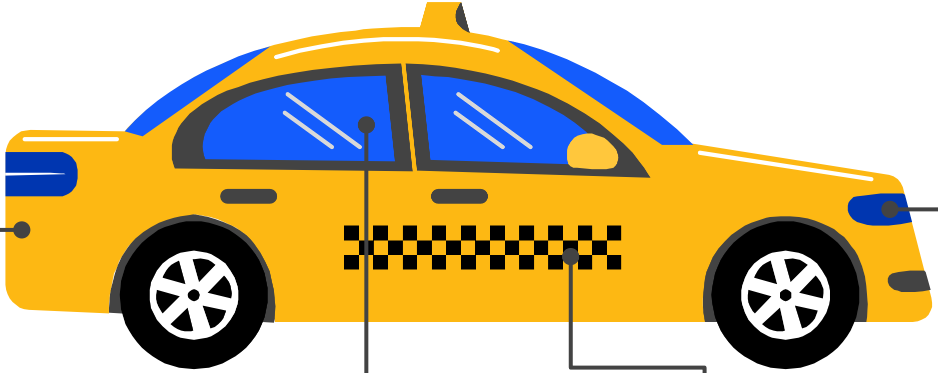
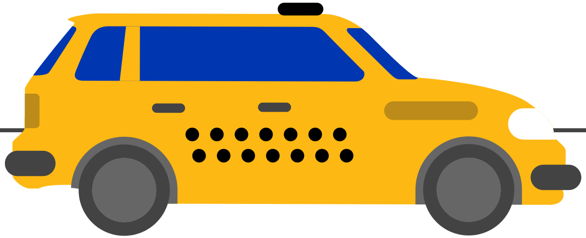
# Final Project

Taxi Demand Prediction on Time Series Data with Autoregressive Moving Average (ARIMA)

# Table of Contents



# Table of Contents



# Overview

With the increasing success of NYC TLC, the demand for better services has also increased in order to be able to serve all incoming taxi requests. They didn't want to allocate too many cars because it would be too expensive. On the other hand, they will lose money if they don't have enough cars to serve all incoming requests.
Thus, the **ARIMA** method will be used as a method to predict the number of occupants in order to serve all incoming requests successfully without paying for unused cars.

