Used Car Price Prediction by Abdil Nayaka Rizky

This project aims to predict used car prices using the regression method. I will analyze used car price data and build a model that can estimate prices based on existing features.

```
pip install scikit-learn
```

```
Collecting scikit-learn
  Downloading scikit_learn-1.5.2-cp311-cp311-win_amd64.whl.metadata (13 kB)
Requirement already satisfied: numpy>=1.19.5 in c:\users\yusza\appdata\local\programs\python\python311\lib\site-packages (from sc
Collecting scipy>=1.6.0 (from scikit-learn)
  Downloading scipy-1.14.1-cp311-cp311-win_amd64.whl.metadata (60 kB)
Collecting joblib>=1.2.0 (from scikit-learn)
  Downloading joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Downloading threadpoolctl-3.5.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.5.2-cp311-cp311-win_amd64.whl (11.0 MB)
   ---------------------------------------- 0.0/11.0 MB ? eta -:--:--
   --- ------------------------------------ 1.0/11.0 MB 7.1 MB/s eta 0:00:02
   ------- -------------------------------- 2.1/11.0 MB 4.9 MB/s eta 0:00:02
   ---------- ----------------------------- 2.9/11.0 MB 4.4 MB/s eta 0:00:02
   ------------ --------------------------- 3.7/11.0 MB 4.4 MB/s eta 0:00:02
   ---------------- ----------------------- 4.5/11.0 MB 4.3 MB/s eta 0:00:02
   -------------------- ------------------- 5.2/11.0 MB 4.2 MB/s eta 0:00:02
   ---------------------- ----------------- 6.0/11.0 MB 4.1 MB/s eta 0:00:02
   ------------------------ --------------- 6.8/11.0 MB 4.2 MB/s eta 0:00:02
   --------------------------- ------------ 7.6/11.0 MB 4.1 MB/s eta 0:00:01
   ----------------------------- ---------- 8.4/11.0 MB 4.1 MB/s eta 0:00:01
   -------------------------------- ------- 9.2/11.0 MB 4.0 MB/s eta 0:00:01
   ---------------------------------- ----- 10.0/11.0 MB 4.1 MB/s eta 0:00:01
   ------------------------------------  11.0/11.0 MB 4.0 MB/s eta 0:00:01
   ---------------------------------------- 11.0/11.0 MB 3.8 MB/s eta 0:00:00
Downloading joblib-1.4.2-py3-none-any.whl (301 kB)
Downloading scipy-1.14.1-cp311-cp311-win_amd64.whl (44.8 MB)
   ---------------------------------------- 0.0/44.8 MB ? eta -:--:--
   - -------------------------------------- 1.3/44.8 MB 5.6 MB/s eta 0:00:08
   - -------------------------------------- 2.1/44.8 MB 4.7 MB/s eta 0:00:10
   -- ------------------------------------- 2.9/44.8 MB 4.4 MB/s eta 0:00:10
   --- ------------------------------------ 3.7/44.8 MB 4.3 MB/s eta 0:00:10
   --- ------------------------------------ 4.5/44.8 MB 4.2 MB/s eta 0:00:10
   ---- ----------------------------------- 5.2/44.8 MB 4.1 MB/s eta 0:00:10
   ----- ---------------------------------- 6.0/44.8 MB 4.1 MB/s eta 0:00:10
   ------ --------------------------------- 6.8/44.8 MB 4.1 MB/s eta 0:00:10
   ------ --------------------------------- 7.6/44.8 MB 4.0 MB/s eta 0:00:10
   ------- -------------------------------- 8.7/44.8 MB 4.0 MB/s eta 0:00:09
   -------- ------------------------------- 9.4/44.8 MB 4.0 MB/s eta 0:00:09
   --------- ------------------------------ 10.2/44.8 MB 4.0 MB/s eta 0:00:09
   --------- ------------------------------ 11.0/44.8 MB 4.0 MB/s eta 0:00:09
   ---------- ----------------------------- 11.8/44.8 MB 4.0 MB/s eta 0:00:09
   ----------- ---------------------------- 12.8/44.8 MB 4.0 MB/s eta 0:00:08
   ----------- ---------------------------- 13.6/44.8 MB 4.0 MB/s eta 0:00:08
   ------------ --------------------------- 14.4/44.8 MB 4.0 MB/s eta 0:00:08
   ------------- -------------------------- 15.2/44.8 MB 4.0 MB/s eta 0:00:08
   -------------- ------------------------- 16.0/44.8 MB 4.0 MB/s eta 0:00:08
   -------------- ------------------------- 17.0/44.8 MB 4.0 MB/s eta 0:00:07
   --------------- ------------------------ 17.8/44.8 MB 4.0 MB/s eta 0:00:07
   ---------------- ----------------------- 18.6/44.8 MB 4.0 MB/s eta 0:00:07
   ----------------- ---------------------- 19.7/44.8 MB 4.0 MB/s eta 0:00:07
   ------------------ --------------------- 20.4/44.8 MB 4.0 MB/s eta 0:00:07
   ------------------ --------------------- 21.2/44.8 MB 4.0 MB/s eta 0:00:06
   ------------------- -------------------- 22.0/44.8 MB 4.0 MB/s eta 0:00:06
   -------------------- ------------------- 22.8/44.8 MB 4.0 MB/s eta 0:00:06
   --------------------- ------------------ 23.6/44.8 MB 4.0 MB/s eta 0:00:06
   ---------------------- ----------------- 24.6/44.8 MB 3.9 MB/s eta 0:00:06
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


# reading dataset
data = pd.read_csv('used_car_prices.csv')

# Show data
data.head()
```

