

↓ *Final Project*

Prediksi Jumlah
Penumpang Taxi NYC TLC
Dengan Menggunakan
*Autoregressive Moving
Average (ARIMA)*



Permasalahan

Dengan semakin suksesnya NYC TLC, permintaan akan pelayanan yang lebih baik juga meningkat agar dapat melayani semua permintaan taksi yang masuk. Mereka tidak ingin mengalokasikan terlalu banyak mobil karena akan terlalu mahal. Di sisi lain, mereka akan merugi jika tidak memiliki cukup mobil untuk melayani semua permintaan yang masuk. Dengan demikian, akan digunakan metode ARIMA dalam memprediksi jumlah penumpang agar melayani semua permintaan yang masuk dengan sukses tanpa membayar mobil yang tidak terpakai.

Autoregressive Moving Average (ARIMA)

Model ARIMA merupakan gabungan antara model AR (Autoregressive) yaitu suatu model yang menjelaskan pergerakan suatu variabel melalui variabel itu sendiri di masa lalu dan model MA (Moving Average) yaitu model yang melihat pergerakan variabelnya melalui residualnya di masa lalu. ARIMA cocok untuk digunakan dalam peramalan time series jangka pendek. Metode ini juga telah banyak diterapkan untuk peramalan dalam berbagai bidang. ARIMA memiliki tingkat keakuratan peramalan yang cukup tinggi karena setelah mengalami tingkat pengukuran kesalahan peramalan MAE (*mean absolute error*) nilainya mendekati nol

Data Cleaning

Informasi Data Trips

```
[7] 1 trips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54225 entries, 0 to 54224
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   vendor_id        54225 non-null   int64  
 1   pickup_datetime  54225 non-null   object  
 2   dropoff_datetime 54225 non-null   object  
 3   passenger_count  53015 non-null   float64 
 4   trip_distance    54225 non-null   float64 
 5   rate_code         53015 non-null   float64 
 6   store_and_fwd_flag 53015 non-null   object  
 7   payment_type     54225 non-null   int64  
 8   fare_amount      54225 non-null   float64 
 9   extra            54225 non-null   float64 
 10  mta_tax          54225 non-null   float64 
 11  tip_amount       54225 non-null   float64 
 12  tolls_amount     54225 non-null   float64 
 13  imp_surcharge   54225 non-null   float64 
 14  airport_fee      53015 non-null   float64 
 15  total_amount     54225 non-null   float64 
 16  pickup_location_id 54225 non-null   int64  
 17  dropoff_location_id 54225 non-null   int64  
 18  data_file_year   54225 non-null   int64  
 19  data_file_month  54225 non-null   int64  
dtypes: float64(11), int64(6), object(3)
memory usage: 8.3+ MB
```

Jumlah Missing Value Data Trips

```
[9] 1 trips.isna().sum().sort_values(ascending=False)
```

	passenger_count	1210
rate_code	1210	
store_and_fwd_flag	1210	
airport_fee	1210	
vendor_id	0	
tolls_amount	0	
data_file_year	0	
dropoff_location_id	0	
pickup_location_id	0	
total_amount	0	
imp_surcharge	0	
mta_tax	0	
tip_amount	0	
pickup_datetime	0	
extra	0	
fare_amount	0	
payment_type	0	
trip_distance	0	
dropoff_datetime	0	
data_file_month	0	
	dtype: int64	

Kemudian, Hapus Missing Value
Data Trips

Data Cleaning

Ubah tipe data ke datetime

```
1 trips.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 53015 entries, 0 to 54224
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   vendor_id        53015 non-null   object  
 1   pickup_datetime  53015 non-null   datetime64[ns]
 2   dropoff_datetime 53015 non-null   datetime64[ns]
 3   passenger_count  53015 non-null   int64  
 4   trip_distance    53015 non-null   float64 
 5   rate_code         53015 non-null   float64 
 6   store_and_fwd_flag 53015 non-null   object  
 7   payment_type     53015 non-null   int64  
 8   fare_amount      53015 non-null   float64 
 9   extra             53015 non-null   float64 
 10  mta_tax           53015 non-null   float64 
 11  tip_amount       53015 non-null   float64 
 12  tolls_amount     53015 non-null   float64 
 13  imp_surcharge   53015 non-null   float64 
 14  airport_fee      53015 non-null   float64 
 15  total_amount     53015 non-null   float64 
 16  pickup_location_id 53015 non-null   int64  
 17  dropoff_location_id 53015 non-null   int64  
dtypes: datetime64[ns](2), float64(10), int64(4), object(2)
memory usage: 7.7+ MB
```

Mengecek Data Duplikat

```
[11] 1 trips.duplicated().sum()
```

0

Exploratory Data Analysis (EDA)

...