# Data Understanding

**Sources**

**Datasets**

**NOTED:** There are 4 datasets in this project, namely:

- Data set trips

Contains complete data on taxi trips in NYC

- Data set train

Contains the number of taxi passengers in NYC per hour

- Data set sample

Give an example of the predictive data you need to submit

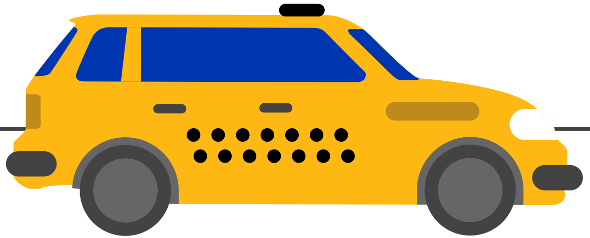
- Data set test

Contains test data to predict at a certain date time

| Field Name | Description |
|---|---|
| vendor_id | A code indicating the TPEP provider that provided the record.<br>• Mobile Technologies<br>• VeriFone Inc |
| pickup_datetime | The date and time when the meter was engaged. |
| dropoff_datetime | The date and time when the meter was disengaged. |
| passenger_count | The Number of passengers in the vehicle. This is a driver-entered value. |
| trip_distance | The elapsed trip distance in miles reported by the taximeter. |
| rate_code | • The final rate code in effect at the end of the trip.<br>Standar rate<br>• JFK<br>• Newark<br>• Nassau or Westchester<br>• Negoitated fare<br>• Group ride |
| payment_type | A numeric code signifying how the passenger paid for the trip.<br>• Credit Card<br>• Cash<br>• No Charge<br>• Dispute<br>• Unknown<br>• Voided Trip |
| store_and_fwd_flag | This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka 'store and forward', because the vehicle did not have a connection to the server.<br>Y = store and forward trip<br>N = not a store and forward trip |
| fare_amount | The time-and-distance fare calculated by the meter. |
| extra | Miscellaneous extras and surcharges. Currentlly, this only includes. The $0.50 and $1 rush hour and overnight charges. |
| mta_tax | 0.50 MTA tax that us automatically triggered based on the metered rate in use. |
| tip_amount | Tip amount - This field is automatically populated for credit card trips. Cash tips are not included. |
| tolls_amount | Total amount of all tolls paid in trip. |
| imp_surcharge | 0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015. |
| airport_fee | |
| total_amount | The total amount charged to passengers. Does not include cash tips. |
| pickup_location_id | As specified by the pickup-point provider. |
| dropoff_location_id | As specified by the dropoff-point provider. |
| data_file_year | The file for a specific year. |
| data_file_month | The file for a specific month. |

# Table of Contents





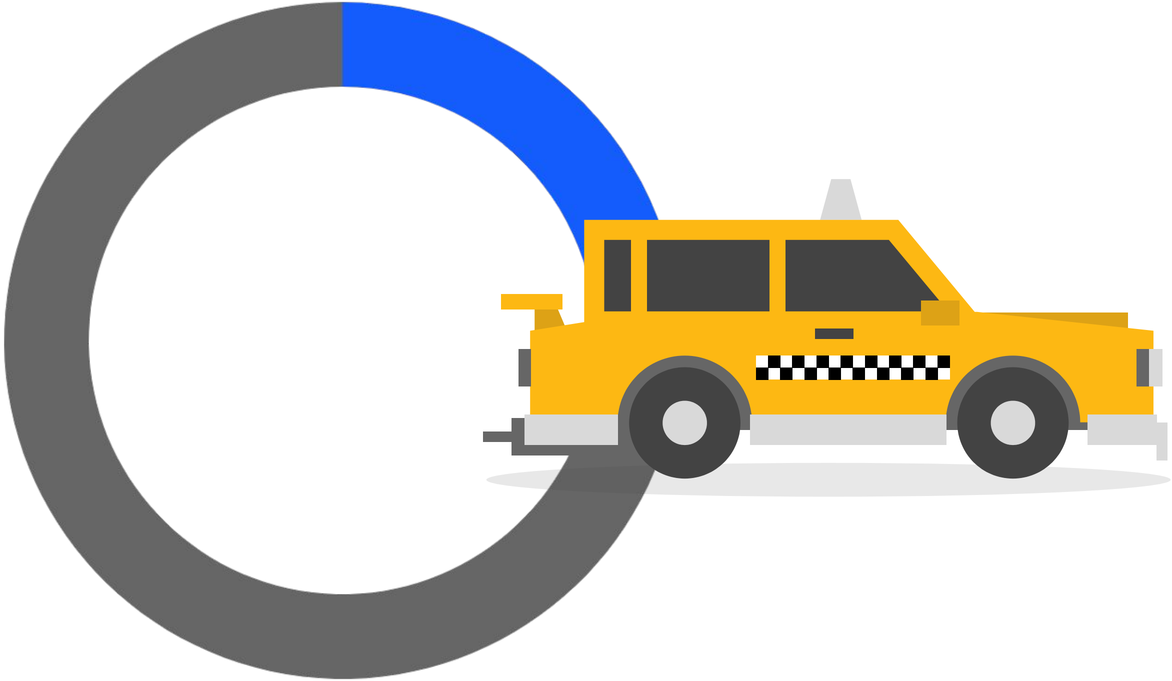
# EDA on Feature Vendor :

