_amount* as defined below.
For each row in the dataset, the value of *total_amount* for that row will be calculated as follows:

total_amount = fare_amount + tip_amount + congestion_surcharge

3.3 Performing Wald’s test, Z-test and t-test

In this step, we want to check how the mean of monthly stats has changed between the two months in your dataset. Apply the Wald’s test, Z-test, and t-test (assume all are applicable) to check whether the mean of *trip_distance* and *fare_amount* are different for the two months. Do this separately for both columns, i.e., you have to compare the mean of stats from month 1 with the mean of stats from month 2 for the *trip_distance* column; repeat the same for the *fare_amount* column. Use MLE for Wald’s test as the estimator; assume for Wald’s estimator purposes that the data is Poisson distributed, and show the derivation of the MLE estimate in your submission.

3.4 Performing K-S test and Permutation test

Infer the equality of distributions between *trip_duration* and *fare_amount* for the dataset using K-S test and Permutation test. For the K-S test, use both 1-sample and 2-sample tests. For the 1-sample test, try Poisson, Geometric, and Binomial. To obtain parameters of these distributions to check against in 1-sample KS, use MME on the *trip_duration* to obtain parameters of the distribution, and then check whether the data for *fare_amount* in your dataset has the distribution with the obtained MME parameters. For the permutation test, use 1000 random permutations. Use a threshold of 0.05 for both K-S test and Permutation test.

3.5 Linear regression

Use *trip_duration* and *trip_distance* to predict *total_amount* using linear regression:

1. Separately ie (*total_amount* vs *trip_duration*) and (*total_amount* vs *trip_distance*)
2. Together ie (*total_amount* vs [*trip_duration*,*trip_distance*])

Calculate SSE and MAPE in both cases. Find β_0, β_1 in each case for Part 1 and β_0, β_1, β_2 for Part 2. Compare both cases and state your observations based on the regression fit.

3.6 Time Series Analysis

Use the *total_amount* column to predict the trip amount on the **last** date of your given data. First, find the median trip amount for each day. Then use this median data from 1st date of the first month till the second last date of the second month (58-61 days). For instance, if your dataset covers 60 days, utilize the median trip amount data of 59 days (according to the methods outlined below) to predict the trip amount for the 60th day. Use the following techniques for your predictions:

1. AR(2)
2. AR(3)
3. EWMA alpha = 0.3
4. SMA (window = 7)

Use MAPE to report the accuracy of the predictions using the median trip amount of the last day as your observed data.

3.7 Chi-Square Test

Use *tip_amount* and *passenger_count* columns for this test. First using median values for each column as a threshold, categorize them into high and low. Now use the Chi Square test to find if they are dependent or independent.

3.8 Bayesian Test

For this test, we will work on the *trip_distance* column on the filter condition:

$0 <$ *trip_distance* $<= 10$ miles

For **only the first month** of the provided data, create 30 samples of 1000 points each chosen randomly from the filtered data (use random states ranging from 1 to 100). Eg: `df_filtered.sample(n=1000, random_state=1`. So each sample (X_i) will have 1000 points and there will be 30 such samples ($X_1, X_2, \ldots X_{30}$)

Assume the frequency distribution of each sample belongs to a Normal Distribution with some unknown mean μ and known standard deviation σ, ie $\{X_1, X_2, \ldots X_{30}\} \sim \text{Norm}(\mu, \sigma^2)$. Assume that the prior of the mean μ, $f(\mu)$, to also be normally distributed with mean $m = 2.25$ and standard deviation $sd = 1.95$

a) Find the posterior distribution of the data via Bayesian Inference: $f(\mu \mid D_1)$ - where D_1 represents the sampled data from the **first** month. You can use the results from A6 Q1a). **Do not** substitute *m, sd,* n, and $\sum X_i$ till the end. Obtain the updated mean (μ_1) and updated standard deviation (σ_1) of the posterior distribution by representing the distribution in its normal form.
   i) Comment on the obtained mean μ_1 and standard deviation σ_1 by representing them in terms of *m*, $\bar{X}$, and *sd* respectively. What happens as n tends towards infinity?
   ii) Now substitute the values of *m, sd,* n, and $\sum X_i$ to obtain the updated mean (μ_1) and standard deviation (σ_1). Assume $\sigma = 2$.

b) Now repeat the same sampling procedure for the **second** month, where again, the frequency of each sample belongs to a Normal Distribution with some unknown mean **μ** and known standard deviation **σ**, ie $\{X_{31}, X_{32}, \ldots X_{60}\} \sim \text{Norm}(\mu, \sigma^2)$. Use the posterior distribution obtained in a) ii), $f(\mu \mid D_1)$, as the new prior distribution i.e., the previous posterior distribution becomes the new prior.
   i) With the assumption that $\sigma = 2$, calculate the updated mean (μ_2) and standard deviation (σ_2) for the new posterior distribution: $f(\mu \mid D_1 \cap D_2)$
   ii) Find the MAP for the obtained posterior distribution in i).