	web-scraper-order	Car Model	Month/Year	Average price	Minimum price	Maximum price
0	1680204632-1	Skoda Octavia A8 2022	2023-03	967,000 EGP	926,000 EGP	1,017,000 EGP
1	1680204632-2	Skoda Octavia A8 2022	2023-02	979,000 EGP	931,000 EGP	1,045,000 EGP
2	1680204632-3	Skoda Octavia A8 2022	2023-01	917,000 EGP	893,000 EGP	950,000 EGP
3	1680204632-4	Skoda Octavia A8 2022	2022-12	881,000 EGP	793,000 EGP	950,000 EGP
4	1680204632-5	Skoda Octavia A8 2022	2022-11	868,000 EGP	789,000 EGP	950,000 EGP

Next steps: Generate code with data | View recommended plots | New interactive sheet

Data Preparation

Data is taken from source kaggle and includes information about car model, price, and month/year. The data is then cleaned by removing missing values and converting the data format as needed.

```
# Remove 'EGP' and convert prices to numeric format
data['Average price'] = data['Average price'].str.replace(' EGP', '').str.replace(',', '').astype(float)
data['Minimum price'] = data['Minimum price'].str.replace(' EGP', '').str.replace(',', '').astype(float)
data['Maximum price'] = data['Maximum price'].str.replace(' EGP', '').str.replace(',', '').astype(float)

# Convert Month/Year to datetime format
data['Month/Year'] = pd.to_datetime(data['Month/Year'], format='%Y-%m')

# Extracting month and year features from Month/Year
data['Month'] = data['Month/Year'].dt.month
data['Year'] = data['Month/Year'].dt.year

# delete columns that are not needed for prediction
data = data.drop(['web-scraper-order', 'Car Model', 'Month/Year'], axis=1)

# View preprocessing results
data.head()
```

	Average price	Minimum price	Maximum price	Month	Year
0	967000.0	926000.0	1017000.0	3.0	2023.0
1	979000.0	931000.0	1045000.0	2.0	2023.0
2	917000.0	893000.0	950000.0	1.0	2023.0
3	881000.0	793000.0	950000.0	12.0	2022.0
4	868000.0	789000.0	950000.0	11.0	2022.0

Next steps: Generate code with data | View recommended plots | New interactive sheet

Modeling

I will split the data into training and testing data, and then train a linear regression model.

```
# Deleting rows containing NaN
data_cleaned = data.dropna()

# Re-separate features and targets after cleaning data
X = data_cleaned[['Month', 'Year', 'Minimum price', 'Maximum price']]
y = data_cleaned['Average price']

# Splitting data into training data and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# model initialization
model = LinearRegression()

# Training the model
model.fit(X_train, y_train)

# Predicting test data
y_pred = model.predict(X_test)


# Model evaluation
mae = mean_absolute_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)

print(f'MAE: {mae}')
print(f'R-squared: {r2}')
```

```
MAE: 1798.9153818518405
R-squared: 0.9996180949113733
```