**71 %**

**VeriFone Inc**

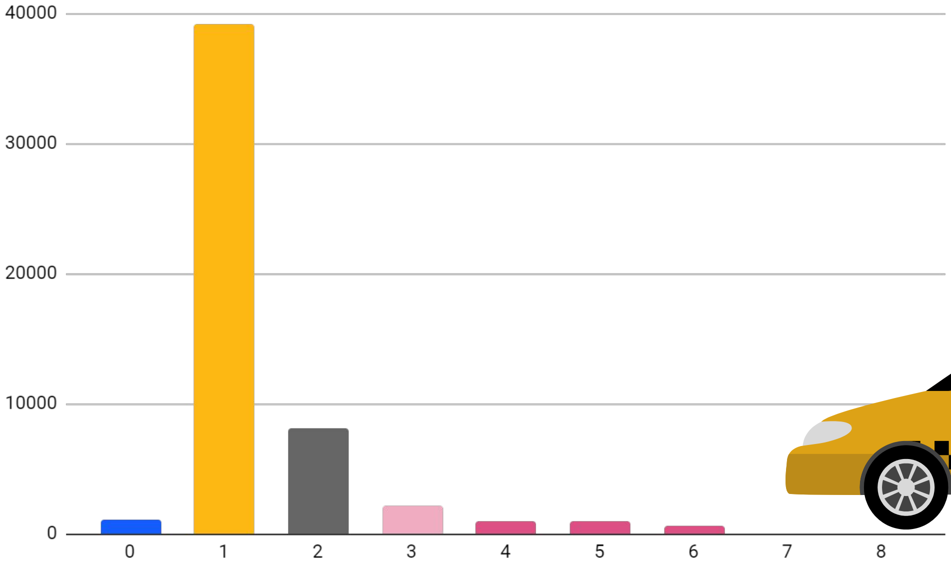
**29 %**

**Mobile Technologies**



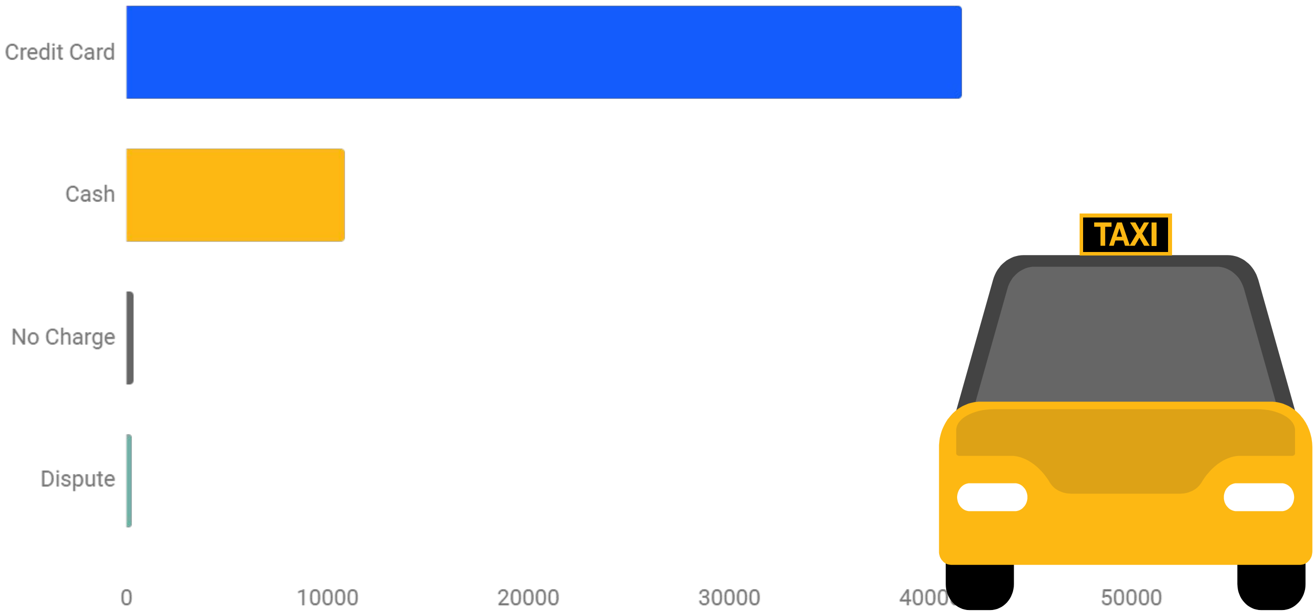
# EDA on Passengers Count :

**Observation :**
The number of frequencies is the most of the number of passengers per car, namely the number of passengers is only 1 person

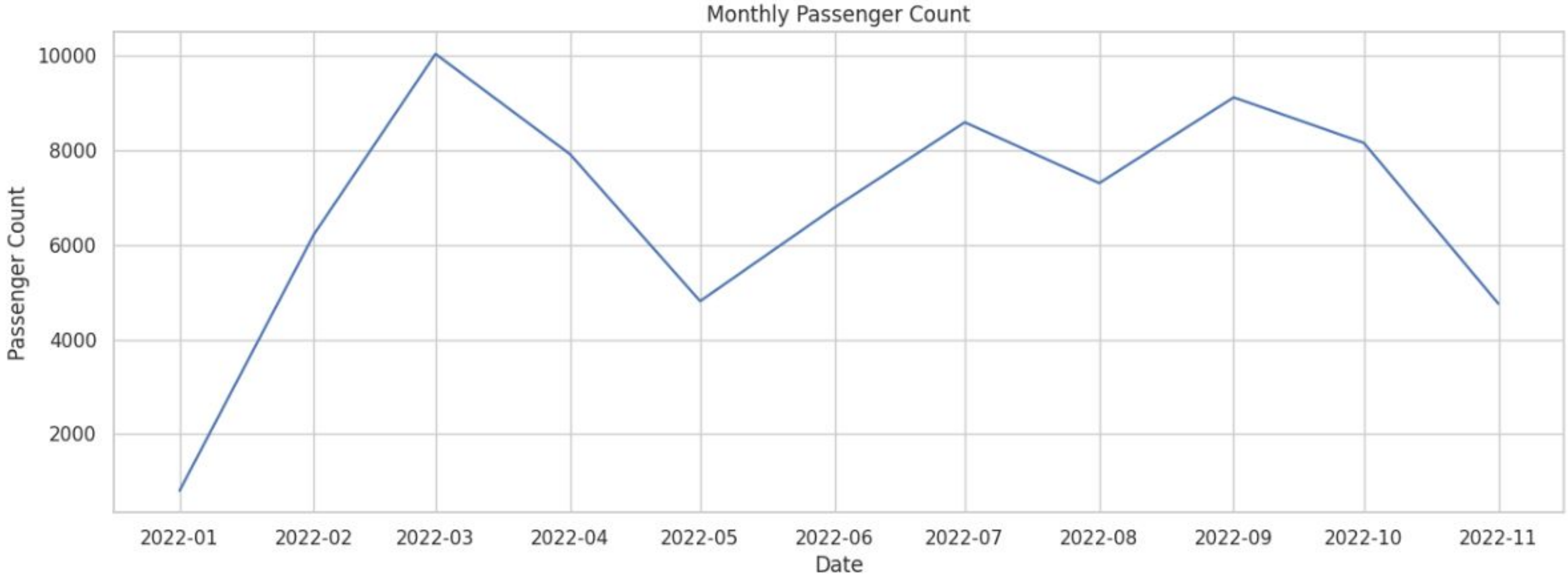



# EDA on Feature Payment Type :

**Observation :**
Based on the graph above, most passengers pay by credit card and a little for free

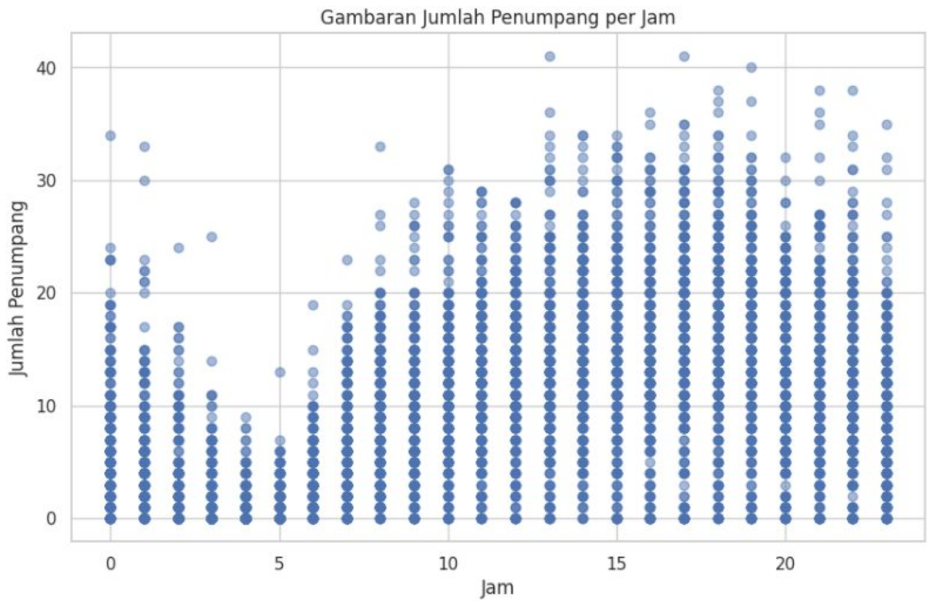


# Monthly Passenger Count :



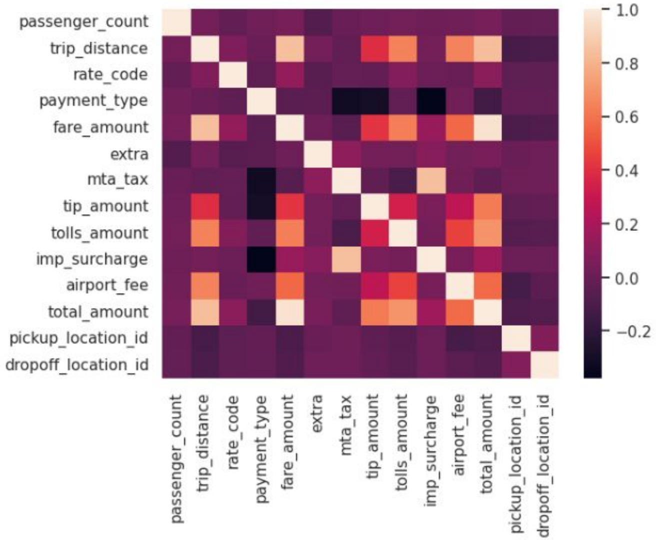

Based on the graph above, it shows that every month during 2022 the number of passengers will **fluctuate**

# An overview of the number of passengers per hour :




Based on the graph above, the busiest hours for pickup passengers are around **14.00-19.00**.

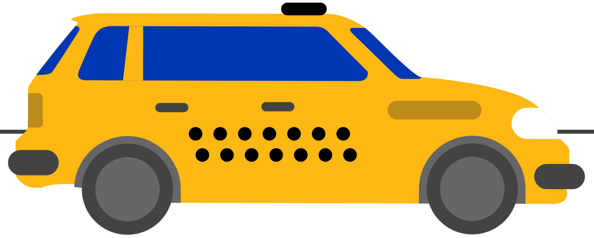
# Correlation for Data Trips :




Based on the results of the correlation between each columns, it can be concluded that if the correlation value is **more than 0.5** then the relationship between columns has a strong effect

