University of Mumbai

PRACTICAL JOURNAL

PSDS2P1: Artificial Intelligence & Machine Learning

(MSc. Computer Science with specialization in Data Science 2021-2023)

SEM-II

Submitted by

Sumon Singh
Roll No. 16
Seat No. 112011

Problems :

1. Min-max and Alpha-beta pruning Algorithm

2. Implement Classification Model

3. Implement Regression Model

4. Implement Find S Algorithm

5. Implement Decision Tree Model

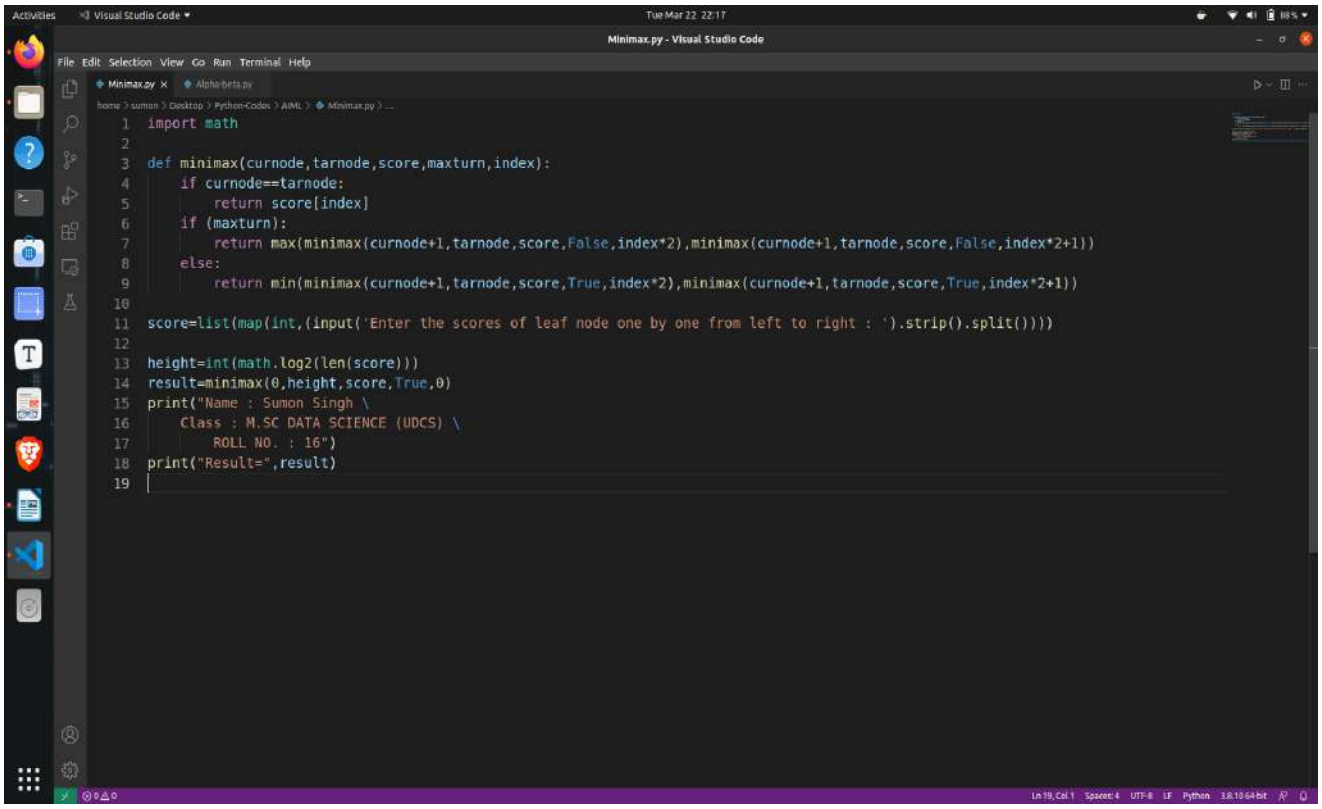
6. Implement Support Vector Machine

7. Implement Distance Based Models

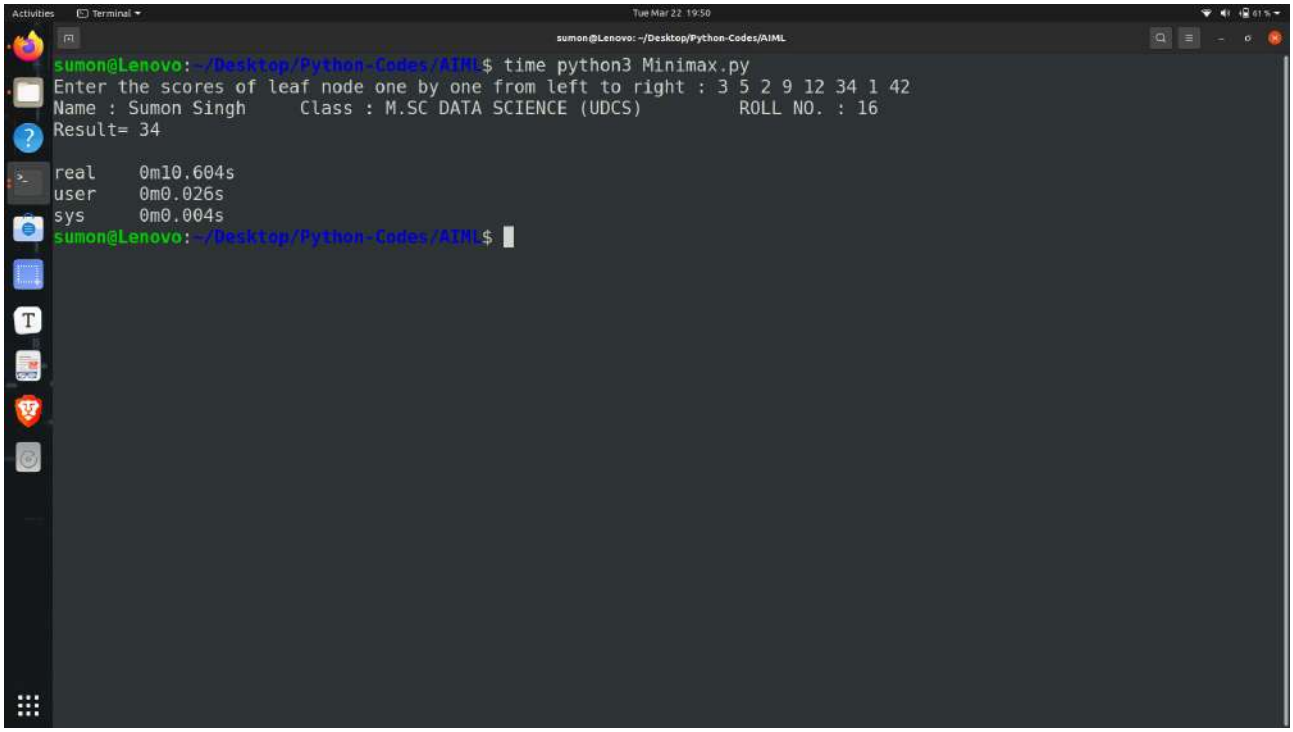
8. Probabilistic Models

NAME : SUMON SINGH ROLL NO. 16 SUBJECT : M.SC (DATA SCIENCE) ALML

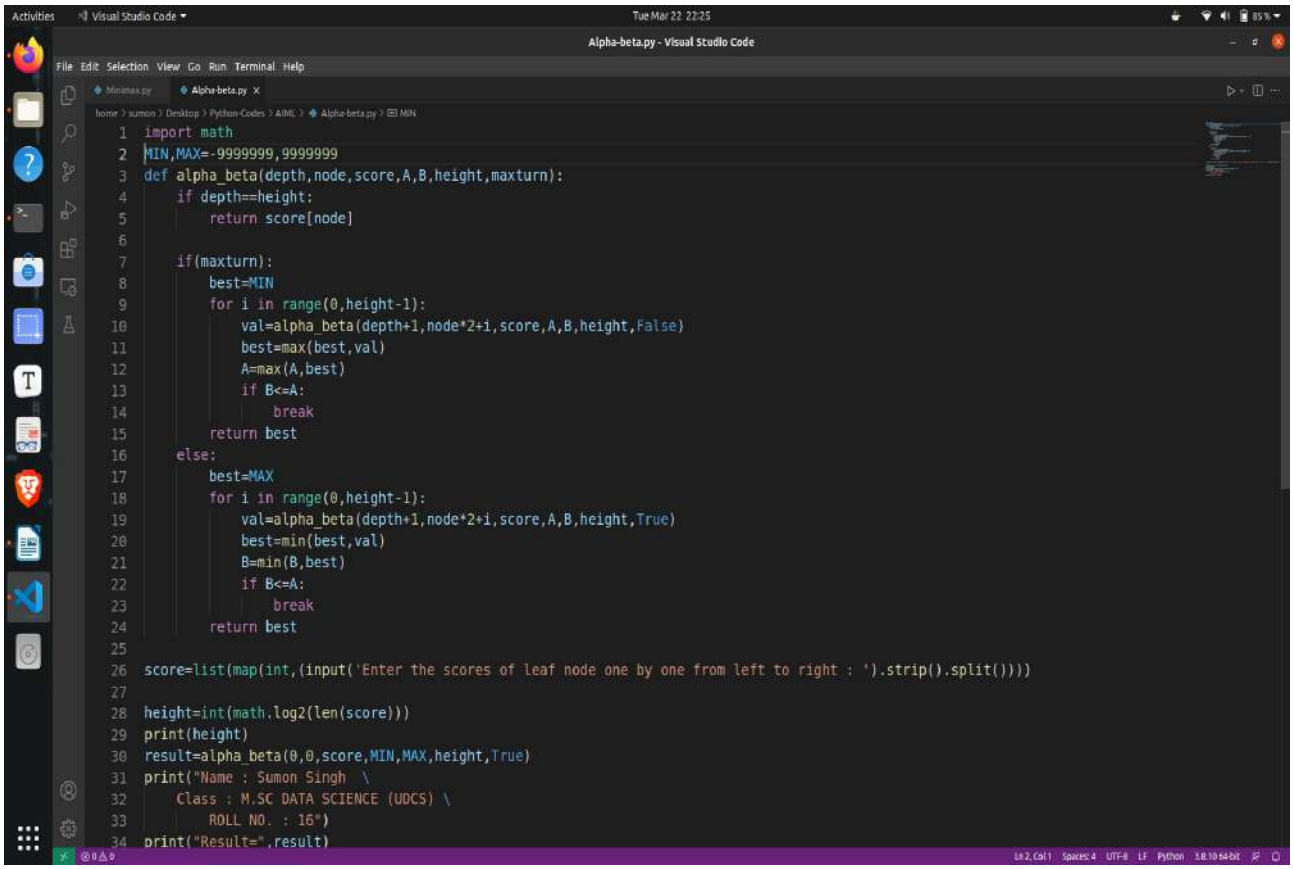
1 . CODE FOR MINIMAX ALGORITHM IN PYTHON



2. OUTPUT FOR MINIMAX CODE :

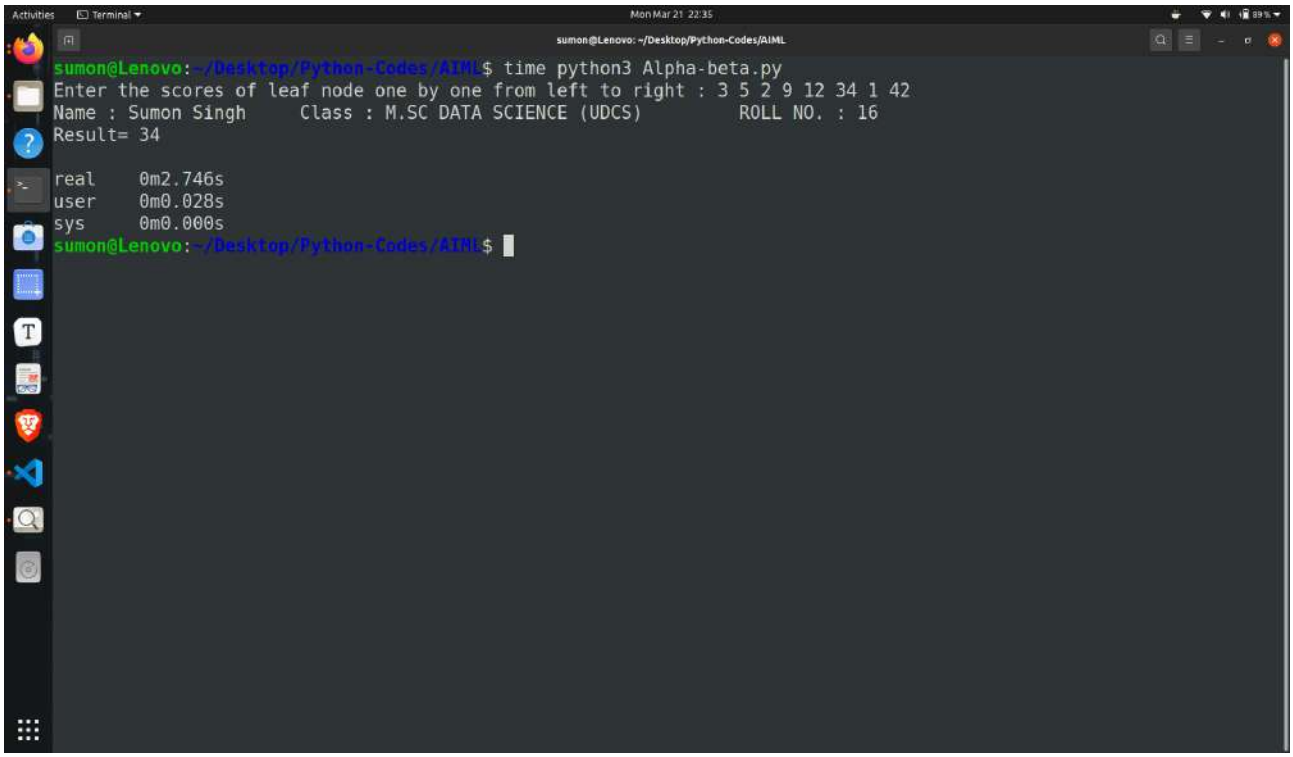


3. CODE FOR MINIMAX ALGORITHM USING ALPHA-BETA PRUNING IN PYTHON



```python
import math
MIN,MAX=-9999999,9999999
def alpha_beta(depth,node,score,A,B,height,maxturn):
    if depth==height:
        return score[node]

    if(maxturn):
        best=MIN
        for i in range(0,height-1):
            val=alpha_beta(depth+1,node*2+i,score,A,B,height,False)
            best=max(best,val)
            A=max(A,best)
            if B<=A:
                break
        return best
    else:
        best=MAX
        for i in range(0,height-1):
            val=alpha_beta(depth+1,node*2+i,score,A,B,height,True)
            best=min(best,val)
            B=min(B,best)
            if B<=A:
                break
        return best

score=list(map(int,(input('Enter the scores of leaf node one by one from left to right : ').strip().split())))

height=int(math.log2(len(score)))
print(height)
result=alpha_beta(0,0,score,MIN,MAX,height,True)
print("Name : Sumon Singh  \
    Class : M.SC DATA SCIENCE (UDCS) \
        ROLL NO. : 16")
print("Result=",result)
```

4. OUTPUT FOR MINIMAX ALGORITHM USING ALPHA-BETA PRUNING



```
sumon@Lenovo:~/Desktop/Python-Codes/AIML$ time python3 Alpha-beta.py
Enter the scores of leaf node one by one from left to right : 3 5 2 9 12 34 1 42
Name : Sumon Singh     Class : M.SC DATA SCIENCE (UDCS)         ROLL NO. : 16
Result= 34

real    0m2.746s
user    0m0.028s
sys     0m0.000s
sumon@Lenovo:~/Desktop/Python-Codes/AIML$
```

Sumon Singh
M.Sc Data Science, Roll No: 16
21.03.22
Semonsingh

AIM:- Write To Implement Minimax Algorithm

Algorithm:- (Pseudocode)

```
function Minimax(CurDepth, CurIndex, MaxPlayer)
    IF CurDepth is the terminal node, THEN
        return, value of current node

    IF MaxPlayer THEN
        return, max(Minimax(CurDepth+1, CurIndex*2+1,
                            false),
                    Minimax(CurDepth+1, CurIndex*2,
                            false))

    ELSE
        return, Min(Minimax(CurDepth+1, CurIndex*2+1,
                            True),
                    Minimax(CurDepth+1, CurIndex*2,
                            True))

    END
```

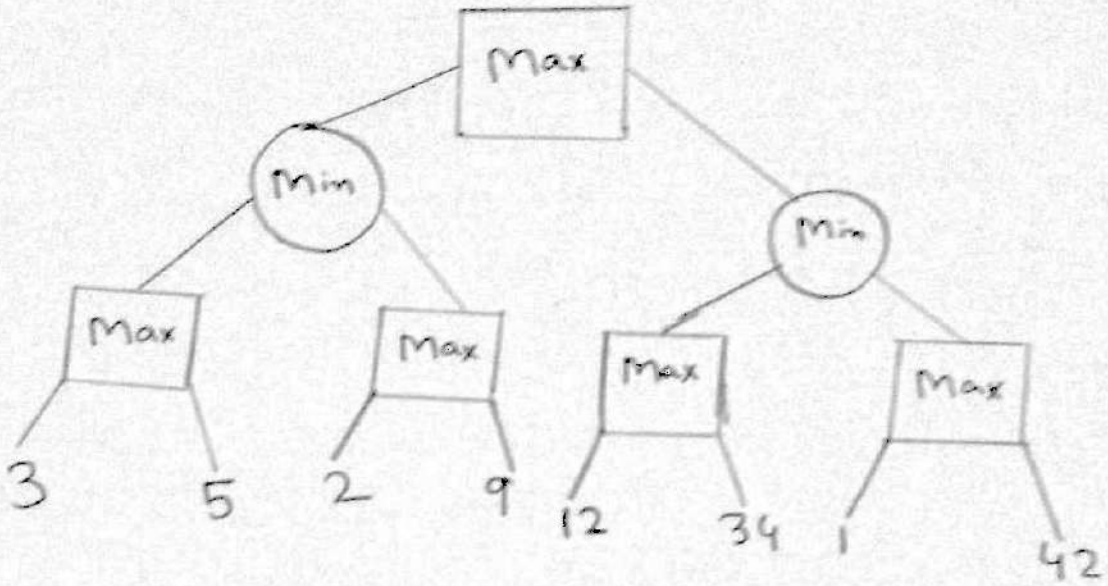
Assumption:- CurDepth, is the current level of tree
CurIndex, is the index of the current node

Sumon Singh, M.Sc Data Science, Roll No: 16

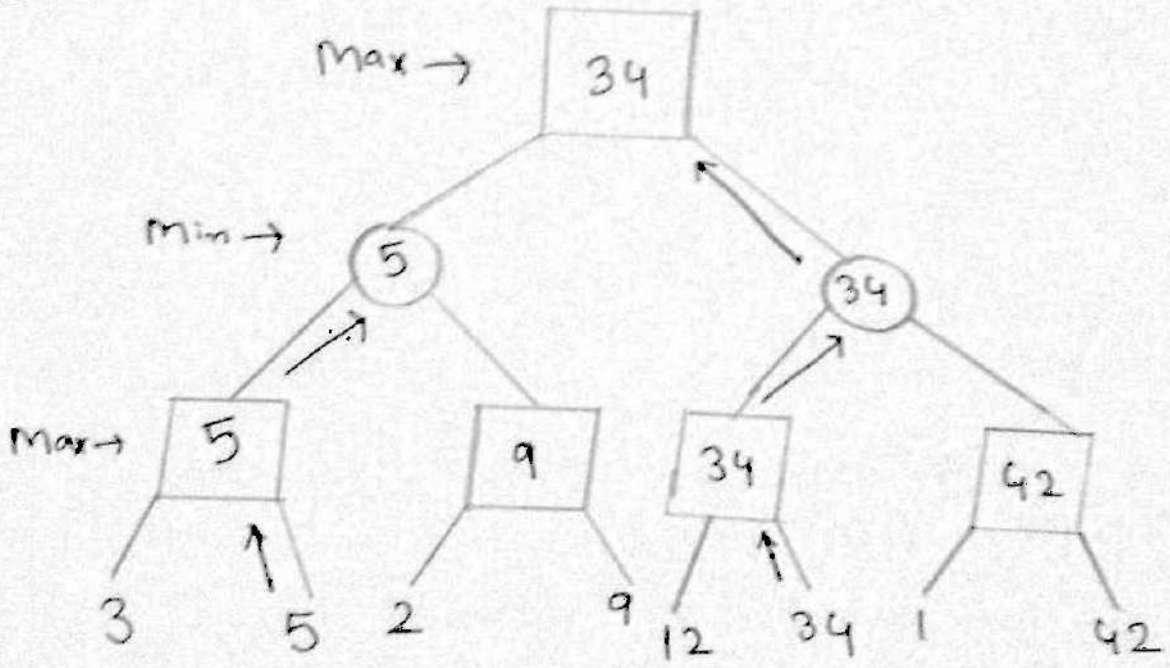
22.3.22

Example :-

Input :- 3 5 2 9 12 34 1 42



Output :-



Sumon Singh
M.Sc Data Science, Roll No 16
22.03.22
Soma Singh

AIM :- Write To Implement Minimax Algorithm Using Alpha-Beta Pruning

Algorithm :- (Pseudocode)

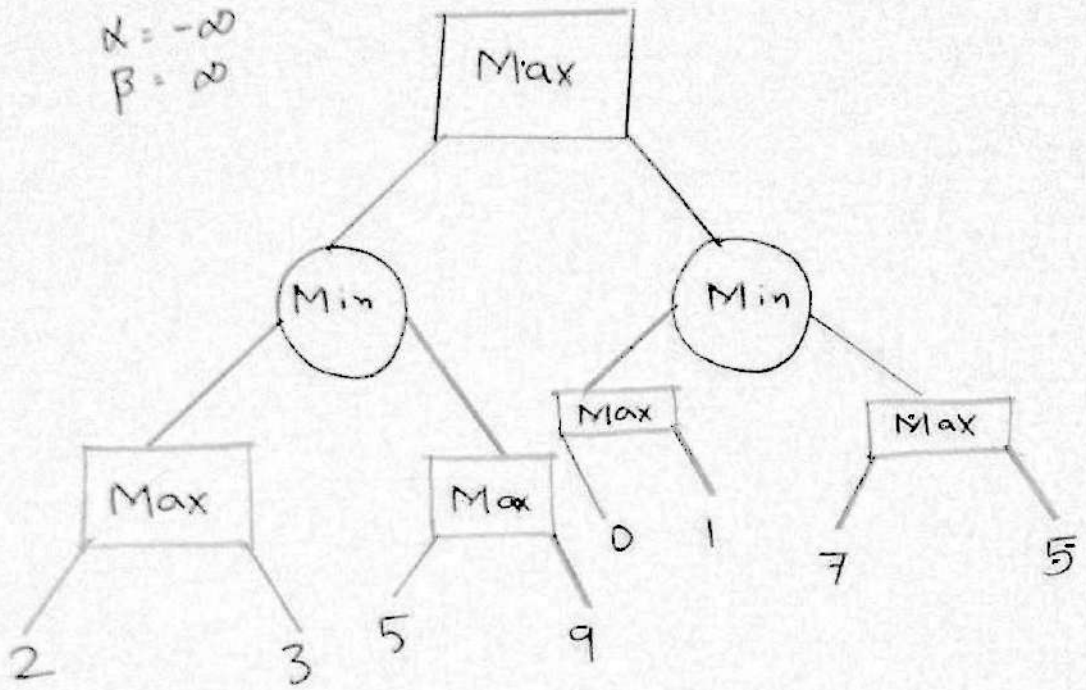
```
function Minimax(node, depth, MaxPlayer, alpha, beta)
    If, node is a leaf node THEN
     RETURN, value of the node

    If, MaxPlayer THEN
        bestval = -INF
       For Each child node THEN
        Value = Minimax(node, depth+1, False, alpha, beta)
       bestval = max(bestval, value)
        alpha = max(alpha, bestval)
        If, beta <= alpha THEN
            break
       RETURN bestval
    ELSE,
        bestval = +INF
        For Each child node THEN
            value = Minimax(node, depth+1, True, alpha, beta)
            bestval = min(bestval, value)
            beta = min(beta, bestval)
            If, beta <= alpha THEN
                break
        RETURN bestval
END)
```