Diagram Batang Jumlah Penumpang

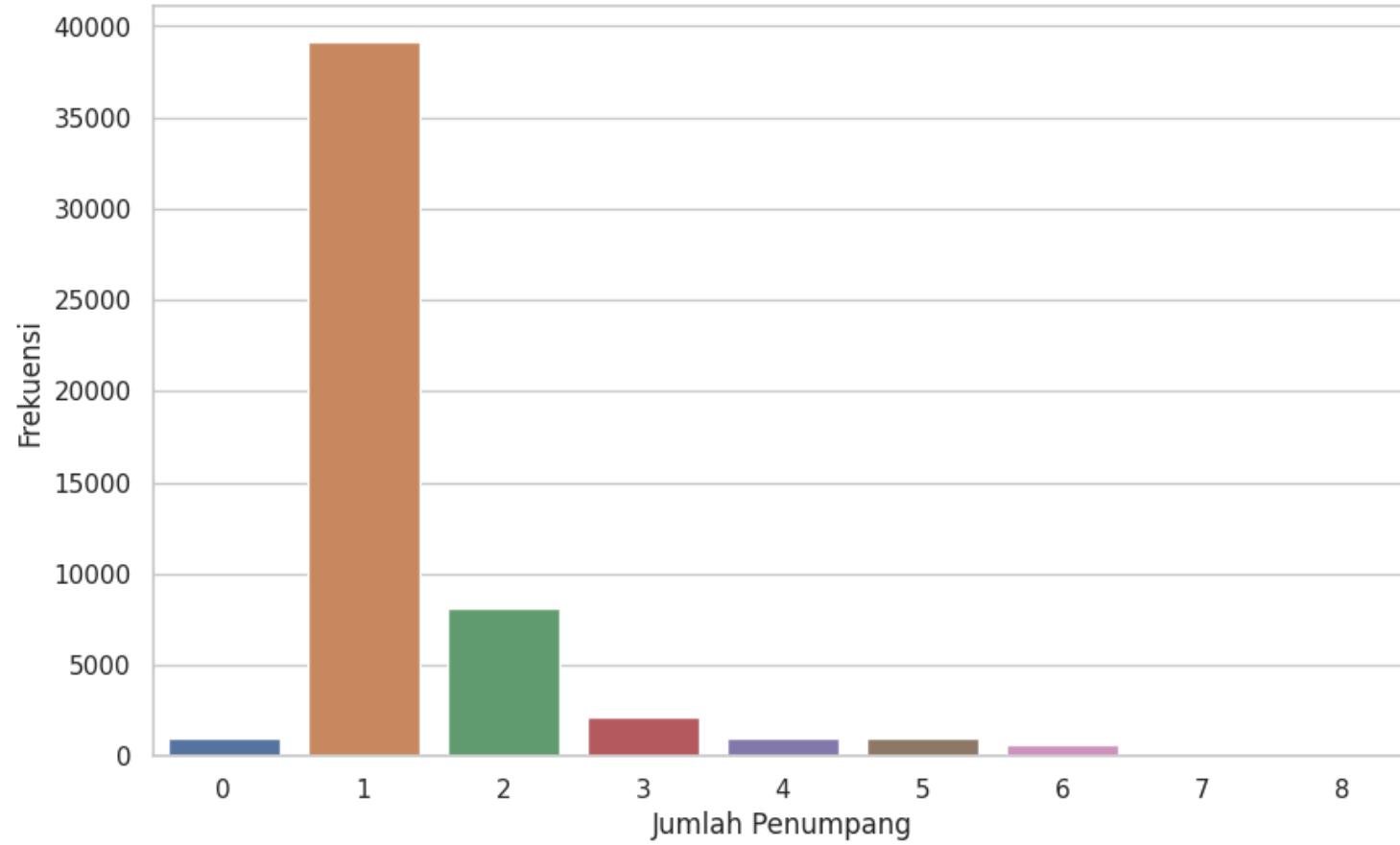