# Table of Contents





# Autoregressive Moving Average :

Test whether it is true that the data_train has fluctuated :

```
from statsmodels.tsa.stattools import adfuller

def ad_test(dataset):
     dftest = adfuller(dataset, autolag = 'AIC')
     print("1. ADF : ",dftest[0])
     print("2. P-Value : ", dftest[1])
     print("3. Num Of Lags : ", dftest[2])
     print("4. Num Of Observations Used For ADF Regression:", dftest[3])
     print("5. Critical Values :")
     for key, val in dftest[4].items():
         print("\t",key, ": ", val)
```

```
ad_test(train['passenger_count'])
```

Result :

```
1. ADF :  -5.234425988209913
2. P-Value :  7.475326613689224e-06
3. Num Of Lags :  33
4. Num Of Observations Used For ADF Regression: 6272
5. Critical Values :
	 1% :  -3.431393045018898
	 5% :  -2.8620009336864833
	 10% :  -2.567015352019056
```

Because here it uses data_train, where this data contains the number of passengers per hour and is one of the summaries of data_trips. And the results show that the time series is **not stationary (fluctuated)**.

# Autoregressive Moving Average :

Choose the best Autoregressive Moving Average (ARIMA) model based on data_train using auto_arima or hereinafter referred to as Historical Data :

```
from pmdarima import auto_arima
  # Mengabaikan warning
import warnings
warnings.filterwarnings("ignore")
```

```
stepwise_fit = auto_arima(train['passenger_count'], trace=True,
                          suppress_warnings=True)
stepwise_fit.summary()
```

Result :

```
Best model:  ARIMA(2,1,3)(0,0,0)[0]
Total fit time: 242.520 seconds
```

SARIMAX Results

| | | | |
|---|---|---|---|
| Dep. Variable: | y | No. Observations: | 6306 |
| Model: | SARIMAX(2, 1, 3) | Log Likelihood | -19368.923 |
| Date: | Mon, 28 Aug 2023 | AIC | 38749.847 |
| Time: | 14:47:08 | BIC | 38790.342 |
| Sample: | 0 | HQIC | 38763.874 |
| | - 6306 | | |
| Covariance Type: | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| ar.L1 | 1.9269 | 0.001 | 1517.606 | 0.000 | 1.924 | 1.929 |
| ar.L2 | -0.9945 | 0.001 | -783.067 | 0.000 | -0.997 | -0.992 |
| ma.L1 | -2.7683 | 0.006 | -446.467 | 0.000 | -2.780 | -2.756 |
| ma.L2 | 2.6125 | 0.012 | 214.398 | 0.000 | 2.589 | 2.636 |
| ma.L3 | -0.8395 | 0.006 | -134.866 | 0.000 | -0.852 | -0.827 |
| sigma2 | 27.1528 | 0.378 | 71.844 | 0.000 | 26.412 | 27.894 |

| | | | |
|---|---|---|---|
| Ljung-Box (L1) (Q): | 36.10 | Jarque-Bera (JB): | 924.47 |
| Prob(Q): | 0.00 | Prob(JB): | 0.00 |
| Heteroskedasticity (H): | 1.40 | Skew: | 0.66 |
| Prob(H) (two-sided): | 0.00 | Kurtosis: | 4.33 |

From the results of the pmdarima library (Python AutoRegressive Integrated Moving Average) shows that the best model is with p (Order of Autoregressive - AR), d (Order of Integration - I) and q (Order of Moving Average - MA) worth **(2, 1,3)**.

# Autoregressive Moving Average :

Because data_test only contains the index of the desired datetime to predict, so here I make predictions for the number of passengers using the ARIMA model based on historical data starting from
2022-09-26 08:00:00+00:00 to
2022-11-30 23:00:00+ 00:00