22.3.22
Sumon Singh

Sumon Singh
M.Sc Data Science, Roll No. 16

Example :

Input :- 2 3 5 9 0 1 7 5

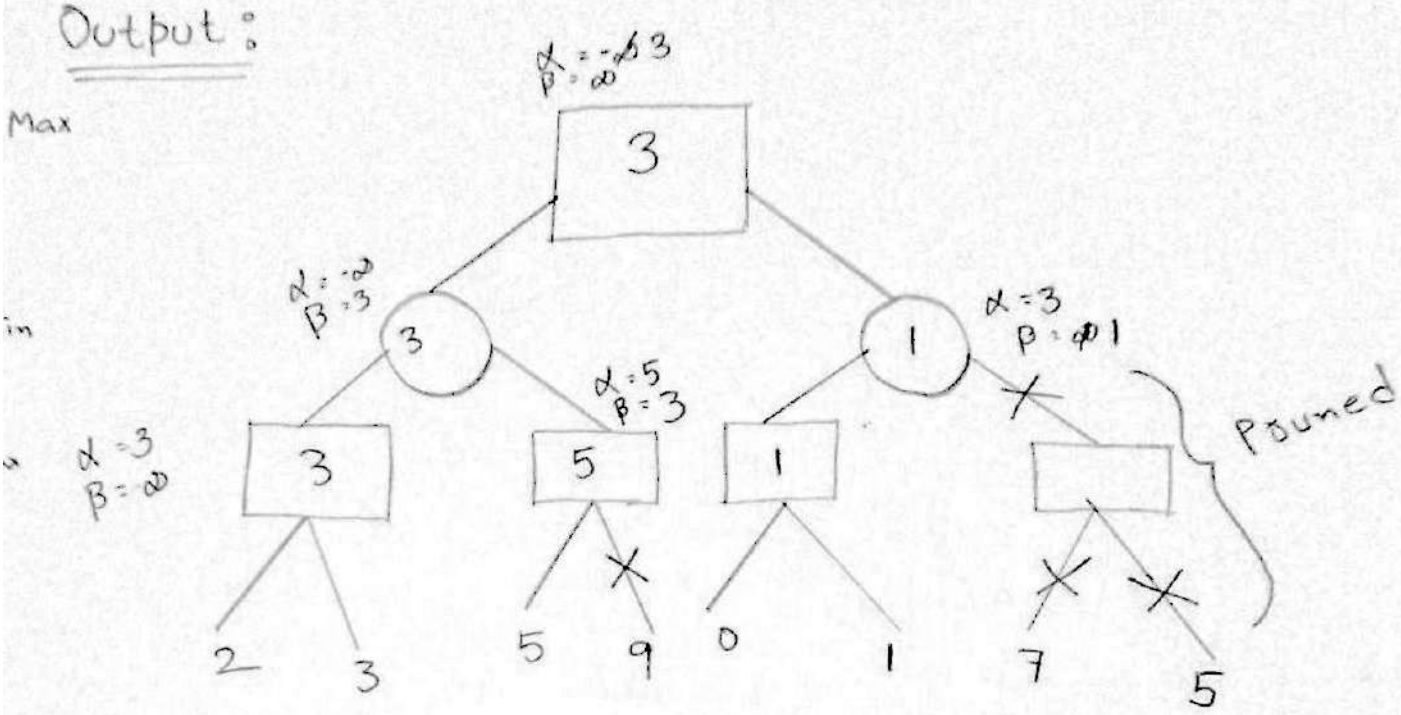


Sumon Singh
M.Sc Data Science, Roll No. 16

22.8.22

Sumon Singh

Output:



* Updating α at Max and β at Min
* Initial value, α = -∞ β = ∞
* Condition for pruning β <= α
* α is max player and β is min player
* 'X' (cross sign) denotes ~~pur~~ pruned branch

Sumon Singh
17.04.22

Sumon Singh

A) Write a program to input dataset and perform Binary classification. Evaluate the model based on classification metrics and infer your result.

1) Dataset Description:-

A) Title: Pima Indians Diabetes Database

B) Describe the data: This dataset consist of two types of diabetes which can be classified based on their features. It's a binary-class classified dataset. This dataset is available in kaggle website.

C) Size and Dimension: This dataset consists of 768 rows and 9 columns.

D) Data type: This dataset consist of real valued data.

E) Features: The dataset consist of 8 attributes and 1 Label data

* Attributes :- Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, Age and DiabetesPedigreeFunction

* Label :- Outcome (Type of diabetes) (Target)

Sumon Singh

~~21~~.

17.04.22

Sumon Singh

2) ML Model Description:

A) Name of model :- Logistic Regression

B) Type of model :- It is a supervised learning model which is used for classification. ~~model~~.

C) Algorithm :-

1) Using linear regression calculate output based on features.

Example: Let us assume, we have a dataset consisting n features $x_1, x_2 \cdots x_n$, of real values.

∴ output will be -

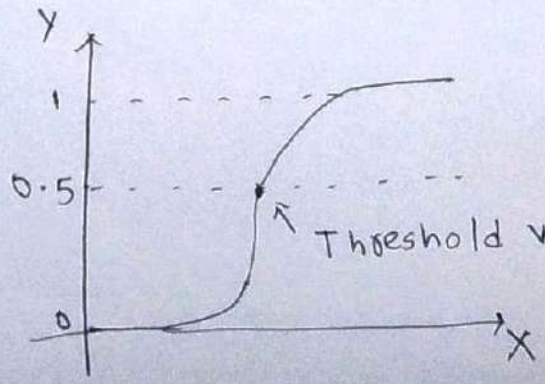
$$y = b_0 + b_1 x_1 + b_2 x_2 \cdots + b_n x_n$$

2) Using Sigmoid function find the ~~prob~~ probablity of the output of previous step.

Sigmoid function : $\frac{1}{1+e^{-y}}$ (y is previous output)

3) If the probablity output is more than 0.5 then the actual classification will be of type 1 else type 0.

D) Draw :-



Sigmoid function

$\frac{1}{1+e^{-y}}$

Sumon Singh
17.04.22

Sumon Singh

3) Result Analysis :-

A) Accuracy - 77%

Our model identifies 77% of the testing data correctly.

B) Precision -

For identifing 0 precision is 0.91
For identifing 1 precision is 0.52

The model works well for identifing Label 0 but it's very much low for identifing Label 1.

c) Recall -

Recall for identifing 0 is 0.78
Recall for identifing 1 is 0.75

The model identifies 78% '0' correctly among all '0' Labeled data and 75% '1' correctly among all '1' Labeled data.

Activities Firefox Web Browser Sun Apr 17 17:20 98 %

Diabetes_dataset - Jupyt

localhost:8888/notebooks/Diabetes_dataset.ipynb

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

Name : Sumon Singh

Roll No. : 16

In [1]:
```
import datetime

x = datetime.datetime.now()
print('Date-Time : ',x)
```

```
Date-Time :  2022-04-17 16:12:08.278788
```

Load Libraries

In [2]:
```
import numpy as np ## Array Calculation
import pandas as pd ## Manipulate dataset
import seaborn as sns ## Visualization
import matplotlib.pyplot as plt ## Ploting
from sklearn.model_selection import train_test_split ## Data spliting
from sklearn.linear_model import LogisticRegression  ## Classification model
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report ## Result Evaluation
from mlxtend.plotting import plot_confusion_matrix  ## Plot confusion matrix
from sklearn.preprocessing import MinMaxScaler  ## Feature standrization
```

Load Dataset

Source : https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

In [3]:
```
df = pd.read_csv('./diabetes.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

* Dimension of the dataset

In [5]:
```
df.shape
```

Out[5]:
```
(768, 9)
```

* Brief overview of dataset

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

* Check if any missing data in present in the dataset

In [7]:
```
df.isna().sum()
```

Out[7]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

* Summary of the dataset

In [8]:
```
df.describe().transpose()
```

* Summary of the dataset

In [8]:
```
df.describe().transpose()
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

* Check the each type of target

In [9]:
```
df['Outcome'].value_counts()
```

Out[9]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

Remove Unrelevant features

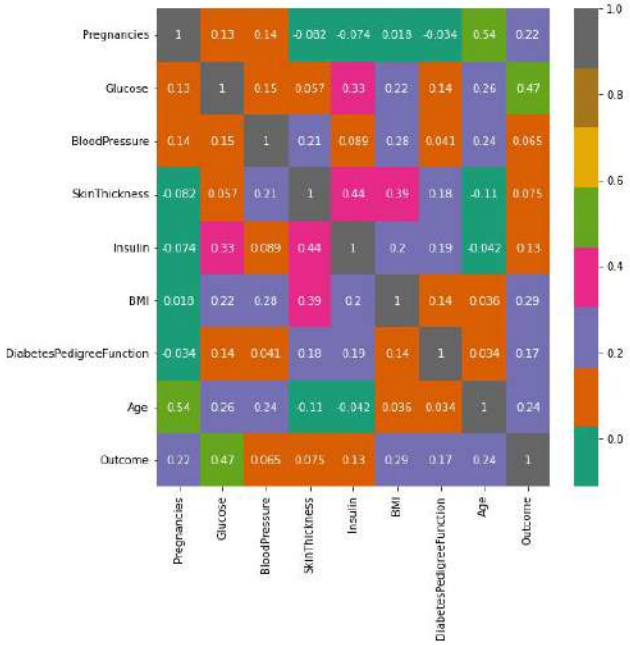
In [10]:
```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True,cmap='Dark2')
```

Out[10]: <AxesSubplot:>

Remove Unrelevant features

In [10]:
```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True,cmap='Dark2')
```

Out[10]: <AxesSubplot:>



*** BloodPressure and SkinThickness are not much correlated with the target so we can remove that features**

In [11]:
```
df.drop(['SkinThickness','BloodPressure'],axis=1,inplace=True)
```

Split features and target

*** BloodPressure and SkinThickness are not much correlated with the target so we can remove that features**

In [11]:
```
df.drop(['SkinThickness','BloodPressure'],axis=1,inplace=True)
```

Spilit features and target

In [12]:
```
X=df.drop('Outcome',axis=1).values
Y=df['Outcome'].values
```

Standarize the data

In [13]:
```
X=MinMaxScaler().fit_transform(X)
```

Spiliting dataset into training and testing dataset

In [14]:
```
train_x,test_x,train_y,test_y=train_test_split(X,Y,test_size=0.3,random_state=100,stratify=Y)
```

Use Logistic Regression model for classification

*** Intialize and fit the model with training dataset**

In [15]:
```
clf=LogisticRegression(random_state=0)
clf.fit(train_x,train_y)
```

Out[15]:
```
LogisticRegression(random_state=0)
```

*** Check how good the model works on the training data**

In [16]:
```
clf.score(train_x,train_y)
```

Out[16]:
```
0.7746741154562383
```

*** Predict the testing data outcome**

In [17]:
```
prediction_val = clf.predict(test_x)
```

*** Check accuracy for testing data**

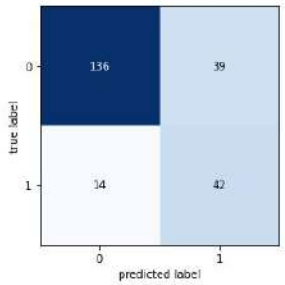
In [18]:
```
accuracy_score(prediction_val,test_y)
```

Out[18]:
```
0.7705627705627706
```

*** Plot the confusion matrix**

In [19]:
```
plot_confusion_matrix(confusion_matrix(prediction_val,test_y))
```

Out[19]:
```
(<Figure size 432x288 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



*** Get result**

In [20]:
```
print(classification_report(prediction_val,test_y))
```

```
              precision    recall  f1-score   support

           0       0.91      0.78      0.84       175
           1       0.52      0.75      0.61        56

    accuracy                           0.77       231
   macro avg       0.71      0.76      0.73       231
weighted avg       0.81      0.77      0.78       231
```

Sumon Singh
17.04.22

Sumon Singh

B) Write a program to input dataset and perform multi-class classification. Evaluate the model based on classification metrics and infer your result.

I) Data Description:

A) Title :- wheat-seeds

B) Describe the data :- This dataset consists of 3 kinds of wheat-seeds which can be classified based on their features. It's a multi-class classified dataset. This dataset is available in the Kaggle website.

C) Size and Dimension :- This dataset consist of 199 rows and 8 columns.

D) Data type :- This dataset is of Real valued data. All the columns except the target column which is 'Type' are float values and the target column is Int values.

E) Features :- This dataset consist of 7 attributes and 1 target.

* Attributes :- Area, Perimeter, Compactness, Kernel_Length, Kernel_Width, Asymmetry_Coeff, Kernel_Groove

* Label (Target) :- Type (which is the type of seed having values 1, 2, 3)

Sumon Singh
17.04.22

2) ML Model Description:

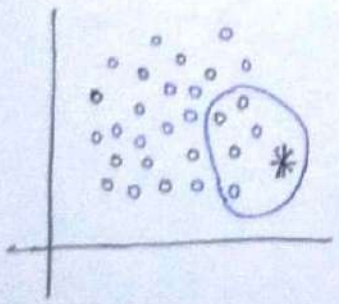
A) Name of the model :- K-Nearest Neighbor (KNN) classifier

B) Type of model :- It's a classifier model used in supervised learning

C) Algorithm :-

1) Select the number K of the neighbors.
2) Calculate the Euclidean distance of K number of neighbors.
3) Take the K nearest neighbors as per the calculated Euclidean distance.
4) Among these K neighbors, count the number of data points in each catagory.
5) Assign the new data points to that catagory for which the number of the neighbor is maximum.

D) Draw :-



o → Catagory black
o → Catagory blue
* → New data point

As the 3 black catagory is near to new data point and 2 blue catagory is near to new data point so the new data point will be of black catagory. (If K neighbors is 5)

Sumon Singh
17.04.22

3) Result Analysis :- The outcomes are 1, 2 and 3.

A) Accuracy: Our model identifies approx. 91% data correctly. which means our model is pretty good.

B) Precission:

For 1 the model's precision is 0.90
For 2 the model's precision is 0.90
For 3 the model's precision is 0.95

The model identifies almost all the Labels quite well.

c) Recall :

For 1 recall is 88%
For 2 recall is 92%
For 3 recall is 95%

The model identifies 88% '1' Labeled data correctly among all '1' Labeled data. Similarly, It identifes 92% '2' and 95% '3' Labeled data respectivly in correct way among all data.

Activities Firefox Web Browser Sun Apr 17 17:29 94 %

seed_dataset - Jupyter N

localhost:8889/notebooks/seed_dataset.ipynb

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

Name : Sumon Singh

Roll No. : 16

In [1]:
```
import datetime

x = datetime.datetime.now()
print('Date-Time : ',x)
```

```
Date-Time :  2022-04-17 16:12:03.757417
```

Load Libraries

In [2]:
```
import numpy as np  ## Array Calculation
import pandas as pd  ## Manipulate dataset
import seaborn as sns  ## Visualization
import matplotlib.pyplot as plt  ## Plotting
from sklearn.model_selection import train_test_split  ## Data spliting
from sklearn.neighbors import KNeighborsClassifier  ## Classification model
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report  ## Result Evaluation
from mlxtend.plotting import plot_confusion_matrix  ## Plot confusion matrix
```

Load Dataset

Source : https://www.kaggle.com/datasets/jmcaro/wheat-seedsuci

In [3]:
```
df = pd.read_csv('./seeds.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kernel.Groove	Type
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1

Explore Dataset

*** Dimension of the dataset**

In [5]:
```
df.shape
```

Out[5]:
```
(199, 8)
```

*** Brief overview of dataset**

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Area             199 non-null    float64
 1   Perimeter        199 non-null    float64
 2   Compactness      199 non-null    float64
 3   Kernel.Length    199 non-null    float64
 4   Kernel.Width     199 non-null    float64
 5   Asymmetry.Coeff  199 non-null    float64
 6   Kernel.Groove    199 non-null    float64
 7   Type             199 non-null    int64
dtypes: float64(7), int64(1)
memory usage: 12.6 KB
```

*** Summary of the dataset**

In [7]:
```
df.describe().transpose()
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Area	199.0	14.918744	2.919976	10.5900	12.3300	14.4300	17.4550	21.1800
Perimeter	199.0	14.595829	1.310445	12.4100	13.4700	14.3700	15.8050	17.2500
Compactness	199.0	0.870811	0.023320	0.8081	0.8571	0.8734	0.8868	0.9183
Kernel.Length	199.0	5.643151	0.443593	4.8990	5.2670	5.5410	6.0020	6.6750
Kernel.Width	199.0	3.265533	0.378322	2.6300	2.9545	3.2450	3.5645	4.0330
Asymmetry.Coeff	199.0	3.699217	1.471102	0.7651	2.5700	3.6310	4.7990	8.3150
Kernel.Groove	199.0	5.420653	0.492718	4.5190	5.0460	5.2280	5.8790	6.5500
Type	199.0	1.994975	0.813382	1.0000	1.0000	2.0000	3.0000	3.0000

*** Check if any missing data in present in the dataset**

*** Check if any missing data in present in the dataset**

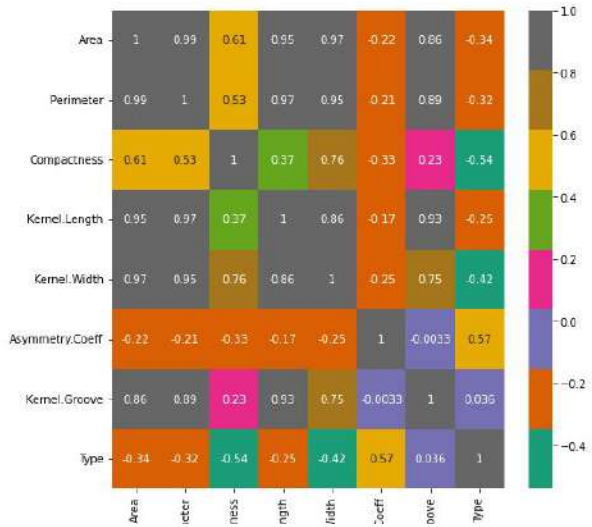
In [8]:
```
df.isna().sum()
```

Out[8]:
```
Area              0
Perimeter         0
Compactness       0
Kernel.Length     0
Kernel.Width      0
Asymmetry.Coeff   0
Kernel.Groove     0
Type              0
dtype: int64
```

*** Heatmap Plot**

In [9]:
```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True,cmap='Dark2')
```

Out[9]: <AxesSubplot:>



*** Check the each type of target**

In [10]:
```
df['Type'].value_counts()
```

Out[10]:
```
2    68
1    66
3    65
Name: Type, dtype: int64
```

Spiliting features and target

In [11]:
```
X=df.drop('Type',axis=1).values
Y=df['Type'].values
```

Spiliting dataset into training and testing dataset

In [12]:
```
train_x,test_x,train_y,test_y=train_test_split(X,Y,test_size=0.3,random_state=100,stratify=Y)
```

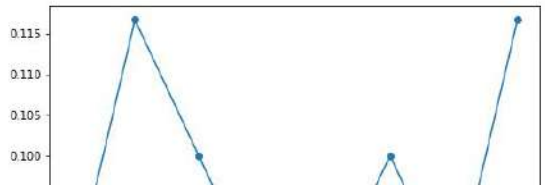
Use KNN model for classification

*** Select value of K for KNN model**

In [13]:
```
k_val=range(3,11)
error=[]
for i in k_val:
    clf = KNeighborsClassifier(n_neighbors=i)
    clf.fit(train_x,train_y)
    p = clf.predict(test_x)
    error.append(np.mean(p!=test_y))
```

In [14]:
```
plt.figure(figsize=(8,5))
plt.plot(k_val,error,marker='o')
```

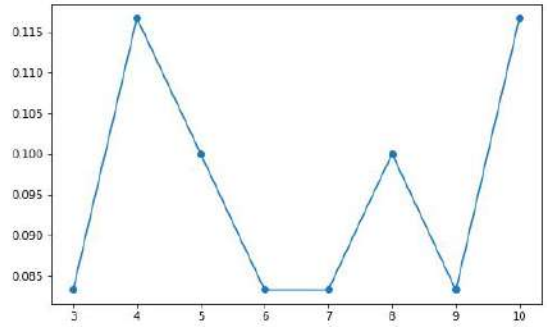
Out[14]:
```
[<matplotlib.lines.Line2D at 0x7f0f4e2f3850>]
```



```
    p = clf.predict(test_x)
    error.append(np.mean(p!=test_y))
```

In [14]:
```
plt.figure(figsize=(8,5))
plt.plot(k_val,error,marker='o')
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x7f0f4e2f3850>]
```



The optimal K value is 3 ,6 ,7 and 9

Create the optimal KNN model

*** Intialize and fit the model with training dataset**

In [15]:
```
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(train_x,train_y)
```

Out[15]:
```
KNeighborsClassifier(n_neighbors=3)
```

*** Check how good the model works on the training data**

In [16]:
```
clf.score(train_x,train_y)
```

Out[16]:
```
0.935251798561151
```

*** Predict the testing data outcome**

Activities Firefox Web Browser Sun Apr 17 17:30 94 %

seed_dataset - Jupyter N

localhost:8889/notebooks/seed_dataset.ipynb

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

In [17]:
```
prediction_val = clf.predict(test_x)
```

Evalute the model

* Check accuracy for testing data

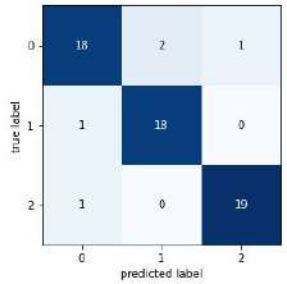
In [18]:
```
accuracy_score(prediction_val,test_y)
```

Out[18]:
```
0.9166666666666666
```

* Plot the confusion matrix

In [19]:
```
plot_confusion_matrix(confusion_matrix(prediction_val,test_y))
```

Out[19]:
```
(<Figure size 432x288 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



* Get result

In [20]:
```
print(classification_report(prediction_val,test_y))
```

