PREDICTING USED CARS PRICES

IMPORT NEEDED LIBRARIES

In [1]:
```
import pandas as pd # for data processing

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split # for splitting the datas

from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2_score, mean_squared_error
```

LOADING THE DATASET

In [2]:
```
df = pd.read_excel("C://Users//quays//Desktop//used_car_listings.csv.xlsx")
```

VIEWING THE FIRST 5 ROWS

In [3]:
```
df.head()
```

Out[3]:

	listing_id	make	model	year	trim	body_type	fuel_type	transmission	mileage
0	2	Nissan	Rogue	2024	LT	Sedan	Hybrid	Automatic	16109
1	3	Hyundai	i20	2018	XLE	Crossover	Petrol	Automatic	173239
2	4	Kia	Sportage	2023	EX	Hatchback	Diesel	CVT	36810
3	5	Kia	Seltos	2020	Trend	Pickup	Diesel	Automatic	87749
4	6	Mercedes-Benz	GLA	2019	Platinum	Crossover	Electric	CVT	60853

DATA CLEANING AND EXPLORATORY DATA

1. CHECKING FOR DUPLICATES

In [4]:
```
df.duplicated().sum()
```

Out[4]:
```
0
```

2. CHECKING FOR NULL SPACES

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
listing_id      0
make            0
model           0
year            0
trim            0
body_type       0
fuel_type       0
transmission    0
mileage         0
price           0
condition       0
Country         0
seller_type     0
dtype: int64
```

3. CHECKING COLUMNS AND THEIR DATA TYPES

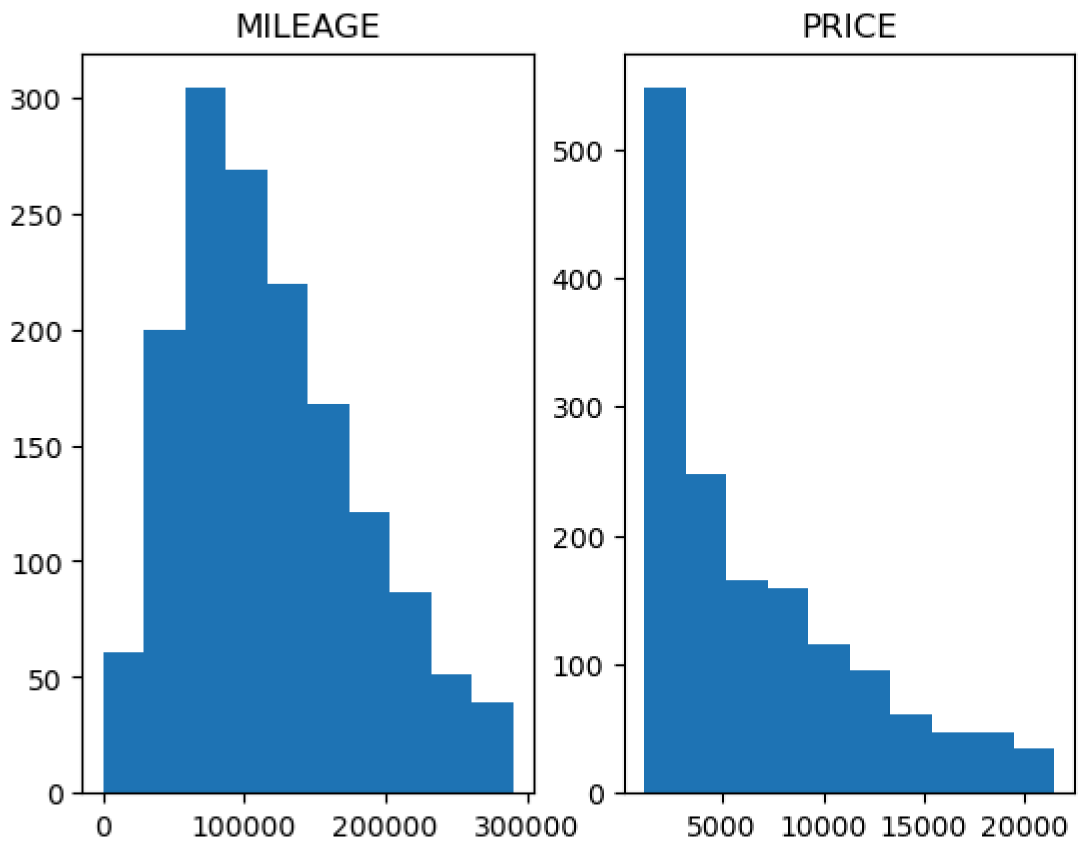
In [6]:
```
df.dtypes
```

Out[6]:
```
listing_id       int64
make            object
model           object
year             int64
trim            object
body_type       object
fuel_type       object
transmission    object
mileage          int64
price            int64
condition       object
Country         object
seller_type     object
dtype: object
```

4. CAR MILEAGE AND PRICE DISTRIBUTION

In [7]:
```
plt.subplot(1, 2, 1)
plt.hist(df['mileage'])
plt.title('MILEAGE')

