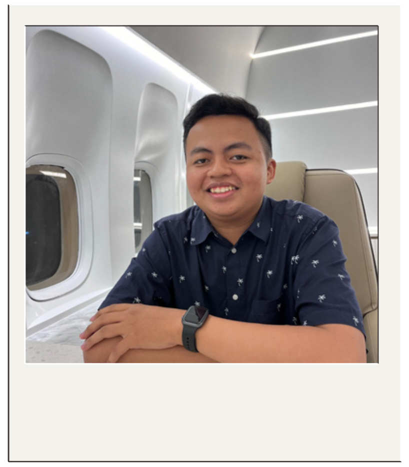
Fraud Detection

Margareth Hamilton

Friday, October 18th 2024

Meet the Team



Alwy Bathia R.

Background and EDA

Daniel Machsimus L.

Feature Engineering

Jason Hermawan

Model Comparison and Conclusion

Background & Problem Statement

Fraud detection adalah proses untuk **mengidentifikasi** aktivitas atau transaksi yang **mencurigakan** dan **berpotensi menipu**, yang **tidak sesuai dengan perilaku normal** atau harapan yang ditetapkan dalam suatu sistem.

Background & Problem Statement

Global Economic Crime Survey 2024

Key findings

55%

Reported procurement fraud is a widespread concern in their country, yet a minority are using available tools to identify or combat it.

42%

Either don't have a third-party risk management programme or don't do any form of risk scoring as part of their programme.

> "Companies have an opportunity to build compliance programmes that support businesses in maintaining trust and building resilience, contributing to the confidence to transform, invest and grow. With the right data and insights, risks can be taken with confidence."
>
> **Ryan Murphy,** Global & US Forensics Leader, Partner, PwC US

