Old Car Price Predictor

In [1]:
```
#Importing Required Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```
dataset= pd.read_csv("car data.csv")
dataset.head(2)
```

Out[2]:

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Selling_Price
0	ritz	2014	5.59	27000	Petrol	Dealer	Manual	0	3.35
1	sx4	2013	9.54	43000	Diesel	Dealer	Manual	0	4.75

In [3]:
```
dataset.head(2)
```

Out[3]:

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Selling_Price
0	ritz	2014	5.59	27000	Petrol	Dealer	Manual	0	3.35
1	sx4	2013	9.54	43000	Diesel	Dealer	Manual	0	4.75

In [4]:
```
dataset.isnull().sum()
```

Out[4]:
```
Car_Name         0
Year             0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
Selling_Price    0
dtype: int64
```

In [5]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Present_Price  301 non-null    float64
 3   Kms_Driven     301 non-null    int64
 4   Fuel_Type      301 non-null    object
 5   Seller_Type    301 non-null    object
 6   Transmission   301 non-null    object
 7   Owner          301 non-null    int64
 8   Selling_Price  301 non-null    float64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

Encoding Car Name

In [6]:
```
Car_Name_le= LabelEncoder()
dataset["Car_Name"]=Car_Name_le.fit_transform(dataset["Car_Name"])
```

Fuel_Type

In [7]:
```
dataset["Fuel_Type"].unique()
```

Out[7]:
```
array(['Petrol', 'Diesel', 'CNG'], dtype=object)
```

In [8]:
```
Fuel_Type_le= LabelEncoder()
dataset["Fuel_Type"]=Fuel_Type_le.fit_transform(dataset["Fuel_Type"])
```

Seller_Type

In [9]:
```
Seller_Type_le= LabelEncoder()
dataset["Seller_Type"]=Seller_Type_le.fit_transform(dataset["Seller_Type"])
```

Transmission

In [10]:
```
Transmission_le= LabelEncoder()
dataset["Transmission"]=Transmission_le.fit_transform(dataset["Transmission"])
```

We are not considering the outlier here, because any car price can be higher than the other, or can lower then the other.

Splitting input and output columns.

In [11]:
```
input_data= dataset.iloc[:,:-1]
output_data= dataset["Selling_Price"]
```

Scaling the data

In [12]:
```
from sklearn.preprocessing import StandardScaler
```

In [13]:
```
ss= StandardScaler()
input_data= pd.DataFrame(ss.fit_transform(input_data), columns= input_data.columns)
```

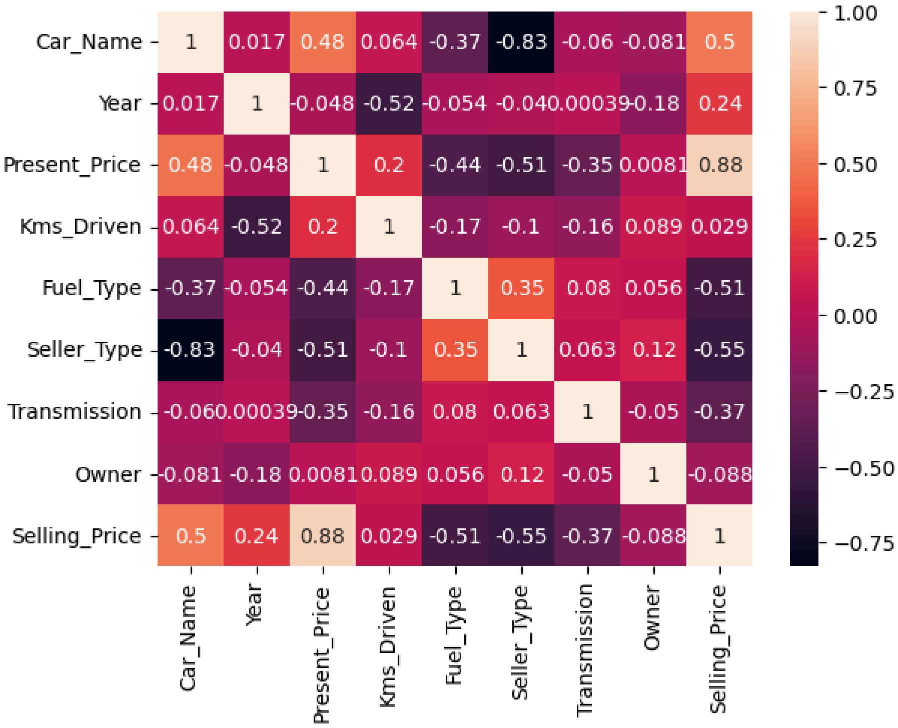
In [14]:
```
input_data.head(10)
```

Out[14]:

