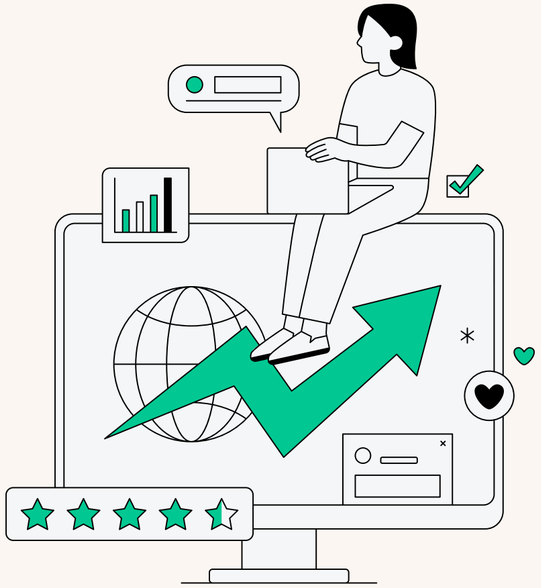
Digital Skola

Uber Fares Prediction using Regression Machine Learning Model

Final Project



BHAVANA ABHIRAMA

Bhavana Abhirama

KELOMPOK 6

Abdul Hadi

Antonius Andre

Anthony A. Ataupah

Febrio Wibowo

Hamsah Hams

Januarizqi Dwi M.

M. Badrul Munir Habibulloh

Defining Problem

Uber memberikan layanan kepada ratusan ribu pelanggan setiap hari. Sekarang, menjadi sangat penting untuk mengelola data mereka dengan baik untuk menghasilkan ide-ide bisnis baru agar mendapatkan hasil terbaik. Pada akhirnya, menjadi sangat penting untuk memperkirakan harga tarif dengan akurat.

Data Understanding

20 ribu data transaksi Uber Ride
tahun 2009 - 2015

Variable dalam dataset berupa :

Key / ID
Identifikasi unik untuk setiap perjalanan

Fare Amount
Biaya setiap perjalanan (USD)

Pickup Date Time
tanggal dan waktu ketika perjalanan dimulai

Passenger Count
jumlah penumpang dalam kendaraan

Pickup Longitude
Koordinat garis horizontal lokasi argo diaktifkan

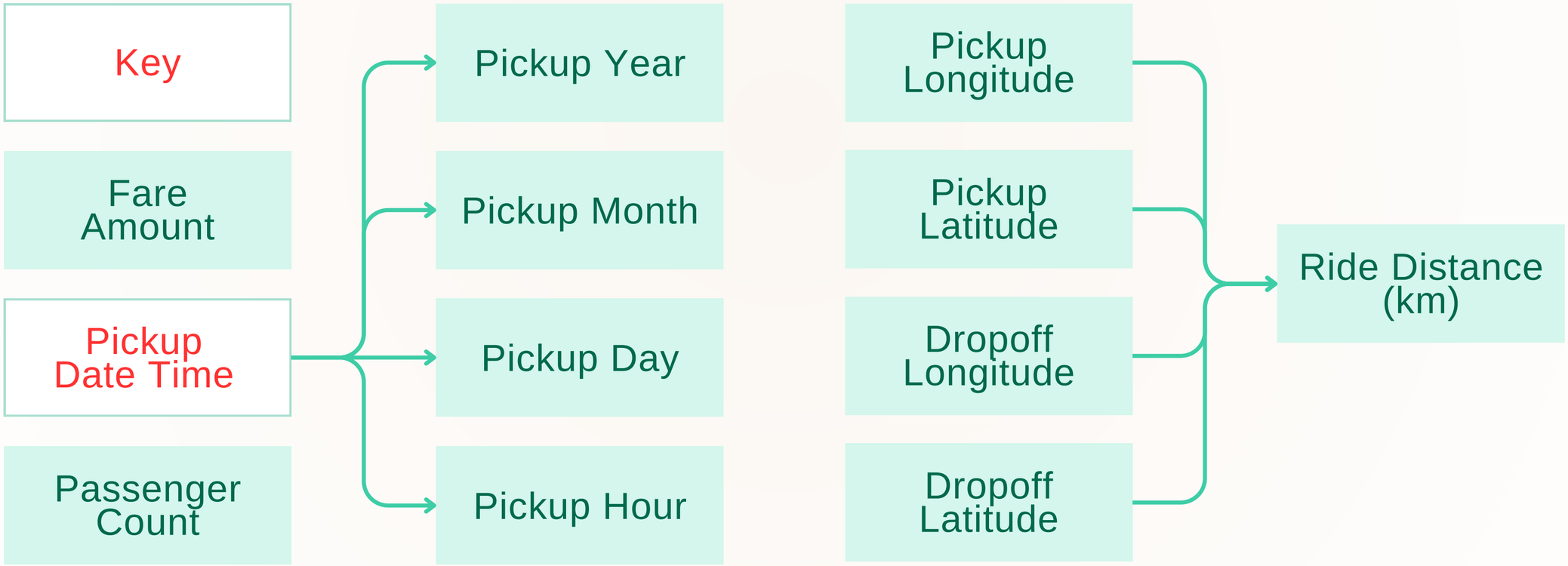
Pickup Latitude
Koordinat garis vertikal lokasi argo diaktifkan

Dropoff Longitude
Koordinat garis horizontal lokasi argo dimatikan

Dropoff Latitude
Koordinat garis vertikal lokasi argo dimatikan

Variable Determination

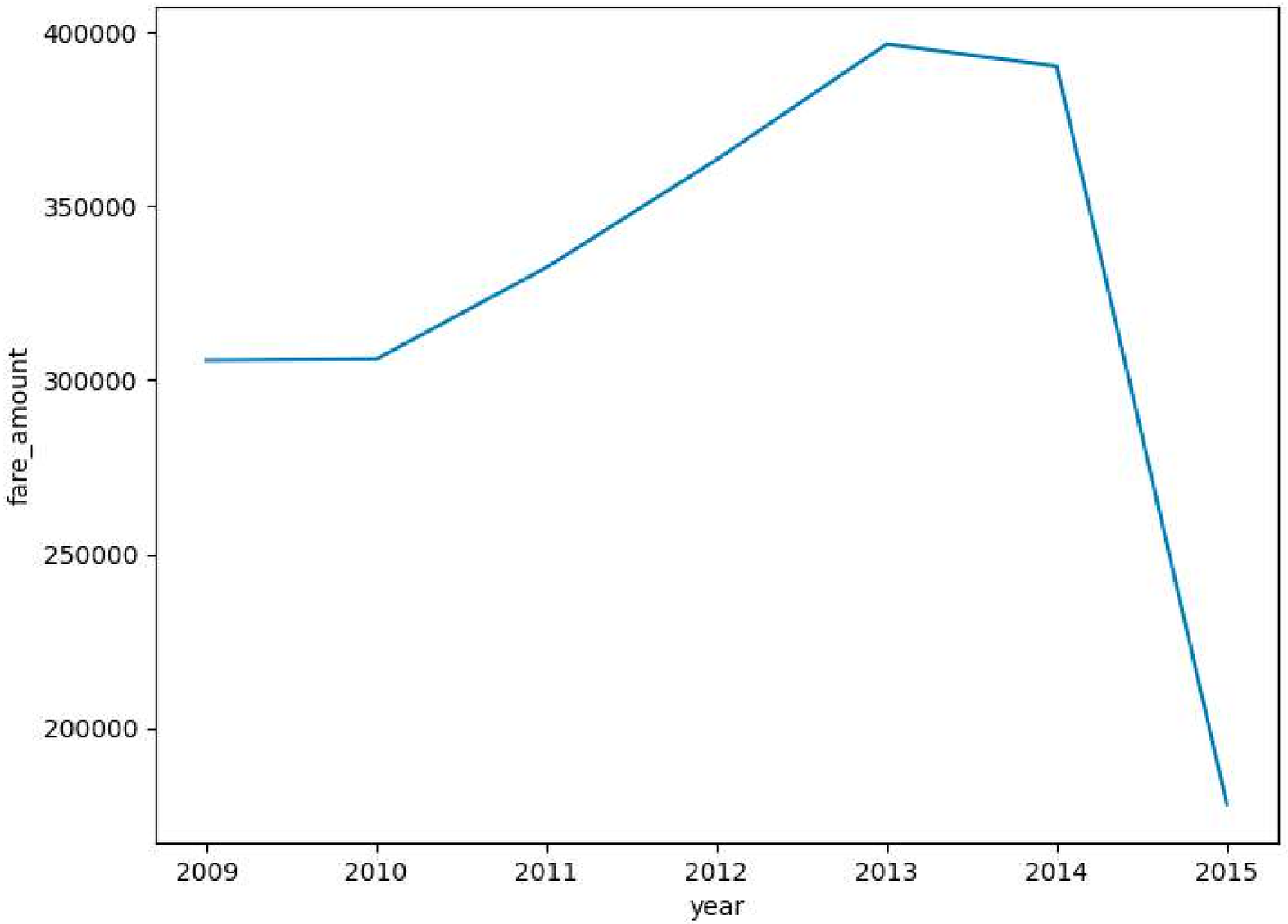
Dengan konteks tujuan : membuat prediksi tarif Uber Ride berdasarkan historical data menggunakan linear regression