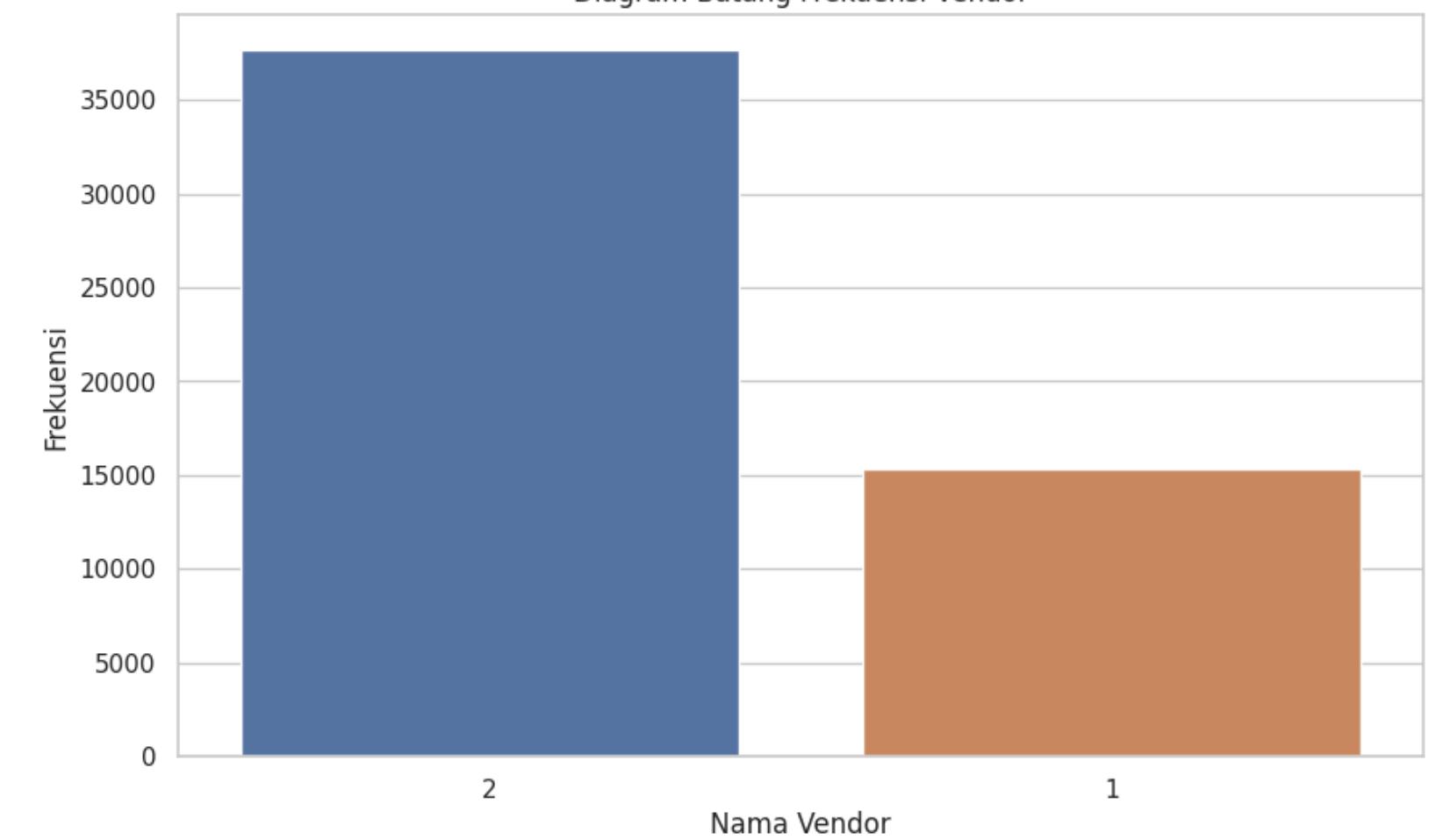