```
              precision    recall  f1-score   support

           1       0.90      0.86      0.88        21
           2       0.90      0.95      0.92        19
           3       0.95      0.95      0.95        20

    accuracy                           0.92        60
   macro avg       0.92      0.92      0.92        60
weighted avg       0.92      0.92      0.92        60
```

Implement Linear Regression

Import Libraries

In [157]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

Load Dataset

In [158]:
```
df = pd.read_csv('./Fish.csv')
```

In [159]:
```
df.head()
```

Out[159]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

Data Exploration

In [160]:
```
df.shape
```

Out[160]: (159, 7)

In [161]:
```
df.isna().sum()
```

Out[161]: Species 0

Data Exploration

In [160]:
```
df.shape
```

Out[160]:
```
(159, 7)
```

In [161]:
```
df.isna().sum()
```

Out[161]:
```
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```

In [162]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Species  159 non-null    object 
 1   Weight   159 non-null    float64
 2   Length1  159 non-null    float64
 3   Length2  159 non-null    float64
 4   Length3  159 non-null    float64
 5   Height   159 non-null    float64
 6   Width    159 non-null    float64
dtypes: float64(6), object(1)
memory usage: 8.8+ KB
```

In [163]:
```
df.describe().transpose()
```

Out[163]:

	count	mean	std	min	25%	50%	75%	max
Weight	159.0	398.326415	357.978317	0.0000	120.00000	273.0000	650.0000	1650.000
Length1	159.0	26.247170	9.996441	7.5000	19.05000	25.2000	32.7000	59.000
Length2	159.0	28.415723	10.716328	8.4000	21.00000	27.3000	35.5000	63.400
Length3	159.0	31.227044	11.610246	8.8000	23.15000	29.4000	39.6500	68.000
Height	159.0	8.970994	4.286208	1.7284	5.94480	7.7860	12.3659	18.957
Width	159.0	4.417486	1.685804	1.0476	3.38565	4.2485	5.5845	8.142

In [164]:
```
df.corr()
```

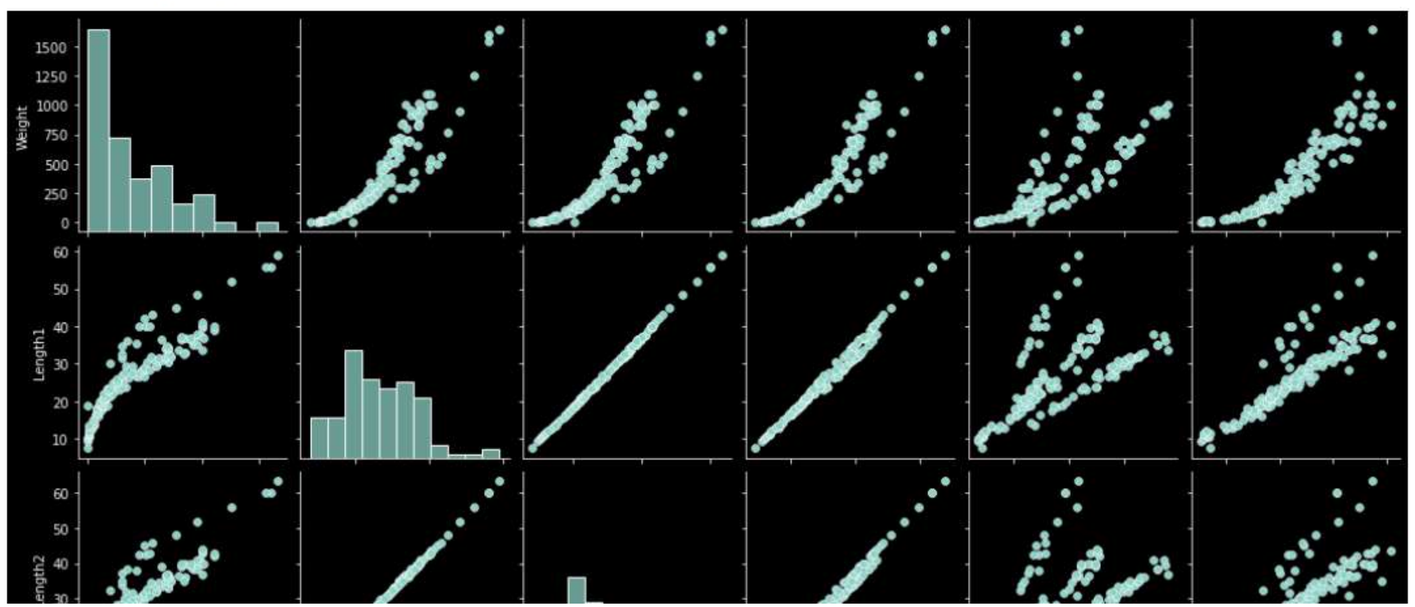
Out[164]:

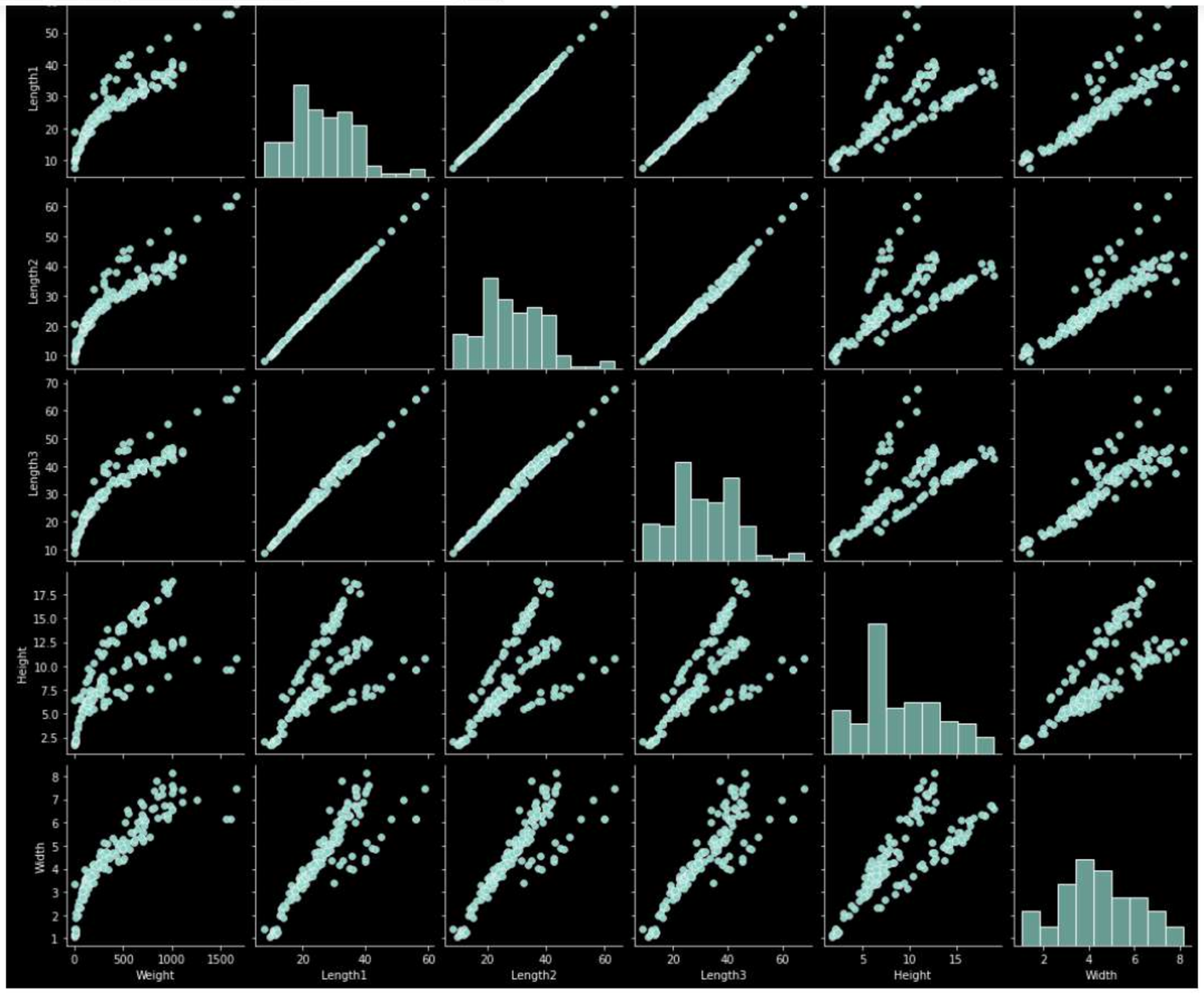
	Weight	Length1	Length2	Length3	Height	Width
Weight	1.000000	0.915712	0.918618	0.923044	0.724345	0.886507
Length1	0.915712	1.000000	0.999517	0.992031	0.625378	0.867050
Length2	0.918618	0.999517	1.000000	0.994103	0.640441	0.873547
Length3	0.923044	0.992031	0.994103	1.000000	0.703409	0.878520
Height	0.724345	0.625378	0.640441	0.703409	1.000000	0.792881
Width	0.886507	0.867050	0.873547	0.878520	0.792881	1.000000

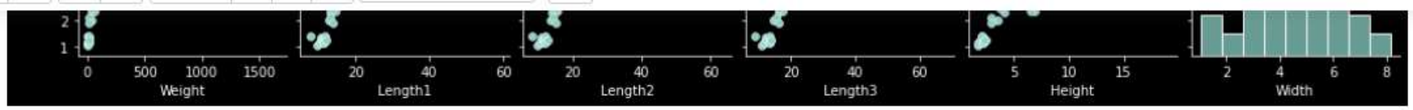
Data Visualization

In [165]:
```
plt.style.use('dark_background')
sns.pairplot(df)
```

Out[165]:
```
<seaborn.axisgrid.PairGrid at 0x7ff6763d39a0>
```

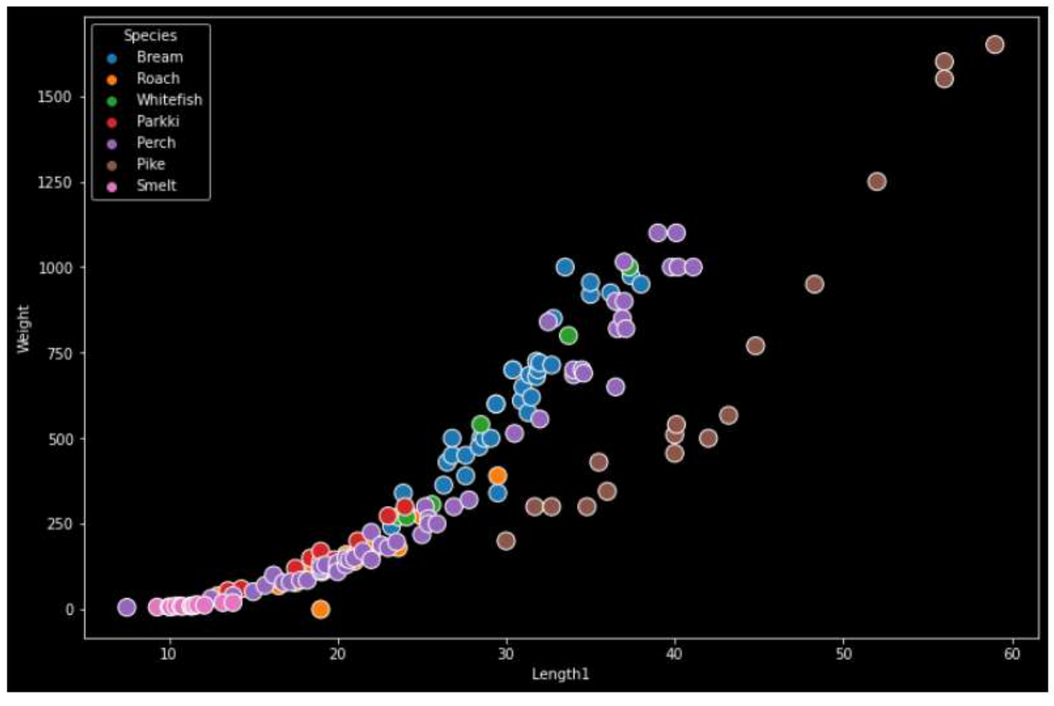






In [166]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Length1',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[166]: <AxesSubplot:xlabel='Length1', ylabel='Weight'>

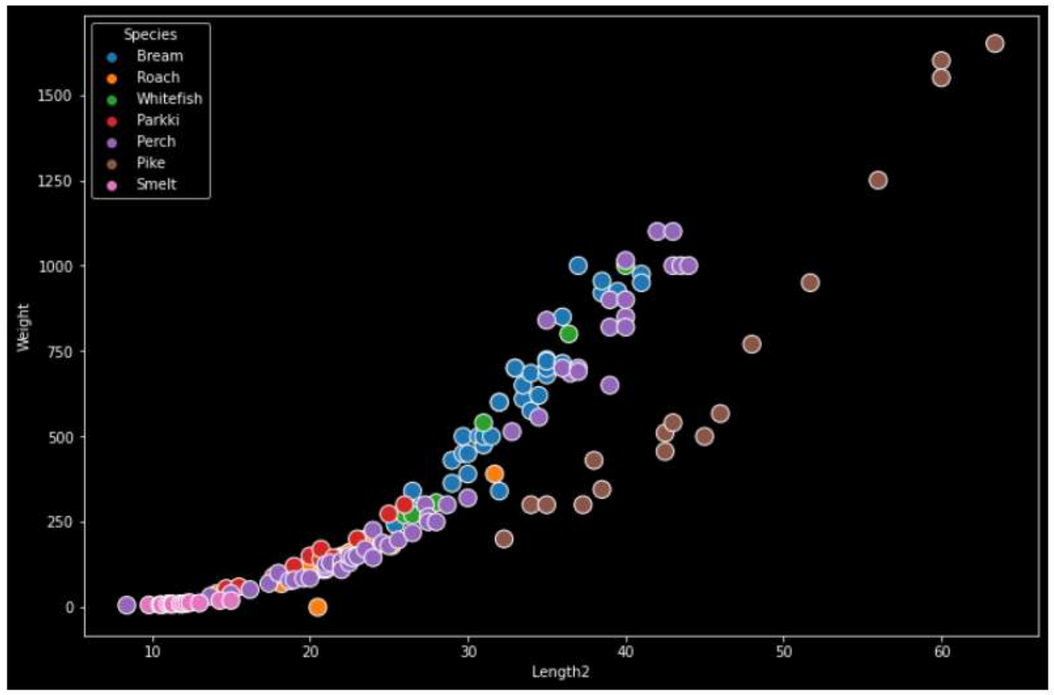


In [167]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Length2',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[167]: <AxesSubplot:xlabel='Length2', ylabel='Weight'>

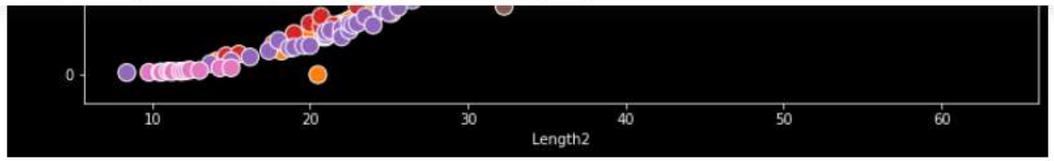
In [167]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Length2',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[167]: <AxesSubplot:xlabel='Length2', ylabel='Weight'>



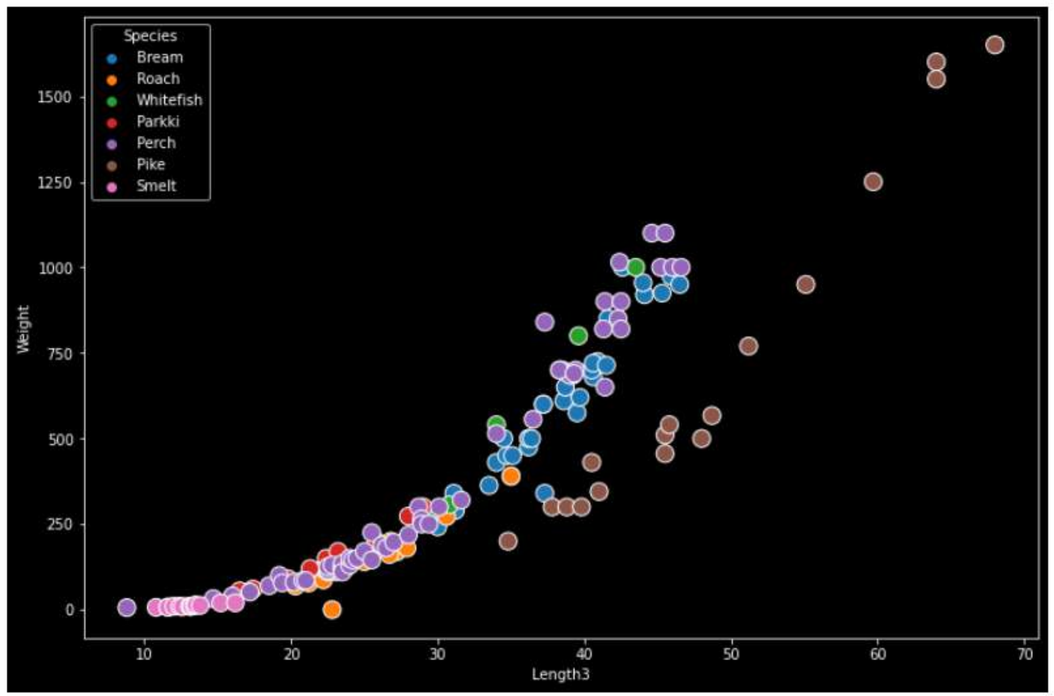
In [168]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Length3',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[168]: <AxesSubplot:xlabel='Length3', ylabel='Weight'>



In [168]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Length3',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[168]: <AxesSubplot:xlabel='Length3', ylabel='Weight'>

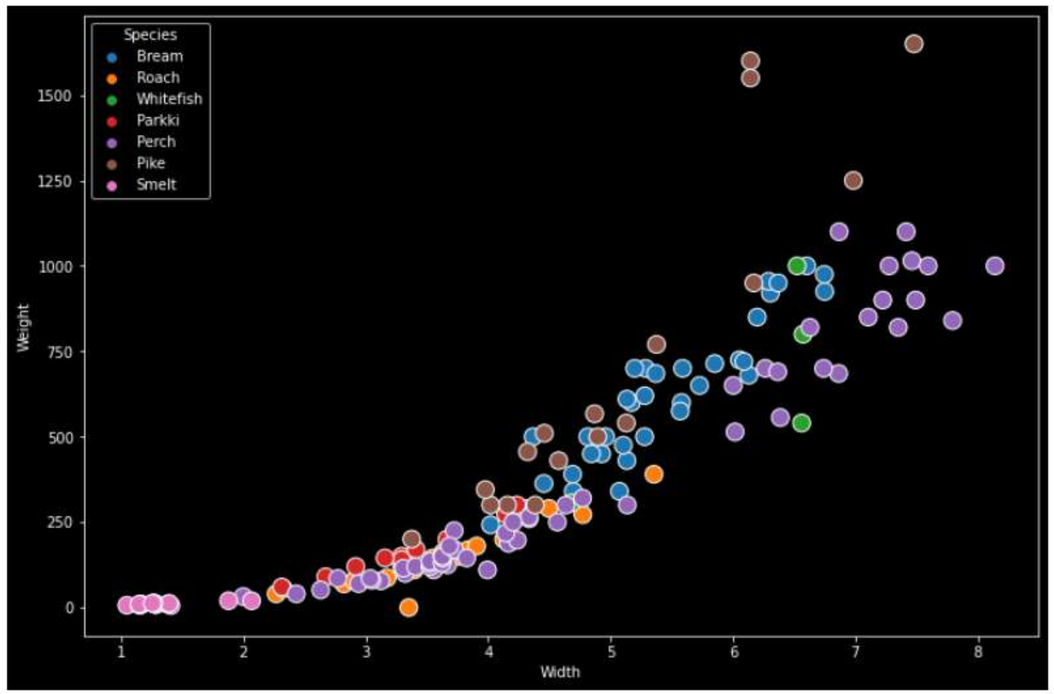


In [169]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Width',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[169]: <AxesSubplot:xlabel='Width', ylabel='Weight'>

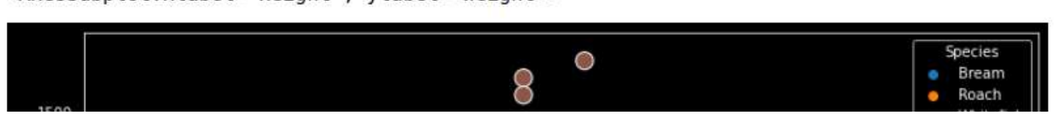
In [169]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Width',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[169]: <AxesSubplot:xlabel='Width', ylabel='Weight'>



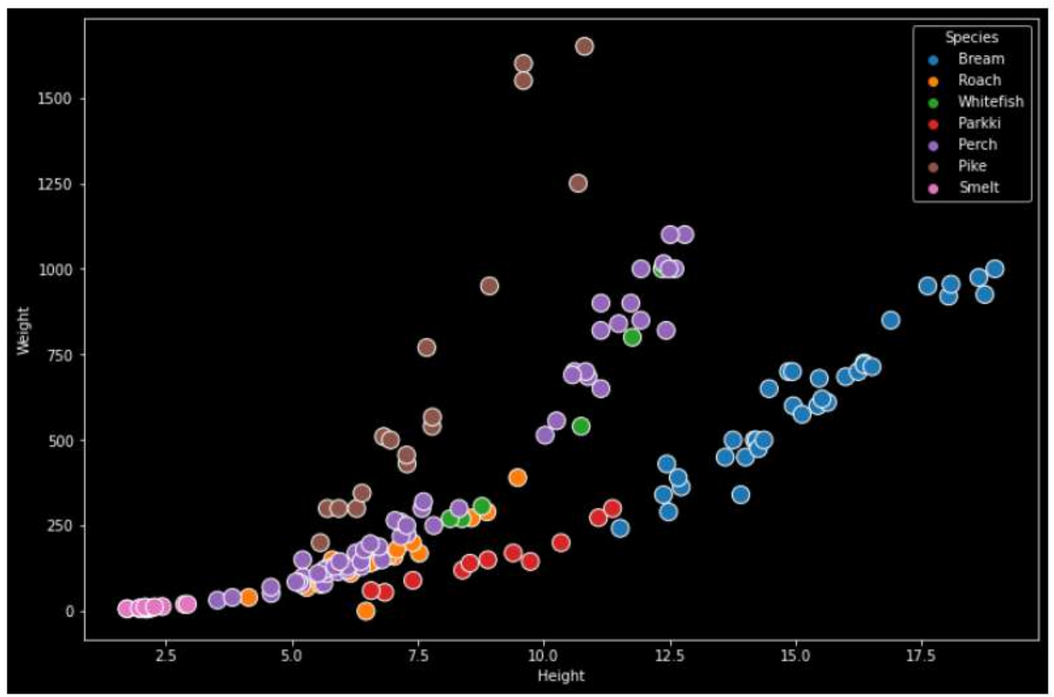
In [170]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Height',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[170]: <AxesSubplot:xlabel='Height', ylabel='Weight'>



In [170]:
```
plt.style.use('dark_background')
plt.figure(figsize=(12,8))
sns.scatterplot(x='Height',y='Weight',hue='Species',data=df,s=150,palette='tab10')
```

Out[170]: <AxesSubplot:xlabel='Height', ylabel='Weight'>



Drop Species column

In [171]:
```
df = df.drop('Species',axis=1)
```

Data Spliting

Drop Species column

In [171]:
```
df = df.drop('Species',axis=1)
```

Data Spliting

In [172]:
```
x=df.drop('Weight',axis=1).values
y=df['Weight'].values
```

In [173]:
```
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.2,random_state=100,shuffle=True)
```

Model

In [174]:
```
lm = LinearRegression()
lm.fit(train_x,train_y)
```

Out[174]:
```
LinearRegression()
```

In [175]:
```
print(f'Intercep : {lm.intercept_}')
print(f'Coefficent : {lm.coef_}')
```