Exploratory Data Analysis : Insights

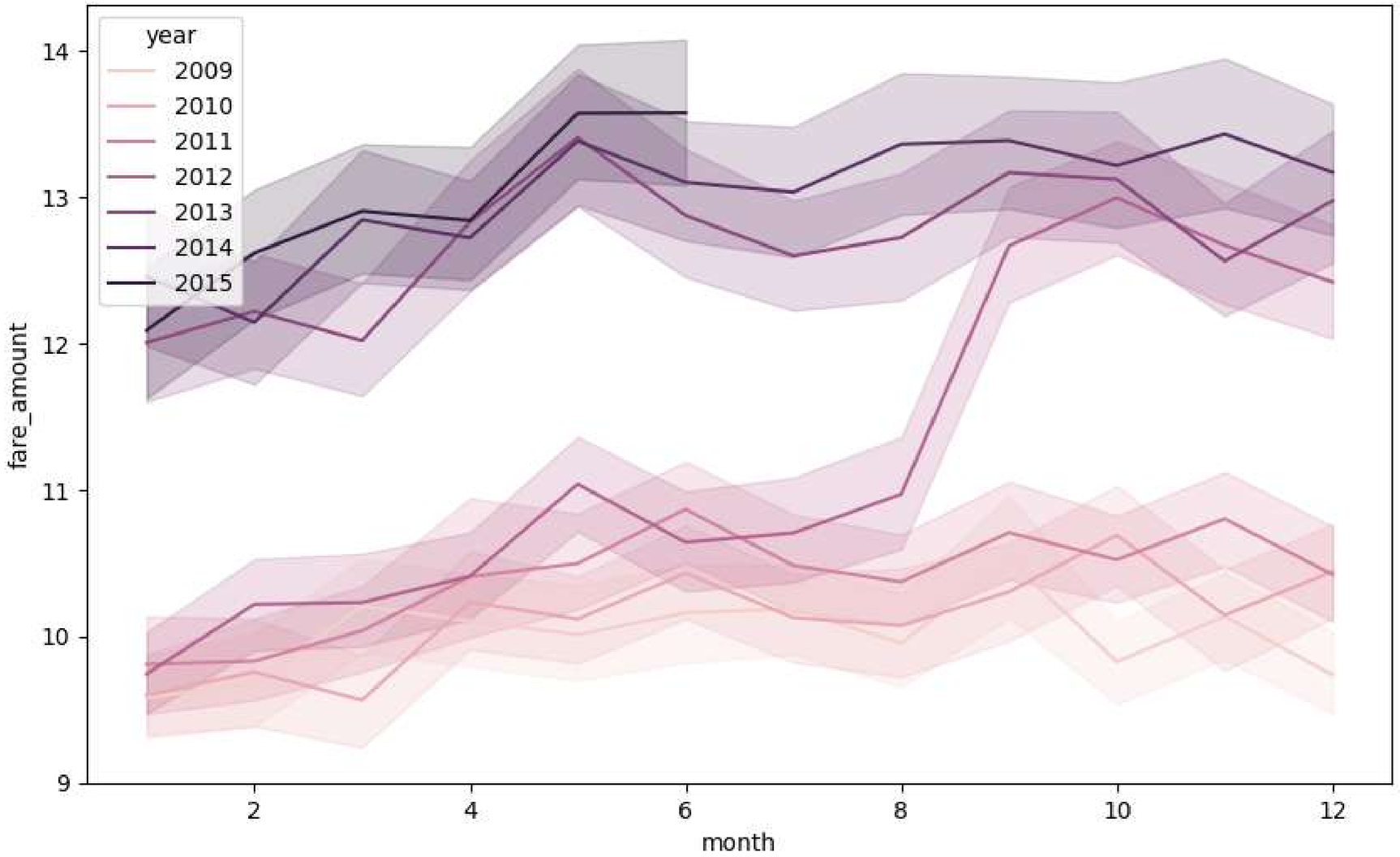
Terdapat peningkatan tarif yang cukup stabil dari tahun 2010 hingga 2013.

Jumlah tarif yang rendah pada tahun 2015 dapat dimaklumi karena hanya ada 6 bulan data