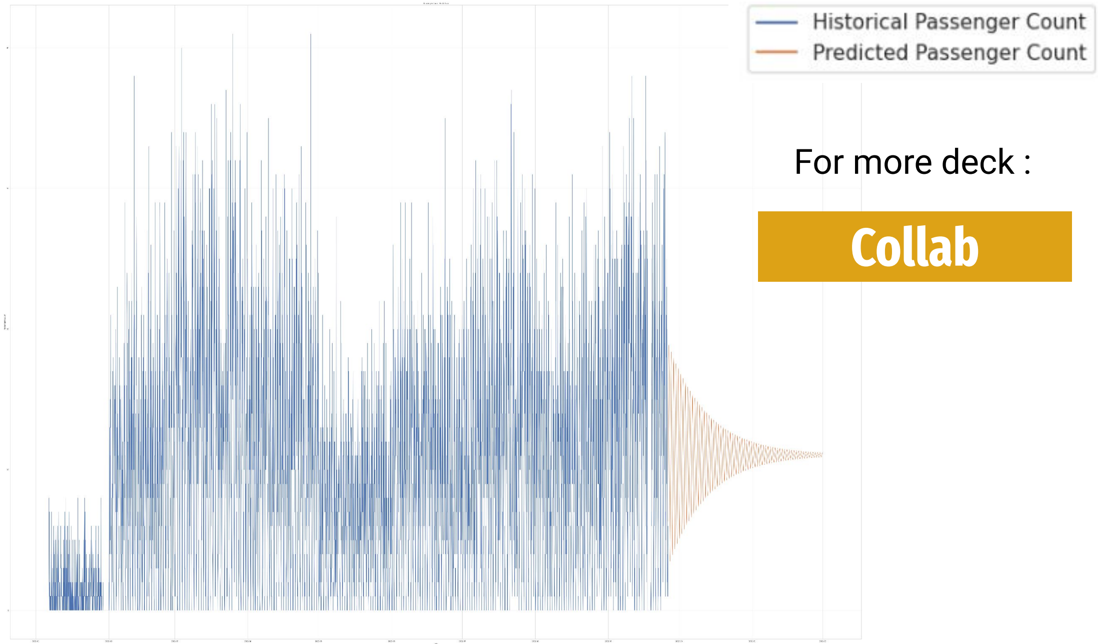
```python
import pandas as pd
import numpy as np
import statsmodels.api as sm

# Tanggal-tanggal yang diberikan dengan zona waktu UTC
start_date = pd.to_datetime('2022-09-26 08:00:00+00:00')
end_date = pd.to_datetime('2022-11-30 23:00:00+00:00')
predicted_index = pd.date_range(start_date, end_date, freq='H')

# Buat DataFrame kosong untuk menyimpan prediksi
predicted_data = pd.DataFrame(index=predicted_index)

# Pilih deret waktu dari data historis
historical_data = train['passenger_count']

# Perluas indeks historis untuk mencakup rentang waktu yang ingin diprediksi
extended_index = historical_data.index.union(predicted_index)

# Reindeks ulang deret waktu historis
data_train = historical_data.reindex(extended_index)

# Fit model ARIMA ke data yang sudah diperluas indeksnya
# Gunakan model ARIMA terbaik yang telah di uji coba sebelum nya dengan p = 2, d = 1, dan q = 3
model = sm.tsa.ARIMA(data_train, order=(2, 1, 3))
results = model.fit()

# Lakukan prediksi untuk rentang waktu yang diinginkan
predictions = results.predict(start=start_date, end=end_date, dynamic=True)
predicted_data['predicted_passenger_count'] = predictions
```

# Autoregressive Moving Average :

Final Result :

| | predicted_passenger_count |
|---|---|
| 2022-09-26 08:00:00+00:00 | 6.332948 |
| 2022-09-26 09:00:00+00:00 | 8.171293 |
| 2022-09-26 10:00:00+00:00 | 10.195562 |
| 2022-09-26 11:00:00+00:00 | 12.267755 |
| 2022-09-26 12:00:00+00:00 | 14.247313 |
| ... | ... |
| 2022-11-30 19:00:00+00:00 | 11.208410 |
| 2022-11-30 20:00:00+00:00 | 11.208503 |
| 2022-11-30 21:00:00+00:00 | 11.198026 |
| 2022-11-30 22:00:00+00:00 | 11.177739 |
| 2022-11-30 23:00:00+00:00 | 11.149067 |

