Bishal Neupane

Bxn200007

Importing necessary libraries

In [242…

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
```

Importing files

In [322…

```
data = pd.read_csv("C:/Users/bisha/OneDrive/Desktop/Fall 2022/Machine Learning/programm
```

In [323…

```
data
```

Out[323…

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70.0	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70.0	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70.0	1	amc rebel sst
4	17.0	8	302.0	140	3449	NaN	70.0	1	ford torino
…	…	…	…	…	…	…	…	…	…
387	27.0	4	140.0	86	2790	15.6	82.0	1	ford mustang gl
388	44.0	4	97.0	52	2130	24.6	82.0	2	vw pickup
389	32.0	4	135.0	84	2295	11.6	82.0	1	dodge rampage
390	28.0	4	120.0	79	2625	18.6	82.0	1	ford ranger
391	31.0	4	119.0	82	2720	19.4	82.0	1	chevy s-10

392 rows × 9 columns

Printing first Five Rows of Data

In [324…
```
data.head()
```

Out[324…

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70.0	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70.0	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70.0	1	amc rebel sst
4	17.0	8	302.0	140	3449	NaN	70.0	1	ford torino

Dimensions of Data

In [325…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 392 entries, 0 to 391
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    int64  
 4   weight        392 non-null    int64  
 5   acceleration  391 non-null    float64
 6   year          390 non-null    float64
 7   origin        392 non-null    int64  
 8   name          392 non-null    object 
dtypes: float64(4), int64(4), object(1)
memory usage: 27.7+ KB
```

In [326…
```
print("Number of Rows: ",len(data))
print("Number of Columns: ", len(data.columns))
```

```
Number of Rows:  392
Number of Columns:  9
```

Data Exploration

In [327…
```
data.describe()
```

Out[327…

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	o
count	392.000000	392.000000	392.000000	392.000000	392.000000	391.000000	390.000000	392.00
mean	23.445918	5.471939	194.411990	104.469388	2977.584184	15.554220	76.010256	1.57
std	7.805007	1.705783	104.644004	38.491160	849.402560	2.750548	3.668093	0.80
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000	1.00
25%	17.000000	4.000000	105.000000	75.000000	2225.250000	13.800000	73.000000	1.00

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	o
50%	22.750000	4.000000	151.000000	93.500000	2803.500000	15.500000	76.000000	1.00
75%	29.000000	8.000000	275.750000	126.000000	3614.750000	17.050000	79.000000	2.00
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.00

The mpg value ranges from 9 mpg (min) to 46.60 mpg (max) which makes an average of 23.445918.

The cylinder value ranges from 3 cylinders (min) to 8 cylinders (max) which makes an average of 5.471939.

The displacement ranges from 68 (min) to 455 (max) which makes an average of 194.411990.

The horsepower ranges from 46(min) to 230 (max) which makes an average of 104.469388.

The weight ranges from 1613 (min) to 5140 (max) which makes an average of 2977.584184.

The acceleration ranges from 8 (min) to 24.80(max) which makes an average of 15.554220.

The year ranges from 70 (min) to 82 (max) which makes an average of 76.010256.

The origin ranges from 1 (min) to 3 (max) which makes an average of 1.576531.

Checking datatypes of the columns

In [328…
```
data.dtypes
```

Out[328…
```
mpg             float64
cylinders         int64
displacement    float64
horsepower        int64
weight            int64
acceleration    float64
year            float64
origin            int64
name             object
dtype: object
```

Converting cylinder column into categorical

In [329…
```
data['cylinders'] = data['cylinders'].astype('category')
#data['cylinders'] = data['cylinders'].cat.codes
data.head()
```

Out[329…

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70.0	1	buick skylark 320

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
2	18.0	8	318.0	150	3436	11.0	70.0	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70.0	1	amc rebel sst
4	17.0	8	302.0	140	3449	NaN	70.0	1	ford torino

Verifying datatypes of Column

In [330…
```
#Checking the datatypes of all columns
data.dtypes
```

Out[330…
```
mpg              float64
cylinders       category
displacement     float64
horsepower         int64
weight             int64
acceleration     float64
year             float64
origin             int64
name              object
dtype: object
```

