Program:-

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegressor
from sklearn.preprocessing import StandardScaler
from google.colab import drive
drive.mount('/content/drive')
```

```
dataset = pd.read_csv("/content/drive/MyDrive/dataset/ford.csv")
dataset.head()
```

#Dataset(head):-

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
0	Fiesta	2017	12000	Automatic	15944	Petrol	150	57.7	1.0
1	Focus	2018	14000	Manual	9083	Petrol	150	57.7	1.0
2	Focus	2017	13000	Manual	12456	Petrol	150	57.7	1.0
3	Fiesta	2019	17500	Manual	10460	Petrol	145	40.3	1.5
4	Fiesta	2019	16500	Automatic	1482	Petrol	145	48.7	1.0

#Dataset(tail):-

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
17961	B-MAX	2017	8999	Manual	16700	Petrol	150	47.1	1.4
17962	B-MAX	2014	7499	Manual	40700	Petrol	30	57.7	1.0
17963	Focus	2015	9999	Manual	7010	Diesel	20	67.3	1.6
17964	KA	2018	8299	Manual	5007	Petrol	145	57.7	1.2
17965	Focus	2015	8299	Manual	5007	Petrol	22	57.7	1.0

#Rows & Columns of the Dataset:-

```
dataset.shape
```

```
(17966, 9)
```

#Info about the Dataset:-

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17966 entries, 0 to 17965
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   model         17966 non-null  object 
 1   year          17966 non-null  int64  
 2   price         17966 non-null  int64  
 3   transmission  17966 non-null  object 
 4   mileage       17966 non-null  int64  
 5   fuelType      17966 non-null  object 
 6   tax           17966 non-null  int64  
 7   mpg           17966 non-null  float64
 8   engineSize    17966 non-null  float64
dtypes: float64(2), int64(4), object(3)
memory usage: 1.2+ MB
```

#Datapreprocessing:-

```
#printing the unique value
print(dataset['transmission'].unique())
print(dataset['fuelType'].unique())
```

```
['Automatic' 'Manual' 'Semi-Auto']
['Petrol' 'Diesel' 'Hybrid' 'Electric' 'Other']
```

```
#Encoding the categorical transmission and fuelType column
dataset.replace({'transmission':{'Automatic':0,'Manual':1,'Semi-Auto':2}},inplace=True)
dataset.replace({'fuelType':{'Petrol':0,'Diesel':1,'Hybrid':2,'Electric':3,'Other':4}},inplace=True)
```

#Checking for the conversion:-

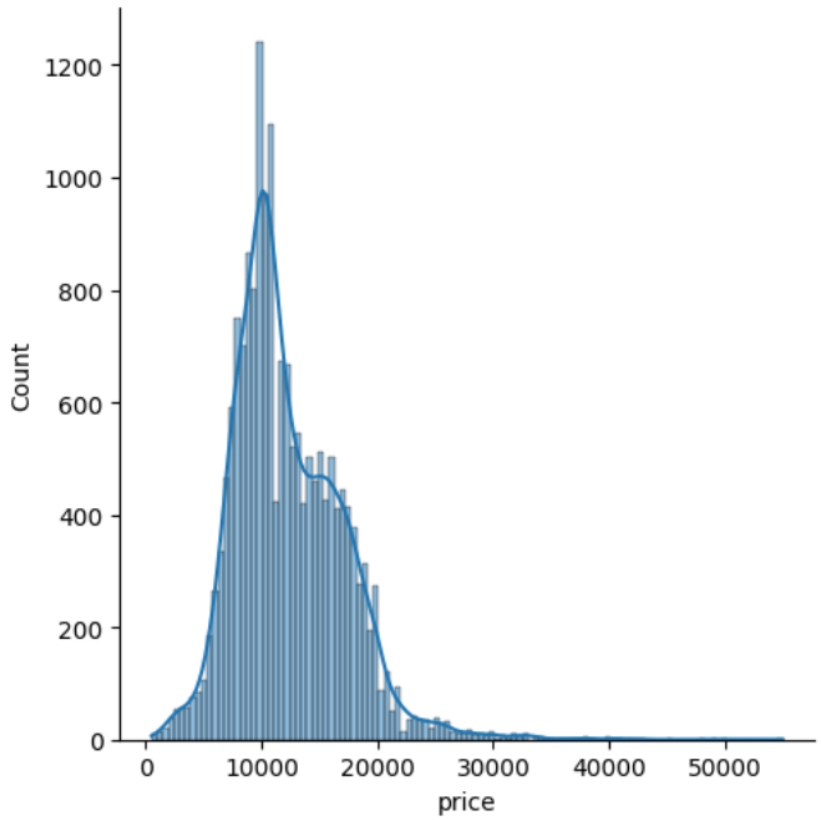
```
dataset.head(5)
```

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
0	Fiesta	2017	12000	0	15944	0	150	57.7	1.0
1	Focus	2018	14000	1	9083	0	150	57.7	1.0
2	Focus	2017	13000	1	12456	0	150	57.7	1.0
3	Fiesta	2019	17500	1	10460	0	145	40.3	1.5
4	Fiesta	2019	16500	0	1482	0	145	48.7	1.0