1576 rows × 1 columns




For more deck :

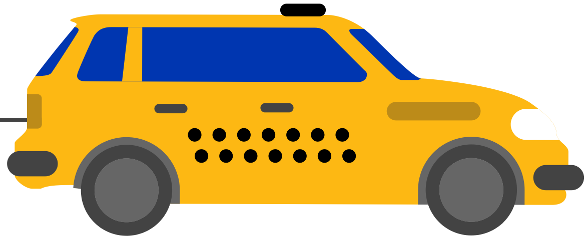
**Collab**

From the overall results of this model I made based on literature studies and my abilities. In time series data, you can use "Weighted MA", "Exponential WMA", "Triple EWMA" as features. Also “Fourier Transform” could provide better features, but in this case it's quite functional.

**If there is an error please I am open to being corrected and reminded.**

# Table of Contents





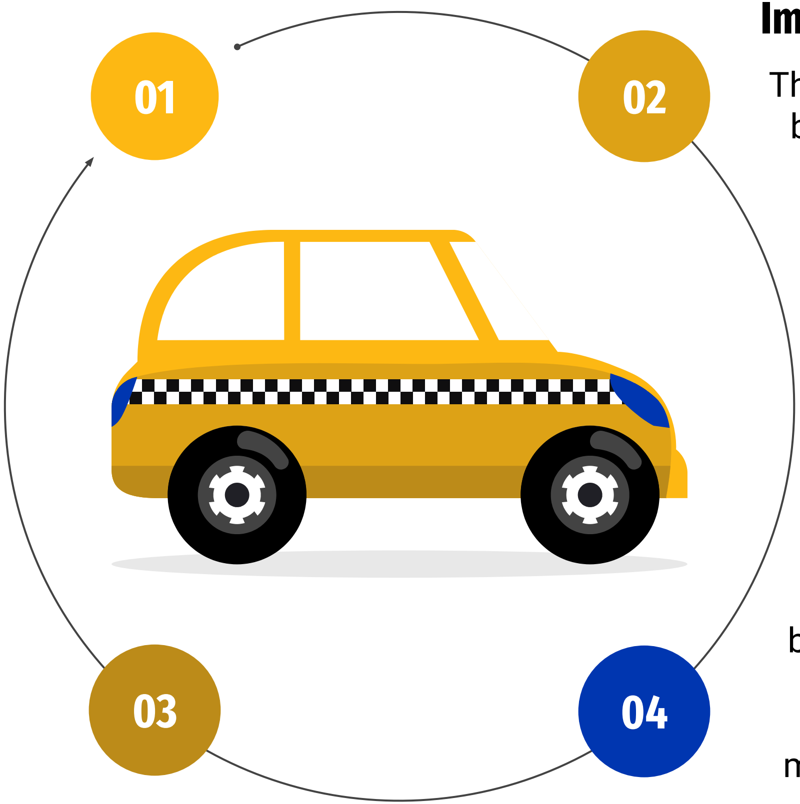
# Business Recommendation

## Fleet Allocation Optimization:

Companies can allocate cars at 10 o'clock and above, because seeing the predictions at that hour there will be an increase in passengers

## Improved Customer Service:

This predictive information can be used to provide customers with more accurate wait time updates. This can increase customer satisfaction by providing realistic expectations.




## Fleet Maintenance:

By predicting future demand, taxi companies can better plan fleet upkeep and maintenance. This helps minimize the impact on service from a damaged fleet.

## Partnerships and Void Ride Mitigation:

Partnerships and Void Ride Mitigation: If possible, consider a partnership between Mobile Technologies and VeriFone Inc with a ridesharing service to help meet demand when taxi fleets are limited.

# Thank You
# for Attention !