In [ ]:
```

```

Converting Origin Column into categorical

In [331…
```
data['origin'] = pd.Categorical(data.origin)
```

Veryfing datatypes of Column

In [332…
```
#Checking the datatypes of all columns
data.dtypes
```

Out[332…
```
mpg              float64
cylinders       category
displacement     float64
horsepower         int64
weight             int64
acceleration     float64
year             float64
origin          category
name              object
dtype: object
```

Dealing with NA's

Dropping row with NA's

In [333…
```
data = data.dropna()
```

Resetting index after dropping rows with NA's

In [334...
```
data = data.reset_index(drop = True)
```

New Dimension After Deleting rows with NA's

In [335...
```
print("Number of Rows: ",len(data))
print("Number of Columns: ", len(data.columns))
```

```
Number of Rows:  389
Number of Columns:  9
```

New Data information after dropping rows with NA's

In [336...
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 389 entries, 0 to 388
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype   
---  ------        --------------  -----   
 0   mpg           389 non-null    float64 
 1   cylinders     389 non-null    category
 2   displacement  389 non-null    float64 
 3   horsepower    389 non-null    int64   
 4   weight        389 non-null    int64   
 5   acceleration  389 non-null    float64 
 6   year          389 non-null    float64 
 7   origin        389 non-null    category
 8   name          389 non-null    object  
dtypes: category(2), float64(4), int64(2), object(1)
memory usage: 22.5+ KB
```

Modification Of Columns

In [337...
```
data['mpg_high'] = np.where(data["mpg"] > data['mpg'].mean(), 1,0)
```

Converting mpg_high into Category

In [338...
```
data['mpg_high'] = pd.Categorical(data.mpg_high)
```

In [339...
```
data.dtypes
```

Out[339...
```
mpg             float64
cylinders      category
displacement    float64
horsepower        int64
weight            int64
acceleration    float64
year            float64
origin         category
name             object
mpg_high       category
dtype: object
```

Deleting mpg column

In [340…
```
del data['mpg']
```

Printing few rows after deleting mpg

In [341…
```
data
```

Out[341…

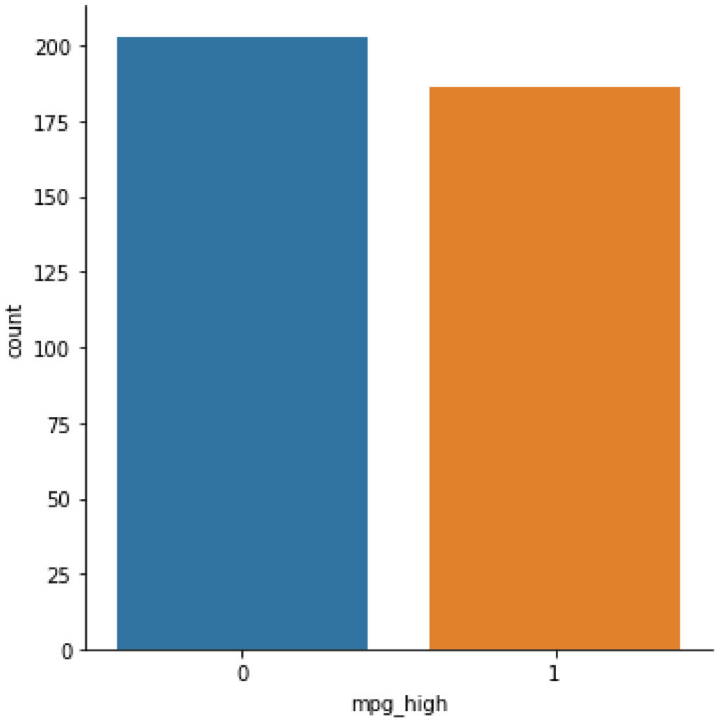
	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	mpg_high
0	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu	0
1	8	350.0	165	3693	11.5	70.0	1	buick skylark 320	0
2	8	318.0	150	3436	11.0	70.0	1	plymouth satellite	0
3	8	304.0	150	3433	12.0	70.0	1	amc rebel sst	0
4	8	454.0	220	4354	9.0	70.0	1	chevrolet impala	0
...	...	...	...	...	...	...	...	...	...
384	4	140.0	86	2790	15.6	82.0	1	ford mustang gl	1
385	4	97.0	52	2130	24.6	82.0	2	vw pickup	1
386	4	135.0	84	2295	11.6	82.0	1	dodge rampage	1
387	4	120.0	79	2625	18.6	82.0	1	ford ranger	1
388	4	119.0	82	2720	19.4	82.0	1	chevy s-10	1

389 rows × 9 columns

Data Exploration With Seaborn

Seaborn Catplot

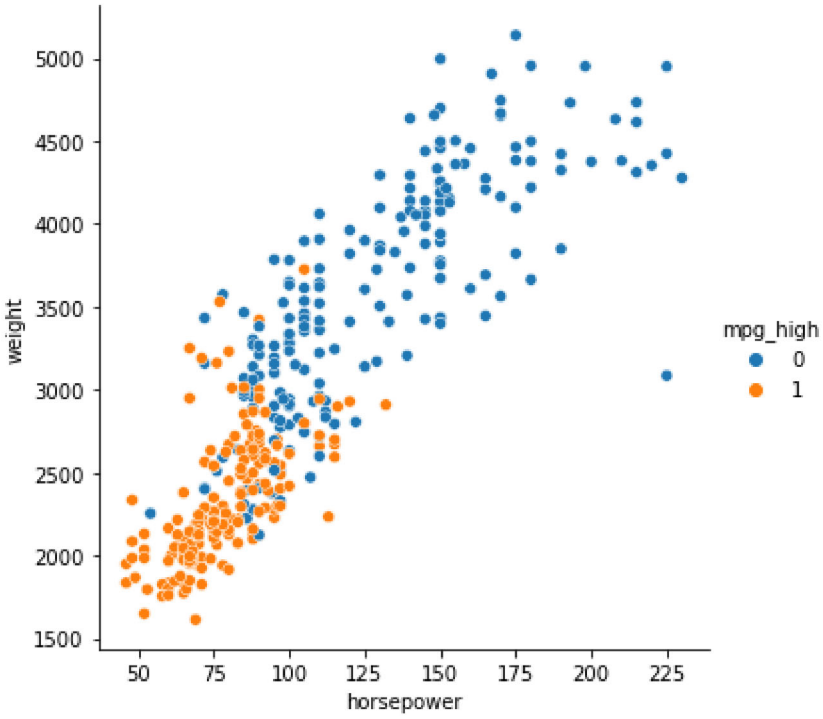
In [342…
```
g = sns.catplot(x="mpg_high",
                kind = "count",
               data= data)
```



Seaborn relplot

In [343…