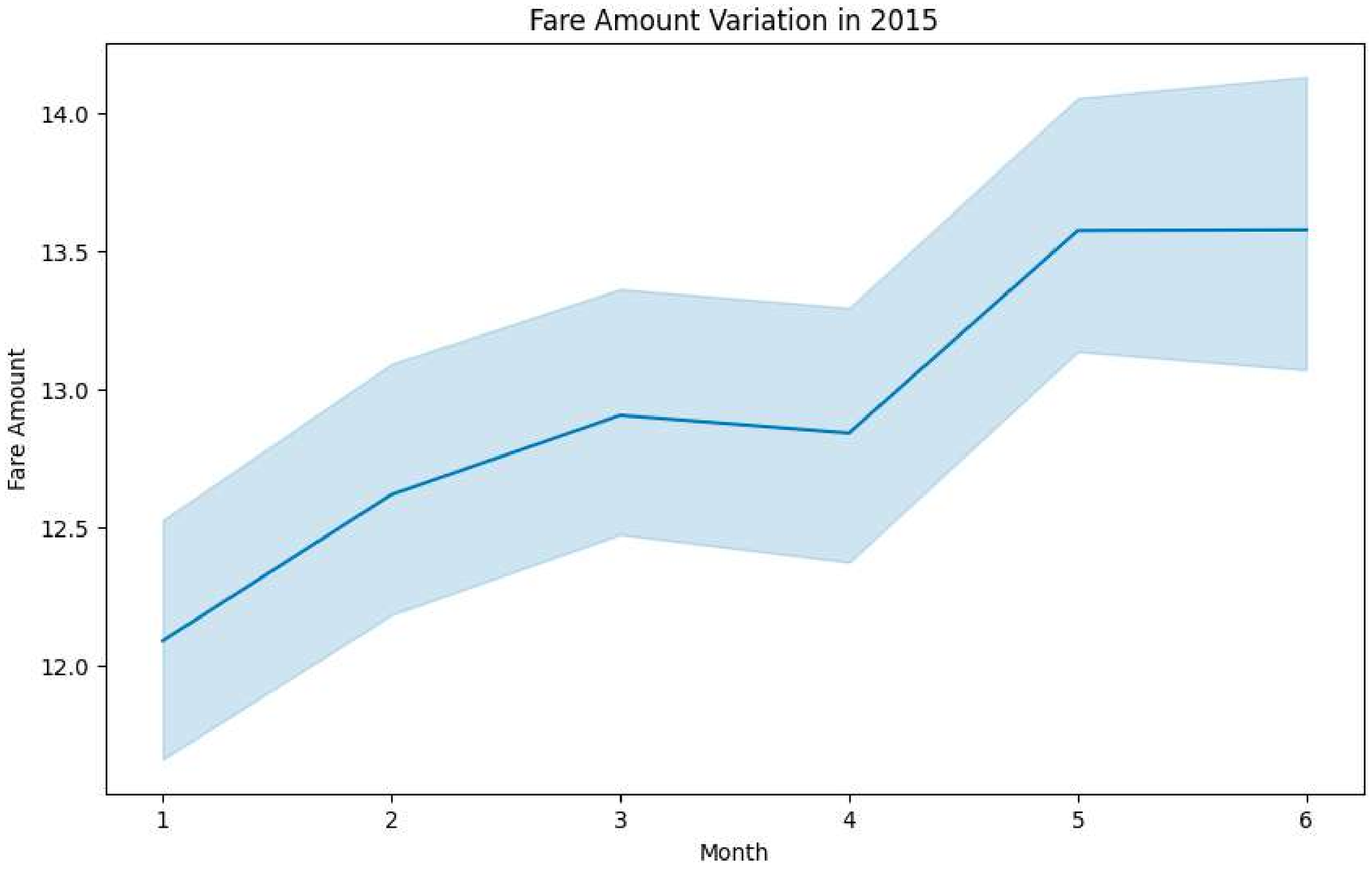


Pada tahun 2012 terdapat kenaikan harga yang sangat signifikan. Kenaikan harga tersebut naik dari bulan 8 sampai bulan 10.

Kenaikan harga pada tahun dan bulan tersebut, berpengaruh pada harga dibulan dan tahun berikutnya.



Pada tahun 2015 (1 tahun terakhir) terdapat kenaikan harga yang cenderung stabil



Data Preparation

Data Quality

- Mengatasi data yang hilang
- Memeriksa duplikasi data
- Membersihkan data yang tidak sesuai dengan variable input
- Menghapus data outlier yang berlebihan

Feature Engineering

- Menambahkan variabel baru yang diperlukan model prediksi
- Mengahapus kolom yang tidak diperlukan oleh model regresi

Data Transformation

- Dilakukan transformasi logaritma pada variabel yang tidak berdistribusi normal
- Memisahkan data train dan data test
- Standarisasi data

Data Preparation

Membersihkan missing value

```
df.dropna(inplace=True)
```

Mengbersihkan data yang tidak sesuai dengan input variabel

```
#Remove negative fare amount values
df_tmp = df_tmp[~(df_tmp["fare_amount"] < 0)]

#Remove impossible longitude and latitude value
unusual_1 = ((df_tmp["pickup_longitude"] >= 180) | (df_tmp["pickup_longitude"] <= -180) |
 (df_tmp["dropoff_longitude"] >= 180) | (df_tmp["dropoff_longitude"] <= -180) | (df_tmp["pickup_latitude"] >= 90) |
  (df_tmp["pickup_latitude"] <= -90) | (df_tmp["dropoff_latitude"] >= 90) | (df_tmp["dropoff_latitude"] <= -90))
df_tmp = df_tmp[~unusual_1]
```

Memeriksa data duplikat

```
df.duplicated().sum()
```

```
0
```

Data Preparation

Melepaskan variabel yang tidak diperlukan model regresi

```
df_tmp = df.drop(columns = ["Unnamed: 0", "key"]).copy()
```

Menambahkan kolom yang membantu dalam membuat model regresi yang lebih baik

```
#Add new columns that represent date and time

df_tmp['year'] = df_tmp.pickup_datetime.dt.year
df_tmp['month'] = df_tmp.pickup_datetime.dt.month
df_tmp["day"] = df_tmp.pickup_datetime.dt.day
df_tmp['weekday'] = df_tmp.pickup_datetime.dt.weekday
df_tmp['hour'] = df_tmp.pickup_datetime.dt.hour
```

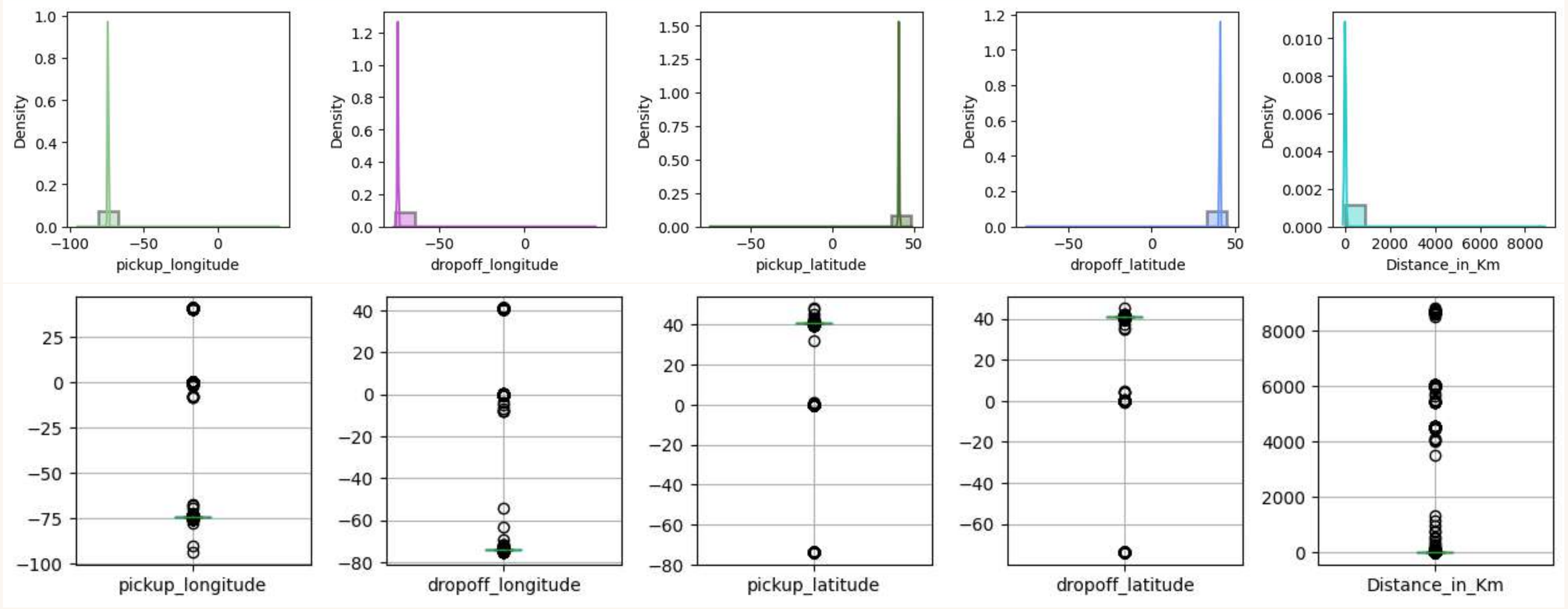
```
#Add new column for distance

distance = []
for a in df_tmp.index :
  loc1=(df_tmp.pickup_latitude[a], df_tmp.pickup_longitude[a])
  loc2=(df_tmp.dropoff_latitude[a], df_tmp.dropoff_longitude[a])
  distance.append(hs.haversine(loc1,loc2,unit=Unit.KILOMETERS))

df_tmp["Distance_in_Km"] = distance
```

Melepaskan variabel yang tidak diperlukan model regresi

```
df_tmp.drop(columns = "pickup_datetime", inplace = True)
```

Data Preparation

Visualisasi data yang menunjukan sebaran data didominasi outlier yang tidak normal

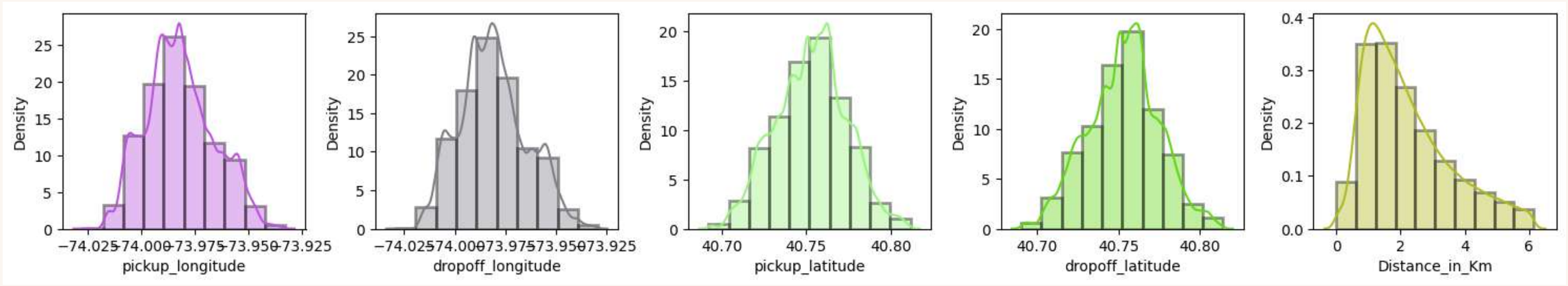


Data Preparation

Menghilangkan outlier

```
for i in nf:
    Q1 = df_tmp2[i].quantile(0.25)
    Q3 = df_tmp2[i].quantile(0.75)
    IQR = Q3 - Q1
    df_tmp2 = df_tmp2[df_tmp2[i] <= (Q3+(1.5*IQR))]
    df_tmp2 = df_tmp2[df_tmp2[i] >= (Q1-(1.5*IQR))]
    df_tmp2 = df_tmp2.reset_index(drop=True)
```

Visualisasi sebaran data setelah outlier dihilangkan



Data Preparation

Menghilangkan data yang tidak sesuai dengan input variabel

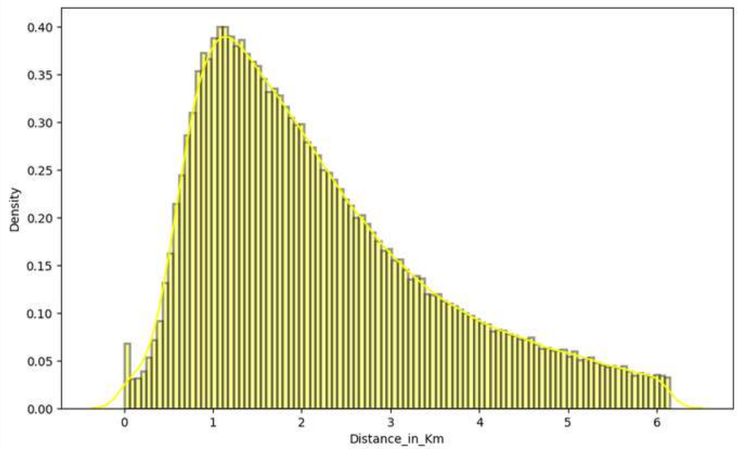
```
#Remove negative fare amount values
df_tmp = df_tmp[~(df_tmp["fare_amount"] < 0)]

#Remove rows with 0 distance
df_tmp = df_tmp[~(df_tmp["Distance_in_Km"] == 0)]
```

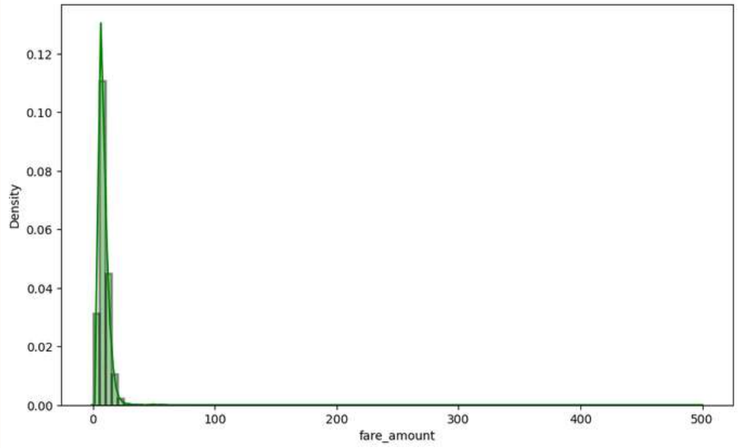
Transformasi logaritma pada data yang tidak berdistribusi normal

```
#Logarithm transformation on Distance
df_tmp2["Distance_in_Km"] = np.log1p(df_tmp2["Distance_in_Km"])

#Logarithm transformation on fare amount
df_tmp2["fare_amount"] = np.log1p(df_tmp2["fare_amount"])
```

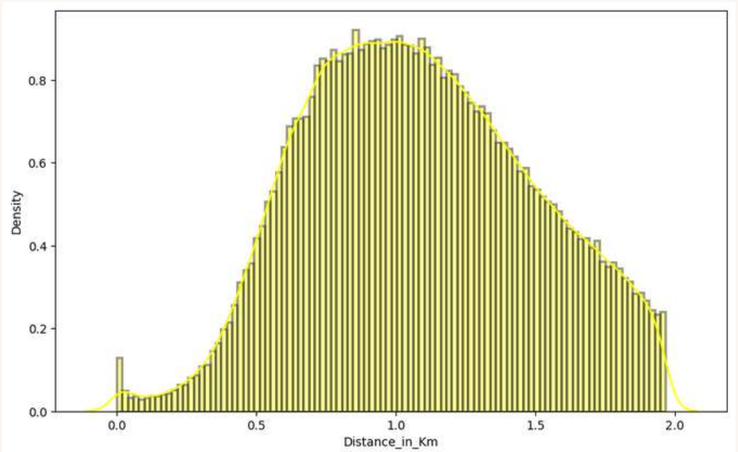
Data Preparation

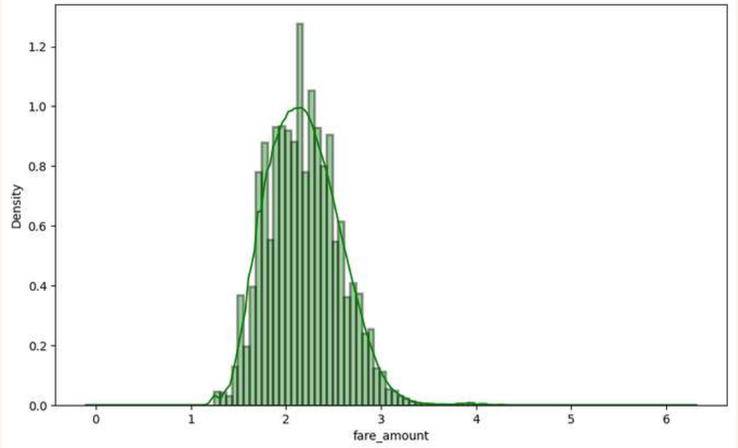
Sebaran data sebelum transformasi logaritma