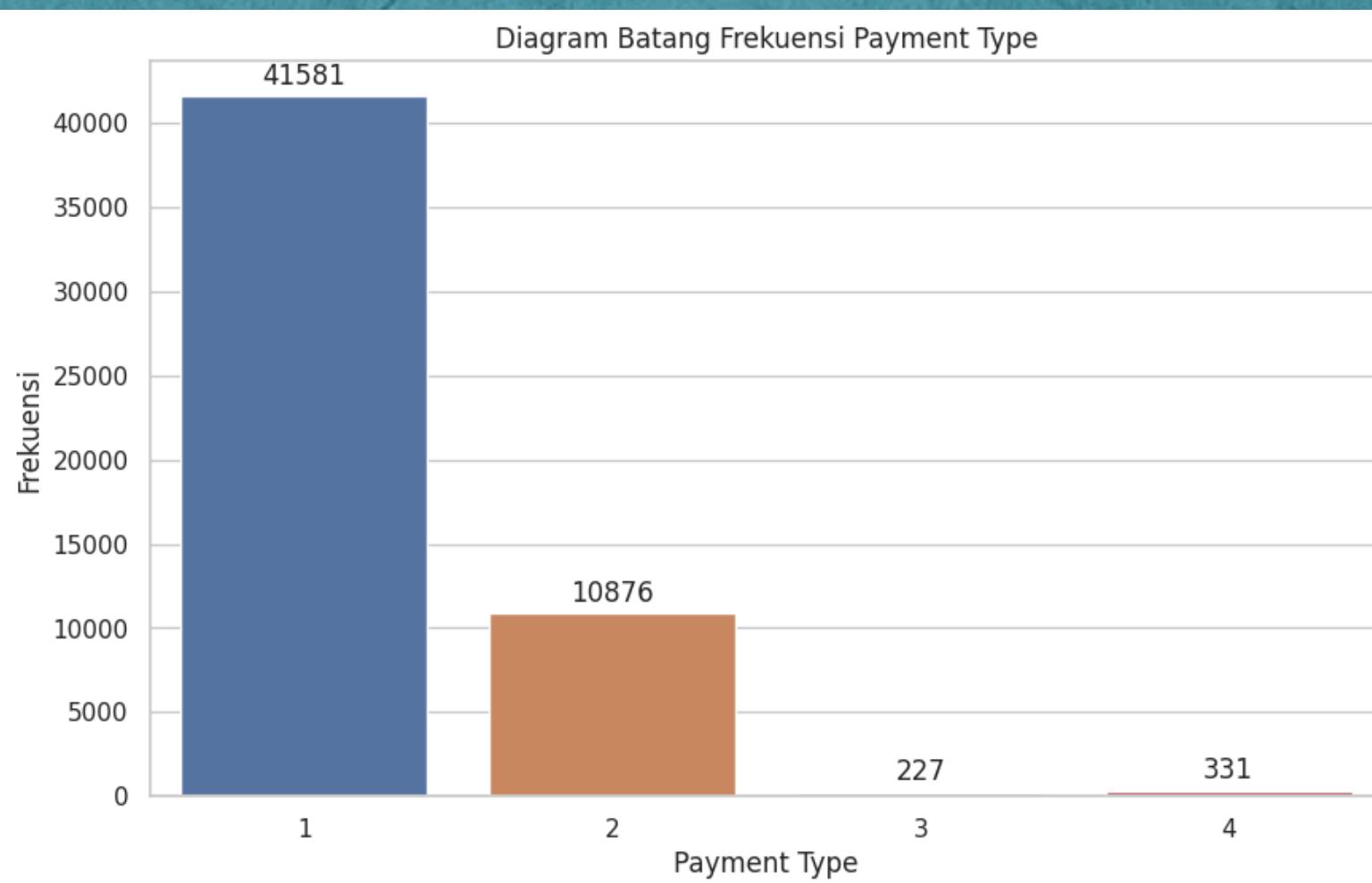
Diagram Batang Frekuensi Vendor



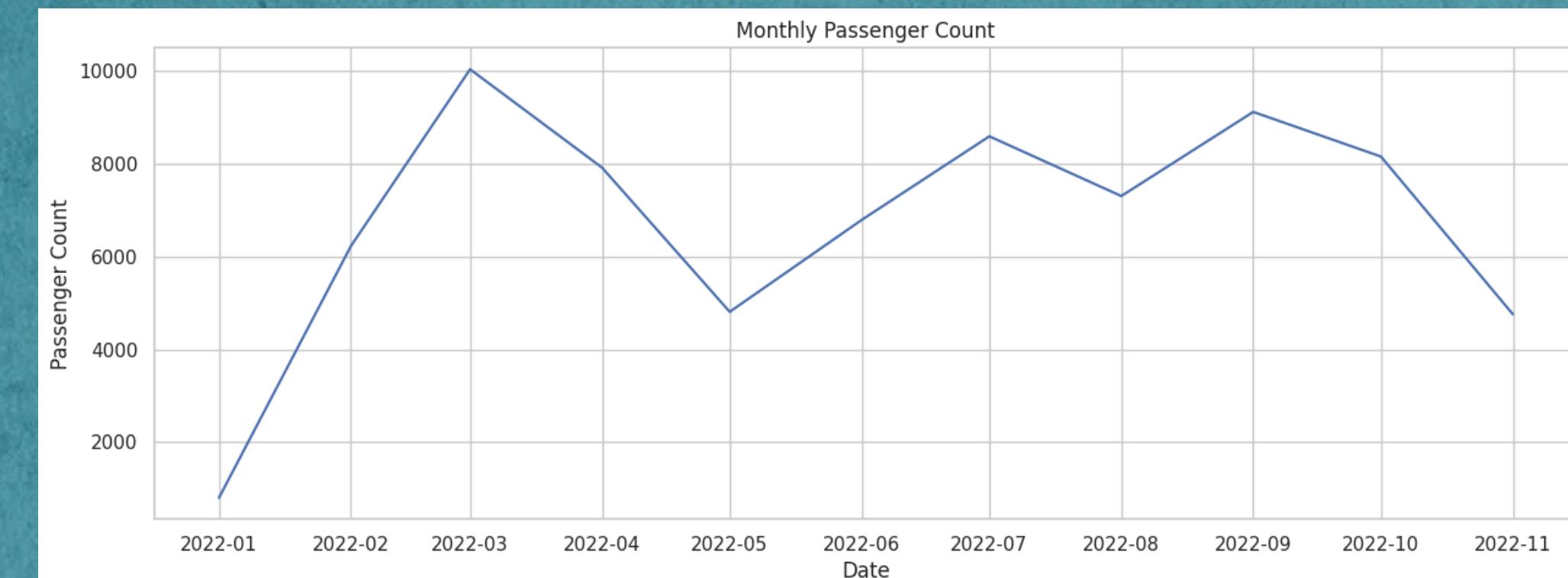
Jumlah frekuensi paling banyak yaitu penumpang nya hanya berjumlah 1 orang