```
f = sns.relplot(data= data, x='horsepower' , y = 'weight', hue = 'mpg_high' )
```



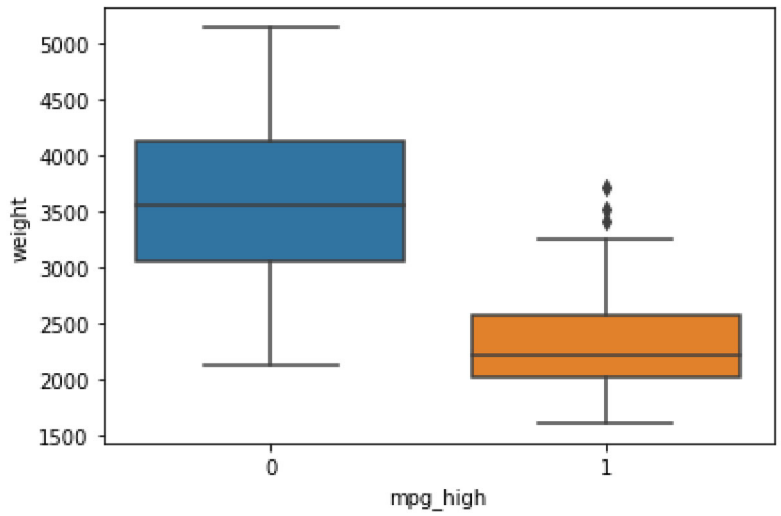
Seaborn boxplot

In [344…