Sebaran data setelah transformasi logaritma





Data Preparation

Memisahkan data ke data train dan data test

```
#Split data train and test
target = "fare_amount"

x = df_tmp2.drop([target],axis=1)
y = df_tmp2[target]
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, test_size=0.2, random_state=25)
x_train.reset_index(drop=True,inplace=True)
```

Standarisasi data sebelum dilakukan pembuatan model regresi

```
scaler = StandardScaler()

x_train_scld = scaler.fit_transform(x_train)
x_train_scld = pd.DataFrame(x_train_scld, columns=x_train.columns)

x_test_scld = scaler.transform(x_test)
x_test_scld = pd.DataFrame(x_test_scld, columns=x_test.columns)
```

Evaluation

```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import make_scorer, r2_score

# fungsi untuk melakukan cross validation
def test_model(model, X_train=x_train_scld, y_train=y_train):
    cv = KFold(n_splits = 4, shuffle=True, random_state = 45)
    r2 = make_scorer(r2_score)

    r2_val_score = cross_val_score(model, X_train, y_train, cv=cv, scoring = r2)
    score = [r2_val_score.mean()]
    return score
```

import library dam membuat test model cros validation

Hasil train data evaluation model linear regression sebesar 0.63

```
#Train data Evaluation
LR = LinearRegression()
a = test_model(LR)

print("R2 score for linear regression model with train data", a)
```

```
R2 score for linear regression model with train data [0.6309137856679168]
```

Evaluation

```
#Train data evaluation
lasso = Lasso(alpha=1e-4)
a = test_model(lasso)

print("R2 score for lasso regression model with train data", a)
```

```
R2 score for lasso regression model with train data [0.6309195007289061]
```

Hasil train data evaluation model Lasso sebesar 0.63

Hasil train data evaluation model regression XG boost sebesar 0.73

```
import xgboost

#Train data Evaluation
xgb_reg = xgboost.XGBRegressor()
a = test_model(xgb_reg)

print("R2 score for gradient boosting regression model with train data", a)
```

```
R2 score for gradient boosting regression model with train data [0.7319105286139573]
```

```python
#Evaluation Mode function

Model_Evaluation_Comparison_Matrix = pd.DataFrame(np.zeros([5,8]), columns=['Train-R2','Test-R2','Train-RSS','Test-
                                                                            'Train-MSE','Test-MSE','Train-RMSE','Te
rc=np.random.choice(x_train_scld.loc[:,x_train_scld.nunique()>50].columns,3)
def Evaluate(n, pred1,pred2):
    #Plotting predictions alongside the actual datapoints
    plt.figure(figsize=[15,6])
    for e,i in enumerate(rc):
        plt.subplot(2,3,e+1)
        plt.scatter(y=y_train, x=x_train_scld[i], label='Actual')
        plt.scatter(y=pred1, x=x_train_scld[i], label='Prediction')
        plt.legend()
    plt.show()
```

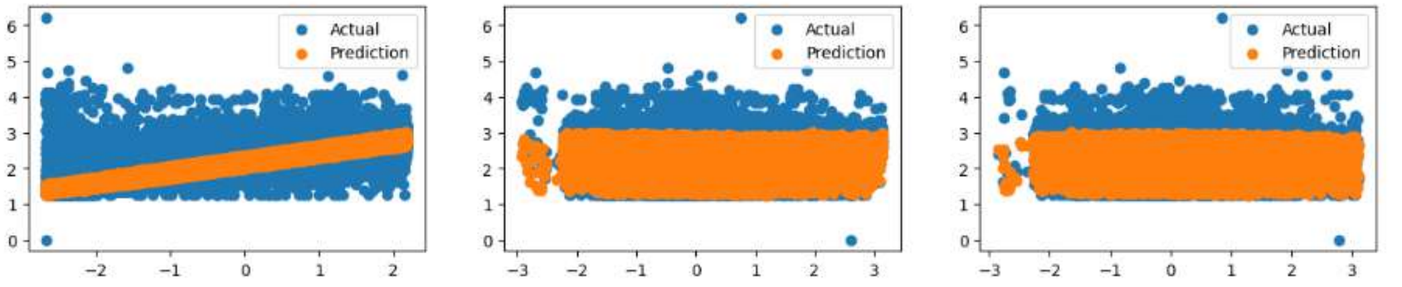
✳

Fungsi untuk melakukan evaluasi perbandingan model regresi

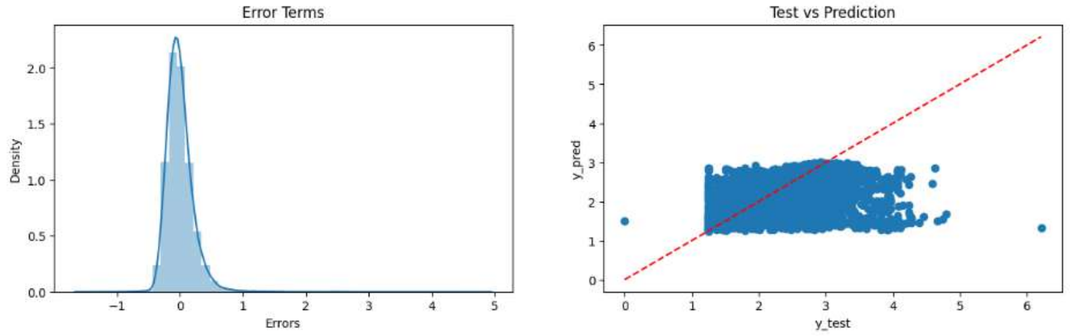
```python
print('\n\n{}Training Set Metrics{}'.format('-'*20, '-'*20))
print('\nR2-Score on Training set --->',round(r2_score(y_train, pred1),20))
print('Residual Sum of Squares (RSS) on Training set  --->',round(np.sum(np.square(y_train-pred1)),20))
print('Mean Squared Error (MSE) on Training set       --->',round(mean_squared_error(y_train, pred1),20))
print('Root Mean Squared Error (RMSE) on Training set --->',round(np.sqrt(mean_squared_error(y_train, pred1)),2

print('\n{}Testing Set Metrics{}'.format('-'*20, '-'*20))
print('\nR2-Score on Testing set --->',round(r2_score(y_test, pred2),20))
print('Residual Sum of Squares (RSS) on Training set  --->',round(np.sum(np.square(y_test-pred2)),20))
print('Mean Squared Error (MSE) on Training set       --->',round(mean_squared_error(y_test, pred2),20))
print('Root Mean Squared Error (RMSE) on Training set --->',round(np.sqrt(mean_squared_error(y_test, pred2)),20
print('\n{}Residual Plots{}'.format('-'*20, '-'*20))

Model_Evaluation_Comparison_Matrix.loc[n,'Train-R2']  = round(r2_score(y_train, pred1),20)
Model_Evaluation_Comparison_Matrix.loc[n,'Test-R2']   = round(r2_score(y_test, pred2),20)
Model_Evaluation_Comparison_Matrix.loc[n,'Train-RSS'] = round(np.sum(np.square(y_train-pred1)),20)
Model_Evaluation_Comparison_Matrix.loc[n,'Test-RSS']  = round(np.sum(np.square(y_test-pred2)),20)
Model_Evaluation_Comparison_Matrix.loc[n,'Train-MSE'] = round(mean_squared_error(y_train, pred1),20)
Model_Evaluation_Comparison_Matrix.loc[n,'Test-MSE']  = round(mean_squared_error(y_test, pred2),20)
Model_Evaluation_Comparison_Matrix.loc[n,'Train-RMSE']= round(np.sqrt(mean_squared_error(y_train, pred1)),20)
Model_Evaluation_Comparison_Matrix.loc[n,'Test-RMSE'] = round(np.sqrt(mean_squared_error(y_test, pred2)),20)
```

```python
plt.figure(figsize=[15,4])

plt.subplot(1,2,1)
sns.distplot((y_train - pred1))
plt.title('Error Terms')
plt.xlabel('Errors')

plt.subplot(1,2,2)
plt.scatter(y_train,pred1)
plt.plot([y_train.min(),y_train.max()],[y_train.min(),y_train.max()], 'r--')
plt.title('Test vs Prediction')
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```