```
Intercep : -508.4596714942162
Coefficent : [ 72.05073579 -25.63762345 -19.04539816  22.52267905  30.38242903]
```

In [176]:
```
pred = lm.predict(test_x)
```

Accuracy

In [177]:
```
r2_score(pred,test_y)
```

Out[177]:
```
0.8954100039412451
```

In [178]:
```
mean_absolute_error(pred,test_y)
```

Out[178]:
```
94.66620334873912
```

In [179]:
```
mean_squared_error(pred,test_y)
```

In [172]:
```
x=df.drop('Weight',axis=1).values
y=df['Weight'].values
```

In [173]:
```
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.2,random_state=100,shuffle=True)
```

Model

In [174]:
```
lm = LinearRegression()
lm.fit(train_x,train_y)
```

Out[174]:
```
LinearRegression()
```

In [175]:
```
print(f'Intercep : {lm.intercept_}')
print(f'Coefficent : {lm.coef_}')
```

```
Intercep : -508.4596714942162
Coefficent : [ 72.05073579 -25.63762345 -19.04539816  22.52267905  30.38242903]
```

In [176]:
```
pred = lm.predict(test_x)
```

Accuracy

In [177]:
```
r2_score(pred,test_y)
```

Out[177]:
```
0.8954100039412451
```

In [178]:
```
mean_absolute_error(pred,test_y)
```

Out[178]:
```
94.66620334873912
```

In [179]:
```
mean_squared_error(pred,test_y)
```

Out[179]:
```
12283.928623592332
```

Sumon Singh
30.4.22

AIM :- Write a program to implement Linear regression algorithm

1) Dataset Description:

A) Name of dataset:- Fish.csv

B) Description : This dataset consist of data of various kind of fishs. The data are weight, Length, Height etc. This dataset is a numerical dataset. The data set is available in Kaggle website.

c) Size: The dataset has 159 rows and 7 columns

D) Attributes: This dataset has 6 features.

a) Species: Name of the fish

b) ~~cte~~ Length1: Vertical Length of fish

c) Length 2: diagonal Length of fish

d) Length 3: Cross Length of fish

e) Height: height of fish

f) Width: Width of fish

E) Label: The target of this dataset is weight of the fish which is a numeric data in Gram.

Sumon Singh
30.4.22

2) Model Evaluation :-

A) Name :- Multiple linear regression

B) Type :- It's a supervised learning model which is used for regression type problem.

C) Algorithm :-

Input :- Data samples of traing dataset

Step1: Create n-dimension graph if the input dataset has n columns

Step2: plot all the attributes data points on the graph and generate an equation of straight line such that the output and the target data has the minimum error that means generate a straight line of best fit.

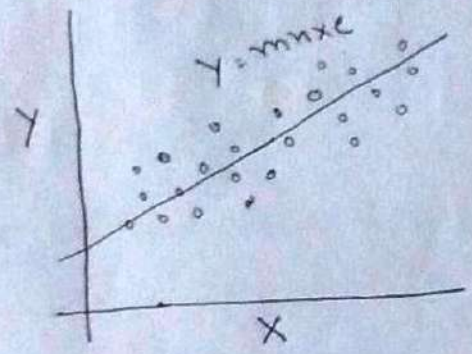
So, the equation will be :-

$$m_1 x_n + m_2 x_{n-1} + \ldots\ldots + b \cdot y$$

Step3: Based on the generated best fit line and the values of intercepts and slope, new datapoints are tested.

Sumon Singh
30.4.22

D) Draw :-



X → feature
Y → target

3) Result Analysis :-

a) r2 score :- 0.89, that means 89% of ~~target~~ predicte data is almost explained by the features correctly.

b) mean absolute error :- 94.667, that means the absolute error average of predicted and target data is 94.66

c) mean squared error :- 12283.9287, that means average square error of predicted and target data is 12283.9287

Implement Polynomial Regression

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import PolynomialFeatures,StandardScaler,MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

Load Dataset

In [2]:
```
df = pd.read_excel('./Real estate valuation data set.xlsx')
```

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

Out[3]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.916667	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.916667	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583333	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500000	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833333	5.0	390.56840	5	24.97937	121.54245	43.1

Drop non-important columns and rename columns

In [4]:
```
df.columns
```

Out[4]:
```
Index(['No', 'X1 transaction date', 'X2 house age',
```

4	5	2012.833333	5.0	390.56840	5	24.97937	121.54245	43.1

Drop non-important columns and rename columns

In [4]:
```
df.columns
```

Out[4]:
```
Index(['No', 'X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'Y house price of unit area'],
      dtype='object')
```

In [5]:
```
df.drop(['No','X1 transaction date'],axis=1,inplace=True)
```

In [6]:
```
df.head()
```

Out[6]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	32.0	84.87882	10	24.98298	121.54024	37.9
1	19.5	306.59470	9	24.98034	121.53951	42.2
2	13.3	561.98450	5	24.98746	121.54391	47.3
3	13.3	561.98450	5	24.98746	121.54391	54.8
4	5.0	390.56840	5	24.97937	121.54245	43.1

In [7]:
```
df.columns=['house_age','nearest_station','stores','latitude','longitude','price']
```

Data Exploration

In [8]:
```
df.shape
```

Out[8]:
```
(414, 6)
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
```

Out[8]: (414, 6)

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   house_age        414 non-null    float64
 1   nearest_station  414 non-null    float64
 2   stores           414 non-null    int64  
 3   latitude         414 non-null    float64
 4   longitude        414 non-null    float64
 5   price            414 non-null    float64
dtypes: float64(5), int64(1)
memory usage: 19.5 KB
```

In [10]:
```
df.isna().sum()
```

Out[10]:
```
house_age          0
nearest_station    0
stores             0
latitude           0
longitude          0
price              0
dtype: int64
```

In [11]:
```
df.describe().transpose()
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
house_age	414.0	17.712560	11.392485	0.00000	9.025000	16.10000	28.150000	43.80000
nearest_station	414.0	1083.885689	1262.109595	23.38284	289.324800	492.23130	1454.279000	6488.02100
stores	414.0	4.094203	2.945562	0.00000	1.000000	4.00000	6.000000	10.00000
latitude	414.0	24.969030	0.012410	24.93207	24.963000	24.97110	24.977455	25.01459
longitude	414.0	121.533361	0.015347	121.47353	121.528085	121.53863	121.543305	121.56627
price	414.0	37.980193	13.606488	7.60000	27.700000	38.45000	46.600000	117.50000

In [12]:
```
df.corr()
```

Out[12]:

	house_age	nearest_station	stores	latitude	longitude	price

longitude	414.0	121.533361	0.015347	121.47353	121.528085	121.53863	121.543305	121.56627
price	414.0	37.980193	13.606488	7.60000	27.700000	38.45000	46.600000	117.50000

In [12]:
```
df.corr()
```

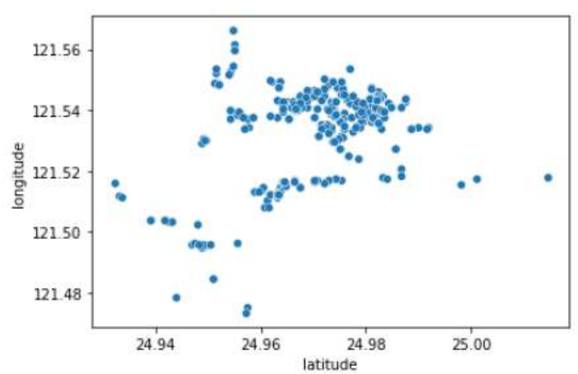
Out[12]:

	house_age	nearest_station	stores	latitude	longitude	price
house_age	1.000000	0.025622	0.049593	0.054420	-0.048520	-0.210567
nearest_station	0.025622	1.000000	-0.602519	-0.591067	-0.806317	-0.673613
stores	0.049593	-0.602519	1.000000	0.444143	0.449099	0.571005
latitude	0.054420	-0.591067	0.444143	1.000000	0.412924	0.546307
longitude	-0.048520	-0.806317	0.449099	0.412924	1.000000	0.523287
price	-0.210567	-0.673613	0.571005	0.546307	0.523287	1.000000

Data Visualization

In [13]:
```
sns.scatterplot(y='longitude',x='latitude',data=df)
```

Out[13]: <AxesSubplot:xlabel='latitude', ylabel='longitude'>

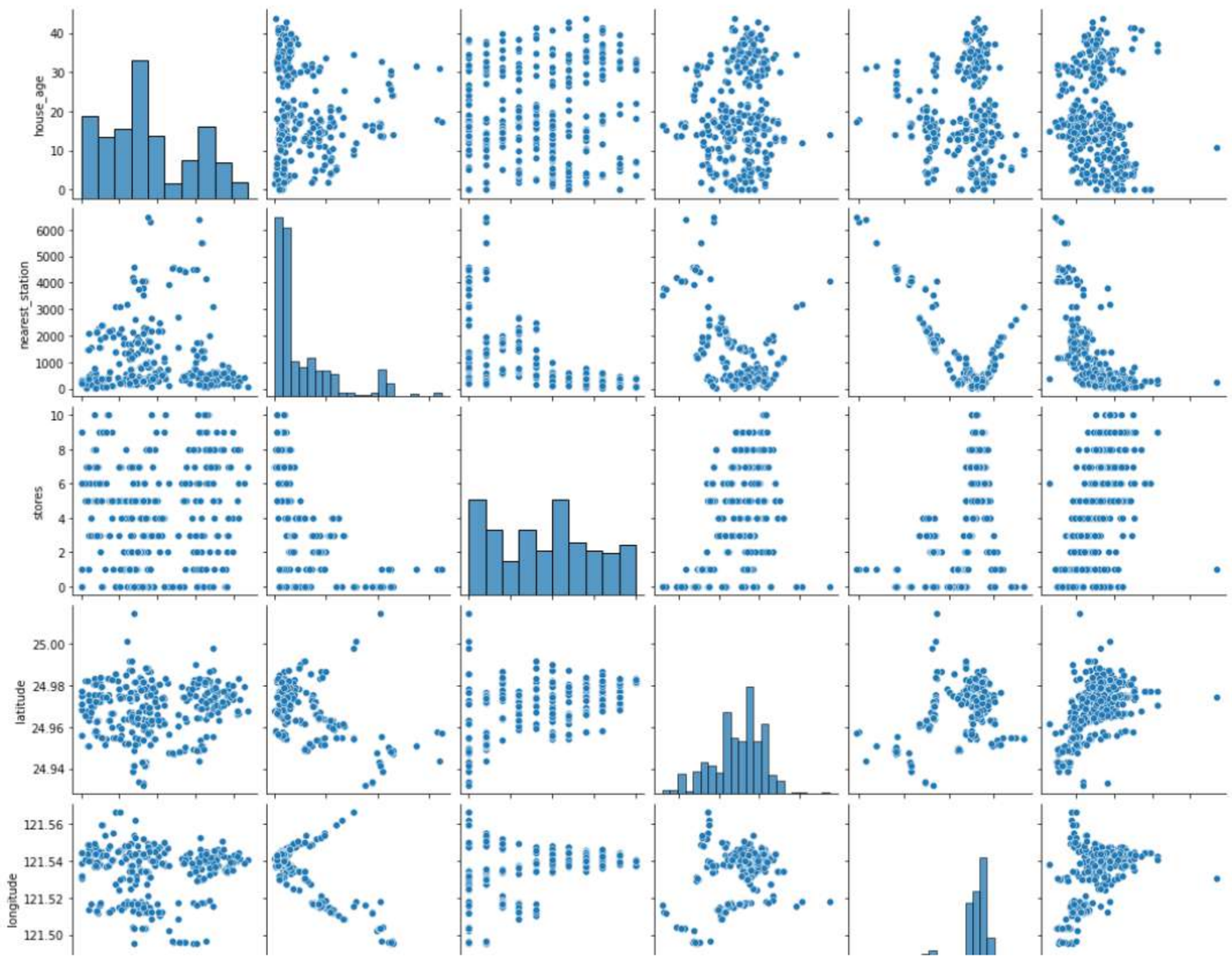


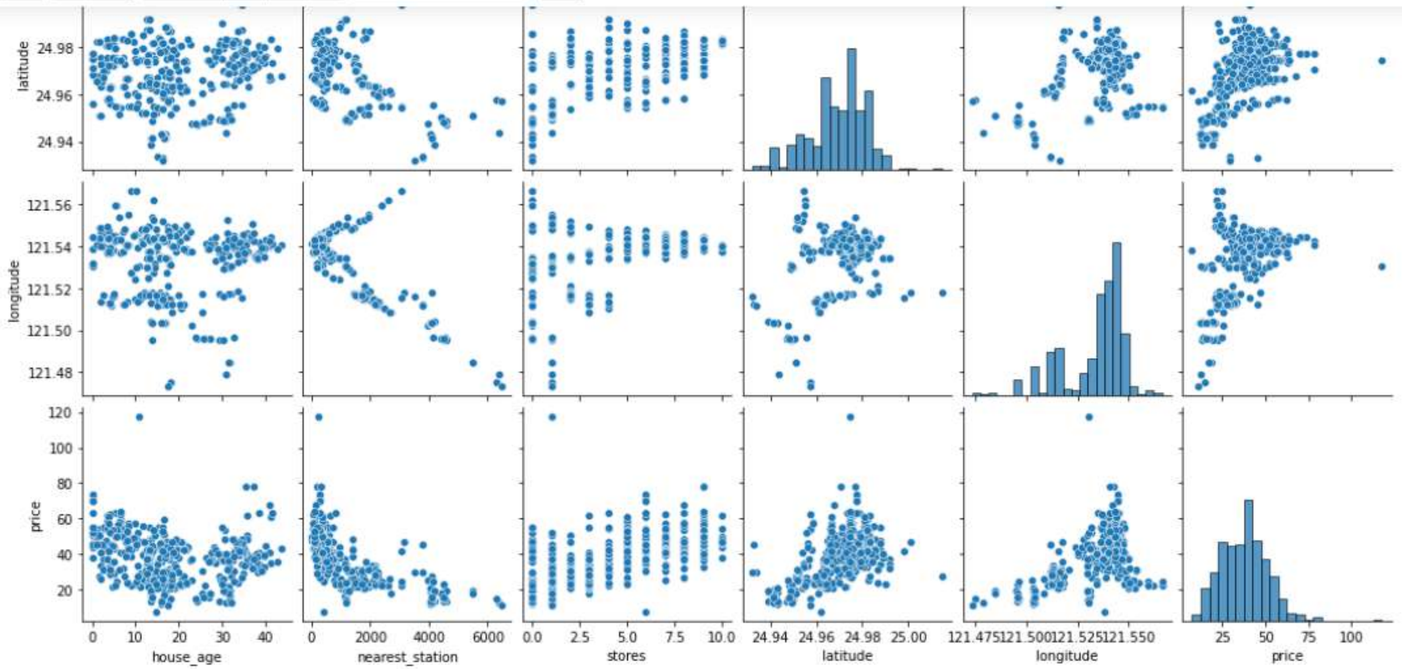
In [14]:
```
sns.pairplot(df)
```

Out[14]: <seaborn.axisgrid.PairGrid at 0x7ff9e40e4e50>

In [14]:
```
sns.pairplot(df)
```

Out[14]: <seaborn.axisgrid.PairGrid at 0x7ff9e40e4e50>





Data spilt

In [15]:
```
x=df.drop('price',axis=1).values
y=df['price'].values
```

In [16]:
```
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.2,random_state=100,shuffle=True)
```

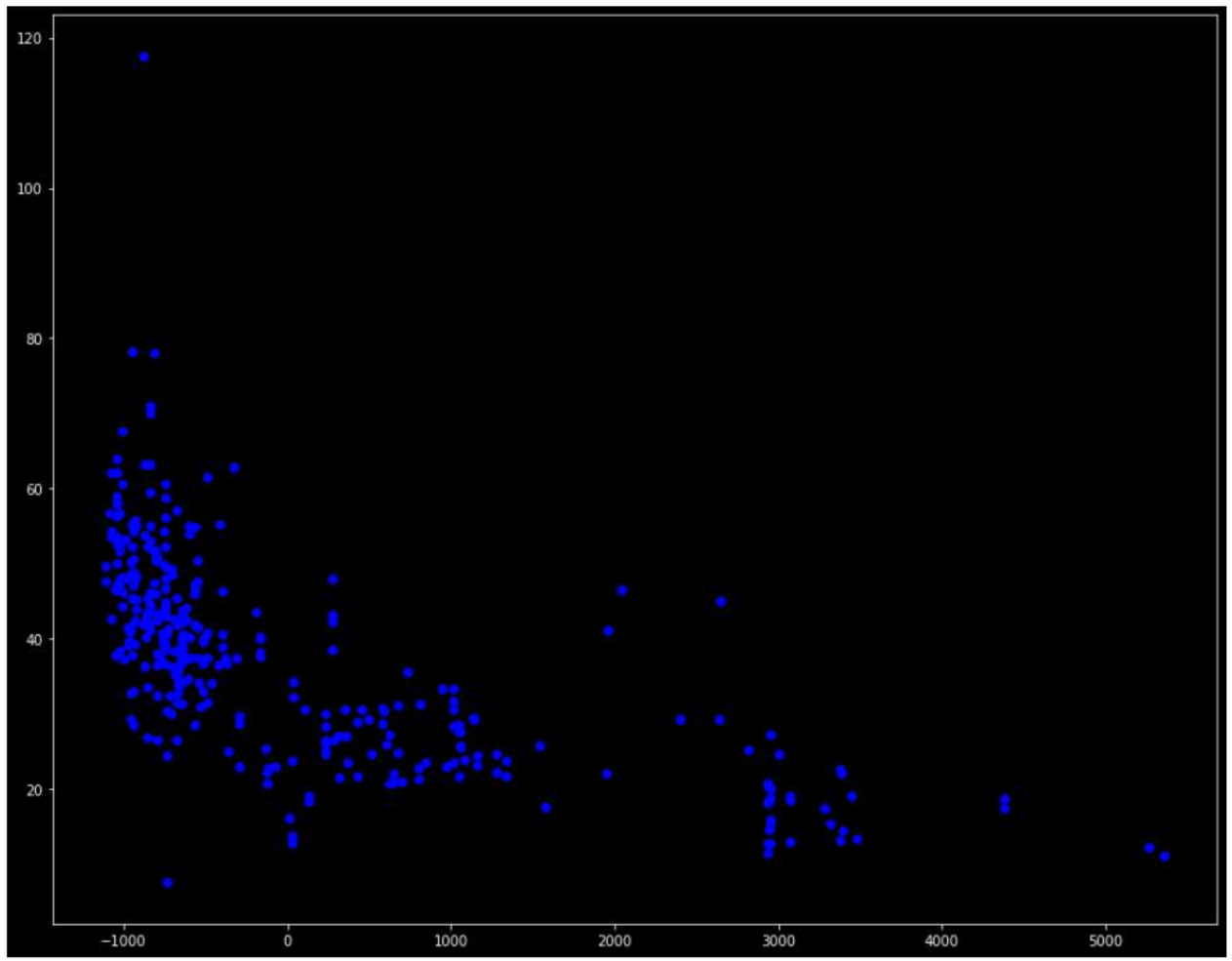
Data Engineering

In [17]:
```
pca=PCA(n_components=1)
train_x_pca=pca.fit_transform(train_x)
```

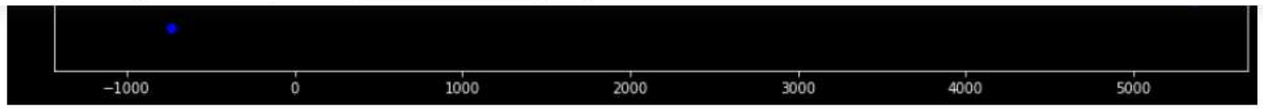
In [18]:
```
train_x_pca=train_x_pca.flatten()
```

In [19]:
```
plt.style.use('dark_background')
plt.figure(figsize=(15,12))
plt.plot(train_x_pca,train_y,'bo')
```

Out[19]: [<matplotlib.lines.Line2D at 0x7ff9d8fd0580>]



In [20]:
```
poly=PolynomialFeatures(degree=2)
```



In [20]:
```
poly=PolynomialFeatures(degree=2)
new_train_x=poly.fit_transform(train_x)
```

In [21]:
```
new_test_x=poly.transform(test_x)
```

Model

In [22]:
```
lm = LinearRegression().fit(new_train_x,train_y)
```

In [27]:
```
lm.score(new_train_x,train_y)
```

Out[27]: 0.6725663998506808

In [23]:
```
pred=lm.predict(new_test_x)
```

Accuracy

In [24]:
```
r2_score(pred,test_y)
```

Out[24]: 0.6881712261218402

In [25]:
```
mean_absolute_error(pred,test_y)
```

Out[25]: 5.076961015088552

In [26]:
```
mean_squared_error(pred,test_y)
```

Out[26]: 38.62497213303055

Sumon Singh
30.4.22

AIM :- Write a program to implement polynomial regression

1) Dataset Description:

A) Name of dataset: Real_estate_valuation_dataset.xlsx

B) Description: This dataset consist of data of houses based on which we evaluate prices of the houses. The dataset is available on kaggle website

c) Size :- This dataset consist of ~~414~~ 414 rows and 8 columns

D) Attributes :- This dataset has ~~8 features~~ 7 features, all of them are continious values.

a) NO. :- Serial No
b) transaction date
c) house age
d) distance to the nearest MRT station
e) Number of stores
f) Latitude
g) Longitude

30.4.22
Sumon Singh

E) Label :- The target data is price of the house which is generated based on features of the house.

2) Model Evaluation :-

A) Name :- Polynomial regression

B) Type :- This is a supervised learning model, used for regression type problem

C) Algorithm :-

Input :- features and target data of training dataset

step 1: Create n-dimension graph and plot all the datapoints ~~of the~~ on the graph

step 2: plot a best fit polynomial line on the datapoint of order 2 or more so that the generated output and the target data has the least error

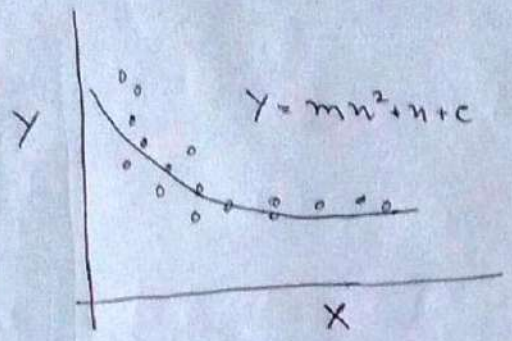
equation: $y = b_0 + b_1 x + b_2 x^2 + \dots b_n x^n$

where y is target, b_0 is intercept and $b_1, b_2 \dots b_n$ are slope

step 3: Based on the generated ~~graf~~ polynomial regression curve new data points are tested.

Sumon Singh
30.4.22

D) Draw :-



X → feature
Y → target

3) Result Analysis :-

* In this dataset we have applied 2 degree polynomial regression and the accuracy is 68%, that means predicted data are correctly recognized by the features.

* we have also tried other model such as Linear regression where accuracy is 57% and polynomial regression more than 2 degree is failing to be a good model so degree 2 polynomial is best for this dataset.

Implement Find-S Algorithm

Load Libraries

In [1]:
```
import pandas as pd
```

Load Dataset

In [2]:
```
df = pd.read_csv('./tennis.csv')
```

Explore Dataset

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

Out[3]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes

In [4]:
```
df.shape
```

Out[4]:
```
(14, 5)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   outlook   14 non-null     object
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   outlook   14 non-null     object
 1   temp      14 non-null     object
 2   humidity  14 non-null     object
 3   windy     14 non-null     bool  
 4   play      14 non-null     object
dtypes: bool(1), object(4)
memory usage: 590.0+ bytes
```

In [16]:
```
for i in df.columns:
    print(f'{i} : {df[i].unique()}')
```