```
sns.boxplot(x="mpg_high",
                y="weight",
                data=data)
```

```
<AxesSubplot:xlabel='mpg_high', ylabel='weight'>
```

Out[344...



Lesson Learned from the graphs

Seaborn Catplot: It shows the count of the two values of mgp_high. The data seems to be pretty good based on division between two classes of mpg_high which are 0 and 1

Sns relplot: The graph of relplot shows that there is some kind of linear relationship between horsepower and weight of vehicles. Also, vehicles with low 0 mgp_high have high horsepower and more weight. This clarifys that vehicle with less weight and less horsepower have more mpg.

Sns Boxplot: From the above boxplot, it can be seen that median lies in in between 3000 to 4000 when mpg_high =0 and lies between 2000 to 2500 when mpg_high = 1. Also, data with mpg high == 1 has some outliers.

Label Encoding for string and Category datatypes

In [345...

```
from sklearn import preprocessing
def label_encoder(df):
    columnsForEncoding = list(df.select_dtypes(include=['category','object']))
    encoder = preprocessing.LabelEncoder()
    for column in columnsForEncoding:
        df[column] = encoder.fit_transform(df[column])
    return df
data = label_encoder(data);
```

Spliting into Train and Test

Columns without mpg_high

In [346...

```
data_without_label = data.loc[:, data.columns != 'mpg_high']
```

Spliting data into train and test

In [347...

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_without_label,data['mpg_high']
```

Dimensions of Test and train

In [348...
```
print("Number of Rows of Train: ",len(X_train))
print("Number of Columns Train : ", len(X_train.columns))
print("Number of rows of trainLabels: ", len(y_train))

print("Number of Rows of Test: ",len(X_test))
print("Number of Columns of Test : ", len(X_test.columns))
print("Number of rows of testLabels: ", len(y_test))
```

```
Number of Rows of Train:  311
Number of Columns Train :  8
Number of rows of trainLabels:  311
Number of Rows of Test:  78
Number of Columns of Test :  8
Number of rows of testLabels:  78
```

In [349...
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Logistic Regression

In [350...
```
from sklearn.linear_model import LogisticRegression
```

In [351...
```
logRegression = LogisticRegression(solver = "lbfgs")
```

In [352...
```
logRegression.fit(X_train, y_train)
```

Out[352... LogisticRegression()

In [353...
```
predicted_value = logRegression.predict(X_test)
test_accuracy = accuracy_score(y_test, predicted_value)
print("The Accuracy for Test dataset is {}".format(test_accuracy*100))
```

```
The Accuracy for Test dataset is 88.46153846153845
```

In [354...
```
print(classification_report(predicted_value,y_test))
```

```
              precision    recall  f1-score   support

           0       0.84      0.98      0.90        43
           1       0.96      0.77      0.86        35

    accuracy                           0.88        78
   macro avg       0.90      0.87      0.88        78
weighted avg       0.90      0.88      0.88        78
```

Decision Tree

Decision Tree With Gini

In [355…
```
 Decision_tree_with_gini = DecisionTreeClassifier(criterion = "gini",
            random_state = 100,max_depth=3, min_samples_leaf=5)

    # Performing training
Decision_tree_with_gini.fit(X_train, y_train)
```