#Creating a distribution plot (histogram) for the 'price' column in your dataset

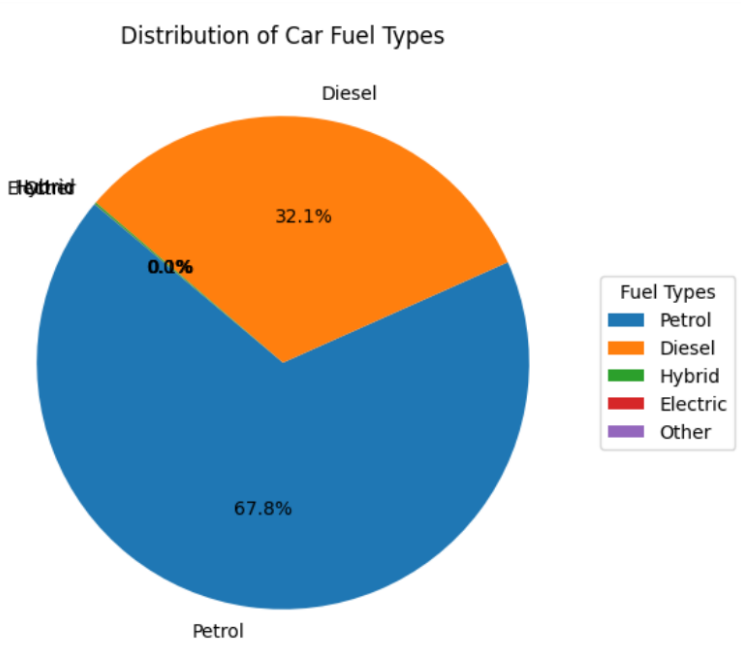
```
# Create a distribution plot for the 'price' column in the 'dataset'
# The 'kde=True' parameter adds a Kernel Density Estimate plot
import seaborn as sns
sns.displot(dataset['price'],kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7c90c68a8be0>
```



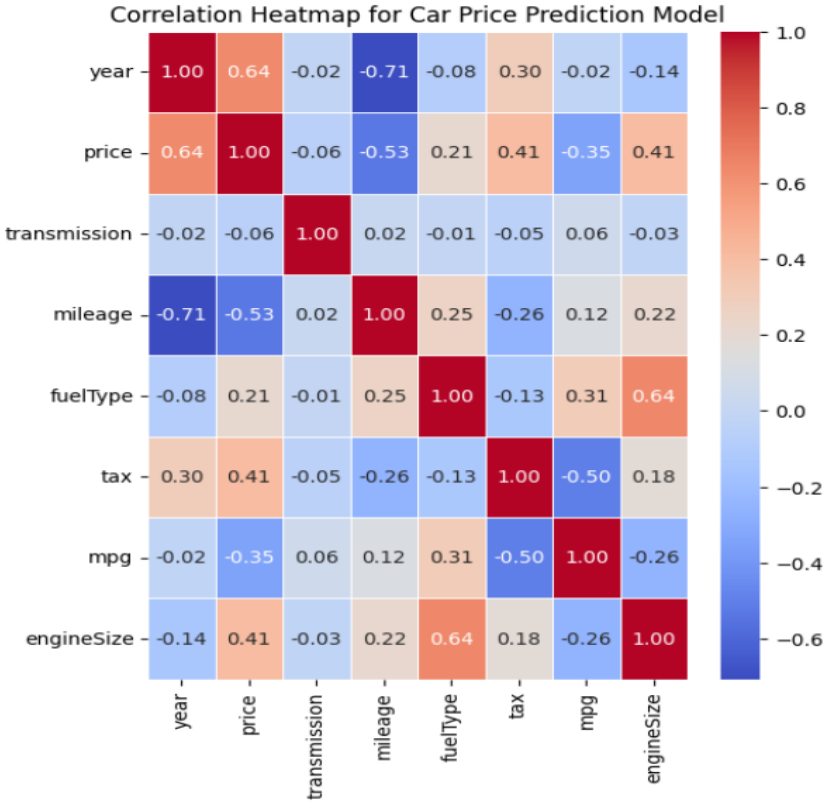
#Distribution of Car's Fuel Type

```
# Count the occurrences of each fuel type
import matplotlib.pyplot as plt
fuel_counts = dataset['fuelType'].value_counts()
# Plotting a pie chart
plt.figure(figsize=(6, 6))
plt.pie(fuel_counts, labels=fuel_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Car Fuel Types')
plt.legend(fuel_counts.index, title="Fuel Types", loc="center left", bbox_to_anchor=(1, 0.5))
plt.show()
```