Background & Problem Statement

Ojol di Makassar Kuras ATM Korban Berisi Rp 36 Juta Ditangkap

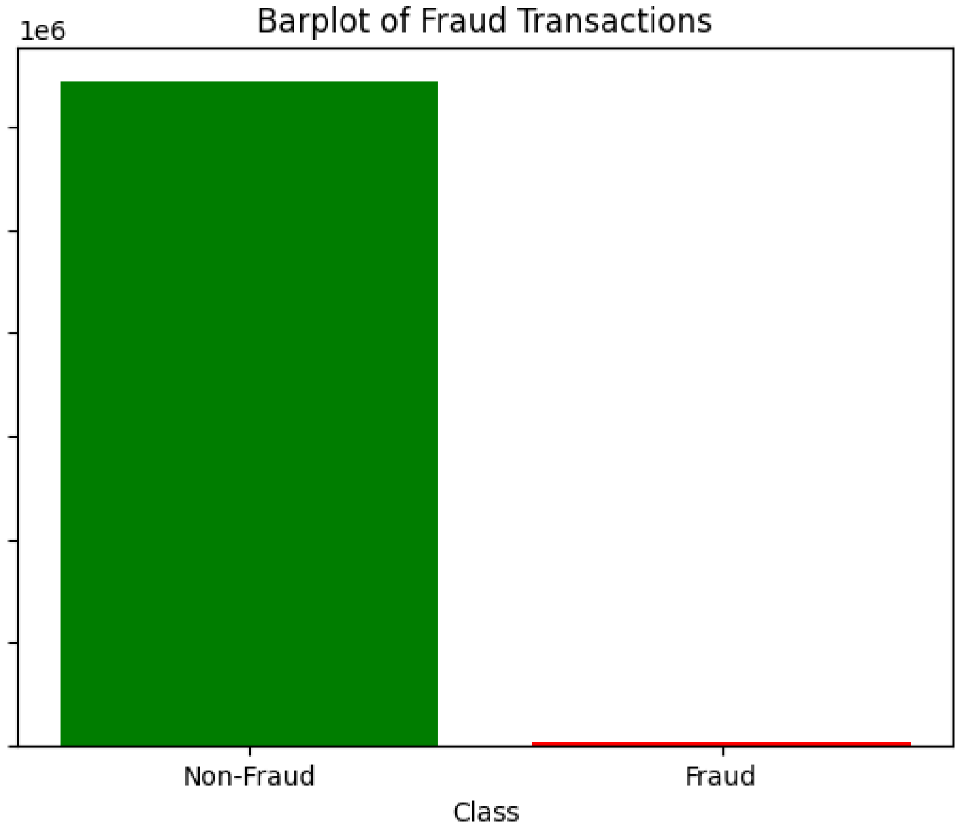
Reinhard Soplantila - detikSulsel

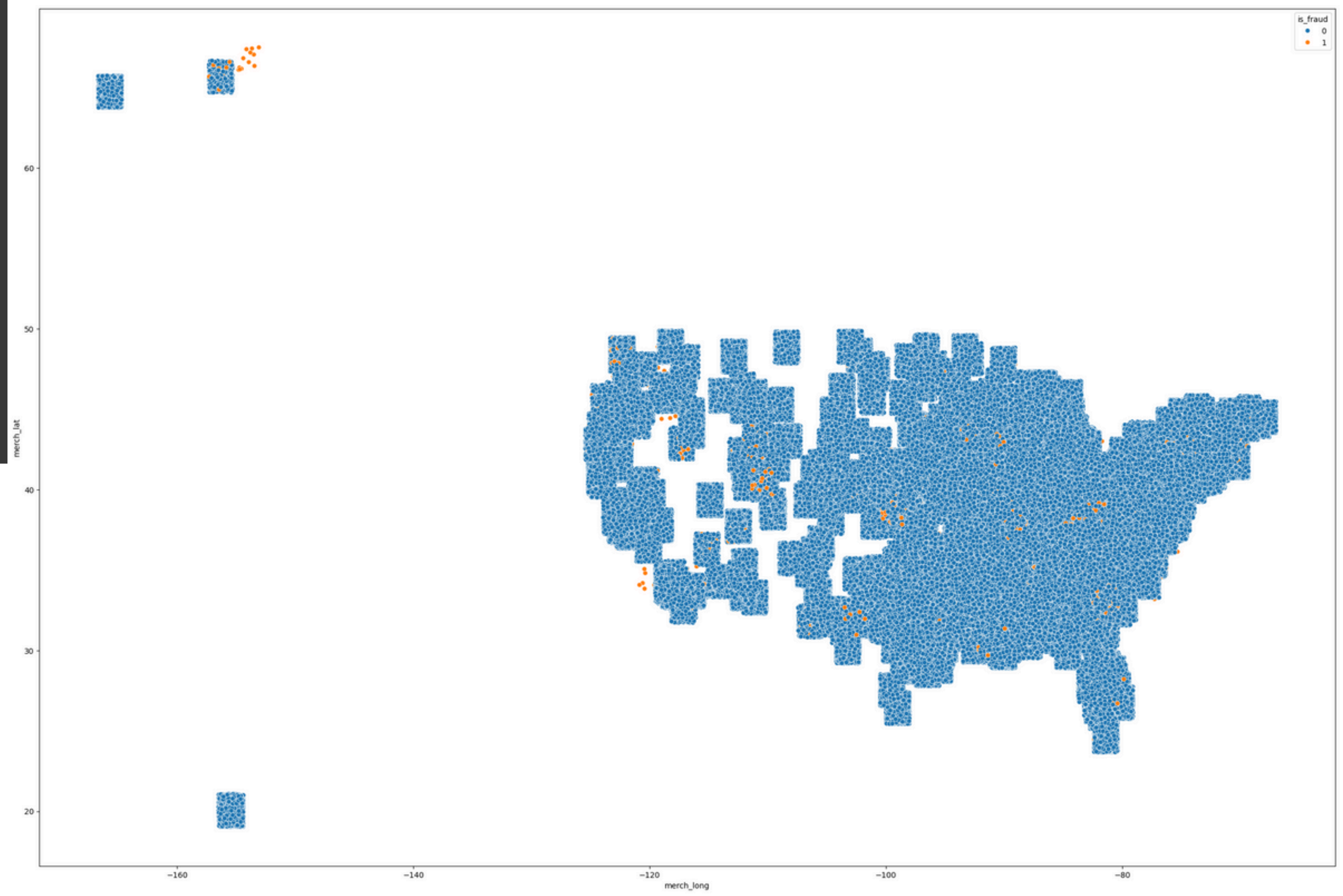
Rabu, 16 Okt 2024 21:31 WIB

Exploratory Data Analysis

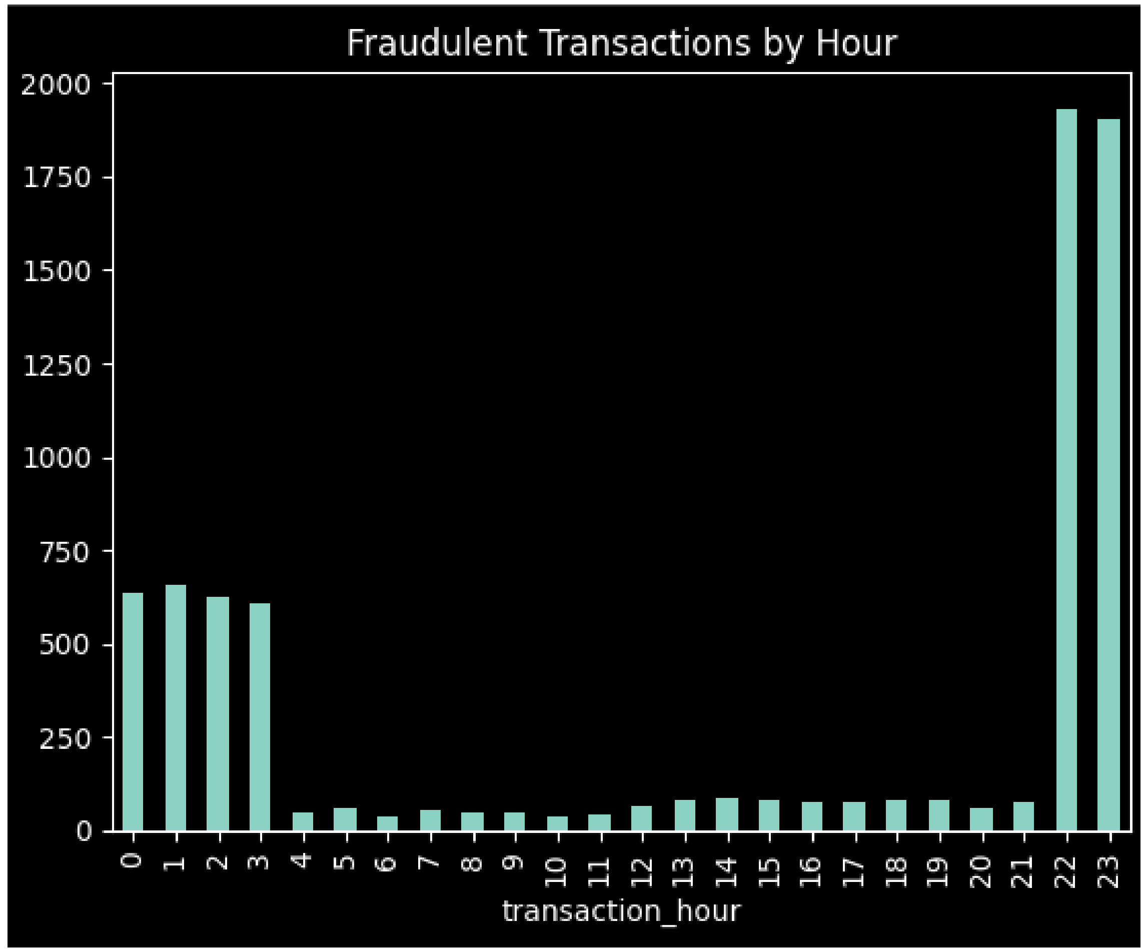
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 23 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   Unnamed: 0             1296675 non-null  int64
 1   trans_date_trans_time  1296675 non-null  object
 2   cc_num                 1296675 non-null  int64
 3   merchant               1296675 non-null  object
 4   category               1296675 non-null  object
 5   amt                    1296675 non-null  float64
 6   first                  1296675 non-null  object