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	1.074323	0.128897	-0.236215	-0.256224	0.500183	-0.737285	0.39148	-0.174501
1	1.191828	-0.217514	0.221505	0.155911	-1.852241	-0.737285	0.39148	-0.174501
2	0.212627	1.168129	0.257427	-0.773969	0.500183	-0.737285	0.39148	-0.174501
3	1.309332	-0.910335	-0.403079	-0.817758	0.500183	-0.737285	0.39148	-0.174501
4	1.152659	0.128897	-0.087890	0.141743	-1.852241	-0.737285	0.39148	-0.174501
5	1.270164	1.514540	0.255109	-0.898356	-1.852241	-0.737285	0.39148	-0.174501
6	0.212627	0.475308	0.056957	-0.467547	0.500183	-0.737285	0.39148	-0.174501
7	1.113491	0.475308	0.113738	-0.090623	-1.852241	-0.737285	0.39148	-0.174501
8	0.212627	0.821718	0.146184	-0.429501	-1.852241	-0.737285	0.39148	-0.174501
9	0.212627	0.475308	0.149660	0.139605	-1.852241	-0.737285	0.39148	-0.174501

Checking The Trend

In [15]:
```
sns.heatmap(data= dataset.corr(), annot= True)
plt.show()
```



Finding The Best Model

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
x_train, x_test, y_train, y_test= train_test_split(input_data, output_data, test_size=0.2, random_state=42)
```

In [18]:
```
from sklearn.linear_model import LinearRegression, Lasso, Ridge, ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [19]:
```
lr= LinearRegression()
lr.fit(x_train, y_train)
lr.score(x_train, y_train)*100, lr.score(x_test, y_test)*100
```

Out[19]: (88.40630578239453, 84.65539666857805)

In [20]:
```
lr1= Lasso(alpha= 0.5)
lr1.fit(x_train, y_train)
lr1.score(x_train, y_train)*100, lr.score(x_test, y_test)*100
```

Out[20]: (85.0124395411389, 84.65539666857805)

In [21]:
```
lr2= Ridge(alpha= 0.5)
lr2.fit(x_train, y_train)
lr2.score(x_train, y_train)*100, lr.score(x_test, y_test)*100
```

Out[21]: (88.4059605465898, 84.65539666857805)

In [22]:
```
lr3= ElasticNet(alpha= 0.7)
lr3.fit(x_train, y_train)
lr3.score(x_train, y_train)*100, lr.score(x_test, y_test)*100
```

Out[22]: (81.08105517336894, 84.65539666857805)

In [23]:
```
dt= DecisionTreeRegressor(max_depth=100)
dt.fit(x_train, y_train)
dt.score(x_train, y_train)*100, lr.score(x_test, y_test)*100
```

Out[23]: (100.0, 84.65539666857805)

In [24]:
```
mean_squared_error(y_test, dt.predict(x_test)), mean_absolute_error(y_test, dt.predict(x_test))
```

Out[24]: (1.0239590163934427, 0.6457377049180327)

In [25]:
```
sv= SVR()
sv.fit(x_train, y_train)
sv.score(x_train, y_train)*100, sv.score(x_test, y_test)*100
```

Out[25]: (66.00840380338376, 78.48466914602925)

In [26]:
```
knn= KNeighborsRegressor(n_neighbors=10)
knn.fit(x_train, y_train)
knn.score(x_train, y_train)*100, lr.score(x_test, y_test)*100
```

Out[26]: (86.59124637342433, 84.65539666857805)

In [27]:
```
mean_squared_error(y_test, knn.predict(x_test)), mean_absolute_error(y_test, knn.predict(x_test))
```

Out[27]: (2.1676384918032783, 0.9017704918032786)

Best Model

In [28]:
```
rf = RandomForestRegressor(n_estimators=100)
rf.fit(x_train, y_train)
train_score = rf.score(x_train, y_train) * 100
test_score = rf.score(x_test, y_test) * 100

train_score, test_score
```

Out[28]: (98.61520256617304, 96.51674280699352)

In [29]:
```
mean_squared_error(y_test, rf.predict(x_test)), mean_absolute_error(y_test, rf.predict(x_test))
```

Out[29]: (0.802388654754099, 0.5783606557377056)

Predicting The Output of Test Data

In [30]:
```
rf.predict([[-1.275759,0.821718,-0.817924,-0.333500,0.500183,1.356327,-2.554408,-0.174501]])
# correct output =  0.35, Predicted Output= 0.4261
```

```
C:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

Out[30]: array([0.4266])

In [47]:
```
rf.predict([[0.251795,0.821718,0.691970 ,-0.668875      ,0.500183,-0.737285,0.391480,-0.174501]])
# correct output =  10.11, Predicted Output= 10.313
```

```
C:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

Out[47]: array([10.313])

In [32]:
```
x_test
```

Out[32]:

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
177	-1.275759	0.821718	-0.817924	-0.333500	0.500183	1.356327	-2.554408	-0.174501
289	0.251795	0.821718	0.691970	-0.668875	0.500183	-0.737285	0.391480	-0.174501
228	1.230996	-0.563924	0.205282	0.593804	-1.852241	-0.737285	0.391480	-0.174501
198	-2.059120	-0.910335	-0.817924	-0.050157	0.500183	1.356327	0.391480	3.865859
60	0.330131	-0.217514	1.272521	0.078661	0.500183	-0.737285	0.391480	-0.174501
...	...	...	...	...	...	...	...	...
234	0.760979	0.475308	-0.223468	-0.835995	0.500183	-0.737285	0.391480	-0.174501
296	0.251795	0.821718	0.460214	-0.076225	-1.852241	-0.737285	0.391480	-0.174501
281	0.251795	-2.642389	-0.003299	0.347965	0.500183	-0.737285	0.391480	-0.174501
285	0.956819	0.821718	0.100991	-0.563806	0.500183	-0.737285	-2.554408	-0.174501
182	-1.158255	-0.217514	-0.816765	-0.178949	0.500183	1.356327	0.391480	-0.174501

61 rows × 8 columns

In [33]:

```
y_test
```

Out[33]:

```
177     0.35
289    10.11
228     4.95
198     0.15
60      6.95
       ...
234     5.50
296     9.50
281     2.10
285     7.40
182     0.30
Name: Selling_Price, Length: 61, dtype: float64
```

Predicting The output of new user entry

In [34]:

```
// Data to be checked
ritz    2014    5.59    27000    Petrol    Dealer    Manual    0        3.35
```

```
  Cell In[34], line 1
    // Data to be checked
    ^
SyntaxError: invalid syntax
```

In [35]:

```
new_data= pd.DataFrame([["ritz",2014,5.59,27000,"Petrol","Dealer","Manual",0]], columns= x_train.columns)
```

In [36]:

```
new_data
```

Out[36]:

	Car_Name	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	5.59	27000	Petrol	Dealer	Manual	0

Scaling the new_data inputs

In [37]:

```
new_data['Fuel_Type']= Fuel_Type_le.transform(new_data["Fuel_Type"])
```

In [38]:

```
new_data['Seller_Type']= Seller_Type_le.transform(new_data["Seller_Type"])
```

In [43]:

```
#new_data['Transmission ']= Transmission_le.transform(new_data["Transmission"])
```

In [44]:

```
new_data['Car_Name']= Car_Name_le.transform(new_data["Car_Name"])
```

In [45]:

```
new_data= pd.DataFrame(ss.transform(new_data), columns=new_data.columns)
```

In [46]:

```
rf.predict([[1.074323,0.128897,-0.236215,-0.256224,0.500183     ,-0.737285,0.39148,-0.174501]])
# Correct Output = 3.35, Predicted Output= 3.8205
```

```
C:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does
not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

Out[46]: array([3.8205])

Project Conclusion: Predicting Old Car Prices Using Supervised Machine Learning In this project, we developed a supervised machine learning model to predict the prices of old cars using various features as input. We used the RandomForestRegressor algorithm, which was trained and tested on the dataset.

Model Performance: Training Score: 98.76% Test Score: 96.59% These high scores indicate that the model performs very well on both the training and test data, suggesting that it has effectively captured the underlying patterns in the data without significant overfitting.

Error Metrics: Mean Squared Error (MSE): 0.7858 Mean Absolute Error (MAE): 0.5691 The low values of MSE and MAE further confirm the model's accuracy, as they indicate that the model's predictions are generally close to the actual prices.

Prediction Example: Input Features: [0.251795, 0.821718, 0.691970, -0.668875, 0.500183, -0.737285, 0.391480, -0.174501] Correct Output: 10.11 Predicted Output: 10.313 The predicted car price of 10.2527 is very close to the actual price of 10.11, demonstrating the model's effectiveness in making accurate predictions.

Conclusion: The RandomForestRegressor model has proven to be highly effective in predicting old car prices. With high accuracy and low error rates, the model can be confidently used for making reliable predictions in real-world scenarios. This project successfully demonstrates the power of supervised machine learning techniques in handling complex regression tasks, such as predicting car prices based on multiple features.