Vendor paling banyak yaitu VeriFone Inc

Exploratory Data Analysis (EDA)

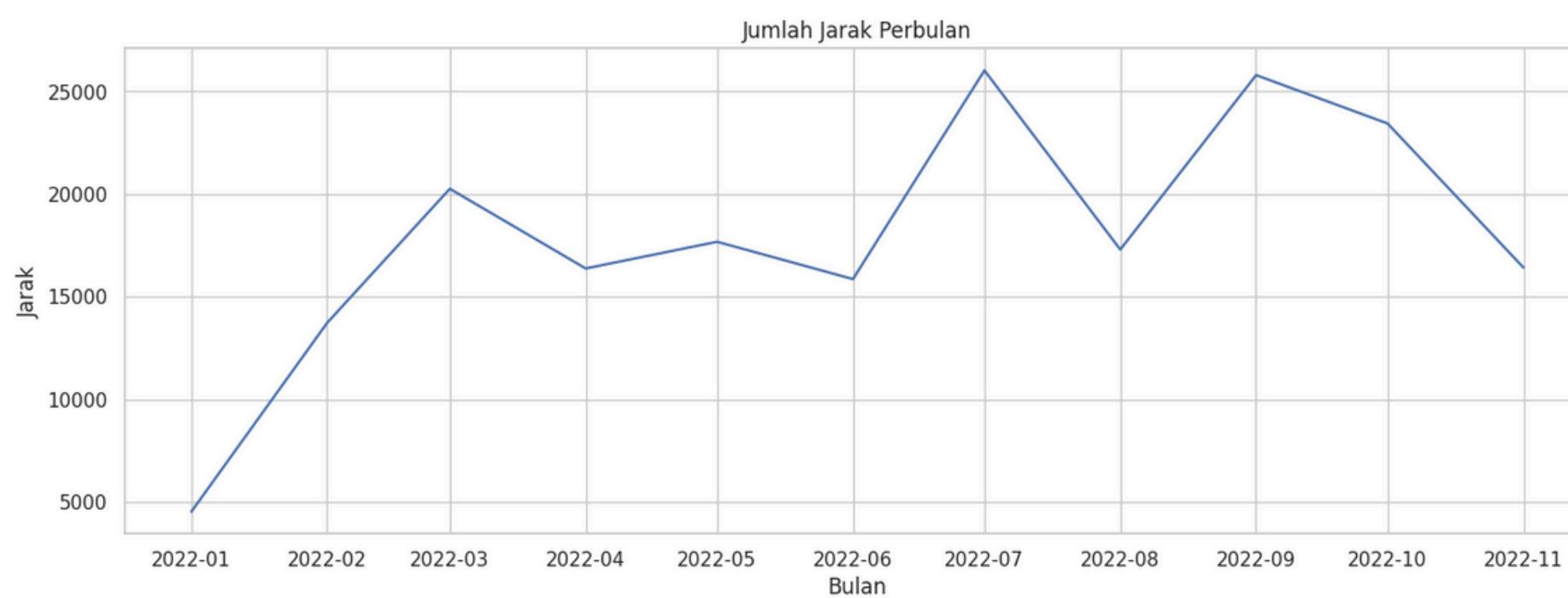


Berdasarkan grafik di atas bahwa penumpang kebanyakan membayar memakai credit card dan paling sedikit yaitu gratis