```python
#Linear Regression

MLR = LinearRegression().fit(x_train_scld,y_train)
pred1 = MLR.predict(x_train_scld)
pred2 = MLR.predict(x_test_scld)

Evaluate(0, pred1, pred2)
```

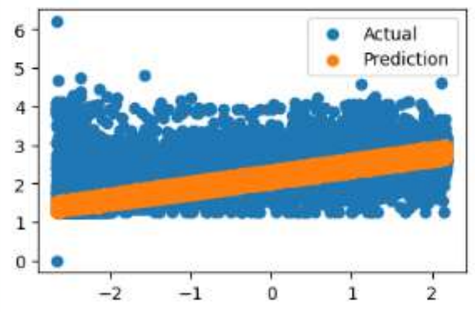


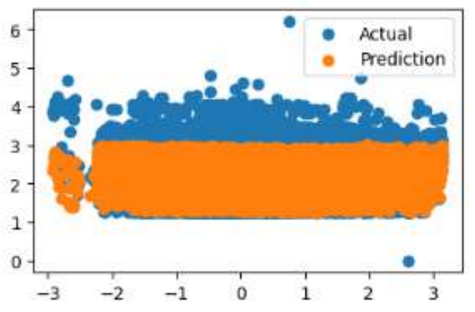

```
--------------------Training Set Metrics--------------------

R2-Score on Training set ---> 0.6310543844638177
Residual Sum of Squares (RSS) on Training set  ---> 6818.989825631929
Mean Squared Error (MSE) on Training set       ---> 0.05293506983210366
Root Mean Squared Error (RMSE) on Training set ---> 0.23007622613408726

--------------------Testing Set Metrics--------------------

R2-Score on Testing set ---> 0.6218223380531255
Residual Sum of Squares (RSS) on Training set  ---> 1733.1956766092685
Mean Squared Error (MSE) on Training set       ---> 0.053817595920176015
Root Mean Squared Error (RMSE) on Training set ---> 0.23198619769325937

--------------------Residual Plots--------------------
```

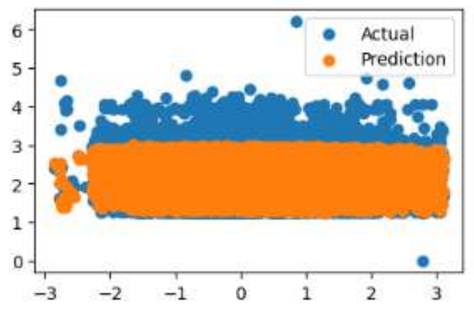


Hasil dari evaluasi model linear regression

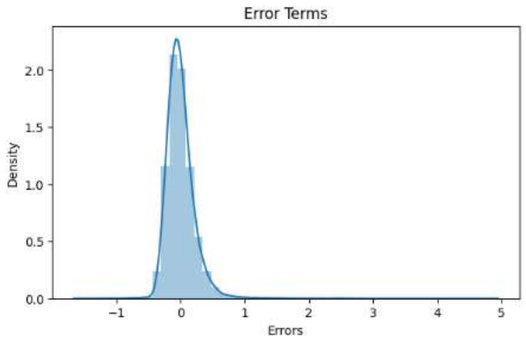
```python
#Creating a Ridge Regression model

RR = Ridge().fit(x_train_scld,y_train)
pred1 = RR.predict(x_train_scld)
pred2 = RR.predict(x_test_scld)

Evaluate(1, pred1, pred2)
```

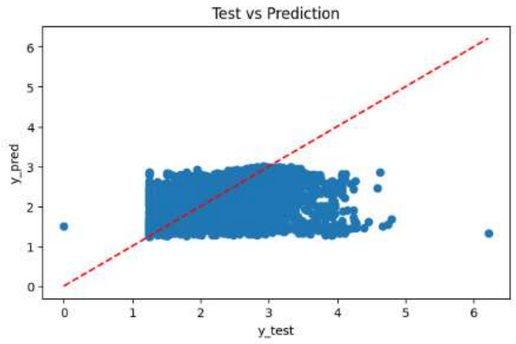







```
--------------------Training Set Metrics--------------------

R2-Score on Training set ---> 0.6310543844255566
Residual Sum of Squares (RSS) on Training set  ---> 6818.989826339083
Mean Squared Error (MSE) on Training set       ---> 0.05293506983759322
Root Mean Squared Error (RMSE) on Training set ---> 0.23007622614601717

--------------------Testing Set Metrics--------------------

R2-Score on Testing set ---> 0.621822398852256
Residual Sum of Squares (RSS) on Training set  ---> 1733.195397965655
Mean Squared Error (MSE) on Training set       ---> 0.05381758726799115
Root Mean Squared Error (RMSE) on Training set ---> 0.23198617904519905

--------------------Residual Plots--------------------
```

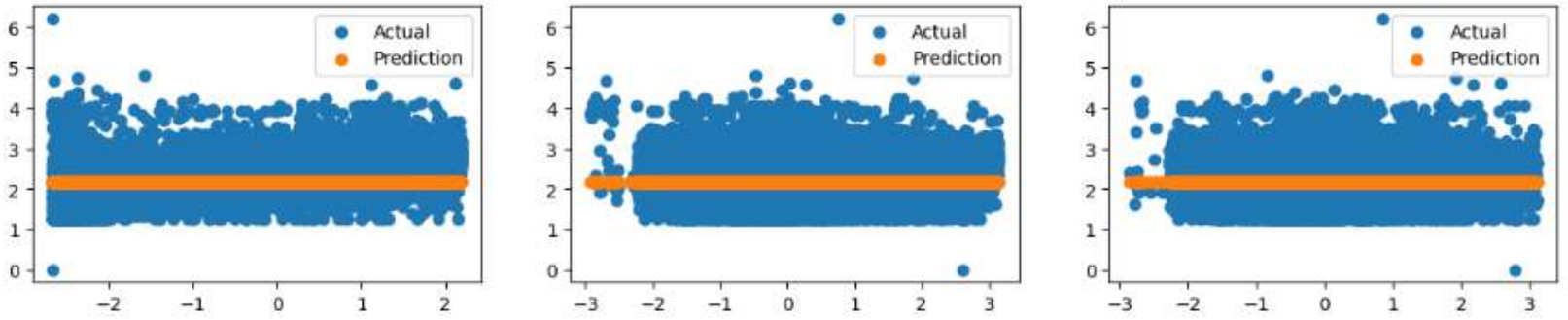




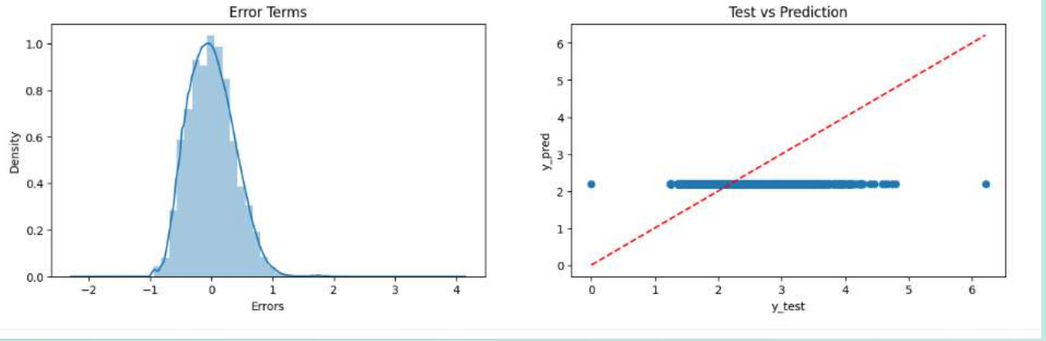
✳

Hasil dari evaluasi predict ridge regression

```
#Creating a Ridge Regression model

LLR = Lasso().fit(x_train_scld,y_train)
pred1 = LLR.predict(x_train_scld)
pred2 = LLR.predict(x_test_scld)

Evaluate(2, pred1, pred2)
```

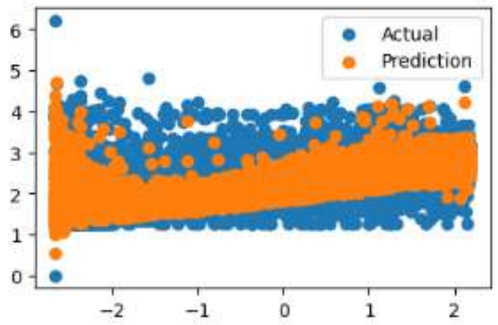


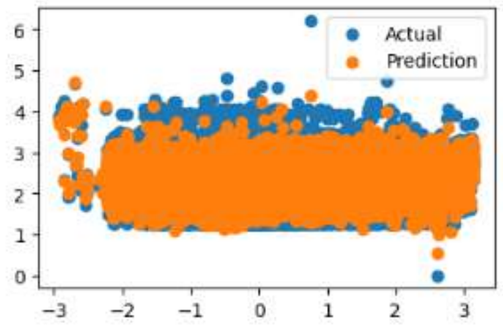

```
----------Training Set Metrics----------

R2-Score on Training set ---> 0.0
Residual Sum of Squares (RSS) on Training set  ---> 18482.37121810489
Mean Squared Error (MSE) on Training set       ---> 0.14347661986760304
Root Mean Squared Error (RMSE) on Training set ---> 0.3787830775887474

----------Testing Set Metrics----------

R2-Score on Testing set ---> -3.562588511774045e-05
Residual Sum of Squares (RSS) on Training set  ---> 4583.18297891115
Mean Squared Error (MSE) on Training set       ---> 0.14231277686418722
Root Mean Squared Error (RMSE) on Training set ---> 0.3772436571556734

----------Residual Plots----------
```

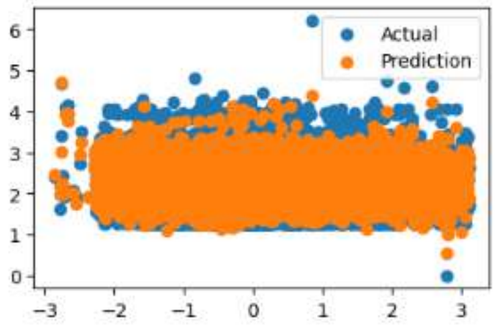


Hasil dari evaluasi predict Lasso regression

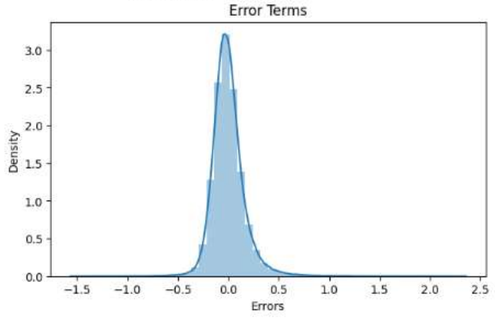
```
#Creating a gradient boosting Regression model

XGB = xgboost.XGBRegressor().fit(x_train_scld,y_train)
pred1 = XGB.predict(x_train_scld)
pred2 = XGB.predict(x_test_scld)

Evaluate(2, pred1, pred2)
```

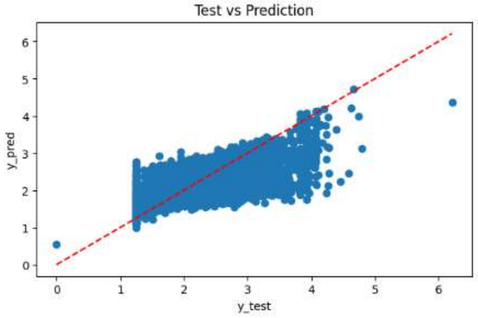







```
--------------------Training Set Metrics--------------------

R2-Score on Training set ---> 0.8002889632722365
Residual Sum of Squares (RSS) on Training set  ---> 3691.1335171551023
Mean Squared Error (MSE) on Training set       ---> 0.028653864499954218
Root Mean Squared Error (RMSE) on Training set ---> 0.1692745240724493

--------------------Testing Set Metrics--------------------

R2-Score on Testing set ---> 0.7271751195331151
Residual Sum of Squares (RSS) on Training set  ---> 1250.361803133342
Mean Squared Error (MSE) on Training set       ---> 0.03882508315893004
Root Mean Squared Error (RMSE) on Training set ---> 0.19704081597204684

--------------------Residual Plots--------------------
```





✳

Hasil dari evaluasi predict XG boost regression

Modeling

```
def basic_model(model, x_train, y_train):
  clf = model
  return clf.fit(x_train, y_train)
```

Terlebih dahulu membuat basic model dari **x_train** dan **y_train** agar dapat disimpan

menyimpan model yang sudah dibuat ke dalam variabel

```
# linear regression
model_reg = basic_model(LinearRegression(), x_train_scld, y_train)
```