```
outlook : ['sunny' 'overcast' 'rainy']
temp : ['hot' 'mild' 'cool']
humidity : ['high' 'normal']
windy : [False  True]
play : ['no' 'yes']
```

Split Dataset Into Attributes And Target

In [6]:
```
result=df['play'].values
```

In [7]:
```
attributes=df.drop('play',axis=1).values
```

Initialization Of Specific Hypothesis

In [8]:
```
H=['0']*attributes.shape[1]
```

In [9]:
```
print(f'Initial Hypothesis is : {H}')
```

```
Initial Hypothesis is : ['0', '0', '0', '0']
```

Implement The Logic Of Find-S Algorithm

In [9]:
```
print(f'Initial Hypothesis is : {H}')
```

```
Initial Hypothesis is : ['0', '0', '0', '0']
```

Implement The Logic Of Find-S Algorithm

In [10]:
```
for i in range(attributes.shape[0]):
    if result[i]=='yes':
        for j in range(attributes.shape[1]):
            if H[j]=='0':
                H[j]=attributes[i][j]
            elif H[j]!=attributes[i][j]:
                H[j]='?'
    print(f'For Step-{i} : {H}')
```

```
For Step-0 : ['0', '0', '0', '0']
For Step-1 : ['0', '0', '0', '0']
For Step-2 : ['overcast', 'hot', 'high', False]
For Step-3 : ['?', '?', 'high', False]
For Step-4 : ['?', '?', '?', False]
For Step-5 : ['?', '?', '?', False]
For Step-6 : ['?', '?', '?', '?']
For Step-7 : ['?', '?', '?', '?']
For Step-8 : ['?', '?', '?', '?']
For Step-9 : ['?', '?', '?', '?']
For Step-10 : ['?', '?', '?', '?']
For Step-11 : ['?', '?', '?', '?']
For Step-12 : ['?', '?', '?', '?']
For Step-13 : ['?', '?', '?', '?']
```

Final General Hypothesis

In [11]:
```
print(f'Final Hypothesis is : {H}')
```

```
Final Hypothesis is : ['?', '?', '?', '?']
```

Sumon Singh

Sumon Singh

30.4.22

AIM:- Implement and demonstrate the Find-S algorithm for finding the most specific hypothesis based on a given dataset. Read the traing data from a csv file or any other file format.

1) Dataset Description :-

A) Name of dataset :- tennis.csv

B) Description : This dataset consist of ~~wheat~~ weather condition based on few parameters and choose whether someone will go out for playing tennis on a certain weather. This is a binary class dataset. The dataset is available on Kaggle website.

C) Size :- The dataset consist of 14 rows and 5 columns.

D) Attributes :- This dataset has 4 features or attributes based on which output is generated. The attributes are-

1) Outlook :- This feature takes 3 values
   a) Sunny b) Overcast c) Rainy

2) temp :- This feature takes 3 values
   a) hot b) cool c) mild

3) humidity :- This feature takes 2 values
   a) high b) normal

4) windy :- This feature takes 2 values
   a) False b) True

Sumonsingh

E) Label : This dataset has a target which is play, it's a catagorical binary output which can be either yes or no.

F) Type of dataset : This is a catagorical dataset which has binary output.

2) Find-S algorithm :-

A) It is a basic concept learning algorithm in machine learning.

B) The find-S algorithm finds the most specific hypothesis that fits all the positive example. The algorithm only works on the positive training samples of the dataset.

C) Algorithm :-

Step 1 : Initialize h to the most specific hypothesis in H

Step 2 : for each positive training instance x
for each attribute constraint a, in h
If the constraint a, is satisfied by x
Then do nothing
Else replace a, in h by the next more general constraint that is satisfied by x

Step 3 : Output hypothesis h

Sumon Singh
30.4.22

30.4.22

Result analysis:-

The final specific hypothesis for our training dataset is (?,?,?,?) which indicates that the attributes are capable of accepting any value

Implement DecisionTree

Load Libraries

In [29]:
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder,OneHotEncoder
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from mlxtend.plotting import plot_confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Dataset

In [30]:
```
df = pd.read_csv('./drug200.csv')
```

In [31]:
```
df.head()
```

Out[31]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

Data Exploration

In [32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
```

Data Exploration

In [32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          200 non-null    int64  
 1   Sex          200 non-null    object 
 2   BP           200 non-null    object 
 3   Cholesterol  200 non-null    object 
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object 
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

In [33]:
```
df.isna().sum()
```

Out[33]:
```
Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64
```

In [34]:
```
df['Sex'].value_counts()
```

Out[34]:
```
M    104
F     96
Name: Sex, dtype: int64
```

In [35]:
```
df['BP'].value_counts()
```

Out[35]:
```
HIGH      77
LOW       64
NORMAL    59
Name: BP, dtype: int64
```

In [36]:
```
df['Cholesterol'].value_counts()
```

Out[36]:
```
HIGH      103
```

```
Name: BP, dtype: int64
```

In [36]:
```
df['Cholesterol'].value_counts()
```

Out[36]:
```
HIGH      103
NORMAL     97
Name: Cholesterol, dtype: int64
```

In [37]:
```
df['Drug'].value_counts()
```

Out[37]:
```
DrugY    91
drugX    54
drugA    23
drugB    16
drugC    16
Name: Drug, dtype: int64
```

Data Engineering

In [38]:
```
df['Drug']=LabelEncoder().fit_transform(df['Drug'])
```

In [39]:
```
df_object=df.select_dtypes('object')
```

In [40]:
```
df_object=df_object.drop('Sex',axis=1)
```

In [41]:
```
col=df_object.columns
```

In [42]:
```
df_object=OrdinalEncoder().fit_transform(df_object)
```

In [43]:
```
df[col]=df_object
```

In [44]:
```
df
```

Out[44]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	0.0	0.0	25.355	0
1	47	M	1.0	0.0	13.093	3
2	47	M	1.0	0.0	10.114	3
3	28	F	2.0	0.0	7.798	4
4	61	F	1.0	0.0	18.043	0

195	56	F	1.0	0.0	11.567	3
196	16	M	1.0	0.0	12.006	3
197	52	M	2.0	0.0	9.894	4
198	23	M	2.0	1.0	14.020	4
199	40	F	1.0	1.0	11.349	4

200 rows × 6 columns

In [45]:
```
df=pd.get_dummies(df)
```

In [46]:
```
df
```

Out[46]:

	Age	BP	Cholesterol	Na_to_K	Drug	Sex_F	Sex_M
0	23	0.0	0.0	25.355	0	1	0
1	47	1.0	0.0	13.093	3	0	1
2	47	1.0	0.0	10.114	3	0	1
3	28	2.0	0.0	7.798	4	1	0
4	61	1.0	0.0	18.043	0	1	0
...	...	...	...	...	...	...	...
195	56	1.0	0.0	11.567	3	1	0
196	16	1.0	0.0	12.006	3	0	1
197	52	2.0	0.0	9.894	4	0	1
198	23	2.0	1.0	14.020	4	0	1
199	40	1.0	1.0	11.349	4	1	0

200 rows × 7 columns

In [47]:
```
df['Drug'].value_counts()
```

Out[47]:
```
0    91
4    54
1    23
2    16
3    16
Name: Drug, dtype: int64
```

Data Spliting

In [48]:
```
X=df.drop('Drug',axis=1)
Y=df['Drug']
```

In [49]:
```
X
```

Out[49]:

	Age	BP	Cholesterol	Na_to_K	Sex_F	Sex_M
0	23	0.0	0.0	25.355	1	0
1	47	1.0	0.0	13.093	0	1
2	47	1.0	0.0	10.114	0	1
3	28	2.0	0.0	7.798	1	0
4	61	1.0	0.0	18.043	1	0
...	...	...	...	...	...	...
195	56	1.0	0.0	11.567	1	0
196	16	1.0	0.0	12.006	0	1
197	52	2.0	0.0	9.894	0	1
198	23	2.0	1.0	14.020	0	1
199	40	1.0	1.0	11.349	1	0

200 rows × 6 columns

In [50]:
```
train_x,test_x,train_y,test_y=train_test_split(X,Y,random_state=100,test_size=0.3)
```

Model

Select Best Parameters

In [51]:
```
para = {
    'criterion': ['gini','entropy'],
    'max_depth' : range(2,8),
    'max_features' : [2,3,4,5,6],
    'splitter' : ["best", "random"]
}
```

In [52]:
```
grid = GridSearchCV(DecisionTreeClassifier(),para,cv=10)
```

In [53]:
```
grid.fit(train_x,train_y)
```

```
/home/sumon/.local/lib/python3.8/site-packages/sklearn/model_selection/_split.py:666: UserWarning: The least populated class in y has only 8 members, which is less than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
```

Out[53]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 8),
                         'max_features': [2, 3, 4, 5, 6],
                         'splitter': ['best', 'random']})
```

In [54]:
```
grid.best_score_
```

Out[54]:
```
0.9928571428571429
```

In [55]:
```
grid.best_params_
```

Out[55]:
```
{'criterion': 'gini', 'max_depth': 4, 'max_features': 6, 'splitter': 'best'}
```

Create Model Using Best Parameters

In [56]:
```
model = DecisionTreeClassifier(max_depth=4,criterion='gini',max_features=6,splitter='best').fit(train_x,train_y)
```

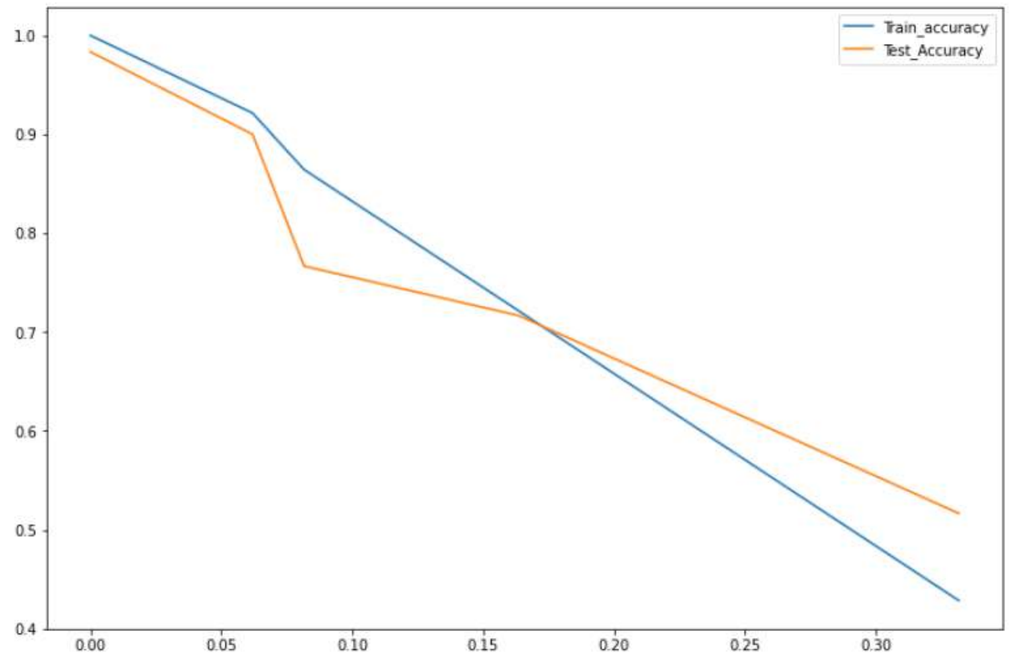
In [57]:
```
path = model.cost_complexity_pruning_path(train_x,train_y)
alphas=path['ccp_alphas']
alphas
```

Out[57]:
```
array([0.        , 0.06195055, 0.08163265, 0.16357339, 0.33160714])
```

In [58]:
```
accuracy_test=[]
accuracy_train=[]
for i in alphas:
    clf=DecisionTreeClassifier(ccp_alpha=i).fit(train_x,train_y)
    pred_test=clf.predict(test_x)
    pred_train=clf.predict(train_x)
    accuracy_test.append(accuracy_score(pred_test,test_y))
    accuracy_train.append(accuracy_score(pred_train,train_y))
```

In [59]:
```
plt.figure(figsize=(12,8))
sns.lineplot(x=alphas,y=accuracy_train,label='Train_accuracy')
sns.lineplot(x=alphas,y=accuracy_test,label='Test_Accuracy')
```

Out[59]: <AxesSubplot:>



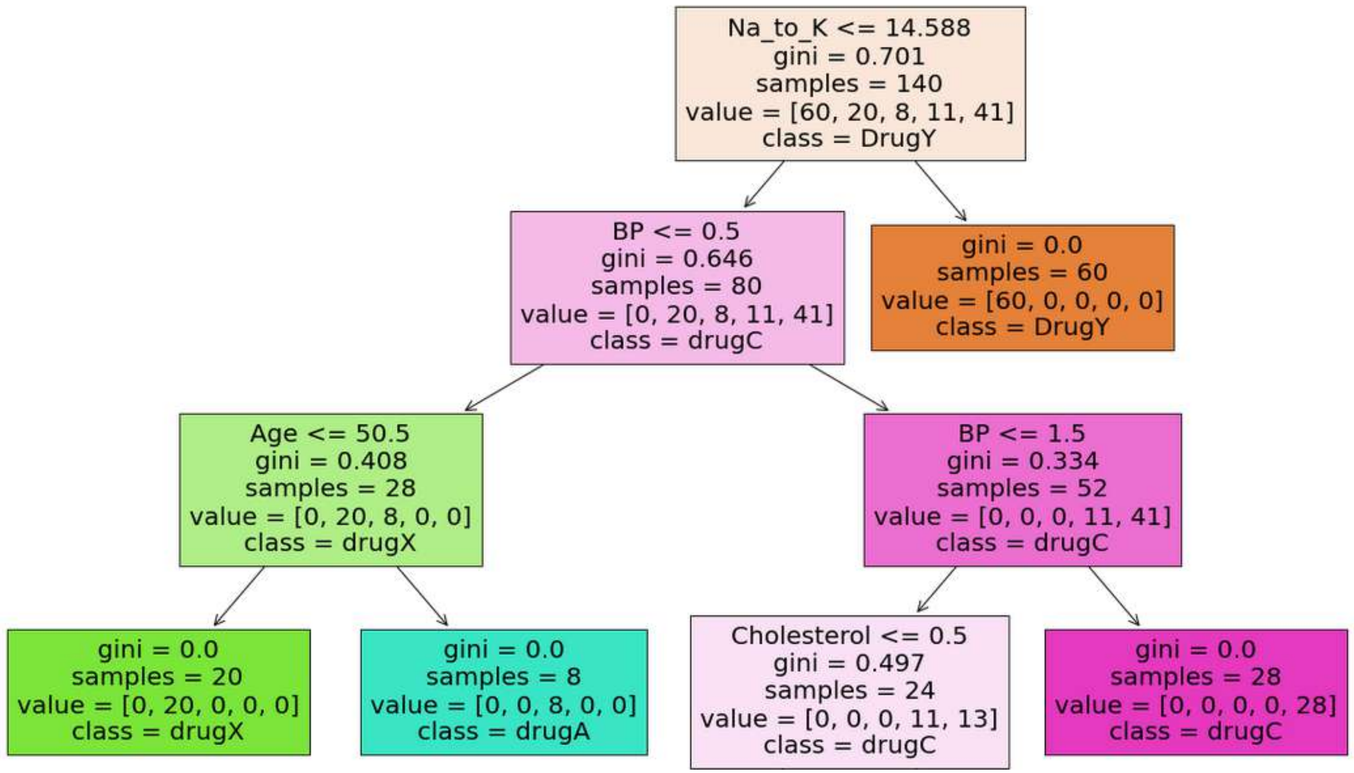
In [60]:
```
import sklearn.tree
```

In [61]:
```
print(sklearn.tree.export_text(model))
```

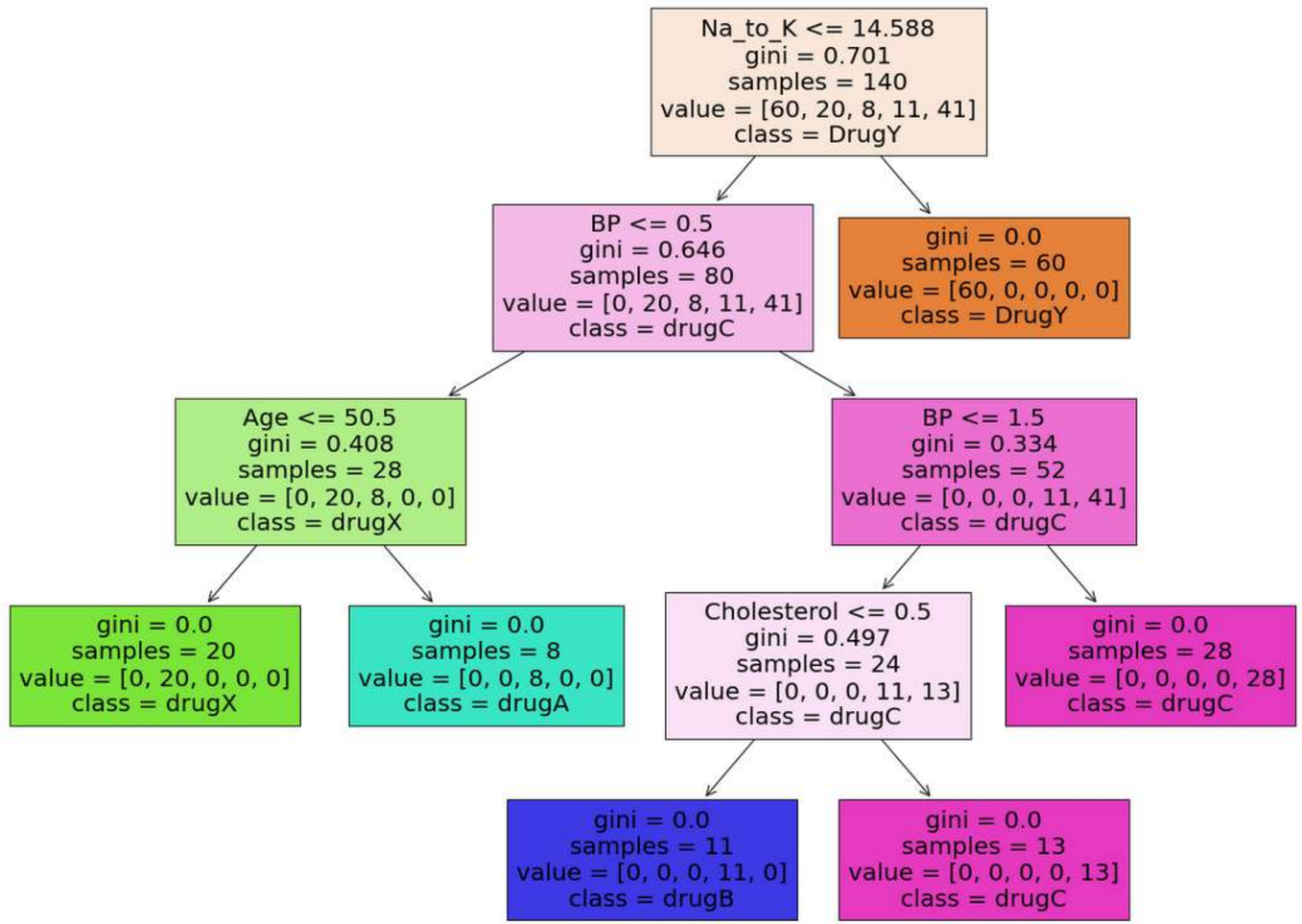
```
|--- feature_3 <= 14.59
|   |--- feature_1 <= 0.50
|   |   |--- feature_0 <= 50.50
|   |   |   |--- class: 1
|   |   |--- feature_0 >  50.50
|   |   |   |--- class: 2
|   |--- feature_1 >  0.50
|   |   |--- feature_1 <= 1.50
|   |   |   |--- feature_2 <= 0.50
|   |   |   |   |--- class: 3
|   |   |   |--- feature_2 >  0.50
|   |   |   |   |--- class: 4
|   |   |--- feature_1 >  1.50
```

In [62]:
```
plt.figure(figsize=(20,15))
plot_tree(model,filled=True,feature_names=X.columns,class_names=['DrugY', 'drugX', 'drugA', 'drugB', 'drugC'])
```

Out[62]:
```
[Text(697.5, 733.86, 'Na_to_K <= 14.588\ngini = 0.701\nsamples = 140\nvalue = [60, 20, 8, 11, 41]\nclass = DrugY'),
 Text(558.0, 570.78, 'BP <= 0.5\ngini = 0.646\nsamples = 80\nvalue = [0, 20, 8, 11, 41]\nclass = drugC'),
 Text(279.0, 407.70000000000005, 'Age <= 50.5\ngini = 0.408\nsamples = 28\nvalue = [0, 20, 8, 0, 0]\nclass = drugX
'),
 Text(139.5, 244.62, 'gini = 0.0\nsamples = 20\nvalue = [0, 20, 0, 0, 0]\nclass = drugX'),
 Text(418.5, 244.62, 'gini = 0.0\nsamples = 8\nvalue = [0, 0, 8, 0, 0]\nclass = drugA'),
 Text(837.0, 407.70000000000005, 'BP <= 1.5\ngini = 0.334\nsamples = 52\nvalue = [0, 0, 0, 11, 41]\nclass = drugC'),
 Text(697.5, 244.62, 'Cholesterol <= 0.5\ngini = 0.497\nsamples = 24\nvalue = [0, 0, 0, 11, 13]\nclass = drugC'),
 Text(558.0, 81.54000000000008, 'gini = 0.0\nsamples = 11\nvalue = [0, 0, 0, 11, 0]\nclass = drugB'),
 Text(837.0, 81.54000000000008, 'gini = 0.0\nsamples = 13\nvalue = [0, 0, 0, 0, 13]\nclass = drugC'),
 Text(976.5, 244.62, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 0, 0, 28]\nclass = drugC'),
 Text(837.0, 570.78, 'gini = 0.0\nsamples = 60\nvalue = [60, 0, 0, 0, 0]\nclass = DrugY')]
```



```
Text(837.0, 81.54000000000008, 'gini = 0.0\nsamples = 13\nvalue = [0, 0, 0, 0, 13]\nclass = drugC'),
Text(976.5, 244.62, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 0, 0, 28]\nclass = drugC'),
Text(837.0, 570.78, 'gini = 0.0\nsamples = 60\nvalue = [60, 0, 0, 0, 0]\nclass = DrugY')]
```



In [63]: `pred = model.predict(test_x)`

In [63]:
```
pred = model.predict(test_x)
```

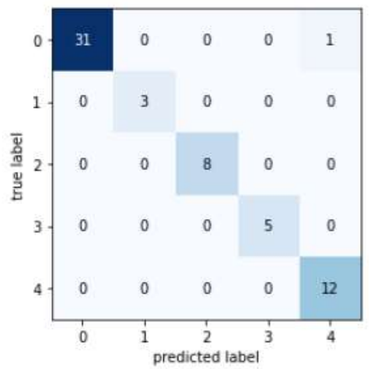
Accuracy

In [64]:
```
accuracy_score(pred,test_y)
```

Out[64]:
```
0.9833333333333333
```

In [65]:
```
plot_confusion_matrix(confusion_matrix(pred,test_y))
```

Out[65]:
```
(<Figure size 432x288 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



In [66]:
```
print(classification_report(pred,test_y))
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        32
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         8
           3       1.00      1.00      1.00         5
           4       0.92      1.00      0.96        12

    accuracy                           0.98        60
   macro avg       0.98      0.99      0.99        60
weighted avg       0.98      0.98      0.98        60
```

Sumon Singh
1.5.22

AIM: write a program to implement Decision tree algorithm

1) Dataset Description:-

A) Name: drug200.csv

B) Description: It is a classifier dataset where based on human feature such as Age, Sex, BP etc which kind of drug is required, ~~getting selected~~ The dataset is available on kaggle website

c) Size: This dataset has 200 rows and ~~5~~ 6 columns

D) Attributes; It has 5 features which are-

a) Age → numeric data

b) Sex → catagorical data (Male, female)

c) BP → Blood pressure which is catagorical data (High, low, Normal)

d) Cholesterol → It is a catagorial data (High, Normal)

e) Na_to_K → It is a continious data

E) Label: The target data is type of drug which is catagorical data (DrugY, drugC, drugX, drugA, drugB)

Sumon Singh
1.5.22

2) Model Evaluation:

A) Name :- Decision Tree classifier

B) Type :- It is a supervised learning model used in classification type problems.