 7   last                   1296675 non-null  object
 8   gender                 1296675 non-null  object
 9   street                 1296675 non-null  object
 10  city                   1296675 non-null  object
 11  state                  1296675 non-null  object
 12  zip                    1296675 non-null  int64
 13  lat                    1296675 non-null  float64
 14  long                   1296675 non-null  float64
 15  city_pop               1296675 non-null  int64
 16  job                    1296675 non-null  object
 17  dob                    1296675 non-null  object
 18  trans_num              1296675 non-null  object
 19  unix_time              1296675 non-null  int64
 20  merch_lat              1296675 non-null  float64
 21  merch_long             1296675 non-null  float64
 22  is_fraud               1296675 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 227.5+ MB
```

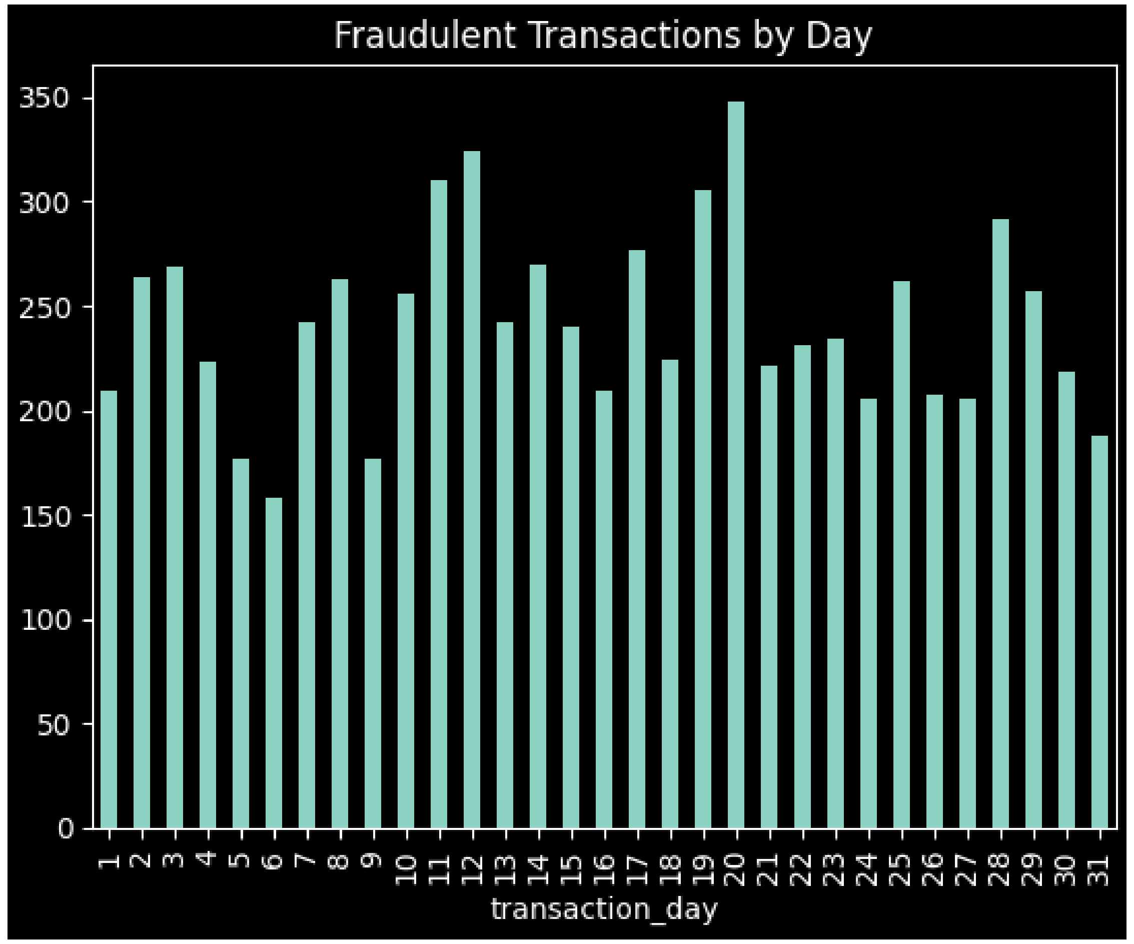
```
Unnamed: 0: 1296675
trans_date_trans_time: 1274791
cc_num: 983
merchant: 693
category: 14
amt: 52928
first: 352
last: 481
gender: 2
street: 983
city: 894
state: 51
zip: 970
lat: 968
long: 969
city_pop: 879
job: 494
dob: 968
trans_num: 1296675
unix_time: 1274823
merch_lat: 1247805
merch_long: 1275745
is_fraud: 2
```

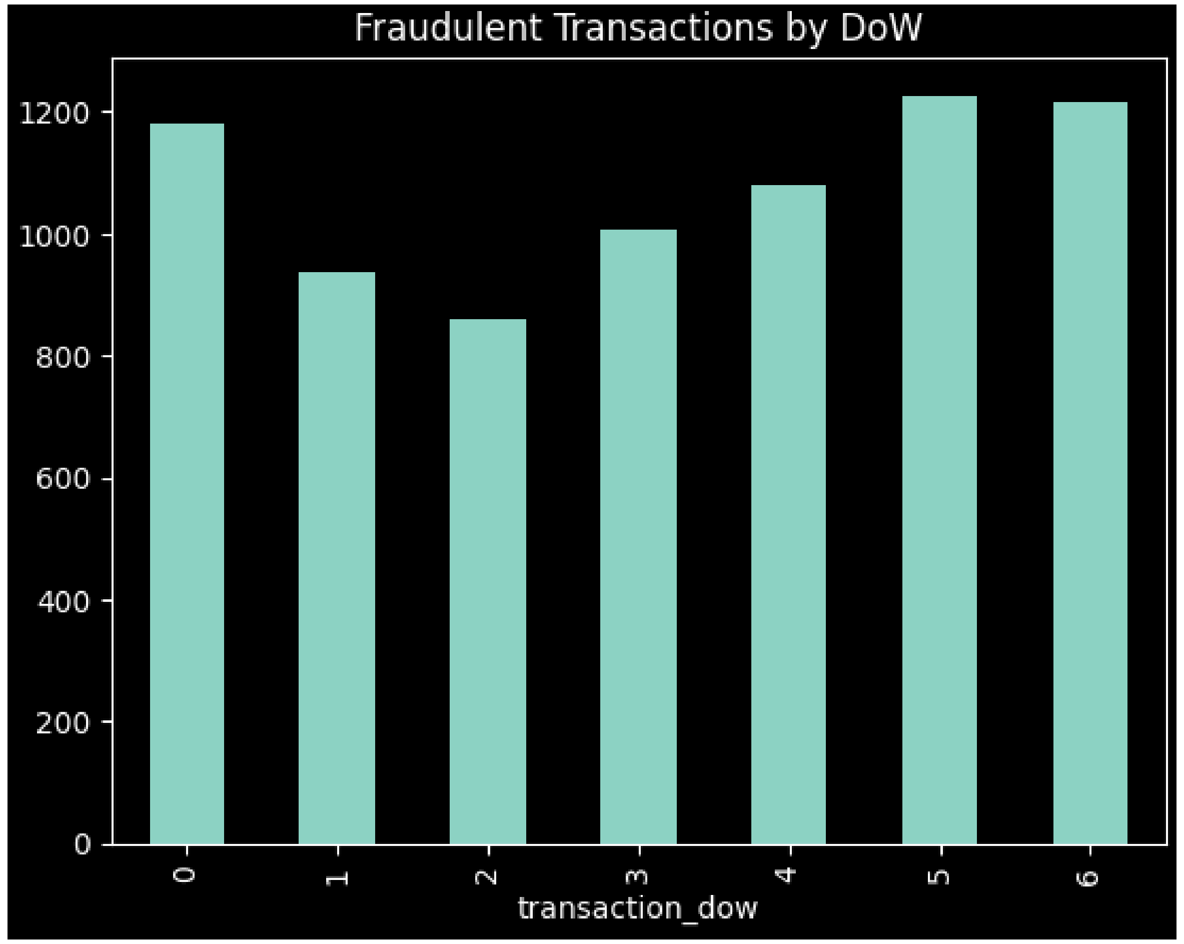




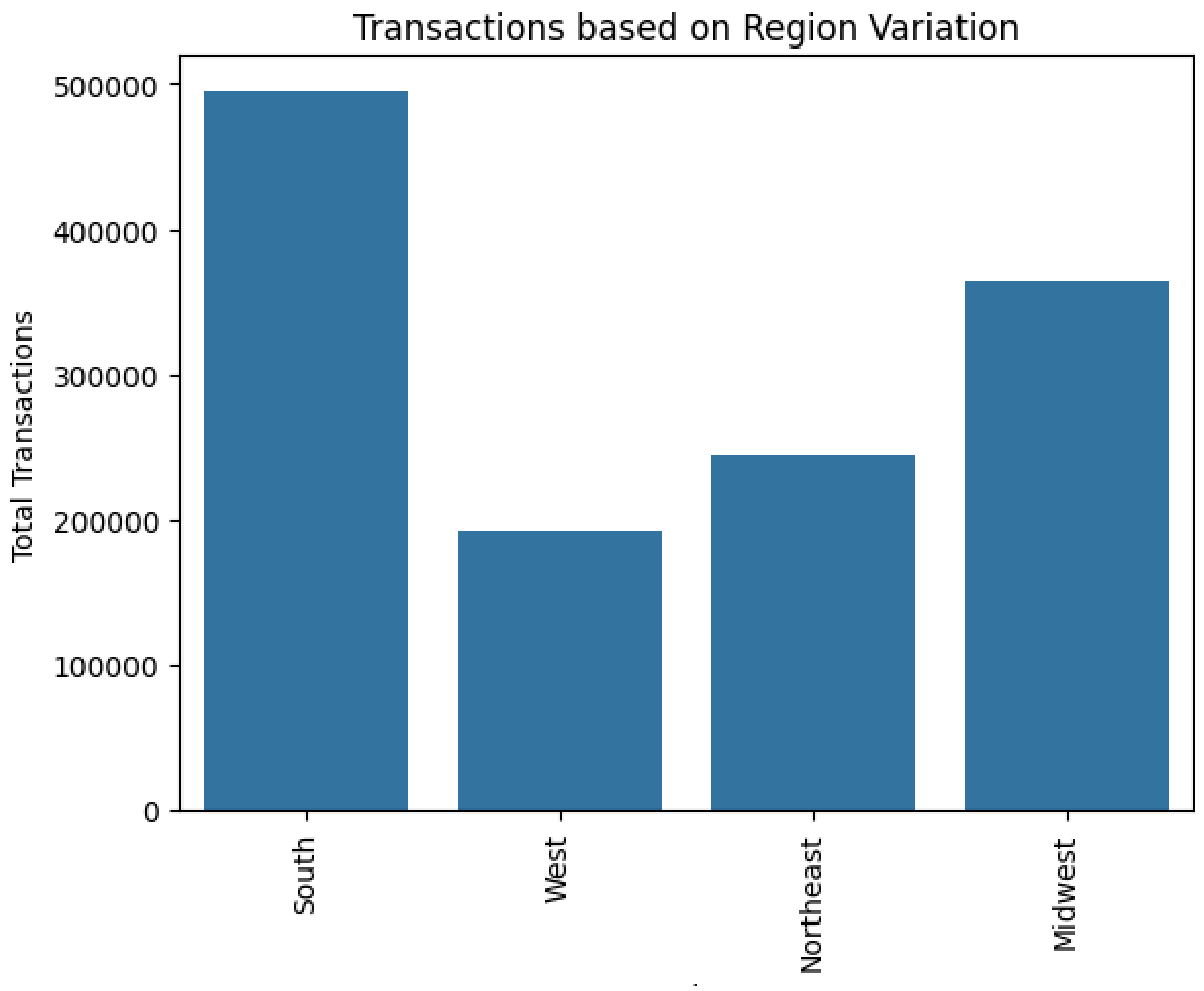
Exploratory Data Analysis

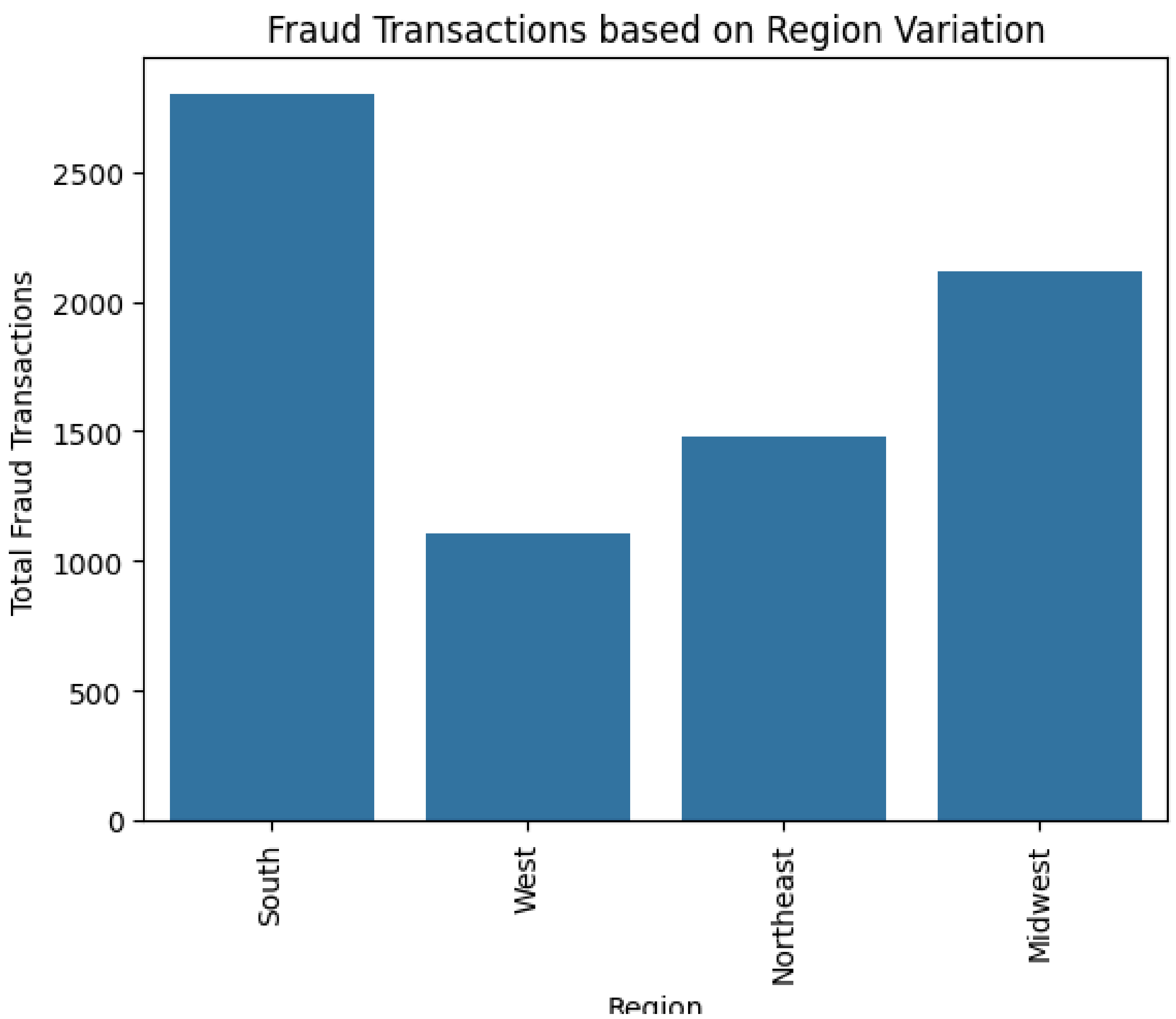






Exploratory Data Analysis





Feature Engineering Addition

Addition: trans_year, trans_month, trans_hour, trans_day