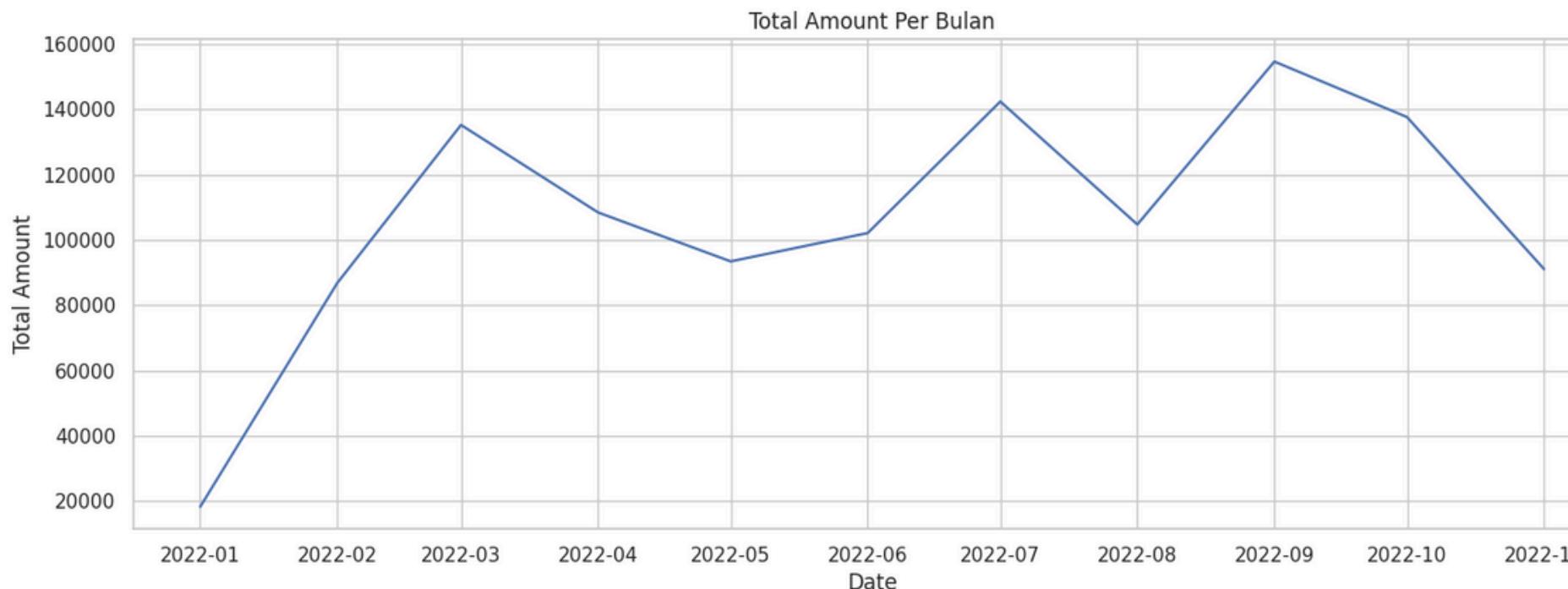


Berdasarkan grafik di atas bahwa penumpang paling banyak yaitu pada bulan Maret dan grafiknya masih fluktuatif

Exploratory Data Analysis (EDA)

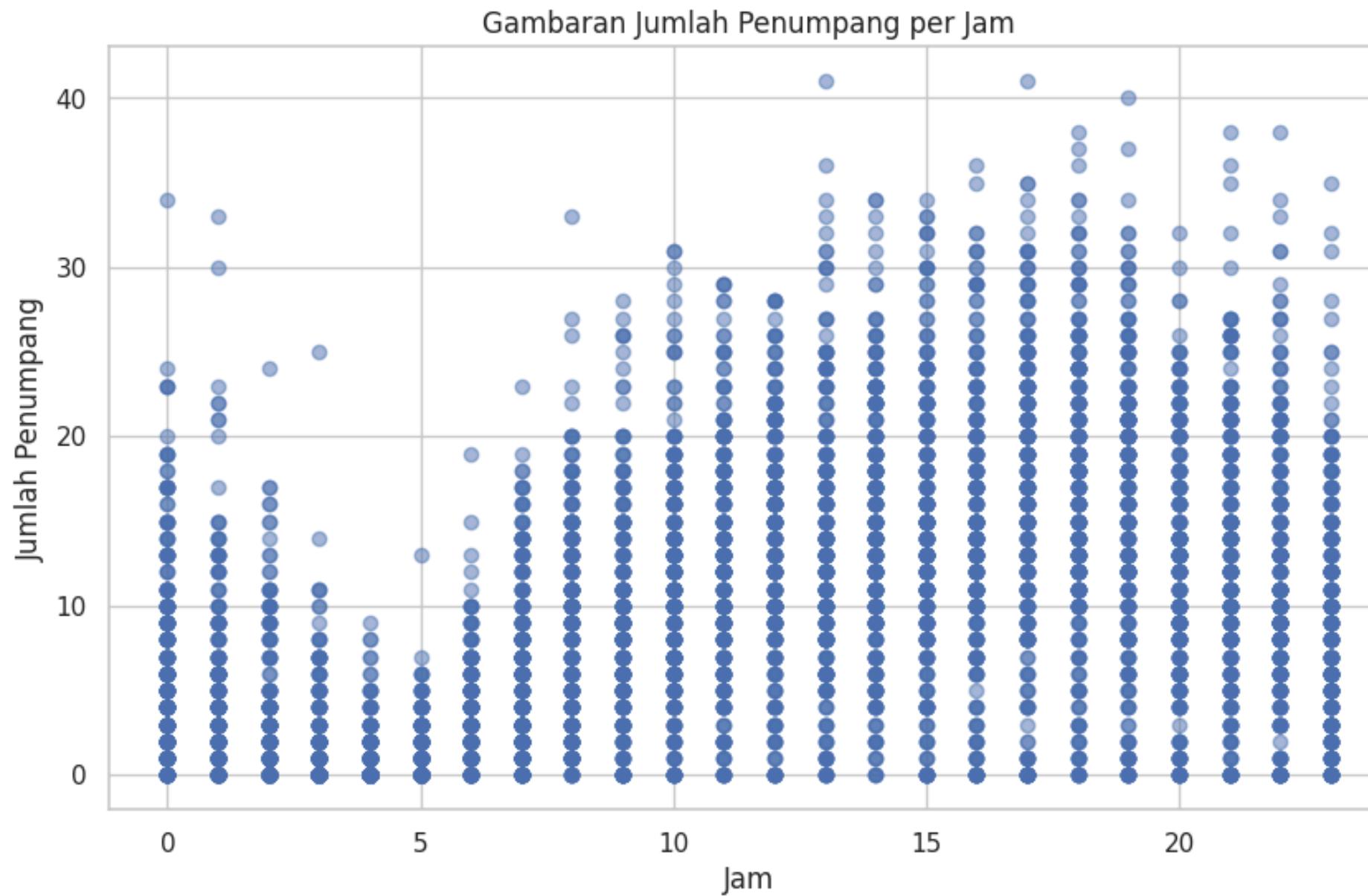


Jarak Tempuh yang paling banyak ditempuh yaitu pada bulan Juli dan September, paling sedikit untuk taxi jarak tempuh dalam sebulan yaitu pada bulan Februari



Grafik di samping menunjukkan bahwa setiap bulan selama Tahun 2022 grafiknya fluktuatif

Exploratory Data Analysis (EDA)



Berdasarkan grafik di atas, jam paling sibuk pickup penumpang yaitu sekitar jam 14-19.

Modelling

Library

```
1 import pandas as pd  
2 import numpy as np  
3 import statsmodels.api as sm
```

Model ARIMA

```
1 # Pilih deret waktu dari data historis  
2 historical_data = train['passenger_count']  
3  
4 # Fit model ARIMA ke data historis  
5 # Gunakan model ARIMA dengan p = 1, d = 0, dan q = 1  
6 model = sm.tsa.ARIMA(historical_data, order=(1, 0, 1))  
7 results = model.fit()  
8  
9 # Lakukan prediksi untuk rentang waktu yang diinginkan  
10 predictions = results.predict(start=start_date, end=end_date, dynamic=True)  
11 predicted_data['predicted_passenger_count'] = predictions
```