C) Algorithm :-

Input: data D; set of features F

Steps: For Grow of Tree

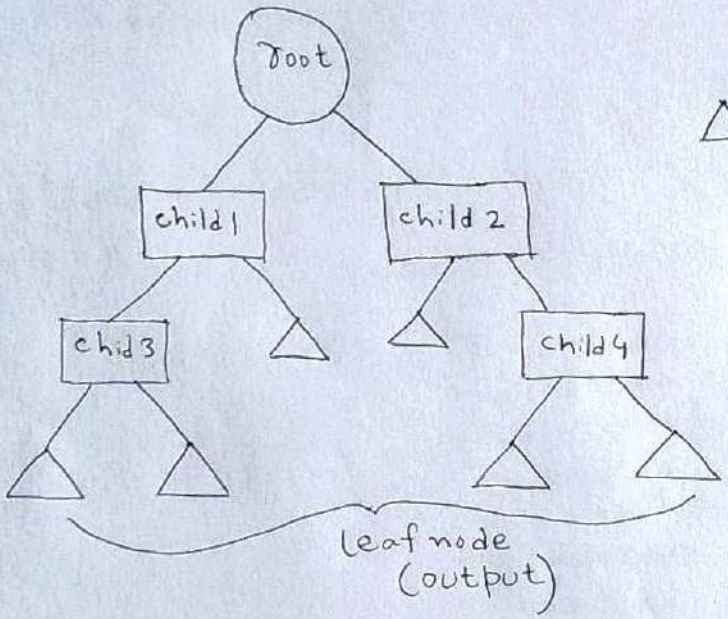
1. If Homogeneous(D) then return Label(D)
2. S ← BestSplit(D,F)
3. split D into subsets D_i according to the iterals in S;
4. for each i do
   a. If $D_i \neq \emptyset$ then $T_i \leftarrow$ Tree(D_i, F)
      else T_i is a leaf labled with Label(D)
   b. end
5. return a tree whose root is labelled with s and whose children are T_i

Steps: For Bestsplit of Tree

1. $I_{min} \leftarrow 1$
2. for each $f \in F$ do
   a. split D into subsets $D_1, \ldots D_i$ according to the values v_i of f;
   b. If $Imp(\{D_1, \ldots D_i\}) < I_{min}$ then
      $I_{min} \leftarrow Imp(\{D_1, \ldots D_i\})$
      $f_{best} \leftarrow f$
   end
   c. end
3. return f_{best}

Sumon Singh
1.5.22

B) Draw :-



△ → Leaf node

A depth 3 tree

3) Result Analysis :-

Accuracy :- 0.983 ; The model is able to classify 98% of data almost correctly so the model works very good on our dataset

Precision :- Except drugX all the drugs are classified perfectly whose precision is 1 whereas precision of drugX is 92%. That means all the Labels are classified very much well.

Recall :- All drugs except DrugY has recall 1, whereas recall of DrugY is 97% that means the model identifies their corresponding labels pretty good.

Implement RandomForestTree

Load Libraries

In [26]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import OneHotEncoder,LabelEncoder,OrdinalEncoder
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV
from xgboost import XGBRegressor
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression,Ridge,Lasso
```

Load Dataset

In [27]:
```
df = pd.read_csv('./insurance.csv')
```

In [28]:
```
df
```

Out[28]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

In [29]:
```
df.isnull().sum()
```

Out[29]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [30]:
```
df.isna().sum()
```

Out[30]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

Data Transformation

In [32]:
```
obj = df.select_dtypes('object')
```

In [33]:
```
obj
```

Out[33]:

Data Transformation

In [32]:
```
obj = df.select_dtypes('object')
```

In [33]:
```
obj
```

Out[33]:

	sex	smoker	region
0	female	yes	southwest
1	male	no	southeast
2	male	no	southeast
3	male	no	northwest
4	male	no	northwest
...	...	...	...
1333	male	no	northwest
1334	female	no	northeast
1335	female	no	southeast
1336	female	no	southwest
1337	female	yes	northwest

1338 rows × 3 columns

In [34]:
```
for i in obj.columns:
    print(obj[i].unique())
```

```
['female' 'male']
['yes' 'no']
['southwest' 'southeast' 'northwest' 'northeast']
```

In [35]:
```
#df_obj = pd.get_dummies(obj,drop_first=True)
tran = LabelEncoder()
obj['sex']=tran.fit_transform(obj['sex'])
obj['smoker']=tran.fit_transform(obj['smoker'])
obj['region']=tran.fit_transform(obj['region'])
```

```
/tmp/ipykernel_8965/2828646237.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  obj['region']=tran.fit_transform(obj['region'])
```

In [36]:
```
df_num = df.select_dtypes(exclude='object')
df_num
```

Out[36]:

	age	bmi	children	charges
0	19	27.900	0	16884.92400
1	18	33.770	1	1725.55230
2	28	33.000	3	4449.46200
3	33	22.705	0	21984.47061
4	32	28.880	0	3866.85520
...	...	...	...	...
1333	50	30.970	3	10600.54830
1334	18	31.920	0	2205.98080
1335	18	36.850	0	1629.83350
1336	21	25.800	0	2007.94500
1337	61	29.070	0	29141.36030

1338 rows × 4 columns

In [37]:
```
df_new = pd.concat([obj,df_num],axis=1)
```

In [38]:
```
df_new
```

Out[38]:

	sex	smoker	region	age	bmi	children	charges
0	0	1	3	19	27.900	0	16884.92400
1	1	0	2	18	33.770	1	1725.55230
2	1	0	2	28	33.000	3	4449.46200
3	1	0	1	33	22.705	0	21984.47061
4	1	0	1	32	28.880	0	3866.85520
...	...	...	...	...	...	...	...
1333	1	0	1	50	30.970	3	10600.54830
1334	0	0	0	18	31.920	0	2205.98080

1338 rows × 7 columns

In [39]:
```
df_new.describe()
```

Out[39]:

	sex	smoker	region	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	0.505232	0.204783	1.515695	39.207025	30.663397	1.094918	13270.422265
std	0.500160	0.403694	1.104885	14.049960	6.098187	1.205493	12110.011237
min	0.000000	0.000000	0.000000	18.000000	15.960000	0.000000	1121.873900
25%	0.000000	0.000000	1.000000	27.000000	26.296250	0.000000	4740.287150
50%	1.000000	0.000000	2.000000	39.000000	30.400000	1.000000	9382.033000
75%	1.000000	0.000000	2.000000	51.000000	34.693750	2.000000	16639.912515
max	1.000000	1.000000	3.000000	64.000000	53.130000	5.000000	63770.428010

In [40]:
```
df_new.corr()['charges']
```

Out[40]:
```
sex         0.057292
smoker      0.787251
region     -0.006208
age         0.299008
bmi         0.198341
children    0.067998
charges     1.000000
Name: charges, dtype: float64
```

In [41]:
```
X=df_new.drop('charges',axis=1)
Y=df_new['charges']
```

In [42]:
```
df_new.dtypes
```

Out[42]:
```
sex           int64
smoker        int64
region        int64
age           int64
bmi         float64
children      int64
charges     float64
dtype: object
```

```
dtype: object
```

Data Spliting

In [67]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
```

RandomForest Tree

In [68]:
```
clf_for = RandomForestRegressor(n_estimators=20)
```

In [69]:
```
cross_val_score(clf_for,X_train,Y_train,cv=10).mean()
```

Out[69]:
```
0.814597012086715
```

In [70]:
```
clf_for.fit(X_train,Y_train)
```

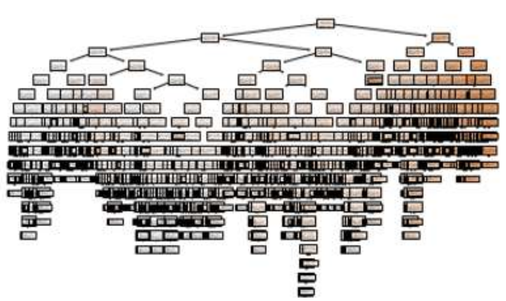
Out[70]:
```
RandomForestRegressor(n_estimators=20)
```

Plot a tree from the forest

In [71]:
```
import sklearn.tree
```

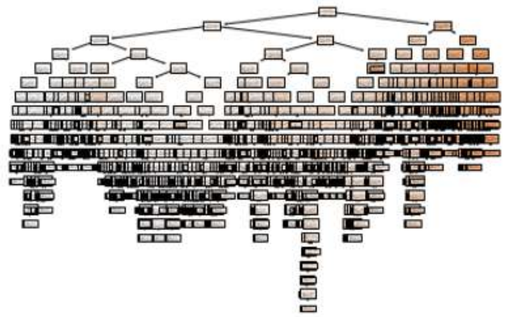
In [72]:
```
sklearn.tree.plot_tree(clf_for.estimators_[15],filled=True)
```

```
Text(242.0406437361944, 123.54545454545455, 'mse = 0.0\nsamples = 1\nvalue = 13451.122'),
Text(242.8858314925844, 123.54545454545455, 'mse = -0.0\nsamples = 1\nvalue = 24513.091'),
Text(242.2519406752919, 143.31272727272727, 'mse = -0.0\nsamples = 1\nvalue = 28923.137'),
Text(248.1418428526349, 153.19636363636363, 'X[3] <= 60.5\nmse = 560159.28\nsamples = 25\nvalue = 13373.88'),
...]
```



In [72]:
```
sklearn.tree.plot_tree(clf_for.estimators_[15],filled=True)
```

```
Text(242.0406437361944, 123.54545454545455, 'mse = 0.0\nsamples = 1\nvalue = 13451.122'),
Text(242.8858314925844, 123.54545454545455, 'mse = -0.0\nsamples = 1\nvalue = 24513.091'),
Text(242.2519406752919, 143.31272727272727, 'mse = -0.0\nsamples = 1\nvalue = 28923.137'),
Text(248.1418428526349, 153.19636363636363, 'X[3] <= 60.5\nmse = 560159.28\nsamples = 25\nvalue = 13373.88'),
...]
```



In [73]:
```
pred=clf_for.predict(X_test)
```

Accuracy

In [74]:
```
r2_score(pred,Y_test)
```

Out[74]: 0.8561859521545946

In [75]:
```
mean_absolute_error(pred,Y_test)
```

Out[75]: 2415.040304353545

In [76]:
```
mean_squared_error(pred,Y_test)
```

Out[76]: 20989200.97864216

Sumon Singh
1.5.22

AIM:- Write a program to implement Random-Forest tree algorithm

A) Dataset Description:-

1) Name:- ~~insurances~~ insurance.csv

2) Description:- This dataset consist of information of people regarding their insurance. Based on few features like age, sex, bmi etc how much should be the insurance charges is being generated. It is a continious dataset. The dataset is available on kaggle website.

3) Size:- This dataset has 1338 rows and 7 columns

4) Attributes:- The dataset has 6 attributes age, sex, bmi, children, smoker, region. Age, bmi and children are numeric data whereas others take catagorical variable

5) Label:- The target data is charges which should be paid by the client based on his features. It's a continious data.

Sumon Singh
1.5.22

B) Model Evaluation:

1) Name:- Random Forest Regressor

2) Type:- It's a supervised ensemble learning model used in regression type problem.

3) Algorithm:-

Input: A dataset with m-features

Step 1: Randomly select K features from total m features, where K<<m

step 2: Among K features, calculate the node d using best split point.

step 3: split the node into children node using best split

step 4: Repeat step 1 to 3 until the required depth of the tree is reached

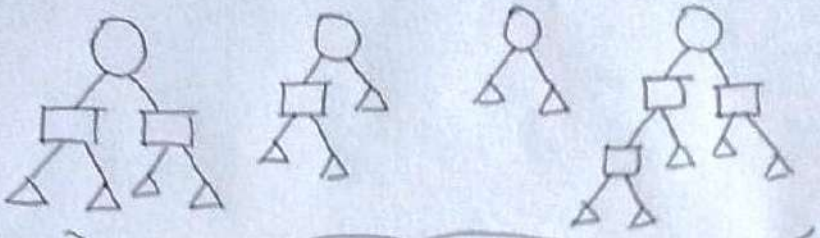
step 5: Repeat step 1 to 4 for n times to create n number of trees of the forest.

step 6: After creation of the forest take the average of ~~all~~ the outputs of all the trees of the forest.

step 7: The average output is the result

Sumon Singh
1.5.22

4) Draw:-



A Randomforesttree of 4 estimeters

c) Result analysis :-

* R2-score of RandomForestRegressor is 85%. that means the model is able to predict 85% of data almost correctly.
* In the model 20 estimeters have been computed for the final output.
* mean squard error of predicted and target data is 22112277.28, as it is a regressor problem so the average is high.

Implement SVM

Load Libraries

In [49]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.preprocessing import MinMaxScaler
from mlxtend.plotting import plot_decision_regions,plot_confusion_matrix
```

Load Dataset

In [50]:
```
df = pd.read_csv('./UniversalBank.csv',index_col='ID')
```

In [51]:
```
df.head()
```

Out[51]:

ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
5	35	8	45	91330	4	1.0	2	0	0	0	0	0	1

Explore Dataset

In [52]:
```
df.shape
```

Out[52]:
```
(5000, 13)
```

In [53]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 1 to 5000
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
```

Out[52]: (5000, 13)

In [53]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 1 to 5000
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Age                 5000 non-null   int64  
 1   Experience          5000 non-null   int64  
 2   Income              5000 non-null   int64  
 3   ZIP Code            5000 non-null   int64  
 4   Family              5000 non-null   int64  
 5   CCAvg               5000 non-null   float64
 6   Education           5000 non-null   int64  
 7   Mortgage            5000 non-null   int64  
 8   Personal Loan       5000 non-null   int64  
 9   Securities Account  5000 non-null   int64  
 10  CD Account          5000 non-null   int64  
 11  Online              5000 non-null   int64  
 12  CreditCard          5000 non-null   int64  
dtypes: float64(1), int64(12)
memory usage: 546.9 KB
```

In [54]:
```
df.isna().sum()
```

Out[54]:
```
Age                 0
Experience          0
Income              0
ZIP Code            0
Family              0
CCAvg               0
Education           0
Mortgage            0
Personal Loan       0
Securities Account  0
CD Account          0
Online              0
CreditCard          0
dtype: int64
```

In [55]:
```
df.describe().transpose()
```

Out[55]:

	count	mean	std	min	25%	50%	75%	max
Age	5000.0	45.338400	11.463166	23.0	35.0	45.0	55.0	67.0
Experience	5000.0	20.104600	11.467954	-3.0	10.0	20.0	30.0	43.0
Income	5000.0	73.774200	46.033729	8.0	39.0	64.0	98.0	224.0
ZIP Code	5000.0	93152.503000	2121.852197	9307.0	91911.0	93437.0	94608.0	96651.0
Family	5000.0	2.396400	1.147663	1.0	1.0	2.0	3.0	4.0

Out[55]:

	count	mean	std	min	25%	50%	75%	max
Age	5000.0	45.338400	11.463166	23.0	35.0	45.0	55.0	67.0
Experience	5000.0	20.104600	11.467954	-3.0	10.0	20.0	30.0	43.0
Income	5000.0	73.774200	46.033729	8.0	39.0	64.0	98.0	224.0
ZIP Code	5000.0	93152.503000	2121.852197	9307.0	91911.0	93437.0	94608.0	96651.0
Family	5000.0	2.396400	1.147663	1.0	1.0	2.0	3.0	4.0
CCAvg	5000.0	1.937938	1.747659	0.0	0.7	1.5	2.5	10.0
Education	5000.0	1.881000	0.839869	1.0	1.0	2.0	3.0	3.0
Mortgage	5000.0	56.498800	101.713802	0.0	0.0	0.0	101.0	635.0
Personal Loan	5000.0	0.096000	0.294621	0.0	0.0	0.0	0.0	1.0
Securities Account	5000.0	0.104400	0.305809	0.0	0.0	0.0	0.0	1.0
CD Account	5000.0	0.060400	0.238250	0.0	0.0	0.0	0.0	1.0
Online	5000.0	0.596800	0.490589	0.0	0.0	1.0	1.0	1.0
CreditCard	5000.0	0.294000	0.455637	0.0	0.0	0.0	1.0	1.0

In [56]:
```
df.corr()['CreditCard']
```

Out[56]:
```
Age                 0.007681
Experience          0.008967
Income             -0.002385
ZIP Code            0.007691
Family              0.011588
CCAvg              -0.006689
Education          -0.011014
Mortgage           -0.007231
Personal Loan       0.002802
Securities Account -0.015028
CD Account          0.278644
Online              0.004210
CreditCard          1.000000
Name: CreditCard, dtype: float64
```

Data Engineering

Data Spliting

In [57]:
```
X=df.drop('CreditCard',axis=1)
Y=df['CreditCard']
```

In [58]:
```
train_x,test_x,train_y,test_y=train_test_split(X,Y,test_size=0.2,random_state=100,stratify=Y)
```

In [58]:
```
train_x,test_x,train_y,test_y=train_test_split(X,Y,test_size=0.2,random_state=100,stratify=Y)
```

Drop non-important features

In [59]:
```
df.drop('ZIP Code',axis=1,inplace=True)
```

In [60]:
```
scale = MinMaxScaler().fit(train_x)
```

In [61]:
```
train_x=scale.transform(train_x)
test_x=scale.transform(test_x)
```

Select Best Parameters For SVC Model

In [62]:
```
para = {
    'kernel':['linear', 'poly', 'rbf'],
    'C' : [0,1,10,50],
    'gamma':[0.1, 0.2, 0.5]
}
```

In [63]:
```
grid=GridSearchCV(SVC(),para,cv=10)
```

In [64]:
```
grid.fit(train_x,train_y)
```

```
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/sumon/.local/lib/python3.8/site-packages/sklearn/svm/_base.py", line 226, in fit
    fit(X, y, sample_weight, solver_type, kernel, random_seed=seed)
  File "/home/sumon/.local/lib/python3.8/site-packages/sklearn/svm/_base.py", line 277, in _dense_fit
    self._probB, self.fit_status_ = libsvm.fit(
  File "sklearn/svm/_libsvm.pyx", line 192, in sklearn.svm._libsvm.fit
ValueError: C <= 0

  warnings.warn("Estimator fit failed. The score on this train-test"

/home/sumon/.local/lib/python3.8/site-packages/sklearn/model_selection/_search.py:922: UserWarning: One or more of
the test scores are non-finite: [    nan     nan     nan     nan     nan     nan     nan     nan     nan
 0.74125 0.7395  0.74125 0.74125 0.745   0.74475 0.74125 0.74575 0.7455
 0.74125 0.745   0.746   0.74125 0.74575 0.746   0.74125 0.744   0.745
 0.74125 0.746   0.746   0.74125 0.7455  0.74525 0.74125 0.7395  0.741  ]
  warnings.warn(
```

Out[64]:
```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': [0, 1, 10, 50], 'gamma': [0.1, 0.2, 0.5],
                         'kernel': ['linear', 'poly', 'rbf']})
```

In [65]:
```
grid.best_score_
```

Out[65]:
```
0.7460000000000001
```

In [65]:
```
grid.best_score_
```
Out[65]:
```
0.7460000000000001
```

In [66]:
```
grid.best_params_
```
Out[66]:
```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

Model

In [67]:
```
clf = SVC(C=10,gamma=0.1,kernel='rbf')
```

In [68]:
```
clf.fit(train_x,train_y)
```
Out[68]:
```
SVC(C=10, gamma=0.1)
```

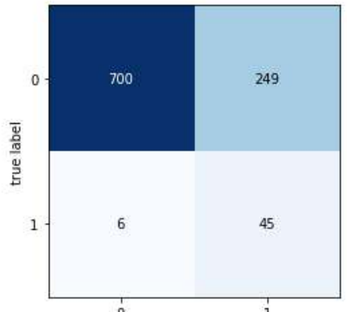
In [69]:
```
clf.score(train_x,train_y)
```
Out[69]:
```
0.746
```

Prediction and Accuracy

In [70]:
```
pred=clf.predict(test_x)
```

In [71]:
```
accuracy_score(pred,test_y)
```
Out[71]:
```
0.745
```

In [72]:
```
plot_confusion_matrix(confusion_matrix(pred,test_y))
```
Out[72]:
```
(<Figure size 432x288 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



Prediction and Accuracy

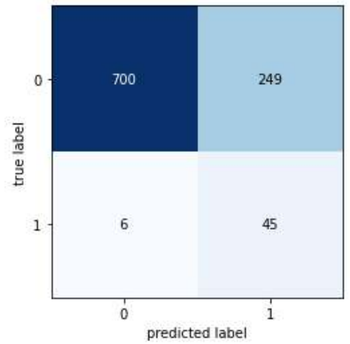
In [70]:
```
pred=clf.predict(test_x)
```

In [71]:
```
accuracy_score(pred,test_y)
```

Out[71]: 0.745

In [72]:
```
plot_confusion_matrix(confusion_matrix(pred,test_y))
```

Out[72]:
```
(<Figure size 432x288 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



In [73]:
```
print(classification_report(pred,test_y))
```

```
              precision    recall  f1-score   support

           0       0.99      0.74      0.85       949
           1       0.15      0.88      0.26        51

    accuracy                           0.74      1000
   macro avg       0.57      0.81      0.55      1000
weighted avg       0.95      0.74      0.82      1000
```

Sumon Singh
3.5.22

Write a program to implement Support Vector Machine (LSVM/Kernal/Soft Margin SVM)

1) Data Description :-

A) Name :- UniversalBank.csv

B) Description :- This is a data of a bank, of the people who has applied for a creditcard and based on the situation ~~whether~~ whether they have got the creditcard or not. This dataset is available in kaggle website.