```
dft3['trans_date']=dft3.trans_date_trans_time.dt.date
dft3['trans_time']=dft3.trans_date_trans_time.dt.time
dft3.trans_date = pd.to_datetime(dft3['trans_date'],format='%Y-%m-%d')
```

```
dft3['trans_year']=dft3.trans_date.dt.year
dft3['trans_month']=dft3.trans_date.dt.month
dft3['trans_hour']=dft3.trans_date_trans_time.dt.hour
dft3['trans_day']=dft3.trans_date.dt.day

dft3.head(2)
```

Addition: age

```
from datetime import datetime
dft3['age'] = dft3['trans_date_trans_time'].dt.year - dft3['dob'].dt.year
dft3['age'] -= ((dft3['trans_date_trans_time'].dt.month < dft3['dob'].dt.month) |
               ((dft3['trans_date_trans_time'].dt.month == dft3['dob'].dt.month) &
                (dft3['trans_date_trans_time'].dt.day < dft3['dob'].dt.day))).astype(int)
```

Feature Engineering Addition

Addition: hist_trans_30d

```
df_hist_trans_30d = \
    dft3 \
    .groupby(['cc_num'])['val_for_agg']\
    .rolling('30D')\
    .count()\
    .shift()\
    .reset_index()\
    .fillna(0)

df_hist_trans_30d.columns = ['cc_num','trans_date_trans_time','hist_trans_30d']
```

Addition: hist_amt_avg_30d

```
df_hist_amt_avg_30d = \
    dft3 \
    .groupby(['cc_num'])['amt']\
    .rolling('30D')\
    .mean()\
    .shift(1)\
    .reset_index()\
    .fillna(0)

df_hist_amt_avg_30d.columns = ['cc_num','trans_date_trans_time','hist_amt_avg_30d']
```

Skema beberapa *feature engineering* di sini menggunakan ***aggregation model***, yaitu **menjumlahkan fitur** selama periode tertentu [4].

[1] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams, "Transaction aggregation as a strategy for credit card fraud detection," Data Mining and Knowledge Discovery, vol. 18, no. 1, pp. 30–55, Jul. 2008, doi: https://doi.org/10.1007/s10618-008-0116-z.
[4] I. Sadgali, N. Sael, and F. Benabbou, "Adaptive Model for Credit Card Fraud Detection," International Journal of Interactive Mobile Technologies (iJIM), vol. 14, no. 03, p. 54, Feb. 2020, doi: https://doi.org/10.3991/ijim.v14i03.11763.

Feature Engineering Addition

Addition: hist_trans_24h

```
df_hist_trans_24h = \
    dft3 \
    .groupby(['cc_num'])['val_for_agg']\
    .rolling('24H')\
    .count()\
    .shift()\
    .reset_index()\
    .fillna(0)

df_hist_trans_24h.columns = ['cc_num','trans_date_trans_time','hist_trans_24h']
```

Addition: hist_amt_avg_24h

```
df_hist_amt_avg_24h = \
    dft3 \
    .groupby(['cc_num'])['amt']\
    .rolling('30D')\
    .mean()\
    .shift(1)\
    .reset_index()\
    .fillna(0)

df_hist_amt_avg_24h.columns = ['cc_num','trans_date_trans_time','hist_amt_avg_24h']
```

Frekuensi ataupun volume transaksi itu penting untuk fraud detection [1].

Deteksinya bisa menggunakan *transaction-level classification*, misalnya dengan menjumlahkan **frekuensi transaksi** selama satu hari terakhir ataupun seminggu terakhir [1]. Jha et. al. membuat feature engineering yang mengagregatkan jumlah transaksi selama 30 hari terakhir [2].

[1] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams, “Transaction aggregation as a strategy for credit card fraud detection,” Data Mining and Knowledge Discovery, vol. 18, no. 1, pp. 30–55, Jul. 2008, doi: https://doi.org/10.1007/s10618-008-0116-z.
[2] S. Jha, M. Guillen, and J. Christopher Westland, “Employing transaction aggregation strategy to detect credit card fraud,” Expert Systems with Applications, vol. 39, no. 16, pp. 12650–12657, Nov. 2012, doi: https://doi.org/10.1016/j.eswa.2012.05.018.

Feature Engineering **Addition**

Addition: distance_cust_store

```
from math import radians, cos, sin, asin, sqrt
```

```
def haversine_distance(lat, long, merch_lat, merch_long):

    # The math module contains a function named
    # radians which converts from degrees to radians.
    long = np.radians(long)
    merch_long = np.radians(merch_long)
    lat = np.radians(lat)
    merch_lat = np.radians(merch_lat)

    # Haversine formula
    dlon = merch_long - long
    dlat = merch_lat - lat
    a = np.sin(dlat / 2)**2 + np.cos(lat) * np.cos(merch_lat) * np.sin(dlon / 2)**2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))
    #c = 2 * np.arctan2(a**0.5, (1-a)**0.5)

    # Radius of earth in kilometers.
    r = 6371
    d=  c * r
    # calculate the result
    return round(d,2)
```

```
dft3['distance_cust_store']= dft3[['lat', 'long', 'merch_lat', 'merch_long']].apply(
                                lambda x:haversine_distance(x[0], x[1], x[2], x[3]), axis=1)
```

Fraud juga dapat dideteksi dengan indikator lain, yang disebut '***collision***' atau '***high velocity***' yang menggambarkan kejadian 2 atau lebih transaksi yang terjadi dalam kurun waktu yang hampir sama pada **lokasi geografis** yang **berjauhan** [1].

Pada data, ingin diekstrak **jarak** antara **pengguna kartu kredit** dengan ***merchant*** **(toko)** tempat ia bertransaksi menggunakan ***Haversian distance***.

Haversian distance = jarak dua titik pada suatu bidang bola jika diketahui posisi ***latitude*** dan ***longitude***-nya. Maria et. al. menggunakan Haversian distance untuk mengukur jarak antarfasilitas umum selain menggunakan Euclidean distance [3].

[1] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams, "Transaction aggregation as a strategy for credit card fraud detection," Data Mining and Knowledge Discovery, vol. 18, no. 1, pp. 30–55, Jul. 2008, doi: https://doi.org/10.1007/s10618-008-0116-z.
[3] E. Maria, E. Budiman, Haviluddin, and M. Taruk, "Measure distance locating nearest public facilities using Haversine and Euclidean Methods," Journal of Physics: Conference Series, vol. 1450, no. 1, p. 012080, Feb. 2020, doi: https://doi.org/10.1088/1742-6596/1450/1/012080.

Model Comparison

Pada bagian ini, kami mencoba membandingkan beberapa model untuk mencari model yang terbaik.

Berikut beberapa model yang kami Gunakan:

1. Random Forest
2. Logistic Regression
3. Adaboost

Random Forest

4.6.2 Evaluation