Penentuan Rentang Prediksi

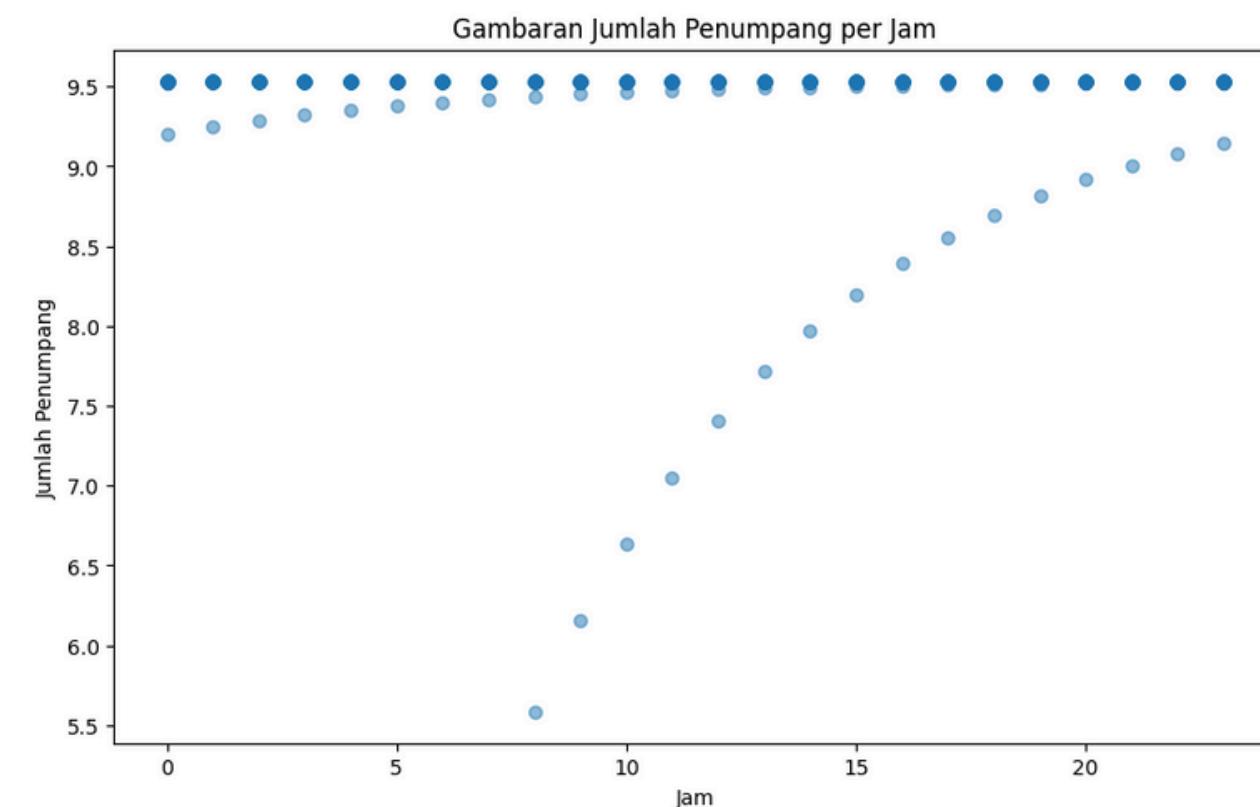
```
1 # Memprediksi dari 2022-09-26 08:00:00+00:00 sampai 2022-11-30 23:00:00+00:00  
2 start_date = pd.to_datetime('2022-09-26 08:00:00+00:00')  
3 end_date = pd.to_datetime('2022-11-30 23:00:00+00:00')  
4 predicted_index = pd.date_range(start_date, end_date, freq='H')  
5  
6 # Buat DataFrame kosong untuk menyimpan prediksi  
7 predicted_data = pd.DataFrame(index=predicted_index)
```

Menentukan nilai p, d, q berdasarkan hasil literatur karena keterbatasan waktu, maka langsung ditentukan p = 1, d = 0, dan q = 1

Hasil Prediksi

1 predicted_data
predicted_passenger_count
2022-09-26 08:00:00+00:00 5.585620
2022-09-26 09:00:00+00:00 6.151727
2022-09-26 10:00:00+00:00 6.636643
2022-09-26 11:00:00+00:00 7.052013
2022-09-26 12:00:00+00:00 7.407812
...
2022-11-30 19:00:00+00:00 9.532851
2022-11-30 20:00:00+00:00 9.532851
2022-11-30 21:00:00+00:00 9.532851
2022-11-30 22:00:00+00:00 9.532851
2022-11-30 23:00:00+00:00 9.532851
1576 rows × 1 columns

Hasil prediksi ini belum dilakukan evaluasi model karena keterbatasan waktu yang tidak cukup untuk melakukan evaluasi model, maka langsung coba submit ke kaggle untuk melihat hasilnya. Hasil di kaggle yaitu 2.86742 untuk 70% data test



Prediksi kedepannya bahwa jumlah penumpang akan terus meningkat di jam 10 keatas

Rekomendasi

- Perusahaan NYC TLC bisa pertahankan kerjasama dengan vendor VeriFone Inc karena vendor tersebut paling banyak mengindikasikan penyedia LPEP
- Perusahaan bisa lebih variasikan transaksi pembayaran non tunai/credit card agar penumpang lebih mudah untuk membayar
- Perusahaan NYC TLC bisa mulai dari tanggal 26 September sampai 30 November bisa lebih ditingkatkan lagi jumlah taxi di jam 10 keatas karena berdasarkan prediksi, jam tersebut mengalami peningkatan jumlah penumpang.





Thank you!