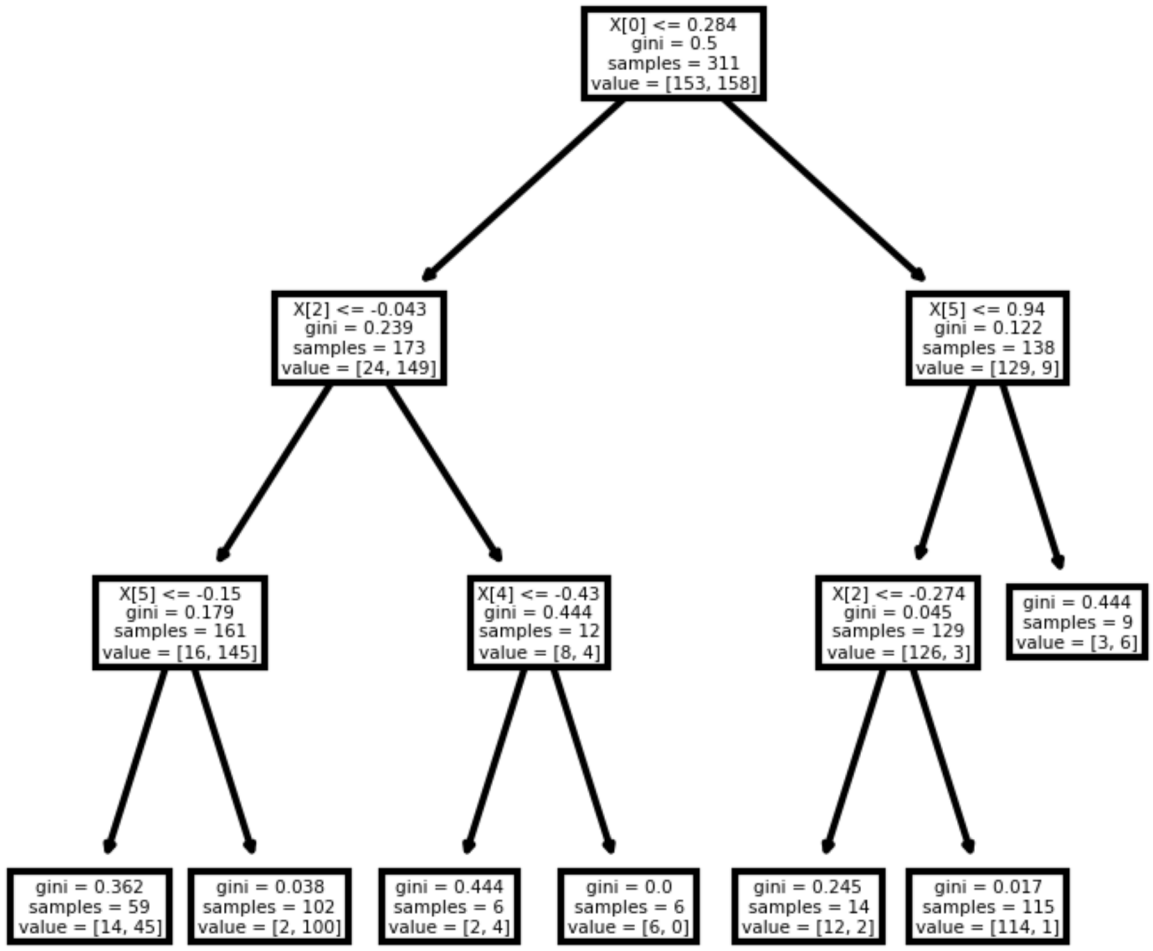
Out[355… DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)

In [356…
```
predicted_value = Decision_tree_with_gini.predict(X_test);
```

In [357…
```
print(classification_report(predicted_value,y_test))
```

```
              precision    recall  f1-score   support

           0       0.80      0.98      0.88        41
           1       0.96      0.73      0.83        37

    accuracy                           0.86        78
   macro avg       0.88      0.85      0.85        78
weighted avg       0.88      0.86      0.86        78
```

In [358…
```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (3,3), dpi=300)
tree.plot_tree(Decision_tree_with_gini);
```



Decision Tree with Entropy as criterion

In [359…
```
Decision_tree_with_entropy = DecisionTreeClassifier(criterion = "entropy",
            random_state = 100,max_depth=3, min_samples_leaf=5)

    # Performing training
Decision_tree_with_entropy.fit(X_train, y_train)
```