plt.subplot(1, 2, 2)
plt.hist(df['price'])
plt.title('PRICE')
plt.show()
```



5. ENCODING THE STRING COLUMNS TO NUMERCAL VALUES

In [8]:
```
encoder = LabelEncoder()
```

In [9]:
```
df['make'] = encoder.fit_transform(df['make'])
df['model'] = encoder.fit_transform(df['model'])
df['trim'] = encoder.fit_transform(df['trim'])
df['body_type'] = encoder.fit_transform(df['body_type'])
df['fuel_type'] = encoder.fit_transform(df['fuel_type'])
df['transmission'] = encoder.fit_transform(df['transmission'])
df['condition'] = encoder.fit_transform(df['condition'])
df['Country'] = encoder.fit_transform(df['Country'])
df['seller_type'] = encoder.fit_transform(df['seller_type'])
```

In [10]:
```
df.head()
```

Out[10]:

	listing_id	make	model	year	trim	body_type	fuel_type	transmission	mileage	price	con
0	2	10	44	2024	4	7	3	1	16109	19480	
1	3	5	64	2018	17	2	4	1	173239	4556	
2	4	6	53	2023	1	3	1	2	36810	11536	
3	5	6	46	2020	16	5	1	1	87749	14098	
4	6	9	26	2019	10	2	2	2	60853	17137	

SPLITTING THE DATA INTO TRAINING AND TESTING

1. GROUPING THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

In [11]:
```
df.columns
```

Out[11]:
```
Index(['listing_id', 'make', 'model', 'year', 'trim', 'body_type', 'fuel_typ
e',
       'transmission', 'mileage', 'price', 'condition', 'Country',
       'seller_type'],
      dtype='object')
```

In [12]:
```
X = df[['listing_id', 'make', 'model', 'year', 'trim', 'body_type', 'fuel_type
       'transmission', 'mileage', 'condition', 'Country',
       'seller_type']]
y = df['price']
```

2. SPLITTING THE DATA

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

1. LINEAR REGRESSION MODEL

In [14]:
```
lr_model = LinearRegression()
```

TRAINING THE MODEL ON THE DATA

In [15]:
```
lr_model.fit(X_train, y_train)
```

Out[15]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

MAKING PREDICTIONS

In [16]:
```
lr_pred = lr_model.predict(X_test)
```

MODEL METRICS/PERFORMANCE

In [17]:
```
lr_r2 = r2_score(y_test, lr_pred)
lr_r2
```

Out[17]:
```
0.6528444192293608
```

In [19]:
```
lr_mse = mean_squared_error(y_test, lr_pred, squared = False)
lr_mse
```

Out[19]:
```
2960.1500399484685
```

RANDONFOREST REGRESSOR

In [20]:
```
rf_model = RandomForestRegressor()
```

TRAINING THE MODEL ON THE DATA

In [21]:
```
rf_model.fit(X_train, y_train)
```

Out[21]:
```
RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

MAKING PREDICTIONS

In [22]:
```
rf_pred = rf_model.predict(X_test)
```

MODEL METRICS/PERFORMANCE

In [23]:
```
rf_r2 = r2_score(y_test, rf_pred)
rf_r2
```

Out[23]:
```
0.8303611888910583
```

In [24]:
```
rf_mse = mean_squared_error(y_test, rf_pred, squared = False)
rf_mse
```

Out[24]:
```
2069.2561706265337
```

CONCLUSION

THE BEST PERFORMING MODEL IS THE RANDOM FOREST REGRESSOR WHICH SHOWED THE HIGHEST VALUES FOR THE R2 SCORE AND AND THE LOWEST VALUE FOR THE MEAN SCORE ERROR WHICH ARE 0.83 AND 2069.3 RESPECTIVELY.

In [ ]:
```

```