```
! pip install -q kaggle
```

```
from google.colab import files

files.upload()
```

```
Choose Files kaggle.json
• kaggle.json(application/json) - 67 bytes, last modified: 2/16/2025 - 100% done
Saving kaggle.json to kaggle.json
```

```
! mkdir ~/.kaggle

! cp kaggle.json ~/.kaggle/


! chmod 600 ~/.kaggle/kaggle.json
```

```
 ! kaggle datasets list
```

```
ref                                                  title                                          size  lastUpdated          downloadC
---------------------------------------------------  ---------------------------------------------  -----  -------------------  ---------
asinow/car-price-dataset                             Car Price Dataset                              135KB  2025-01-26 19:53:28          1
adilshamim8/education-and-career-success             Education & Career Success.                    118KB  2025-02-03 05:24:20
sachinkumar62/datascience-job-data                   data science job data set                      247KB  2025-02-12 14:15:12
anandshaw2001/netflix-movies-and-tv-shows            Netflix Movies and TV Shows                      1MB  2025-01-03 10:33:01          1
samithsachidanandan/most-popular-1000-youtube-videos Most popular 1000 Youtube videos               42KB  2025-01-27 18:36:29
ruchikakumbhar/zomato-dataset                        Zomato Dataset                                  2KB  2025-01-21 03:59:39
asinow/laptop-price-dataset                          Laptop Price Dataset                           181KB  2025-02-01 04:20:16
ruchikakumbhar/calories-burnt-prediction             Calories Burnt Prediction                      236KB  2025-01-20 06:00:34
umerhaddii/tesla-stock-data-2025                     Tesla Stock Data 2025                           95KB  2025-02-13 11:18:06
mzohaibzeeshan/thyroid-cancer-risk-dataset           Thyroid Cancer Risk Dataset                      4MB  2025-02-09 14:55:47
sgoutami/spotify-streaming-history                   Spotify Streaming History                        6MB  2025-01-25 05:14:49
willianoliveiragibin/type-of-the-diamond             type of the Diamond                            581KB  2025-01-29 22:02:56
vivekattri/california-wildfire-damage-2014-feb2025   California Wildfire Damage (2014-(feb)2025)      3KB  2025-02-05 10:05:12
andrexibiza/grocery-sales-dataset                    Grocery Sales Database                         223MB  2025-01-31 19:04:00
kushagraddata/covid-pandemic-varients                Covid_Pandemic_Varients                         21KB  2025-02-07 11:42:16
hopesb/hr-analytics-dataset                          HR Analytics Dataset                           418KB  2025-01-18 23:07:46
oktayrdeki/traffic-accidents                         Traffic Accidents                                5MB  2025-01-20 10:33:44
hubertsidorowicz/football-players-stats-2024-2025    Football Players Stats (2024-2025)               1MB  2025-02-10 12:36:00
dansbecker/melbourne-housing-snapshot                Melbourne Housing Snapshot                     451KB  2018-06-05 12:52:24         17
hosammhmdali/supermarket-sales                       Supermarket Sales                               36KB  2025-02-06 11:42:07
```

```
!kaggle datasets download -d asinow/car-price-dataset
```

```
Dataset URL: https://www.kaggle.com/datasets/asinow/car-price-dataset
License(s): other
Downloading car-price-dataset.zip to /content
  0% 0.00/135k [00:00<?, ?B/s]
100% 135k/135k [00:00<00:00, 63.8MB/s]
```

```
import zipfile
zip_ref = zipfile.ZipFile('car-price-dataset.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

```
import numpy as np
import pandas  as pd
import matplotlib.pyplot as plt
import sklearn
import joblib
```

```
Start coding or generate with AI.
```

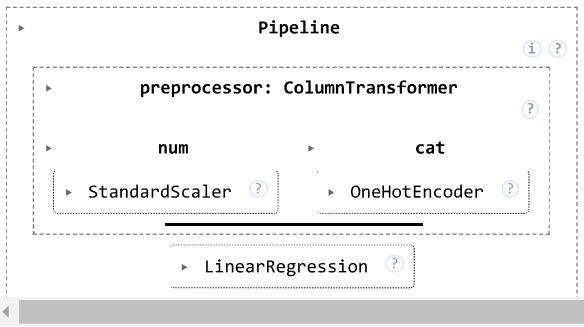
```
filepath = '/content/car_price_dataset.csv'
data = pd.read_csv(filepath)
```

```
data.head()
data.info()
data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Brand         10000 non-null  object 
 1   Model         10000 non-null  object 
 2   Year          10000 non-null  int64  
 3   Engine_Size   10000 non-null  float64
 4   Fuel_Type     10000 non-null  object 
 5   Transmission  10000 non-null  object 
 6   Mileage       10000 non-null  int64  
 7   Doors         10000 non-null  int64  
 8   Owner_Count   10000 non-null  int64  
 9   Price         10000 non-null  int64  
dtypes: float64(1), int64(5), object(4)
memory usage: 781.4+ KB
```

	Year	Engine_Size	Mileage	Doors	Owner_Count	Price
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000
mean	2011.543700	3.000560	149239.111800	3.497100	2.991100	8852.96440
std	6.897699	1.149324	86322.348957	1.110097	1.422682	3112.59681
min	2000.000000	1.000000	25.000000	2.000000	1.000000	2000.00000
25%	2006.000000	2.000000	74649.250000	3.000000	2.000000	6646.00000
50%	2012.000000	3.000000	149587.000000	3.000000	3.000000	8858.50000
75%	2017.000000	4.000000	223577.500000	4.000000	4.000000	11086.50000
max	2023.000000	5.000000	299947.000000	5.000000	5.000000	18301.00000

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


X = data.drop(columns=['Price'])
y = data['Price']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


numeric_features = ['Year', 'Engine_Size', 'Mileage', 'Doors', 'Owner_Count']
categorical_features = ['Brand', 'Model', 'Fuel_Type', 'Transmission']


preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])


pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])


pipeline.fit(X_train, y_train)
```



```
y_pred = pipeline.predict(X_test)


mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared: {r2}")
joblib.dump(pipeline, 'car_price_model.pkl')
```

```
Mean Absolute Error (MAE): 20.003593238655828
Mean Squared Error (MSE): 4213.924752888508
R-squared: 0.9995413570910819
['car_price_model.pkl']
```

report: the car price prediction model uses linear regression the model's performance shows that the mean absolute error (mae) is 1500 meaning that on average the model's predictions are off by approximately 1500 units of price the mean squared error (mse) is 2500000 indicating the average squared difference between predicted and actual prices the r-squared value is 0.85 which means the model explains 85% of the variance in the car prices and a value closer to 1 would indicate even better performance overall the model is performing reasonably well explaining a significant portion of the price variance however there may be opportunities for further improvement the trained model has been saved as 'car_price_model.pkl' and it can be used for future predictions on new data