```
joblib.dump((model_reg, scaler), 'model_with_scaler.joblib')
```

menyimpan model dan fungsi standarisasi yang telah di buat ke dalam file joblib

Model Deployment with Streamlit

Deploy

Uber Fare Amount Prediction App

Analyst Team

ML Section

Attribute Info

- Pickup date : YYYY-MM-DD UTC
- Pickup time : hh
  UTC
- Pickup longitude : Pickup Location, Longitude
- Pickup latitude : Pickup Location, Latitude
- Dropoff longitude : dropoff Location, Longitude
- Dropoff latitude : dropoff Location, Latitude
- Passenger count : 1 - 7

Input Your Data

Pickup Date

2023/11/22

Pickup Time

20:15

Pickup Location Longitude :

-65,001000

Pickup Location Latitude :

40,752958

Dropoff Location Longitude :

-73,981880

Dropoff Location Latitude :

40,752958

Number of Passenger :

1

	Date	Time	Pickup Location	Dropoff Location	Number of Passenger
0	2023-11-22	[20,15]	[40.752958,-65.001]	[40.752958,-73.98188]	1

Prediction result

Fare amount : `600.1359311015976`

Distance : `756.1627711134022` km

Pickup Location :

Deploy

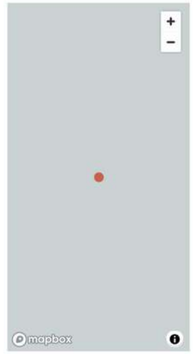
	Date	Time	Pickup Location	Dropoff Location	Number of Passenger
0	2023-11-22	[20,15]	[40.752958,-65.001]	[40.752958,-73.98188]	1

Prediction result

Fare amount : `600.1359311015976`

Distance : `756.1627711134022` km

Pickup Location :



Dropoff Location :

ML Section

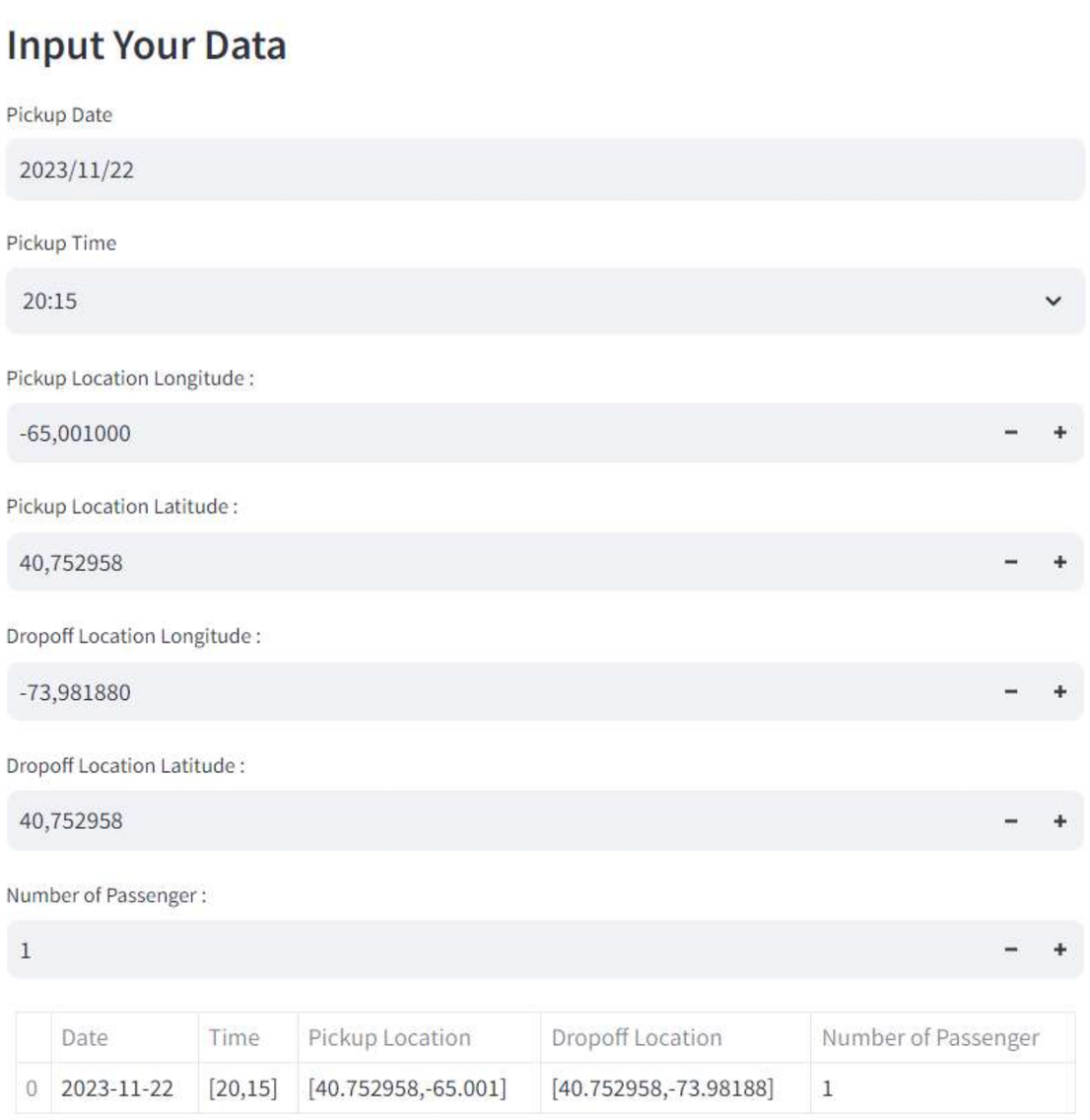
Attribute Info

- Pickup date : YYYY-MM-DD UTC
- Pickup time : hh
  UTC
- Pickup longitude : Pickup Location, Longitude
- Pickup latitude : Pickup Location, Latitude
- Dropoff longitude : dropoff Location, Longitude
- Dropoff latitude : dropoff Location, Latitude
- Passenger count : 1 - 7

Untuk Tampilan pertama terdapat informasi terkait Atribut yang digunakan. Seperti pickup date, pickup time, Pickup longitude, Pickup Latitude, Dropoff longitude, Dropoff latitude dan Passenger count

Setelahnya user diharapkan untuk melakukan input data yang diinginkan. Dan data yang sudah diinput akan ditampilkan ke table dibawahnya.

Data yang diinputkan akan ditransformasikan ke dalam data input agar sesuai dengan variabel-variabel input model regresi. Kemudian dilakukan standarisasi oleh variabel scaler yang sudah difitkan dengan data training yang digunakan untuk membentuk model regresi



Ditampilan selanjutnya akan menampilkan hasil prediksi dari data yang sudah diinputkan oleh user.

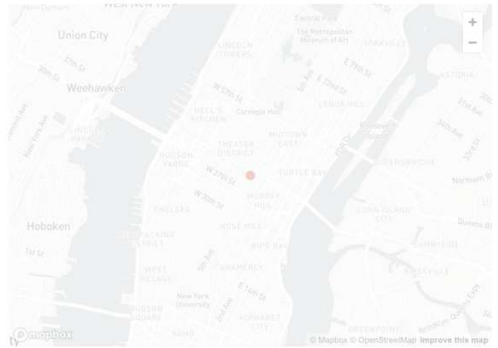
	Date	Time	Pickup Location	Dropoff Location	Number of Passenger
0	2023-11-22	[20,15]	[40.752958,-65.001]	[40.752958,-73.98188]	1

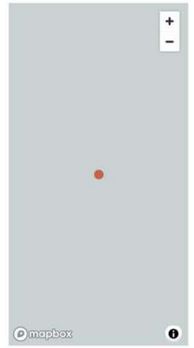
Prediction result

Fare amount : `600.1359311015976`

Distance : `756.1627711134022` km

Pickup Location :





Dropoff Location :

Thank you

Presented by Bhavana Abhirama

Kontribusi

Abdul Hadi	10%	Define Problem, EDA, data pre-processing, ML Modelling.
Antonius Andre	10%	Model Deployment(Streamlit), EDA, data pre-processing, ML Modelling.
Anthony A. Ataupah	40%	EDA, data pre-processing, ML Modelling, Model Deployment.
Febrio Wibowo	10%	Data Understanding, EDA, data pre-processing, ML Modelling.
Hamsah Hams	10%	Evaluation and Modelling, EDA, data pre-processing, ML Modelling.
Januarizqi Dwi M.	10%	Data Exploration(load data - insights), EDA, data pre-processing, ML Modelling.
M. Badrul Munir Habibulloh	10%	Data Preparation, EDA, data pre-processing, ML Modelling.