```
report_dec_tree = classification_report(y_test, rf_predictions)
print(report_dec_tree)
```

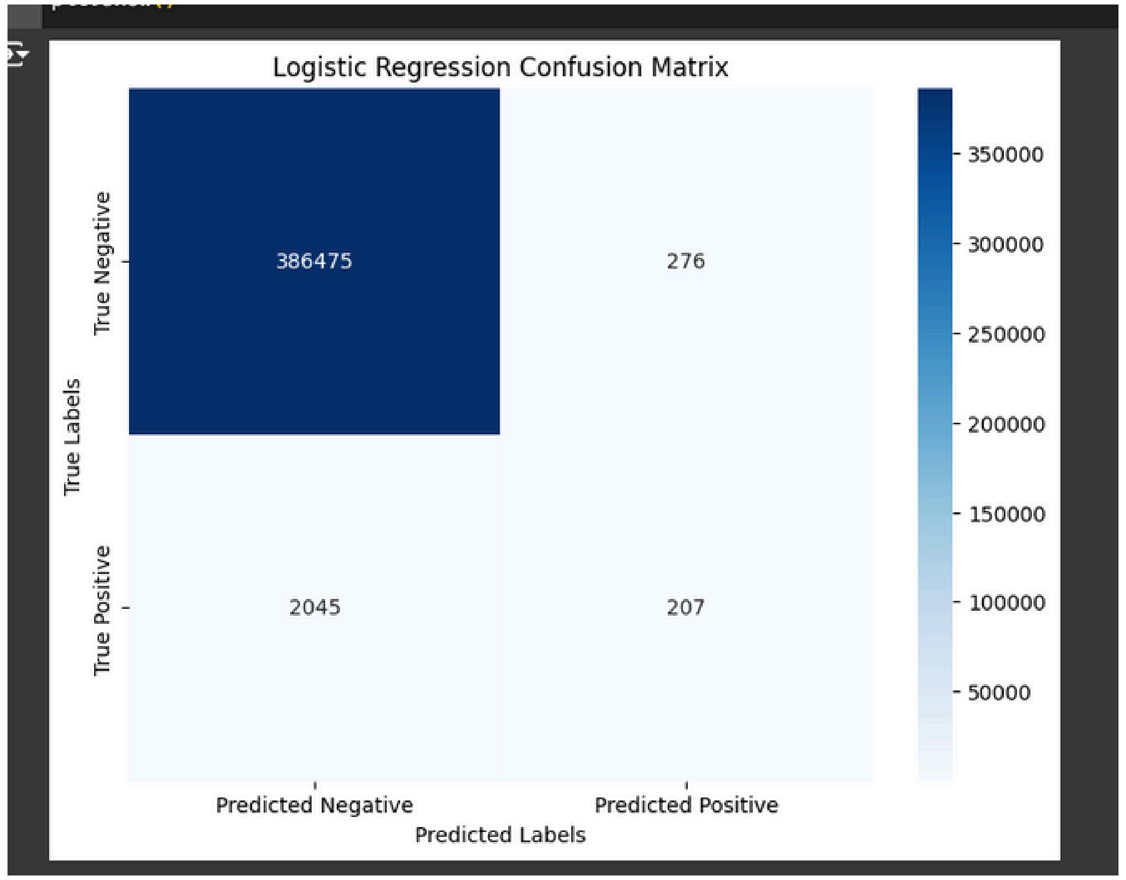
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    386751
           1       1.00      1.00      1.00      2252

    accuracy                           1.00    389003
   macro avg       1.00      1.00      1.00    389003
weighted avg       1.00      1.00      1.00    389003
```

Logistic Regression

4.5.2 Evaluation

```
report_log_reg = classification_report(y_test, log_reg_predictions)
print(report_log_reg)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00    386751
           1       0.43      0.09      0.15      2252

    accuracy                           0.99    389003
   macro avg       0.71      0.55      0.57    389003
weighted avg       0.99      0.99      0.99    389003
```



Adaboost (Decision tree)

4.4.2 Evaluation

```
from sklearn.metrics import classification_report
report_ran = classification_report(y_test, y_pred_ran)
print(report_ran)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    386751
           1       1.00      1.00      1.00      2252

    accuracy                           1.00    389003
   macro avg       1.00      1.00      1.00    389003
weighted avg       1.00      1.00      1.00    389003
```

Model Comparison

Dari berbagai model yang sudah dicoba, hasil terbaik ada di model **Random Forest** dan **Adaboost with Decision Tree**

*Ada penelitian yang mengatakan bahwa random forest adalah salah satu yang terbaik untuk mendeteksi fraud credit card [1].

[1] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams, "Transaction aggregation as a strategy for credit card fraud detection," Data Mining and Knowledge Discovery, vol. 18, no. 1, pp. 30–55, Jul. 2008, doi: https://doi.org/10.1007/s10618-008-0116-z

Conclusion

Kesimpulan yang dapat diambil dari projek Fraud Detection ini adalah, **AI dapat sangat membantu dalam mendeteksi adanya potensi Fraud.**

Sehingga kita dapat melakukan tindakan-tindakan yang diperlukan untuk mencegah atau mengatasi hal-hal yang tidak diinginkan seperti ini.

THANK YOU

Results LSTM Automotive

LSTM

```
MSE Y1: 10.2609963916165
RMSE Y1: 3.20327900620856
MAE Y1: 2.520929156144602

MSE Y2: 0.00993773791228817
RMSE Y2: 0.0996882034760792
MAE Y2: 0.02480681628609697
```