Calculate the correlation matrix for the dataset

```
correlation_matrix = dataset.corr()
# Create a heatmap
plt.figure(figsize=(6, 7))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=.5)
plt.title("Correlation Heatmap for Car Price Prediction Model")
plt.show()
```



Transform (standardize) the data using the computed mean and standard deviation

```
# Create a StandardScaler instance
ss = StandardScaler()
ss.fit(x)
standarized_x = ss.transform(x)
print(standarized_x)
x = standarized_x
```

```
[[ 0.06512772 -2.67003231 -0.38099808 ...  0.59135805 -0.02044162
  -0.81138621]
 [ 0.55286624  0.04135139 -0.73335899 ...  0.59135805 -0.02044162
  -0.81138621]
 [ 0.06512772  0.04135139 -0.56013157 ...  0.59135805 -0.02044162
  -0.81138621]
 ...
 [-0.91034931  0.04135139 -0.83982222 ... -1.50505332  0.92766777
   0.57636151]
 [ 0.55286624  0.04135139 -0.94269045 ...  0.51072684 -0.02044162
  -0.34880364]
 [-0.91034931  0.04135139 -0.94269045 ... -1.47280084 -0.02044162
  -0.81138621]]
```

#Splitting the dataset

```
# Split the data into training and testing sets
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state=42)
print(x.shape,x_train.shape,x_test.shape)
print(y.shape,y_train.shape,y_test.shape)
```

```
(17966, 7) (14372, 7) (3594, 7)
(17966,) (14372,) (3594,)
```

#Model Building

```
#model building
training_score = []
testing_score = []
```

#Function for Model_Predictions

```
def model_prediction(model):
    model.fit(x_train,y_train)
    x_train_pred = model.predict(x_train)
    x_test_pred = model.predict(x_test)
    a = r2_score(y_train,x_train_pred)*100
    b = r2_score(y_test,x_test_pred)*100
    training_score.append(a)
    testing_score.append(b)

    print(f"r2_Score of {model} model on Training Data is:",a)
    print(f"r2_Score of {model} model on Testing Data is:",b)
```

#Model Prediction for Linear Regression

```
model_prediction(LinearRegression())
```

```
r2_Score of LinearRegression() model on Training Data is: 73.56949202165862
r2_Score of LinearRegression() model on Testing Data is: 73.71634634107636
```

#Model Prediction for Gradient Boosting Regressor

```
model_prediction(GradientBoostingRegressor())
```

```
r2_Score of GradientBoostingRegressor() model on Training Data is: 89.90114252003612
r2_Score of GradientBoostingRegressor() model on Testing Data is: 88.62935683860421
```

#Model Prediction for Random Forest Regressor

```
model_prediction(RandomForestRegressor())
```

```
r2_Score of RandomForestRegressor() model on Training Data is: 98.68878089675587
r2_Score of RandomForestRegressor() model on Testing Data is: 90.77612139855874
```

#Model Prediction for Decision Tree Regressor

```
model_prediction(DecisionTreeRegressor())
```

```
r2_Score of DecisionTreeRegressor() model on Training Data is: 99.9600561008004
r2_Score of DecisionTreeRegressor() model on Testing Data is: 86.2348809121382
```

```
models = ["Linear Regression","Decision Tree","Random Forest","Gradient Boost"]
```

#Making Dataframe for all Predicted models

```
df = pd.DataFrame({"Algorithms":models,
                   "Training Score":training_score,
                   "Testing Score":testing_score})
```

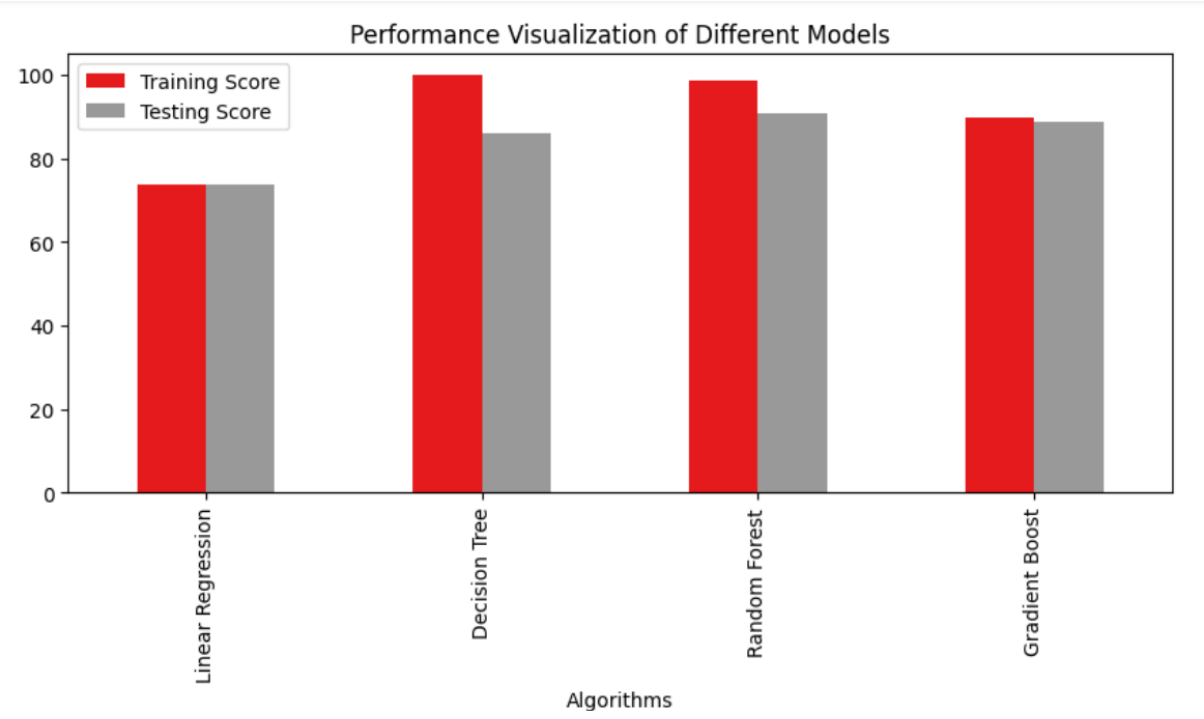
#R2 Score of all Predicted Models

```
df
```

	Algorithms	Training Score	Testing Score
0	Linear Regression	73.569492	73.716346
1	Decision Tree	99.960056	86.110966
2	Random Forest	98.696954	90.791966
3	Gradient Boost	89.901143	88.630838

#Plotting Bar for Visualization

```
# Use the plot method of DataFrame to create a bar plot
df.plot(x="Algorithms",y=["Training Score","Testing Score"], figsize=(10,4),kind="bar",
        title="Performance Visualization of Different Models",colormap="Set1")
plt.show()
```



Conclusion : Based on the various Model's Prediction Decision Tree predicts better on this dataset

CAR PRICE PREDICTION MODEL

Aim: The aim of our project is to predict car prices based on various features using different regression techniques (Linear Regression, Random Forest Regressor, Gradient Boosting Regressor, Decision Tree Regressor).

Steps Involved:

1. Utilize the pandas read_csv() function to import the car dataset.
2. Filter the data to include only relevant features and records, focusing on aspects that contribute significantly to car prices.
3. Check for any null values in the dataset and replace them with appropriate values, such as the mean or median for numerical features.
4. Create a heatmap to visualize the correlation between different features. Remove features with high correlation to avoid multicollinearity.
5. Split the dataset into training and testing sets to train and evaluate the regression models.
6. Regression Models: Apply various regression algorithms to predict car prices:
    - Linear Regression
    - Random Forest Regression
    - Gradient Boosting Regression
    - Decision Tree Regression
7. **Model Evaluation:** Assess the accuracy of each model using metrics R-squared metrics.

    Conclusion: Summarize the findings and identify the most effective regression model for predicting car prices.

Dataset consist of 17966 rows * 9 columns consists of the following :-

model year price transmission mileage fuelType tax mpg engineSize

MACHINE LEARNING
MINI PROJECT

Indira Gandhi Delhi Technical University for Women

SUBMITTED BY -
Kusum Sharma
00602102023
1st Sem M.Tech CSE AI

SUBMITTED TO –

Prof. Seeja .K.R