C) Attributes :- The dataset has 12 attributes -

a) Age :- numeric data
b) Experience :- numeric data
c) Income :- numeric data
d) Zip code :- numeric data
e) Family :- numeric data
f) CCAvg :- Continous data
g) Education :- Catagorical data
h) Mortgage :- catagorical data
i) Personal Loan :- catagorical data
j) Securities Account :- Catagorical data
k) CID Account :- Catagorical data
l) Online :- Catagorical data

D) Label :- The target data is CreditCard which is a catagorical binary data. 0 indicates the client didn't get credit card and 1 indicates client has got a credit card.

Sumon Singh
3.5.22

E) Size: The dataset consist of 5000 rows and 13 columns.

2) Model Evaluation:

A) Name :- Support Vector classifier

B) Type :- This is a classifier type supervised model which classify the datapoints based on hyperplane.

c) Algorithm :-

Input: D=[X,Y], X is array of features, Y is label

function train_svm(X, Y, ephocs)

Step 1. Initalize learning_rate

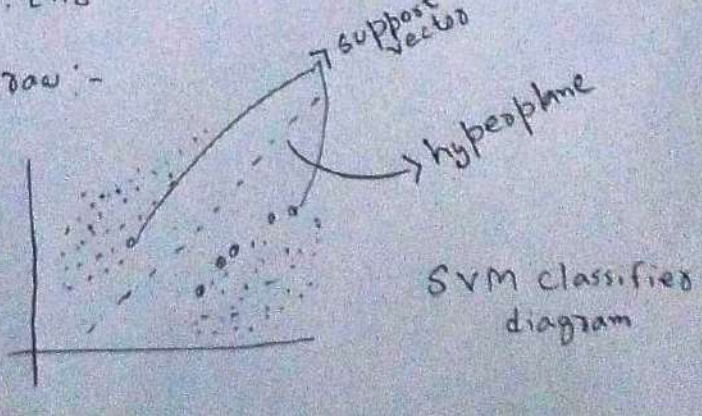
step 2. for learning_rate in ephocs
error=0

step 3. for i in X
if(Y[i]*X[i]*w)<1 then
update w = w+learning_rate*((X[i]*Y[i])*
(-2*(1/ephocs)*w)

else
update w = w+learning_rate*(-2*(1/ephocs)*w)
endif
end

Ste 4. End

D) Draw :-



SVM classifier diagram

Suman Singh
3 8 22

3) Result Analysis :-

In this experiment I have used support vector classifier with kernel rbf and soft margin 10 as these shows the best result after cross-validation.

The accuracy of the model for this dataset is 74.5% which is moderate but if we deep dive and analize that the model is capable for identifing '0' output very well as the precision is 0.99 and recall is 0.74 but for output '1' the model fails badly the precision is only 0.15 and recall is 0.88.

Finally by looking into f-score for output '0' which is 0.85 and for output '1' which is 0.26 we can conclude the model fails for classifing '1' but perform well for classifing '0's.

Implement KNN Algorithm

Load Libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from sklearn.preprocessing import StandardScaler,MinMaxScaler,OneHotEncoder,LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.utils import shuffle
from mlxtend.plotting import plot_confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('./Social_Network_Ads.csv')
```

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

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [4]:
```
df.shape
```

Out[4]:
```
(400, 5)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Out[4]: (400, 5)

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [7]:
```
encoder=LabelEncoder()
df['Gender']=encoder.fit_transform(df['Gender'])
```

In [8]:
```
df.drop('User ID',axis=1,inplace=True)
```

In [9]:
```
col=df.columns
col
```

Out[9]: Index(['Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')

In [10]:
```
X=df.drop('Purchased',axis=1)
Y=df['Purchased']
```

In [11]:
```
minmax=MinMaxScaler()
data=minmax.fit_transform(X)
```

In [12]:
```
scale=StandardScaler()
data=scale.fit_transform(data)
```

In [13]:
```
df = pd.DataFrame(data)
df = pd.concat([df,Y],axis=1)
```

Activities Firefox Web Browser Sun May 8 14:44 78%

File Edit View History Bookmarks Tools Help

KNN_Classifier - Jupyter

localhost:8895/notebooks/KNN_Classifier.ipynb

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [13]:
```
df = pd.DataFrame(data)
df = pd.concat([df,Y],axis=1)
df.columns=col
```

In [14]:
```
df.tail()
```

Out[14]:

	Gender	Age	EstimatedSalary	Purchased
395	-0.980196	0.797057	-0.844019	1
396	1.020204	1.274623	-1.372587	1
397	-0.980196	1.179110	-1.460681	1
398	1.020204	-0.158074	-1.078938	0
399	-0.980196	1.083596	-0.990844	1

In [15]:
```
df.describe()
```

Out[15]:

	Gender	Age	EstimatedSalary	Purchased
count	4.000000e+02	4.000000e+02	4.000000e+02	400.000000
mean	8.881784e-18	-2.842171e-16	5.329071e-17	0.357500
std	1.001252e+00	1.001252e+00	1.001252e+00	0.479864
min	-9.801961e-01	-1.877311e+00	-1.607506e+00	0.000000
25%	-9.801961e-01	-7.550313e-01	-7.852897e-01	0.000000
50%	-9.801961e-01	-6.256110e-02	7.561451e-03	0.000000
75%	1.020204e+00	7.970571e-01	5.361289e-01	1.000000
max	1.020204e+00	2.134241e+00	2.356750e+00	1.000000

In [16]:
```
df = shuffle(df,random_state=56).reset_index(drop=True)
```

In [17]:
```
X=df.drop('Purchased',axis=1)
Y=df['Purchased']
```

In [18]:
```
train_x,test_x,train_y,test_y=train_test_split(X,Y,
                                               test_size=0.2,random_state=100)
```

In [19]:
```
clf=KNeighborsClassifier(n_neighbors=5)
clf.fit(train_x,train_y)
```

Out[19]:
```
KNeighborsClassifier()
```

In [20]:
```
p=clf.predict(test_x)
```

Activities Firefox Web Browser Sun May 8 14:44 78 %

File Edit View History Bookmarks Tools Help

KNN_Classifier - Jupyter

localhost:8895/notebooks/KNN_Classifier.ipynb

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Out[19]:
```
KNeighborsClassifier()
```

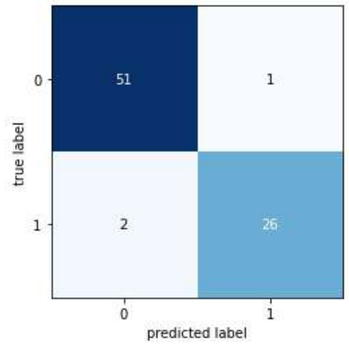
In [20]:
```
p=clf.predict(test_x)
```

In [21]:
```
accuracy_score(test_y,p)
```

Out[21]:
```
0.9625
```

In [22]:
```
plot_confusion_matrix(confusion_matrix(test_y,p))
```

Out[22]:
```
(<Figure size 432x288 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



In [23]:
```
print(classification_report(test_y,p))
```

```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97        52
           1       0.96      0.93      0.95        28

    accuracy                           0.96        80
   macro avg       0.96      0.95      0.96        80
weighted avg       0.96      0.96      0.96        80
```

Sumon Singh
07.05.22

Aim: Write a program to implement KNN Algorithm

1) Dataset Description:

A) Name : Social Network Ads

B) Description: This is a catagorical dataset to determine whether a user purchased a particular product, based on the user's gender, age and salary. The dataset is available on Kaggle website.

c) Size : The dataset consists of 400 rows and 5 columns.

D) Attributes : The dataset has 4 attributes

1) User ID : Unique numbers of the User
2) Gender : Gender of the user
3) Age : Age of the user
4) EstimatedSalary : Salary of User

E) Label : The target is 'Purchased' which is 0 or 1, if 0 then the user didn't bought the product else bought it.

Sumon Singh
07.05.22

Sumon Singh

2) ML Model Description:

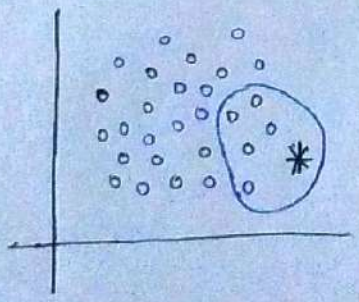
A) Name of the model :- K-Nearest Neighbor (KNN) classifier

B) Type of model :- It's a classifier model used in supervised learning

C) Algorithm :-

1) Select the number K of the neighbors.
2) Calculate the Euclidean distance of K ~~no~~ number of neighbors.
3) Take the K nearest neighbors as per the calculated Euclidean distance.
4) Among these K neighbors, count the number of data points in each catagory.
5) Assign the new data points to that catagory for which the number of the neighbor is maximum.

D) Draw :-



o → Catagory black
o → catagory blue
* → New data point

As the 3 black catagory is near to new data point and 2 blue catagory is near to new data point so the new data point will be of black catagory. (If K neighbors is 5

Sumon Singh
07.05.22

3) Result Analysis: The outcomes are 0 and 1.

A) Accuracy: Our model identifies 0.9625 data correctly which means our model is pretty good.

B) Precission:

For 0 the model's precision is 0.96
For 1 the model's precision is 0.96

c) Recall:

For 0 the recall is 0.98 and for 1 the recall is 0.93

* The model identifies 98% '0' Labled data correctly among all '0' labled data, similarly It identifies 93% '1' Labeled data correctly among all '1' Labled data.

So, our model works very good on the dataset.

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

Implement clustering on US_Arrest Dataset

Import Libraries

In [1]:
```
import numpy as np
from sklearn.cluster import AgglomerativeClustering,KMeans
from sklearn.model_selection import train_test_split
from scipy.cluster.hierarchy import dendrogram,linkage
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
```

Import Dataset

In [2]:
```
df = pd.read_csv('./US_Arrest/USArrests.csv')
```

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

Out[3]:

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
0	Alabama	13.2	236	58	21.2
1	Alaska	10.0	263	48	44.5
2	Arizona	8.1	294	80	31.0
3	Arkansas	8.8	190	50	19.5
4	California	9.0	276	91	40.6

Data Exploration

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   50 non-null     object
 1   Murder       50 non-null     float64
```

Data Exploration

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  50 non-null     object 
 1   Murder      50 non-null     float64
 2   Assault     50 non-null     int64  
 3   UrbanPop    50 non-null     int64  
 4   Rape        50 non-null     float64
dtypes: float64(2), int64(2), object(1)
memory usage: 2.1+ KB
```

In [5]:
```
df.shape
```

Out[5]:
```
(50, 5)
```

Data Cleaning

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Unnamed: 0    0
Murder        0
Assault       0
UrbanPop      0
Rape          0
dtype: int64
```

In [7]:
```
df=df.rename({'Unnamed: 0':'Location'},axis=1)
```

In [8]:
```
df.tail()
```

Out[8]:

	Location	Murder	Assault	UrbanPop	Rape
45	Virginia	8.5	156	63	20.7
46	Washington	4.0	145	73	26.2
47	West Virginia	5.7	81	39	9.3
48	Wisconsin	2.6	53	66	10.8
49	Wyoming	6.8	161	60	15.6

Activities Firefox Web Browser Sun May 8 14:45 77%

File Edit View History Bookmarks Tools Help

US_Arrest - Jupyter Note

localhost:8869/notebooks/US_Arrest.ipynb

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [9]:
```
df.drop('Location',axis=1,inplace=True)
```

In [10]:
```
df.describe().transpose()
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
Murder	50.0	7.788	4.355510	0.8	4.075	7.25	11.250	17.4
Assault	50.0	170.760	83.337661	45.0	109.000	159.00	249.000	337.0
UrbanPop	50.0	65.540	14.474763	32.0	54.500	66.00	77.750	91.0
Rape	50.0	21.232	9.366385	7.3	15.075	20.10	26.175	46.0

In [11]:
```
df.corr()
```

Out[11]:

	Murder	Assault	UrbanPop	Rape
Murder	1.000000	0.801873	0.069573	0.563579
Assault	0.801873	1.000000	0.258872	0.665241
UrbanPop	0.069573	0.258872	1.000000	0.411341
Rape	0.563579	0.665241	0.411341	1.000000

In [12]:
```
df.head()
```

Out[12]:

	Murder	Assault	UrbanPop	Rape
0	13.2	236	58	21.2
1	10.0	263	48	44.5
2	8.1	294	80	31.0
3	8.8	190	50	19.5
4	9.0	276	91	40.6

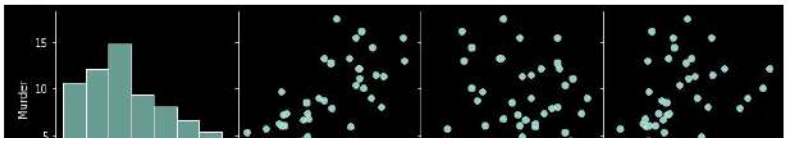
Data Visualization

In [13]:
```
plt.style.use('dark_background')
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True,cmap='tab10')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold')
```

Out[13]:
```
(array([0.5, 1.5, 2.5, 3.5]),
```

In [14]:
```
plt.style.use('dark_background')
plt.figure(figsize=(15,15))
sns.pairplot(df)
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x7fcda51eda30>

<Figure size 1080x1080 with 0 Axes>
```



Activities Firefox Web Browser Sun May 8 14:46 77 %

File Edit View History Bookmarks Tools Help

US_Arrest - Jupyter Note ×

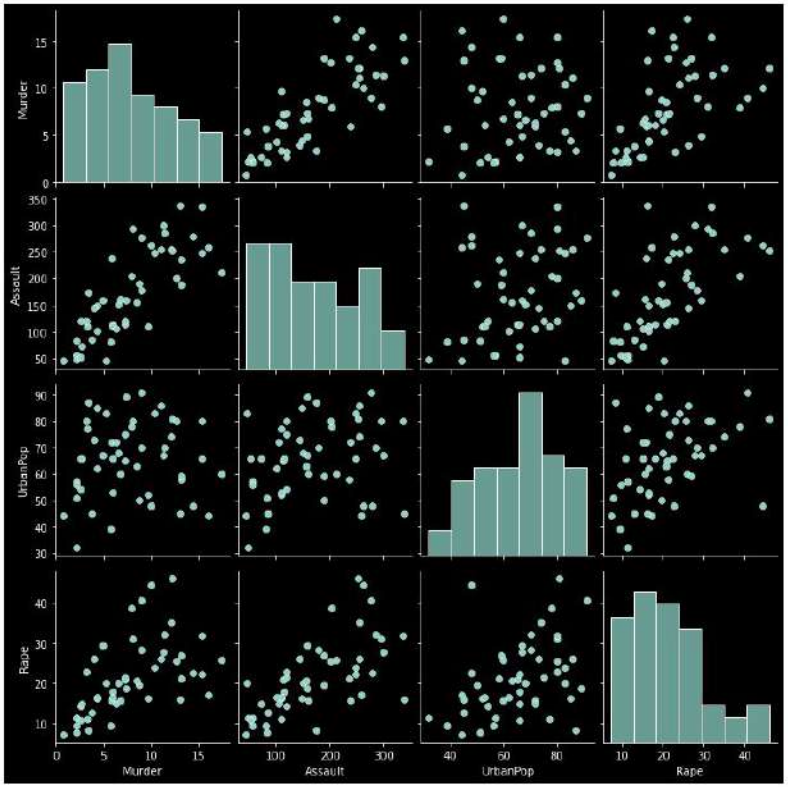
localhost:8869/notebooks/US_Arrest.ipynb

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

Out[14]: <seaborn.axisgrid.PairGrid at 0x7fcda51eda30>

<Figure size 1080x1080 with 0 Axes>



In [15]:
```
plt.style.use('dark_background')
fig=plt.figure(figsize=(15,15))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Murder'], df['UrbanPop'], df['Rape'], c='red', s=150)
```

Activities Firefox Web Browser Sun May 8 14:46 77 %

File Edit View History Bookmarks Tools Help

US_Arrest - Jupyter Note

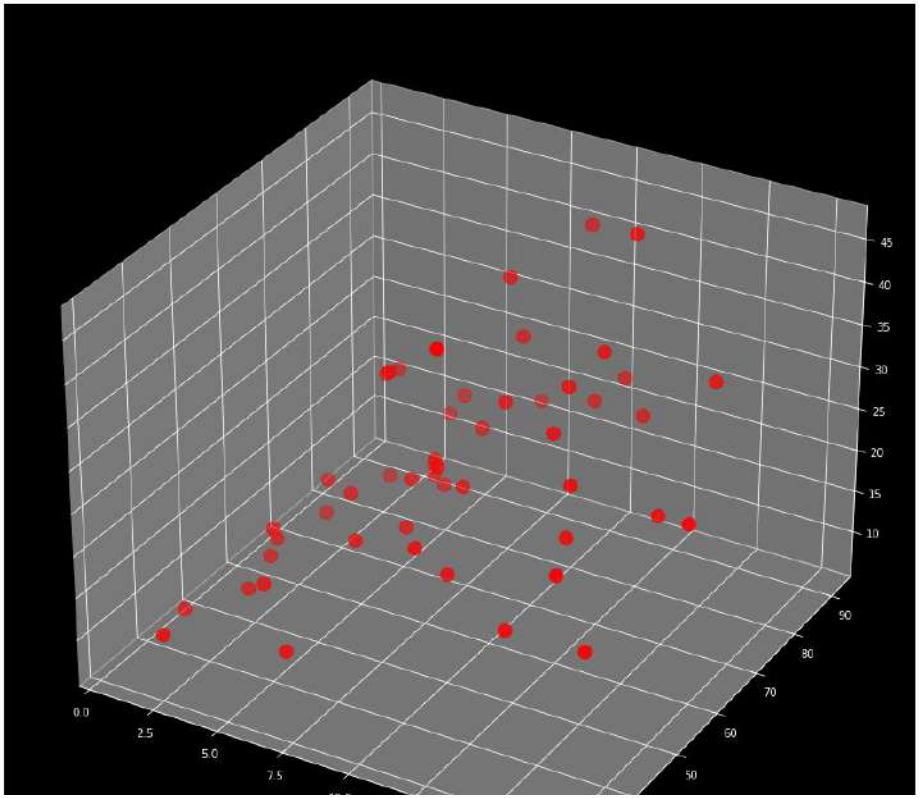
localhost:8869/notebooks/US_Arrest.ipynb

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

Murder Assault UrbanPop Rape

In [15]:
```
plt.style.use('dark_background')
fig=plt.figure(figsize=(15,15))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Murder'], df['UrbanPop'], df['Rape'], c='red', s=150)
plt.show()
```



Activities Firefox Web Browser Sun May 8 14:46 77 %

File Edit View History Bookmarks Tools Help

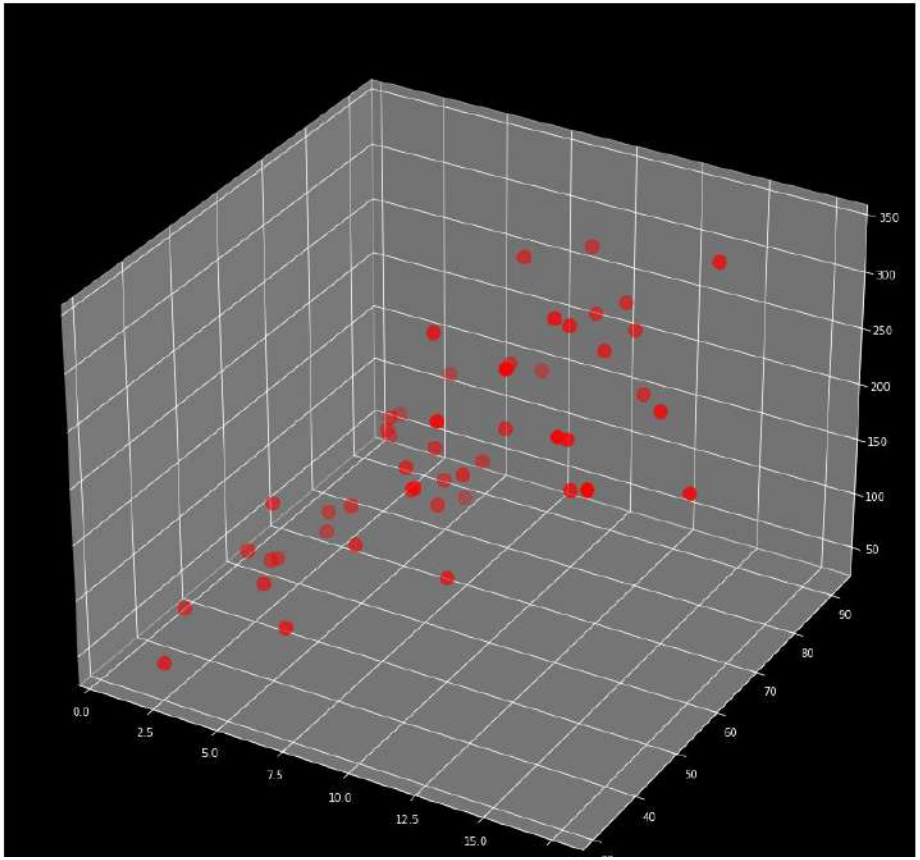
US_Arrest - Jupyter Note

localhost:8869/notebooks/US_Arrest.ipynb

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [16]:
```
plt.style.use('dark_background')
fig=plt.figure(figsize=(15,15))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Murder'], df['UrbanPop'], df['Assault'], c='red', s=150)
plt.show()
```



Activities Firefox Web Browser Sun May 8 14:46 77 %

File Edit View History Bookmarks Tools Help

US_Arrest - Jupyter Note

localhost:8869/notebooks/US_Arrest.ipynb

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

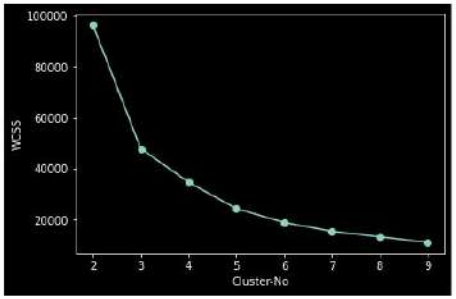
K-Means Model

Elbow Plot For Selecting K

In [17]:
```
K_val=range(2,10)
WCSS=[]
for i in K_val:
    k_model = KMeans(init='k-means++',n_clusters=i)
    cluster=k_model.fit(df)
    WCSS.append(cluster.inertia_)
```

In [18]:
```
plt.plot(K_val,WCSS,marker='o')
plt.xlabel('Cluster-No')
plt.ylabel('WCSS')
```

Out[18]:
```
Text(0, 0.5, 'WCSS')
```



KMeans Model

In [19]:
```
cluster_model_2=KMeans(init='k-means++',n_clusters=4)
cluster_Kmeans=cluster_model_2.fit_predict(df)
```

In [20]:
```
cluster_model_2.labels_
```

Out[20]:
```
array([1, 1, 1, 2, 1, 2, 3, 1, 1, 2, 0, 3, 1, 3, 0, 3, 3, 1, 0, 1, 2, 1,
       0, 1, 2, 3, 3, 1, 0, 2, 1, 1, 1, 0, 3, 2, 2, 3, 2, 1, 0, 2, 2, 3,
       0, 2, 2, 0, 0, 2], dtype=int32)
```

Activities Firefox Web Browser Sun May 8 14:50 75 %

File Edit View History Bookmarks Tools Help

US_Arrest - Jupyter Note

localhost:8869/notebooks/US_Arrest.ipynb

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown

silhouette score for kmeans

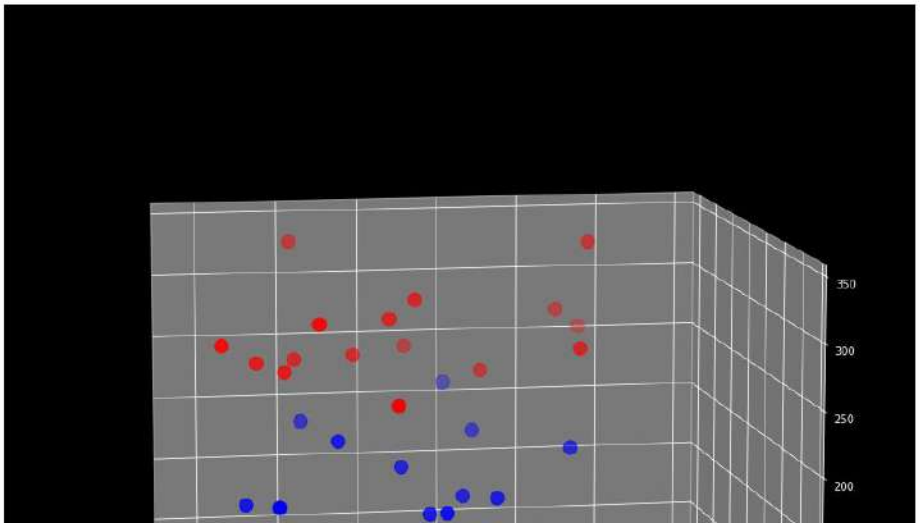
In [21]:
```
silhouette_score(df,cluster_Kmeans)
```

Out[21]: 0.5612332370891496

In [24]:
```
df['Kmeans']=cluster_Kmeans
```

Kmeans Clustering ¶

In [25]:
```
plt.style.use('dark_background')
fig=plt.figure(figsize=(15,15))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Murder'][df['Kmeans']==0], df['UrbanPop'][df['Kmeans']==0], df['Assault'][df['Kmeans']==0], s=150, c='
ax.scatter(df['Murder'][df['Kmeans']==1], df['UrbanPop'][df['Kmeans']==1], df['Assault'][df['Kmeans']==1], s=150 ,c='
ax.scatter(df['Murder'][df['Kmeans']==2], df['UrbanPop'][df['Kmeans']==2], df['Assault'][df['Kmeans']==2], s=150 ,c='
ax.scatter(df['Murder'][df['Kmeans']==3], df['UrbanPop'][df['Kmeans']==3], df['Assault'][df['Kmeans']==3], s=150 ,c='
ax.view_init(10, 170)
```



Activities Firefox Web Browser Sun May 8 14:50 75 %

File Edit View History Bookmarks Tools Help

US_Arrest - Jupyter Note ×

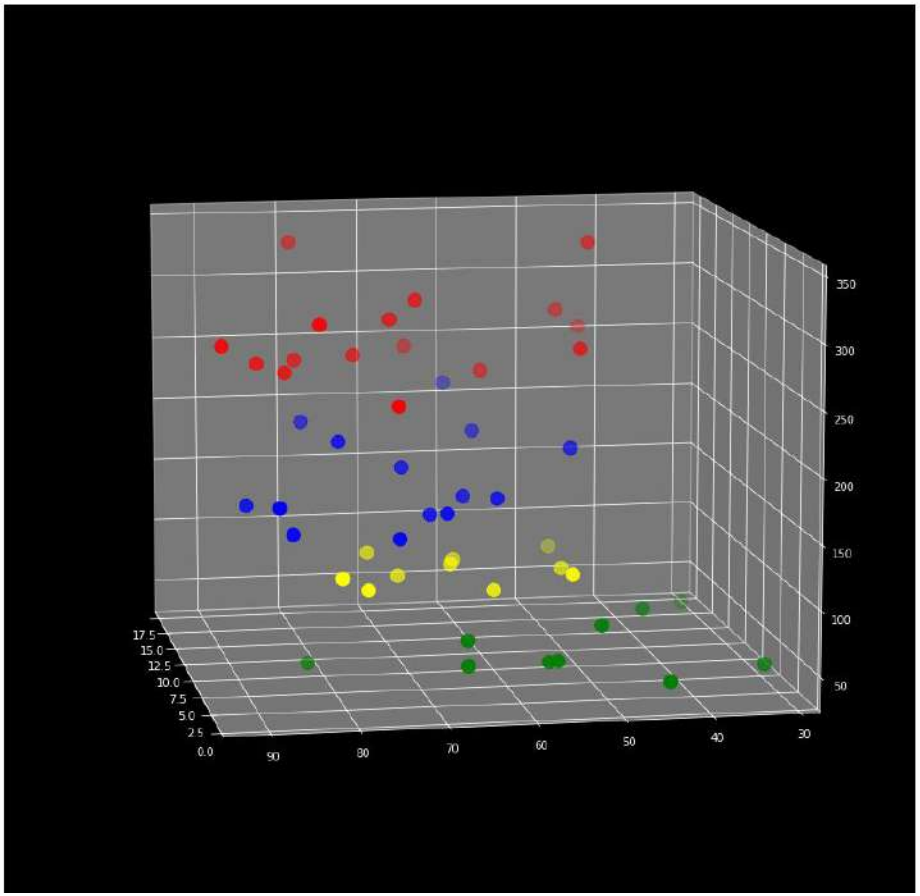
localhost:8869/notebooks/US_Arrest.ipynb

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown

Screenshot