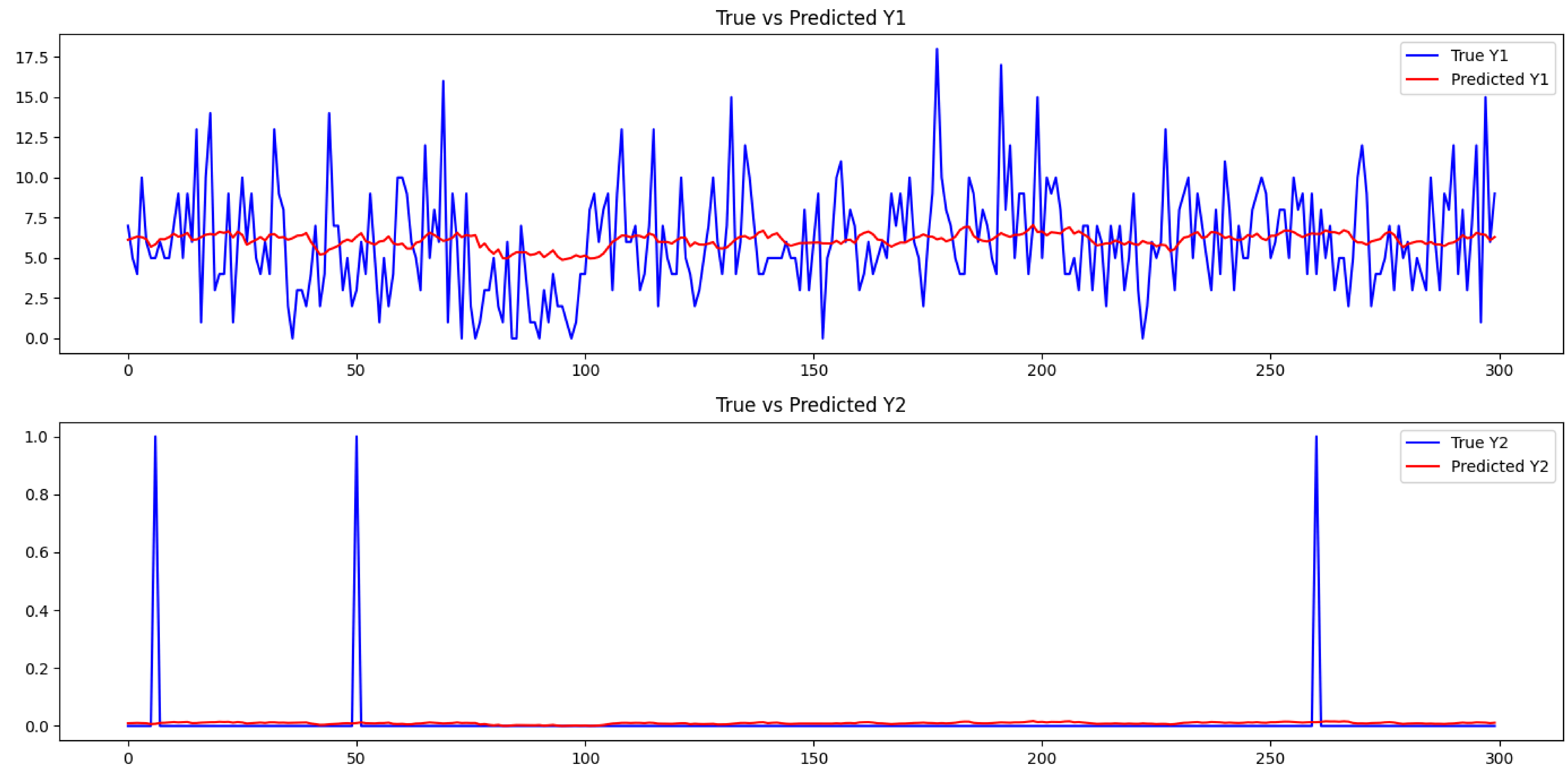
STACKED LSTM

```
MSE Y1: 10.429980638230793
RMSE Y1: 3.2295480547950968
MAE Y1: 2.5710357999801636

MSE Y2: 0.009897322877822181
RMSE Y2: 0.09948528975593418
MAE Y2: 0.019398317643596648
```

GRU

```
MSE Y1: 10.425120469443348
RMSE Y1: 3.2287955137238638
MAE Y1: 2.502937143643697

MSE Y2: 0.010086169353222197
RMSE Y2: 0.10042992259890574
MAE Y2: 0.014172124973071428
```

Results LSTM Automotive



Results LSTM GROCERY 1

LSTM

```
MSE Y1: 352548.77259261766
RMSE Y1: 593.758176863795
MAE Y1: 446.124541015625

MSE Y2: 230.02043617361798
RMSE Y2: 15.166424633829095
MAE Y2: 9.69217856725057
```

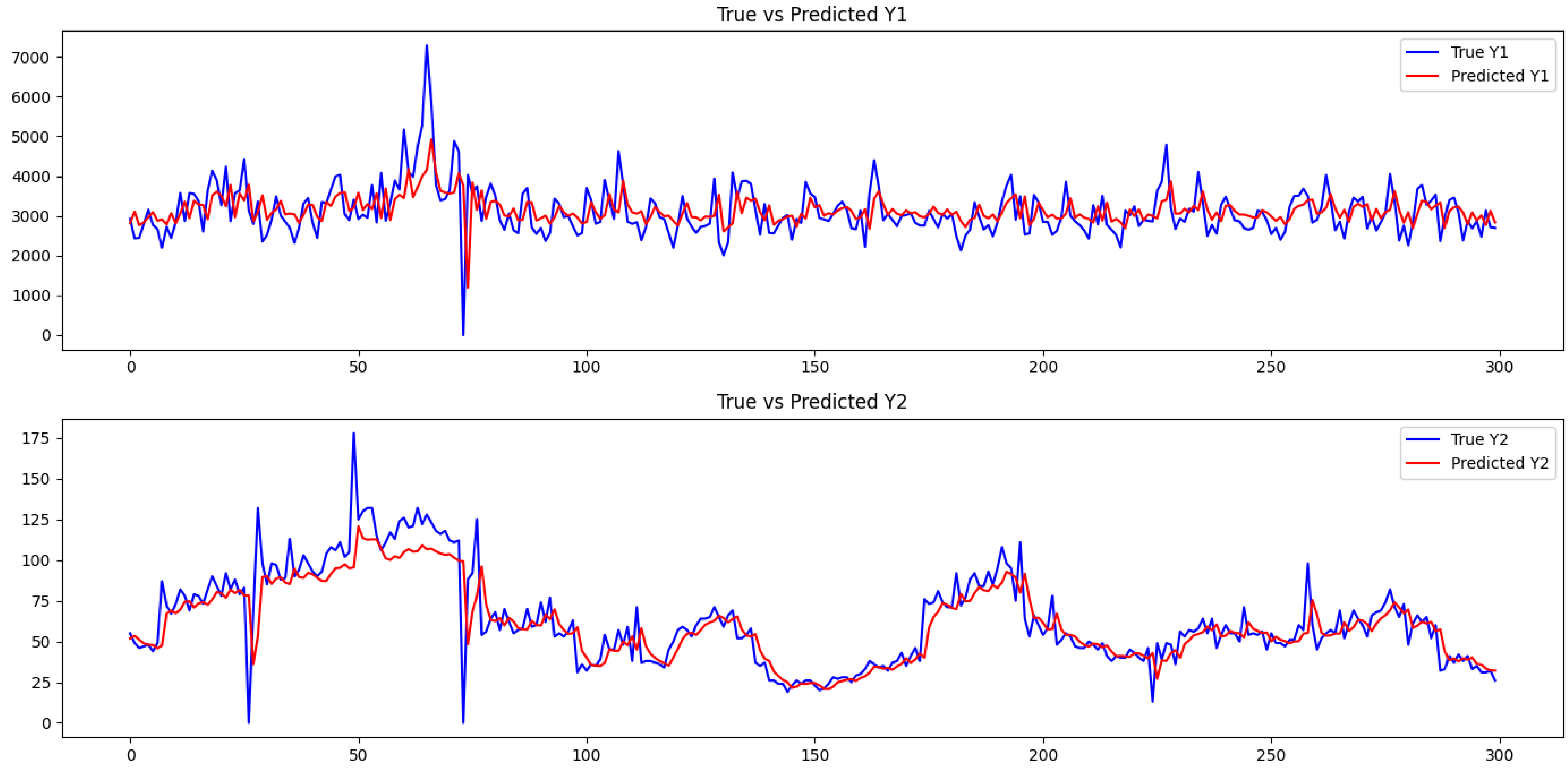
STACKED LSTM

```
MSE Y1: 352983.57617333176
RMSE Y1: 594.1242093816172
MAE Y1: 431.1924104817708

MSE Y2: 338.5816485299347
RMSE Y2: 18.400588265866247
MAE Y2: 13.1119189453125
```

