In [26]:

```
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import r2_score
```

In [3]:

```
data = "taxi_trip_pricing.csv"
df = pd.read_csv(data)

df.head(10)
```

Out[3]:

	Trip_Distance_km	Time_of_Day	Day_of_Week	Passenger_Count	Traffic_Conditions	V
0	19.35	Morning	Weekday	3.0	Low	
1	47.59	Afternoon	Weekday	1.0	High	
2	36.87	Evening	Weekend	1.0	High	
3	30.33	Evening	Weekday	4.0	Low	
4	NaN	Evening	Weekday	3.0	High	
5	8.64	Afternoon	Weekend	2.0	Medium	
6	3.85	Afternoon	Weekday	4.0	High	
7	43.44	Evening	Weekend	3.0	NaN	
8	30.45	Morning	Weekday	3.0	High	
9	35.70	Afternoon	Weekday	2.0	Low	

In [56]:

```
x = df.select_dtypes(include = [np.number])

impute = IterativeImputer()

x = pd.DataFrame(impute.fit_transform(x), columns = x.columns)
```

In [ ]:

In [57]:

```
x2 = df.select_dtypes(exclude = [np.number])
```

In [58]:

```
for col in x2.columns :
    x2[col] = x2[col].fillna(x2[col].mode()[0])
```

In [ ]:

In [59]:

```
encoder = LabelEncoder()
for col in x2.columns :
    x2[col] = encoder.fit_transform(x2[col])
    print(list(encoder.classes_))
```

```
['Afternoon', 'Evening', 'Morning', 'Night']
['Weekday', 'Weekend']
['High', 'Low', 'Medium']
['Clear', 'Rain', 'Snow']
```

In [60]:

```
X = pd.concat([x,x2], axis = 1)
X
```

Out[60]:

	Trip_Distance_km	Passenger_Count	Base_Fare	Per_Km_Rate	Per_Minute_Rate	Trip
0	19.350000	3.0	3.560000	0.800000	0.320000	
1	47.590000	1.0	3.517427	0.620000	0.430000	
2	36.870000	1.0	2.700000	1.210000	0.150000	
3	30.330000	4.0	3.480000	0.510000	0.150000	
4	19.722094	3.0	2.930000	0.630000	0.320000	
...	...	...	...	...	...	
995	5.490000	4.0	2.390000	0.620000	0.490000	
996	45.950000	4.0	3.120000	0.610000	0.248435	
997	7.700000	3.0	2.080000	1.780000	0.302450	
998	47.560000	1.0	2.670000	0.820000	0.170000	
999	22.850000	3.0	4.340000	1.371673	0.230000	

1000 rows × 11 columns

In [61]:

```
def outlier(col):
    Q1 = col.quantile(0.25)
    Q3 = col.quantile(0.75)

    IQR = Q3 - Q1

    Fb = Q1 - 1.5 * IQR
    Fh = Q3 + 1.5 * IQR

    return col.clip(lower = Fb, upper = Fh)

for col in X.columns :
    X[col] = outlier(X[col])


y = X['Trip_Price']
X = X.select_dtypes(include = [np.number])
```

In [62]:

```
X.head()
```

Out[62]:

	Trip_Distance_km	Passenger_Count	Base_Fare	Per_Km_Rate	Per_Minute_Rate	Trip_D
0	19.350000	3.0	3.560000	0.80	0.32	
1	47.590000	1.0	3.517427	0.62	0.43	
2	36.870000	1.0	2.700000	1.21	0.15	
3	30.330000	4.0	3.480000	0.51	0.15	
4	19.722094	3.0	2.930000	0.63	0.32	

In [63]:

```
X = X.drop(columns ='Trip_Price')
```

In [64]:

```
y
```

Out[64]:

```
0      36.262400
1      79.047866
2      52.903200
3      36.469800
4      15.618000
         ...
995    34.404900
996    62.129500
997    33.123600
998    61.209000
999    45.443700
Name: Trip_Price, Length: 1000, dtype: float64
```

In [65]:

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [66]:

```
y_test = y_test.reset_index(drop=True)
```

In [90]:

```
model = xgb.XGBRegressor(objective="reg:squarederror",
    n_estimators=800,
    learning_rate=0.05,
    subsample=0.1,
    colsample_bytree=0.8,
    max_depth=8
)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

In [91]:

```
mse = mean_squared_error(y_test, y_pred)
mse
```

Out[91]:

```
52.618801160394305
```

In [92]:

```
r2 = r2_score(y_test, y_pred)
r2
```

Out[92]:

```
0.9318760984322219
```

In [93]:

```
xx = np.array([35.21, 4, 3.94, 0.56, 0.39, 110.06, 0, 1, 2, 0])
```

```
prd = model.predict([xx])
prd
```

Out[93]:

```
array([70.179535], dtype=float32)
```

In [94]:

```
import pickle

with open('XGboost_regression_Lineaire.pkl', 'wb') as f:
    pickle.dump(model, f)
```

In [ ]:

In [ ]:

In [ ]: