Name: Chayan Gulati

Roll No.: 18CSU054

AUTOMOBILE DATASET

Introduction : Automobile Dataset with different characteristic of an auto. The dataset has total 25 columns and 205 entries. This data set consists of three types of entities:

(a) the specification of an auto in terms of various characteristics,

(b) its assigned insurance risk rating,

(c) its normalized losses in use as compared to other cars. The second rating corresponds to the degree to which the auto is more risky than its price indicates. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is more risky (or less), this symbol is adjusted by moving it up (or down) the scale. Actuarians call this process "symboling". A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.

Attribute Information :

* Symboling
* Normalized-losses
* Make
* Fuel-type
* Aspiration
* Num-of-doors
* Body-style
* Drive-wheels
* Engine-location
* Wheel-base
* Length
* Width
* Height
* Curb-weight
* Engine-type
* Num-of-cylinders
* Engine-size
* Fuel-system
* Bore
* Stroke
* Compression-ratio
* Horsepower
* Peak-rpm
* City-mpg
* Highway-mpg
* Price

Literature Review :

Findings :-

* Vehicle Mileage decrease as increase in Horsepower , engine-size, Curb Weight
* As horsepower increase the engine size increases
* Curb weight increases with the increase in Engine Size
* Price Analysis
* engine size and curb-weight is positively correlated with price
* city-mpg is negatively corelated with price as increase horsepower reduces the mileage

1. Analysis of the data set provides
2. How the data set are distributed
3. Correlation between different fields and how they are related
4. Normalized loss of the manufacturer
5. Symboling : Cars are initially assigned a risk factor symbol associated with its price
6. Mileage : Mileage based on City and Highway driving for various make and attributes
7. Price : Factors affecting Price of the Automobile.
8. Importance of drive wheels and curb weight

Implementation and Experimental Results :

*# Importing Libraries*

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

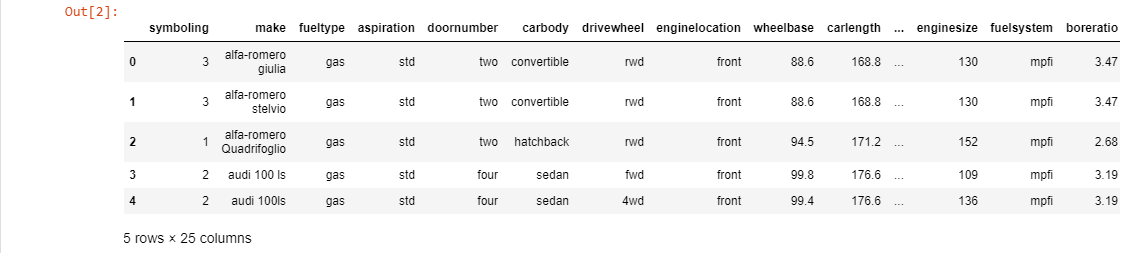
**import** **seaborn** **as** **sns**

In [2]:

*# Read the dataset*

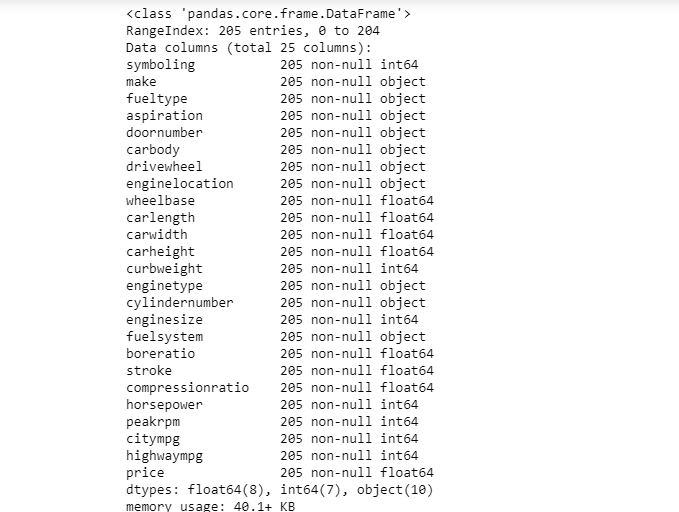
data = pd.read\_csv("AutoData.csv")

data.head()



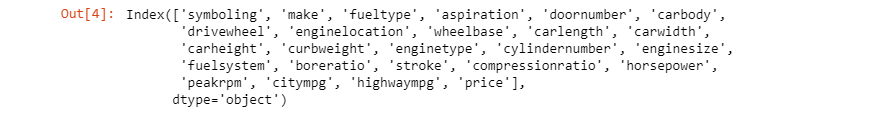
In [3]:

data.info()



In [4]:

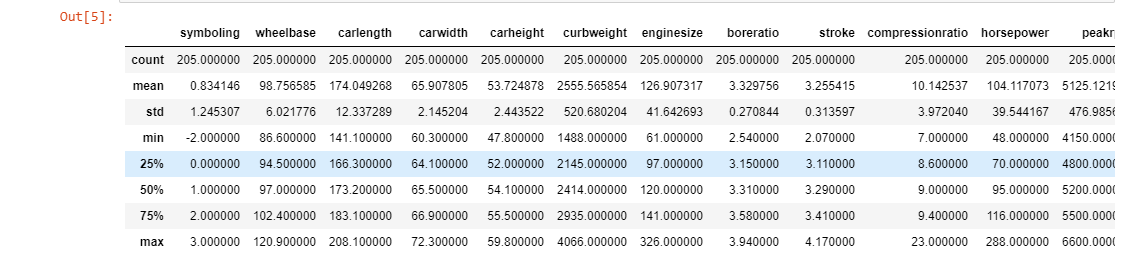
data.columns



Out[4]:

In [5]:

data.describe()



Out[5]:

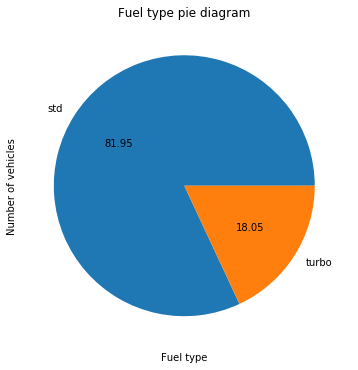
*# Pie-chart reprents the no. of vehicles have fuel type std or turbo*

data['aspiration'].value\_counts().plot.pie(figsize**=**(6, 6), autopct**=**'%.2f')

plt.title("Fuel type pie diagram")

plt.ylabel('Number of vehicles')

plt.xlabel('Fuel type');



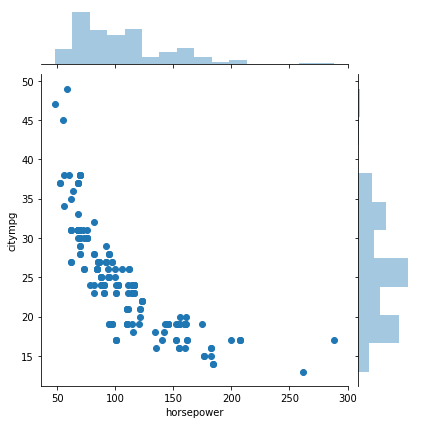
In [6]:

*# The graph depicts that as the horsepower increases the mileage in the city eventually decreases*

sns.jointplot(x='horsepower', y='citympg', data=data)

Out[6]:

<seaborn.axisgrid.JointGrid at 0x13e30fe8fd0>



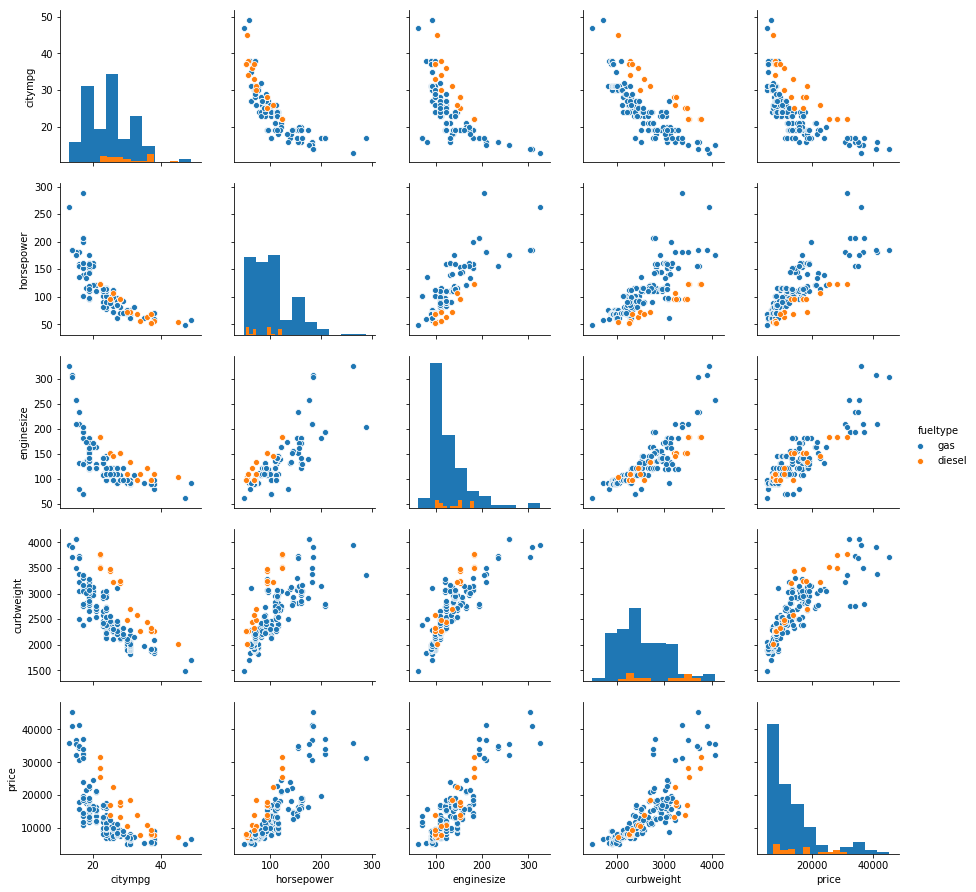
In [31]:

*# Pair plot displays the correlation with following attributes*

sns.pairplot(data[["citympg", "horsepower", "enginesize", "curbweight","price", "fueltype"]], hue**=**"fueltype", diag\_kind**=**"hist")

Out[31]:

<seaborn.axisgrid.PairGrid at 0x13e3433aba8>

**

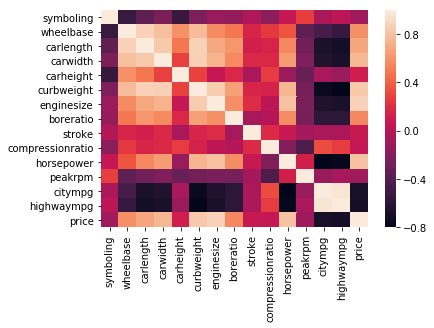
In [7]:

*# Heat map tells us about the correlation among all attributes*

sns.heatmap(data.corr())

Out[7]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x13e31437470>



**Linear Regression**

In [8]:

*# From sckrit-learn library import train\_test\_split to train and split the data*

**from** **sklearn.model\_selection** **import** train\_test\_split

In [9]:

*# In X we have taken 'enginesize' and at y we have target variable 'price'*

X = pd.DataFrame(data.iloc[:,15])

y = pd.DataFrame(data.iloc[:,-1])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

In [10]:

*# Now we apply Linear Regression on our model*

**from** **sklearn.linear\_model** **import** LinearRegression

lr = LinearRegression()

lr.fit(X\_train, y\_train)

Out[10]:

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None,

normalize=False)

In [11]:

*# Check the Intercept where it will cut y-axis and coefficent*

print(f'Intercept : **{lr.intercept\_}**')

print(f'Coefficient : **{lr.coef\_}**')

Intercept : [-7646.43507781]

Coefficient : [[165.76866245]]

In [12]:

*# Predicting the Test set results*

pred = lr.predict(X\_test)

*# To check the error and the accuracy of the model*

**from** **sklearn.metrics** **import** mean\_squared\_error, r2\_score

print(f'Mean squared error : {mean\_squared\_error(y\_test, pred)}')

print(f'R2 score : {r2\_score(y\_test, pred)}')

Mean squared error : 17079275.21406684

R2 score : 0.7534902447506853

**Decision Tree**

In [18]:

*# In X we have taken 'wheelbase','carlength','carwidth','carheight' and at y we have target variable 'enginelocation'*

X1 = pd.DataFrame(data.iloc[:,8:12])

y1 = pd.DataFrame(data.iloc[:,7])

X1\_train, X1\_test, y1\_train, y1\_test = train\_test\_split(X1, y1, test\_size=0.3, random\_state=42)

In [19]:

*# Convert string to boolean*

**from** **sklearn.preprocessing** **import** LabelEncoder

labelencoder=LabelEncoder()

y1.iloc[:,0]=labelencoder.fit\_transform(y1.iloc[:,0].values)

*# Fitting Decision Tree Classification to the Training set*

**from** **sklearn.tree** **import** DecisionTreeClassifier

dtree = DecisionTreeClassifier(criterion='entropy')

dtree.fit(X1\_train, y1\_train)

Out[21]:

DecisionTreeClassifier(class\_weight=None, criterion='entropy', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,

splitter='best')

In [22]:

*# Predicting the Test set results*

predictions = dtree.predict(X1\_test)

*# Making the Confusion Matrix*

**from** **sklearn.metrics** **import** confusion\_matrix, classification\_report

print(confusion\_matrix(y1\_test, predictions))

[[61 0]

[ 0 1]]

In [ ]:

In [29]:

*# To check the accuracy, error, sensitivity, specificity, precision, recall, f\_score of the model*

cm = confusion\_matrix(y1\_test, predictions)

total = cm[0][0]+cm[0][1]+cm[1][0]+cm[1][1]

acc = (cm[0][0]+cm[1][1])/total

print(f'Accuracy: **{acc}**')

error = 1-acc

print(f'Error: **{error}**')

sensitivity = cm[0][0]/cm[0][0]+cm[0][1]

print(f'Sensitivity: **{sensitivity}**')

specificity = cm[1][1]/cm[1][0]+cm[1][1]

print(f'Specificity: **{specificity}**')

precision = cm[0][0]/cm[0][0]+cm[1][0]

print(f'Precision: **{precision}**')

recall = sensitivity

print(f'Recall: **{recall}**')

f\_score = (2\*precision\*recall)/(precision + recall)

print(f'f\_score: **{f\_score}**')

Accuracy: 1.0

Error: 0.0

Sensitivity: 1.0

Specificity: inf

Precision: 1.0

Recall: 1.0

f\_score: 1.0

References : <https://www.kaggle.com/datasets?fileType=csv>

This data set is from the [Univerisity of California Irvine Machine Learning Repository](https://archive.ics.uci.edu/ml/datasets/Automobile) The data was compiled by Jeffrey C. Schlimmer from the following sources:

1) 1985 Model Import Car and Truck Specifications, 1985 Ward's Automotive Yearbook.  
2) Personal Auto Manuals, Insurance Services Office, 160 Water Street, New York, NY 10038  
3) Insurance Collision Report, Insurance Institute for Highway Safety, Watergate 600, Washington, DC 20037