GRU

```
MSE Y1: 343758.5173530601
RMSE Y1: 586.3092335560307
MAE Y1: 415.21027913411456

MSE Y2: 226.99956513133054
RMSE Y2: 15.066504741688782
MAE Y2: 9.029496828715006
```

Results LSTM Grocery 1



Results LSTM BEAUTY

LSTM

```
MSE Y1: 10.2609963916165
RMSE Y1: 3.20327900620856
MAE Y1: 2.520929156144602

MSE Y2: 0.00993773791228817
RMSE Y2: 0.0996882034760792
MAE Y2: 0.02480681628609697
```

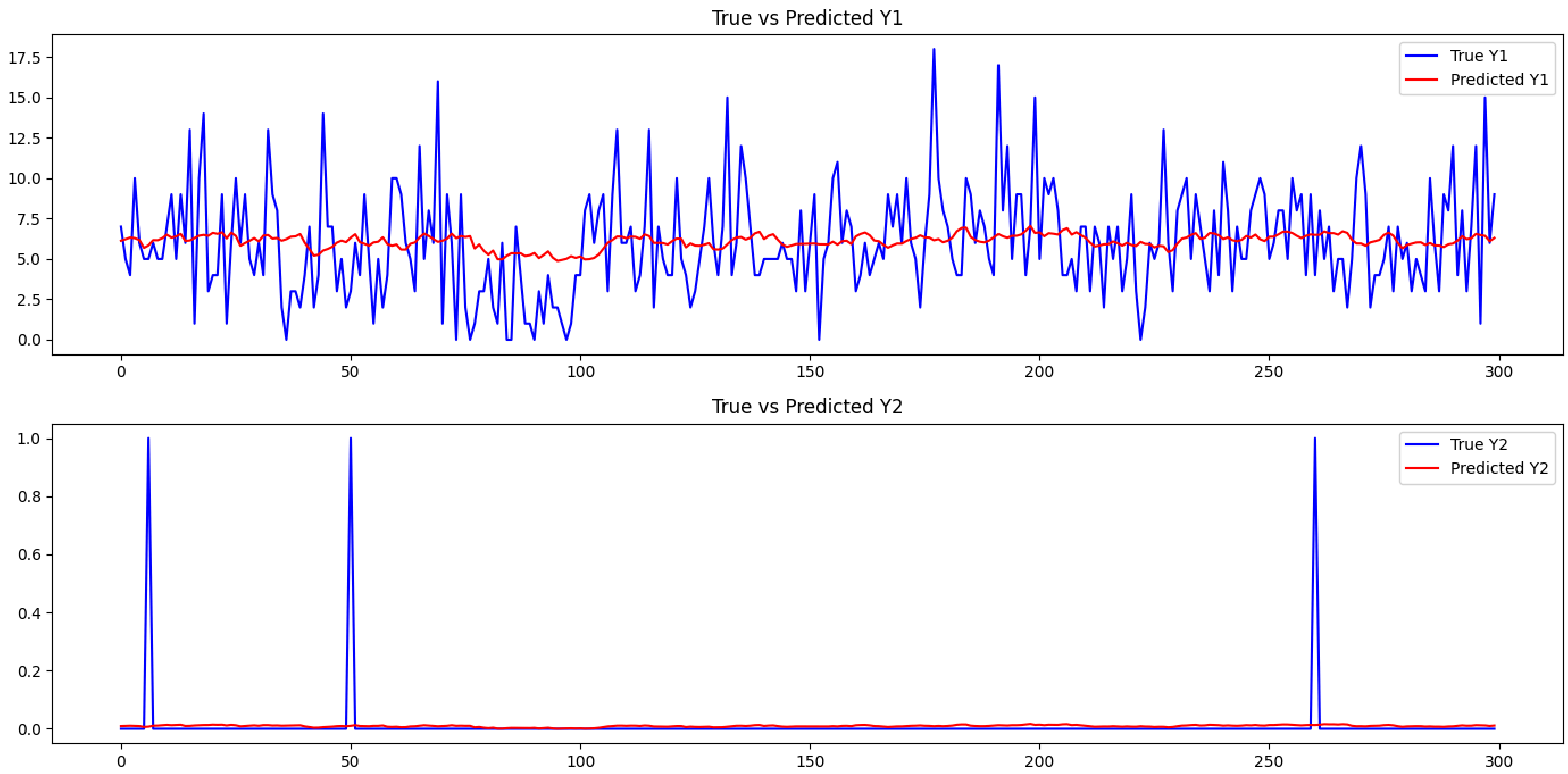
STACKED LSTM

```
MSE Y1: 10.429980638230793
RMSE Y1: 3.2295480547950968
MAE Y1: 2.5710357999801636

MSE Y2: 0.009897322877822181
RMSE Y2: 0.09948528975593418
MAE Y2: 0.019398317643596448
```

GRU

```
MSE Y1: 10.425120469443348
RMSE Y1: 3.2287955137238638
MAE Y1: 2.502937143643697

MSE Y2: 0.010086169353222197
RMSE Y2: 0.10042992259890574
MAE Y2: 0.014172124973071428
```

Results LSTM Automotive



Real-world Application

Examples of how your AI solution has been or could be applied in real scenarios.
Whether your AI solution would be deployed as web application, edge device, CCTV etc.

Future Improvement

Potential limitations of the current solution.
Ideas for further development and improvement.

1. Di sini, penggunaan dcoilwtico sebagai variabel eksogen tidak memiliki dasar statistik yang kuat. Untuk itu, bisa digunakan metode statistik untuk melihat ke-’pantas’-an feature ini sebagai variabel eksogen.
2. Perdalam pehaman tentang data dan hal-hal lain yang memungkinkan untuk dijadikan variabel eksogen.
3. Explorasi Hyperparameter tunning untuk mengoptimalkan model.