```
ax.scatter(df['Murder'][df['Kmeans']==3], df['UrbanPop'][df['Kmeans']==3], df['Assault'][df['Kmeans']==3], s=150 ,c='
ax.view_init(10, 170)
```



Sumon Singh
07.05.22

AIM:-
Write a program to implement K-Means Algorithm

1) Dataset Description :-

A) Name : USArrests.csv

B) Description : This is a data of various types of arrests based on crime, occuared in several states of USA. such as the first data represents there was on average 13.2 arrests due to murder, 236 due to assult, 58 due to Urbanpop and 21.2 due to rape in the state of Alabama. The data is availabe on Kaggle website.

c) Size : The dataset has 50 rows and 5 columns

D) Attributes : The dataset has 5 attributes

1) Location : Name of the states where the arrests occuared

2) Murder : Average arrests for murder in a certain state

3) Assault : Average arrests due to assaults in a certain state.

4) UrbanPop : Average arrests due to urbanpop in a certain state.

5) Rape : Average arrests due to rape in a certain state.

All data except Location is continious datatype and Location is string.

E) Label : As this is a unsupervised Learning so datas are unlabelled and we need to make clusters from these data.

Sumon Singh
07.05.22

2) Model Evaluation :-

A) Name: K-means clustering

B) Type: This is a unsupervised learning model

c) Algorithm:

Input: A unlabelled dataset D with columns x

step 1: Select the number of K to decide the number of clusters. Usually elbow method is used for selecting optimal value of K.

step 2: Select random K points or centroids. It can be taken from the dataset also.

step 3: Assign each data point to their closest centroid, which will be ~~formed~~ form the predefined K clusters

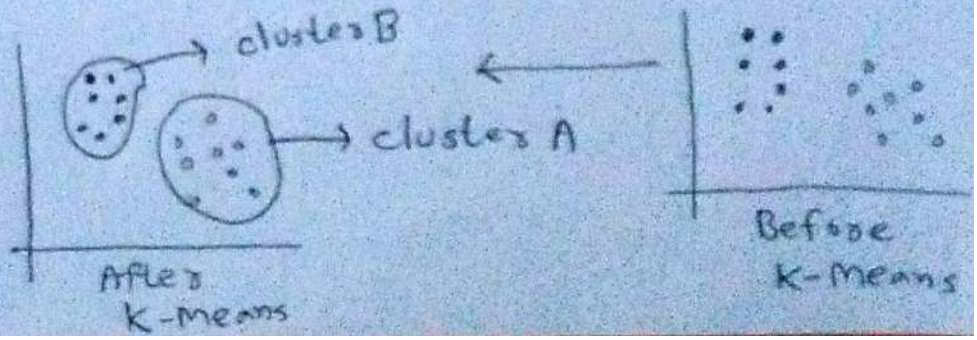
step 4: Calculate variance and place a new centroid of each cluster.

step 5: Repeat step 3 ~~and 4~~

step 6: If there is no reassignment occurs go to finish else go to step 4

step 7: END

D) Draw :-



Sumon Singh
07.05.22

Sumon Singh

3) Result Analysis:

After K-means algorithm implementation the datapoints are clustered into 4 groups and we get a silhouette score of 0.50 which is not that good or not that bad. The clustered labels are 0,1,2,3.

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Implement Logistic Regression

Load Libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from sklearn.preprocessing import StandardScaler,MinMaxScaler,OneHotEncoder,LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.utils import shuffle
from mlxtend.plotting import plot_confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Dataset

In [2]:
```
df = pd.read_csv('./Social_Network_Ads.csv')
```

Dataset Exploration

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

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [4]:
```
df.shape
```

Out[4]:
```
(400, 5)
```

Checking missing data and types

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

Encoding

In [7]:
```
encoder=LabelEncoder()
df['Gender']=encoder.fit_transform(df['Gender'])
```

In [8]:
```
df.drop('User ID',axis=1,inplace=True)
```

In [9]:
```
col=df.columns
col
```

Out[9]:
```
Index(['Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

Data Transformation

In [10]:
```
X=df.drop('Purchased',axis=1)
Y=df['Purchased']
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [11]:
```
minmax=MinMaxScaler()
data=minmax.fit_transform(X)
```

In [12]:
```
scale=StandardScaler()
data=scale.fit_transform(data)
```

In [13]:
```
df = pd.DataFrame(data)
df = pd.concat([df,Y],axis=1)
df.columns=col
```

In [14]:
```
df.tail()
```

Out[14]:

	Gender	Age	EstimatedSalary	Purchased
395	-0.980196	0.797057	-0.844019	1
396	1.020204	1.274623	-1.372587	1
397	-0.980196	1.179110	-1.460681	1
398	1.020204	-0.158074	-1.078938	0
399	-0.980196	1.083596	-0.990844	1

In [15]:
```
df.describe()
```

Out[15]:

	Gender	Age	EstimatedSalary	Purchased
count	4.000000e+02	4.000000e+02	4.000000e+02	400.000000
mean	8.881784e-18	-2.842171e-16	5.329071e-17	0.357500
std	1.001252e+00	1.001252e+00	1.001252e+00	0.479864
min	-9.801961e-01	-1.877311e+00	-1.607506e+00	0.000000
25%	-9.801961e-01	-7.550313e-01	-7.852897e-01	0.000000
50%	-9.801961e-01	-6.256110e-02	7.561451e-03	0.000000
75%	1.020204e+00	7.970571e-01	5.361289e-01	1.000000
max	1.020204e+00	2.134241e+00	2.356750e+00	1.000000

In [16]:
```
df = shuffle(df,random_state=56).reset_index(drop=True)
```

Data Spliting

In [17]:
```
X=df.drop('Purchased',axis=1)
```

Data Spliting

In [17]:
```
X=df.drop('Purchased',axis=1)
Y=df['Purchased']
```

In [18]:
```
train_x,test_x,train_y,test_y=train_test_split(X,Y,
                                               test_size=0.2,random_state=100)
```

Model

In [19]:
```
log_reg=LogisticRegression(random_state=100)
log_reg.fit(train_x,train_y)
```

Out[19]:
```
LogisticRegression(random_state=100)
```

In [20]:
```
p=log_reg.predict(test_x)
```

Accuracy

In [21]:
```
accuracy_score(test_y,p)
```

Out[21]:
```
0.8875
```

In [22]:
```
print(classification_report(test_y,p))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.92        52
           1       0.88      0.79      0.83        28

    accuracy                           0.89        80
   macro avg       0.89      0.86      0.87        80
weighted avg       0.89      0.89      0.89        80
```

In [23]:
```
plot_confusion_matrix(confusion_matrix(test_y,p))
```

Out[23]:
```
(<Figure size 432x288 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```

Activities Firefox Web Browser Sun May 15 18:03 33 %

File Edit View History Bookmarks Tools Help

Logistic_Regression - Jup

localhost:8888/notebooks/Logistic_Regression.ipynb

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [20]:
```
p=log_reg.predict(test_x)
```

Accuracy

In [21]:
```
accuracy_score(test_y,p)
```

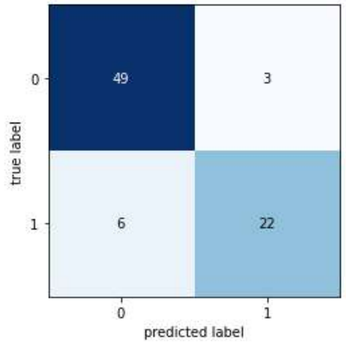
Out[21]:
```
0.8875
```

In [22]:
```
print(classification_report(test_y,p))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.92        52
           1       0.88      0.79      0.83        28

    accuracy                           0.89        80
   macro avg       0.89      0.86      0.87        80
weighted avg       0.89      0.89      0.89        80
```

In [23]:
```
plot_confusion_matrix(confusion_matrix(test_y,p))
```

Out[23]:
```
(<Figure size 432x288 with 1 Axes>,
 <AxesSubplot:xlabel='predicted label', ylabel='true label'>)
```



Sumon Singh
07.05.22

Aim: Write a program to implement Logistic Regression Algorithm

1) Dataset Description:

A) Name: Social Network Ads

B) Description: This is a catagorical dataset to determine whether a user purchased a particular product, based on the user's gender, age and salary. The dataset is available on Kaggle website.

c) Size: The dataset consists of 400 rows and 5 columns.

D) Attributes: The dataset has 4 attributes

1) User ID: Unique numbers of the user
2) Gender: Gender of the user
3) Age: Age of the user
4) EstimatedSalary: Salary of user

E) Label: The target is 'Purchased' which is 0 or 1, if 0 then the user didn't bought the product else bought it.

Sumon Singh
~~2+~~
07.09.22

2) ML Model Description:

A) Name of model :- Logistic Regression

B) Type of model :- It is a supervised learning model which is used for classification.

C) Algorithm :-

1) Using linear regression calculate output based on features.

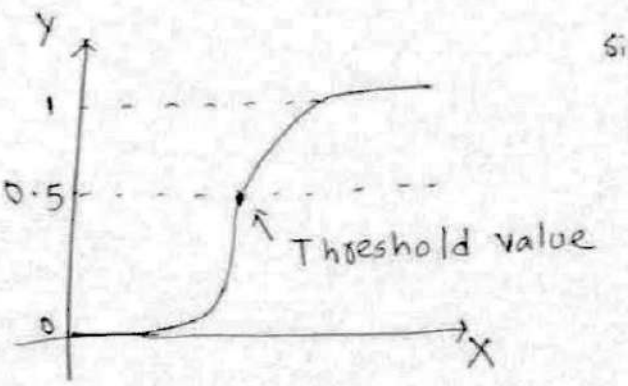
Example: Let us assume, we have a dataset consisting of n features $X_1, X_2 \cdots X_n$, of real values
∴ output will be -

$$y = b_0 + b_1X_1 + b_2X_2 \cdots + b_nX_n$$

2) Using Sigmoid function find the probability of the output of previous step
Sigmoid function : $\frac{1}{1+e^{-y}}$ (y is previous output)

3) If the probablity output is more than 0.5 then the actual classification will be of type 1 else type 0.

D) Draw :-



Sigmoid function
$\frac{1}{1+e^{-y}}$

Sumon Singh
15.05.22

3) Result Analysis : The outcomes are 0 and 1

A) Accuracy : Our model identifies 0.8875 data correctly which means our model is pretty good.

B) Precission :

For 0 the model's precission is 0.89

For 1 the model's precission is 0.88

C) Recall :

For 0 the recall is 0.94 and for 1 the recall is 0.79

D) f1-score :

For 0 the f1-score is 0.92 and for 1 f1-score is 0.83

* The model identifies 94% '0' labeled data correctly among all '0' labeled data, similarly it identifies 79% '1' Labeled data correctly among all '1' labled data

So, our model works fine on the dataset

Implement Gaussian Naive Bayes From Scratch And Using In-Built Model

Load Libraries

In [44]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import make_blobs
from scipy.stats import norm
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
```

Create Dataset

In [45]:
```
X,Y = make_blobs(n_samples=200,n_features=2,centers=2,random_state=100)
```

In [46]:
```
X[:5]
```

Out[46]:
```
array([[-0.71922009,  7.98733855],
       [-0.31298672,  6.43068008],
       [-2.09935677,  6.08209782],
       [-2.67288401,  6.87434687],
       [ 0.74189765, -2.44233865]])
```

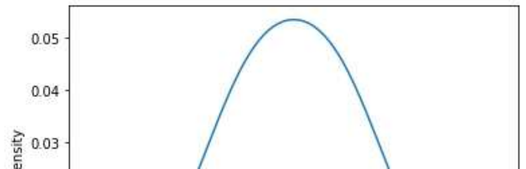
In [47]:
```
Y[:5]
```

Out[47]:
```
array([1, 1, 1, 1, 0])
```

Plot Dataset

In [49]:
```
sns.kdeplot(X[0])
```

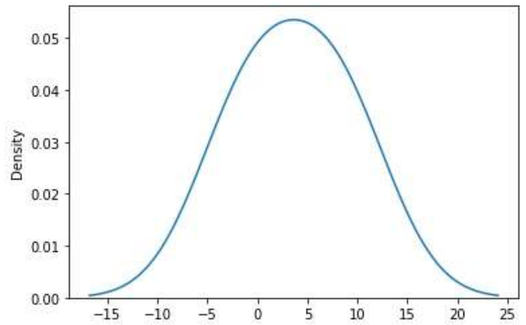
Out[49]:
```
<AxesSubplot:ylabel='Density'>
```



Plot Dataset

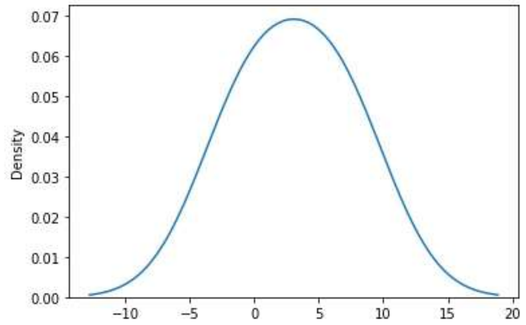
In [49]:
```
sns.kdeplot(X[0])
```

Out[49]: <AxesSubplot:ylabel='Density'>



In [50]:
```
sns.kdeplot(X[1])
```

Out[50]: <AxesSubplot:ylabel='Density'>



Distribution Function

In [51]:
```
def fit_distribution(data):
    mu=np.mean(data)
    st=np.std(data)
```

Distribution Function

In [51]:
```
def fit_distribution(data):
    mu=np.mean(data)
    st=np.std(data)
    print(mu,st)
    dist=norm(mu,st)
    return dist
```

In [52]:
```
def probablity(X,prior,dist1,dist2):
    return prior*dist1.pdf(X[0])*dist2.pdf(X[1])
```

Sort the data into classes

In [53]:
```
xy0=X[Y==0]
xy1=X[Y==1]
```

calculate priors

In [54]:
```
priorxy0=len(xy0)/len(X)
priorxy1=len(xy1)/len(X)
```

Create probablity density function

In [55]:
```
distx1y0=fit_distribution(xy0[:,0])
distx2y0=fit_distribution(xy0[:,1])
```

```
0.8663054747080094 1.0752990278988945
-4.601642581937076 0.9355002483962135
```

In [56]:
```
distx1y1=fit_distribution(xy1[:,0])
distx2y1=fit_distribution(xy1[:,1])
```

```
-1.449846326524923 1.0779386819454677
7.018875126023947 1.0731365954263092
```

Classify samples

We are predicting the 67th datapoint

Classify samples

We are predicting the 67th datapoint

Prediction From Scratch

In [57]:
```
xsample,ysample=X[67],Y[67]
```

In [58]:
```
y0=probablity(xsample,priorxy0,distx1y0,distx2y0)

y1=probablity(xsample,priorxy1,distx1y1,distx2y1)
```

In [59]:
```
if y1>y0:
    output=1
    print('output : ',output)
else:
    output=0
    print('output : ',output)
```

```
output :  1
```

In [60]:
```
if output==ysample:
    print('Correctly Classified')
else:
    print('Incorrectly classified')
```

```
Correctly Classified
```

Prediction Using In-built Model

Model

In [61]:
```
model = GaussianNB().fit(X,Y)
```

In [62]:
```
p=model.predict([xsample])
```

In [63]:
```
p
```

Out[63]:
```
array([1])
```

```
y1=probablity(xsample,priorxy1,distx1y1,distx2y1)
```

In [59]:
```
if y1>y0:
    output=1
    print('output : ',output)
else:
    output=0
    print('output : ',output)
```

```
output :  1
```

In [60]:
```
if output==ysample:
    print('Correctly Classified')
else:
    print('Incorrectly classified')
```

```
Correctly Classified
```

Prediction Using In-built Model

Model

In [61]:
```
model = GaussianNB().fit(X,Y)
```

In [62]:
```
p=model.predict([xsample])
```

In [63]:
```
p
```

Out[63]:
```
array([1])
```

In [64]:
```
if p==ysample:
    print('Correctly Classified')
else:
    print('Incorrectly classified')
```

```
Correctly Classified
```

Suman Singh
15.5.22

Suman Singh

AIM: Write a program to implement Naive Bayes Model and Gaussian Naive Bayes Model from scratch

1) Data Description:

A) Using the make_blobs function of Sklearn a dummy dataset have been created

B) Size: The dataset consist of 100 rows and 3 columns

C) Type: The dataset is a classification type dataset where each feature is of continious.

D) Attributes: The dataset has 2 features created randomly. (attributes are continious data)

E) Target: There is one column which has two label 0 and 1, which are catagorical data

2) Model Evaluation:

A) Name of the model: Gaussian Naive Bayes model

B) Type of model: It's a probablistic model which is used for classification supervised type problems.

C) Working principle:

Input: A dataset $D(X_1, X_2 \ldots, X_n)$ which has X_1, X_2, X_n features and an output Y which has labels.

step1: For each type of label of Y find the prior probablity of each label, using the following equation -

Prior probablity of label C_i of Y target is:- $P(C_i) = \frac{\text{No of } C_i \text{ in } Y}{\text{total no. of } y}$

Sumon Singh
15.5.22

Step 2: For each features $X_1, X_2 \ldots, X_n$ find the mean and standard deviation and create a normal distribution.

Let's say, $\text{mean}(X_i)$ = mean of feature X_i

$\text{std}(X_i)$ = standard deviation of feature X_i

Normal distribution:-

$$f(x_i) = \frac{1}{\text{std}(x_i)\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i - \text{mean}(x_i)}{\text{std}(x_i)}\right)^2}$$

Step 3: Using bayes theorm calculate the probablity for a testing datapoint $t_1, t_2 \ldots t_n$ for each label of Y.

formula: for C_i Label of Y the probablity will be

Prior probablity of C_i * Poobablity density function t_i $(i = 1, 2, \ldots, n)$

Step 4. The probablity which is highest for the label of testing data is the output.

step 4: The label which has highest probablity for testing data will be the output.

step 5: End

Suman Singh
15.5.22

Suman Singh

3) Result Analysis :

Tested the in-built Gaussian Naive Bayes and self written Gaussian model with 67th no. datapoint of the trained dataset and for both the models we are getting similar result.