Out[359…
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=5,
                       random_state=100)
```

In [360…
```
predicted_value = Decision_tree_with_entropy.predict(X_test);
```

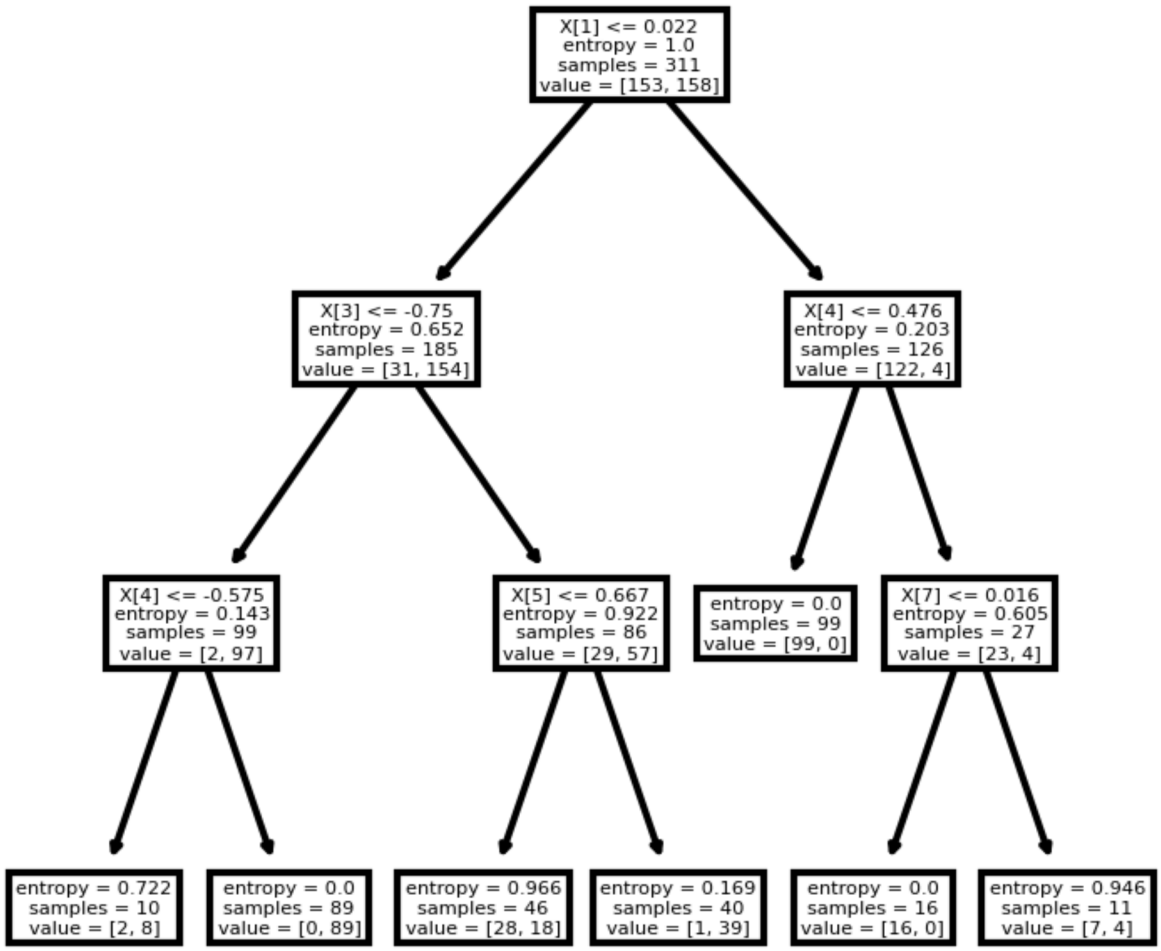
In [361…
```
print(classification_report(predicted_value,y_test))
```

```
              precision    recall  f1-score   support

           0       0.92      0.94      0.93        49
```

```
              1       0.89      0.86      0.88        29

       accuracy                           0.91        78
      macro avg       0.91      0.90      0.90        78
   weighted avg       0.91      0.91      0.91        78
```

In [362…

```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (3,3), dpi=300)
tree.plot_tree(Decision_tree_with_entropy);
```



Neural Network

In [363…

```
neural_network = MLPClassifier(solver='lbfgs',activation = 'relu',
                    hidden_layer_sizes=(389,200,100, 2), random_state=1)

neural_network.fit(X_train, y_train)
```

Out[363…

```
MLPClassifier(hidden_layer_sizes=(389, 200, 100, 2), random_state=1,
              solver='lbfgs')
```

In [364…