Results and Evaluation

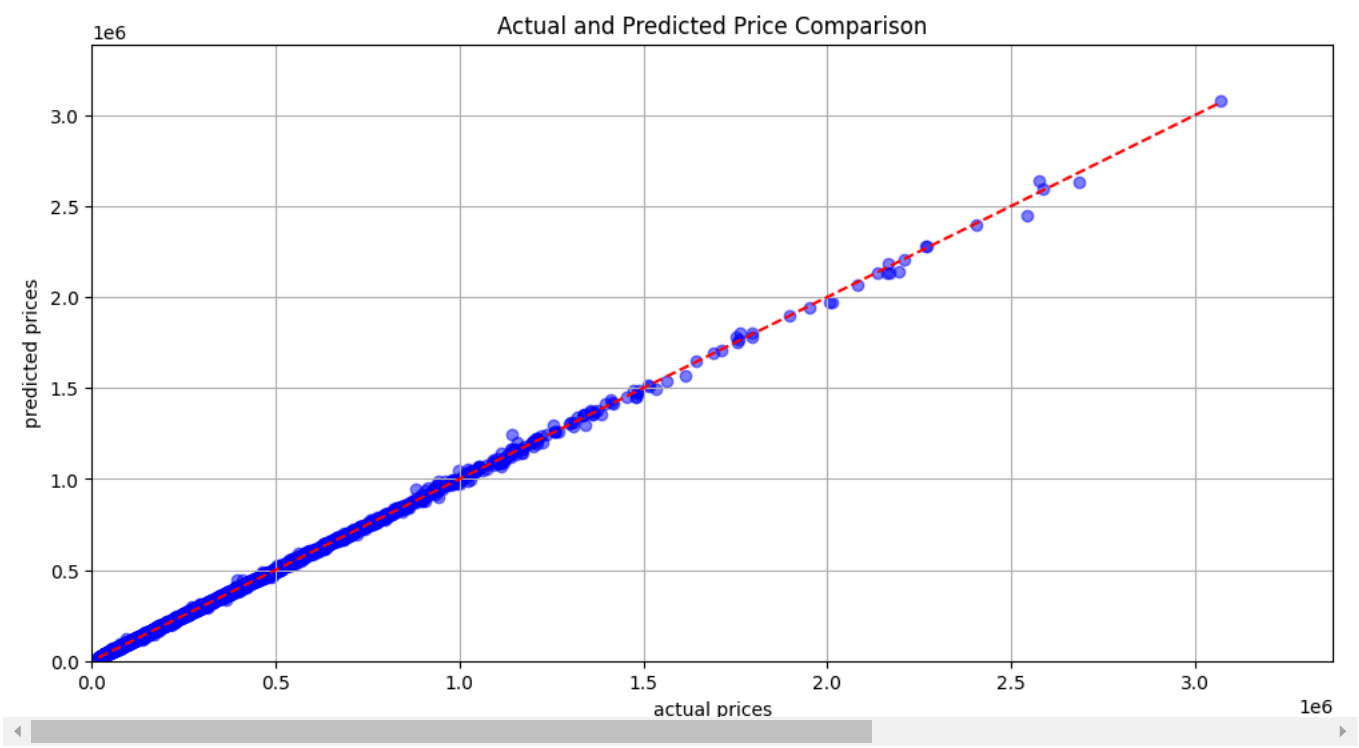
The trained model produced MAE: 1798.92 and R-squared: 0.9996, indicating that the model is quite good at predicting prices.

```
# Displays predicted prices and actual prices
predicted_prices = pd.DataFrame({'actual prices': y_test, 'predicted prices': y_pred})
predicted_prices.head(10)
```

	actual prices	predicted prices
44486	159000.0	158477.625355
65205	76000.0	75056.873658
32322	297000.0	294187.661874
54606	77000.0	77629.930763
56572	48000.0	48582.798987
1192	739000.0	738586.234133
33173	322000.0	325300.939588
69514	38000.0	37794.404638
58584	73000.0	74162.868498
45238	96000.0	99076.197277

Next steps: Generate code with `predicted_prices` | View recommended plots | New interactive sheet

```
import matplotlib.pyplot as plt

# Create a DataFrame to store actual prices and predicted prices.
results = pd.DataFrame({'actual prices': y_test, 'predicted prices': y_pred})

# Setting the plot size
plt.figure(figsize=(12, 6))

# Plot of actual price and predicted price
plt.scatter(results['actual prices'], results['predicted prices'], color='blue', alpha=0.5)
plt.plot([results['actual prices'].min(), results['actual prices'].max()],
         [results['actual prices'].min(), results['actual prices'].max()],
         color='red', linestyle='--')

# Add title and label
plt.title('Actual and Predicted Price Comparison')
plt.xlabel('actual prices')
plt.ylabel('predicted prices')
plt.grid()
plt.xlim(0, results['actual prices'].max() * 1.1)
plt.ylim(0, results['predicted prices'].max() * 1.1)

# Showing plot
plt.show()
```



Conclusion

This project successfully built a model that is able to predict used car prices with high accuracy. For further research, we can try a more complex model or add more features.

Reference

https://www.kaggle.com/datasets/muhammedzidan/car-prices-market?resource=download&select=used_car_prices.csv