```
predicte_value=neural_network.predict(X_test)
print(classification_report(predicted_value,y_test))
```

```
              precision    recall  f1-score   support

           0       0.92      0.94      0.93        49
           1       0.89      0.86      0.88        29

    accuracy                           0.91        78
   macro avg       0.91      0.90      0.90        78
weighted avg       0.91      0.91      0.91        78
```

In [365…

```
neural_network1 = MLPClassifier(activation = 'tanh',solver='lbfgs',
                    hidden_layer_sizes=(389,200,100,50,25,2), random_state=1)

neural_network1.fit(X_train, y_train)
```

Out[365…

```
MLPClassifier(activation='tanh', hidden_layer_sizes=(389, 200, 100, 50, 25, 2),
              random_state=1, solver='lbfgs')
```

In [366…

```
predict_value=neural_network1.predict(X_test)
print(classification_report(predict_value,y_test))
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        44
           1       1.00      0.82      0.90        34

    accuracy                           0.92        78
   macro avg       0.94      0.91      0.92        78
weighted avg       0.93      0.92      0.92        78
```

Comparison between two neural Networks

For the first model, I used relu as the activation function and hidden layers with (389,200,100, 2) neurons and for the second model,I used tanh as the activation function and hidden layers with the (389,200,100,50,25,2) neurons. With this two difference configuration ,I was able to get accuracy of 91 percent with first configuration and 92 percent with second configuration. I think the second model outperforms the first one because it has more hidden layers with more numbers of neurons. Theoritically, relu should perform well than tanh but in our case due to less numbers of hidden layers and numbers of neuron, accuracy and other metrics of second model is better.

Comparison between Different Machine Learning Algorithms

I performed experiment with the logistic regression, Decision tree with gini, Decision tree with entrophy and two different configuration of neural network using the build in algorithms of sklearn. Neural network outperforms other algorithms.

Accuracy Comparison

With my data set, I was able to get the accuracy of about 88 percent using logistic regression, 86 percent with decision tree with gini as criterion and 91 percent with decision tree with entrophy as criterion, 91 and 92 percent with neural network using MLPClassifier of SkLearn. So, looking at this we can say that neural network outperform other algorithms.

Precision Comparison

Also precision for logistic regression is 0.84 and 0.96 for 0 and 1 mpg_high repectively. The precision for Decision tree with gini as criterion is 0.80 and 0.96 for 0 and 1 mpg_high repectively. The precision for Decision tree with entropy as criterion is 0.92 and 0.89 for 0 and 1 mpg_high respectively. The precision for two different neural networks are (0.92, 0.89) and (0.88,1) for 0 and 1 respectively. This shows that second neural network classified all the positive value correctly. There is some kind of trade off for 0 and 1 but in average neural network is outperforming others.

Recall comaprison

The recall for logistic regression is 0.98 and 0.77 for 0 and 1 mpg value respectively. The recall for Decision tree with gini as criterion is 0.98 and 0.73 and the recall for Decision tree with entropy as criterion is 0.94 and 0.86 for 0 and 1 mpg_high respectively. Also, the recall for two neural network models is (0.97, 0.86), (1, 0.82) for 0 and 1 mpg_high respecitvely. Looking at the recall data neural network outperforms other models. Recall for Decision tree with entropy as criterion is almost similar to neural network but also neural network seems to be better.

Reasons for Neural Network Outperforming others

1) Neural network has a lot of neurons and hidden layers and at each neurons bias and weights are calculated based upon the result of previous layer. This process is also known as forward feeding.

2) After the forward feeding, this algorithm goes backpropagation, which gradually optimized the model by decreasing loss functions.

3) There are many parameters in neural network which makes them easier to fit with the data but other algorithms don't have as much free parameters as neural network.

Comparison of Experience of Using R vs Sklearn

Since, I am familiar with python more than R, I like sklearn. Also, R has some advantages over python. Some of the statistical analysis in R was easier. I feel python better for cleaning data and data exploration. Both seems to be good for machine learning and statistical analysis but I would prefer python if the a lot of cleaning and exploration of data needs to be done. I like the summary() function in R.

In [ ]: