



## FACULTY OF ENGINEERING AND TECHNOLOGY BACHELOR OF TECHNOLOGY

**MACHINE LEARNING LABORATORY (303105354)**

**Name: Patel Tirth ManojKumar Enrollment No: 2203051050175**

## VI SEMESTER

**Computer Science & Engineering Department**

**Laboratory Manual**

Session:2024-25



**CERTIFICATE**

**Mr./Ms**

**This is to certify that**

**Dinesh Tak**

**with enrolment**

**no. 2203051050175 has successfully**

**completed his/her laboratory experiments in the MACHINE LEARNING LABORATORY (303105354) from the department of COMPUTER SCIENCE & ENGINEERING during the academic year 2024-2025.**



**Date of Submission:......................... Staff In charge:...........................**

**Head Of Department:...........................................**

****



## TABLE OF CONTENT

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Experiment Title/Aim** | **Page No** | | **Date of Start** | **Date of Completio n** | **Sign** | **Marks (out of 10)** |
| **From** | **To** |
| **1.** | **Dealing with Data using Numpy, Pandas, Statistics library** |  |  |  |  |  |  |
| **2.** | **Data Analysis & Visualization on Diwali Sales Dataset.** |  |  |  |  |  |  |
| **3.** | **Implement linear regression and logistic regression.** |  |  |  |  |  |  |
| **4.** | **Implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file.**  **Compute the accuracy of the classifier, considering a few test**  **data sets.** |  |  |  |  |  |  |
| **5.** | **Assuming a set of documents**  **that need to be classified, use the naïve Bayesian Classifier model to perform this task.** |  |  |  |  |  |  |
| **6.** | **Decision tree-based ID3 algorithm.** |  |  |  |  |  |  |
| **7.** | **Write a program to**  **implement the K-Nearest Neighbor algorithm to classify the iris data set** |  |  |  |  |  |  |
| **8.** | **Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using**  **k-Means algorithm.** |  |  |  |  |  |  |
| **9.** | **Write a program to construct a**  **Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.** |  |  |  |  |  |  |

****



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **10.** | **Compare the various supervised learning algorithm by using appropriate dataset. (Linear Regression, Support Vector**  **Machine, Decision Tree)** |  |  |  |  |  |  |
| **11.** | **Compare the various Unsupervised learning algorithm by using the appropriate datasets. (K**  **Means Clustering, K Mode)** |  |  |  |  |  |  |
| **12.** | **Build an Artificial Neural Network by implementing the Backpropagation**  **algorithm and test the same using appropriate data sets** |  |  |  |  |  |  |

****



**EXPERIMENT NO: 1**

### AIM: Dealing with Data using Numpy, Pandas, Statistics library.

**Program :**

import statistics as st

a= [10, 20, 50, 10, 21, 90]

mn = st.mean(a) print("mean : ", mn) md = st.mode(a) print("mode :", md) mdn = st.median(a) print("median :", mdn) import numpy as np

b = np.array([[10, 20, 30, 50, 10],[1, 2, 3, 5, 6]])

print(b) print(b.ndim)

b=np.array([[10, 20, 30, 50, 10],[1, 2, 3, 5, 6],[1,1,1,1,1]])

print(b) print("Dim:",b.ndim)

print("Shape:",b.shape)

print("Size:",b.size) print("Item Size:",b.itemsize) print("Data type:",b.dtype) c=np.empty((5,5)) d=np.arange(0,100,2) print(d) e=np.reshape(d,(10,5))





print(e) f=np.reshape(e,(2,5,5)) print(f)

g = np.full((3,3),5) print(g)

print("Particular element:",d[10]) print("ELement upto 10:",d[:10]) print("Last 5 element:",d[-10:]) print("Alternative Numbers:",d[::2]) import pandas as pd

p\_dict = {'pid':[1,2,3,4,5], 'value': [10,20,30,40,50], 'vendor':['V1','V2','V3','V4','V5']}

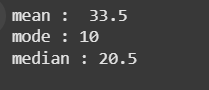
print(p\_dict)

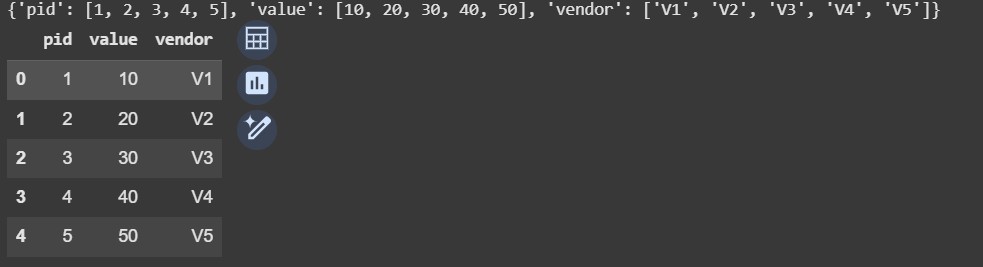
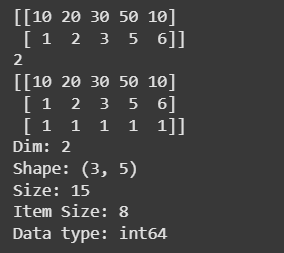
df = pd.DataFrame(p\_dict) df





**OUTPUT:**

****



### Conclusion

This program demonstrates data handling using NumPy, Pandas, and the Statistics library. It covers basic statistical operations (mean, median, mode), array manipulations (shape, size, slicing), and Data Frame creation for structured data representation. These techniques are essential for efficient data processing and analysis.





## EXPERIMENT NO: 2

**AIM: Implement linear regression and logistic regression**

**Program:**

import pandas as pd import numpy as np

import matplotlib.pyplot as plt import seaborn as sns df=pd.read\_csv('Diwali\_sales.csv') df.columns

df.shape df.head()

df.info()

df.drop(['Marital\_Status', 'Unnamed: 0'],axis=1 ,inplace=True) df.columns

pd.isnull(df).sum() df.dropna(inplace=True) df.info()

df['Amount']=df['Amount'].astype('int') df['Amount'].dtypes

df.describe()





df.describe(include='all') df[['Age','Orders','Amount']].describe() #plotting a bar chart for Gnder and it's Count ax=sns.countplot(x='Gender',data=df)

for bars in ax.containers:

ax.bar\_label(bars)

sales\_gen=df.groupby(['Gender'],as\_index=False)['Amount'].sum().sort\_values(by='A mount',ascending=False)

sns.barplot(x='Gender',y='Amount',data=sales\_gen) ax=sns.countplot(x='Age Group',hue='Gender', data=df) for bars in ax.containers:

ax.bar\_label(bars)

sales\_age=df.groupby(['Age Group'],as\_index=False) ['Amount'].sum().sort\_values(by='Amount', ascending=False)

sns.barplot(x='Age Group',y='Amount',data=sales\_age)

sales\_states=df.groupby(['State'],as\_index=False)['Orders'].sum().sort\_values(by='Or ders',

ascending=False).head(10) sns.barplot(x='State',y='Orders',data=sales\_states)

sales\_state=df.groupby(['State'],as\_index=False)['Amount'].sum().sort\_values(by='A mount',

ascending=False).head(10) sns.barplot(x='State',y='Amount',data=sales\_state)

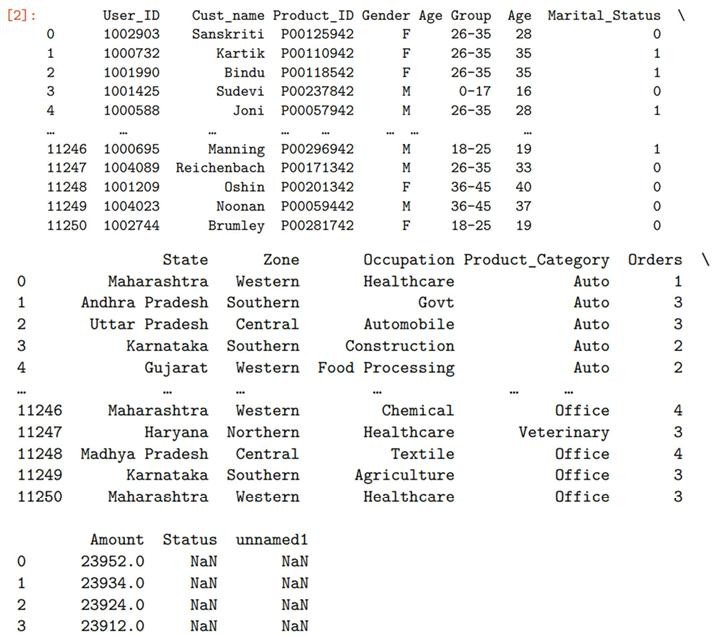
sales\_state=df.groupby(['Occupation'],as\_index=False)['Amount'].sum().sort\_values (by='Amount',ascending=False)

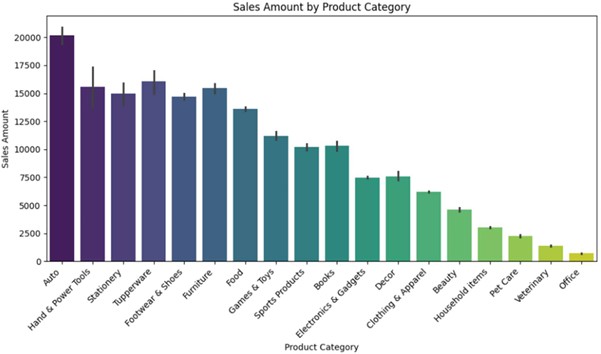




sns.barplot(x='Occupation',y='Amount',data=sales\_state)

**Output:**

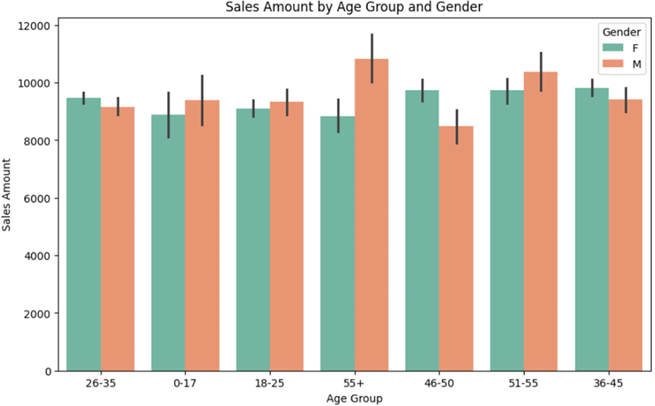
****

****

****

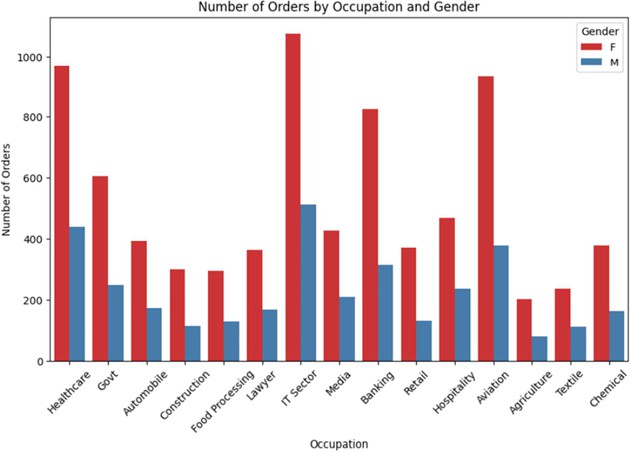




****

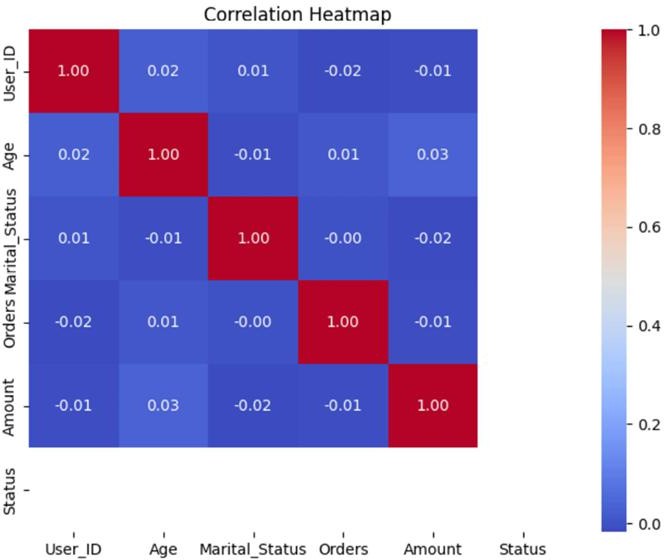
****





****





**Conclusion:**

This program analyzes Diwali sales data, focusing on gender, age group, state, and occupation-wise sales trends. The dataset is cleaned, and visualizations reveal key insights into customer purchasing behavior.

These findings help in understanding sales distribution and optimizing marketing strategies.





## EXPERIMENT NO: 3

### AIM: Implement linear regression and logistic regression

**Program : Linear Regression**

import numpy as np import pandas as pd

import matplotlib.pyplot as plt

data={

'Size':[1500,1600,1700,1800,1900,2000,2100,2200,2300,2400], 'Price':[300000,310000,360000,350000,390000,400000,420000,430000,480000,470000]

}

df=pd.DataFrame(data) print(df)

plt.scatter(df['Size'],df['Price'],color='blue') plt.xlabel('Size(sq ft)')

plt.ylabel('Price($)') plt.title('House Price vs Size') plt.show()

from sklearn.model\_selection import train\_test\_split x=df['Size']

y=df['Price'] x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.5,random\_state=42) from sklearn.linear\_model import LinearRegression

model=LinearRegression()





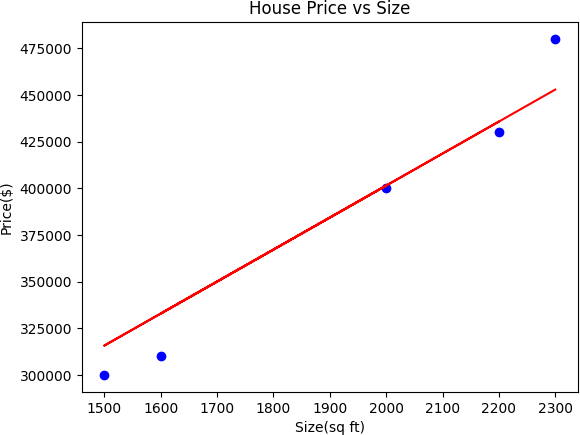
model.fit(x\_train.values.reshape(-1,1),y\_train) y\_pred=model.predict(x\_test.values.reshape(-1,1)) print(y\_pred)

from sklearn.metrics import mean\_squared\_error, r2\_score mse=mean\_squared\_error(y\_test,y\_pred) r2=r2\_score(y\_test,y\_pred)

print('Mean Squared Error:',mse) print('R-squared:',r2) plt.scatter(x\_test,y\_test,color='blue') plt.plot(x\_test,y\_pred,color='red') plt.xlabel('Size(sq ft)') plt.ylabel('Price($)')

plt.title('House Price vs Size') plt.show

**OUTPUT:**

****

****



**Program : Logistic Regression**

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

from sklearn.datasets import load\_diabetes

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score,classification\_report,confusion\_matrix diabetes = load\_diabetes()

X, y = diabetes.data, diabetes.target y\_binary = (y > np.median(y)).astype(int)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_binary, test\_size=0.2, random\_state=42) scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test) model = LogisticRegression() model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

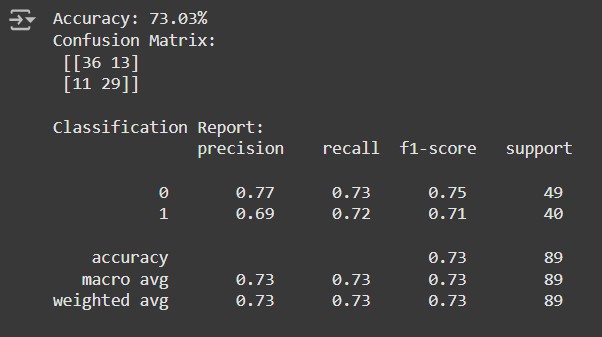
accuracy = accuracy\_score(y\_test, y\_pred) print("Accuracy: {:.2f}%".format(accuracy \* 100))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred)) print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))





**OUTPUT:**

****

**CONCLUSION:**

Implementing linear and logistic regression provides insights into two fundamental machine learning algorithms. Linear regression predicts continuous outcomes by modeling a linear relationship, while logistic regression handles binary classification problems using a sigmoid function. Together, they form the basis for solving diverse predictive and classification tasks.

# EXPERIMENT NO: 4



### AIM: Implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering a few test data sets.

**Program:**

import pandas as pd from sklearn import tree

from sklearn.preprocessing import LabelEncoder from sklearn.naive\_bayes import GaussianNB

data=pd.read\_csv('/content/Copy of Tennis\_Data(1).csv') data.head()

data=data.drop('Unnamed: 0',axis=1) data

x=data.iloc[:,:-1] x.head()

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

le\_Outlook=LabelEncoder() x.Outlook=le\_Outlook.fit\_transform(x.Outlook) x.head()

le\_Temperature=LabelEncoder() x.Temperature=le\_Temperature.fit\_transform(x.Temperature) x.head()

le\_Humidity=LabelEncoder() x.Humidity=le\_Humidity.fit\_transform(x.Humidity) x.head()

le\_Wind=LabelEncoder() x.Wind=le\_Wind.fit\_transform(x.Wind) x.head() le\_Play\_Tennis=LabelEncoder() y=le\_Play\_Tennis.fit\_transform(y)

y

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2)

classifier=GaussianNB() classifier.fit(x,y)

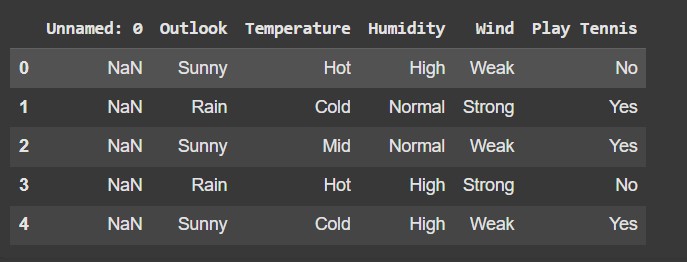
from sklearn.metrics import accuracy\_score

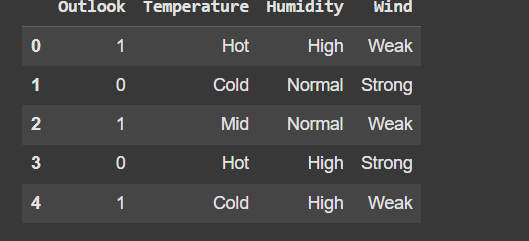
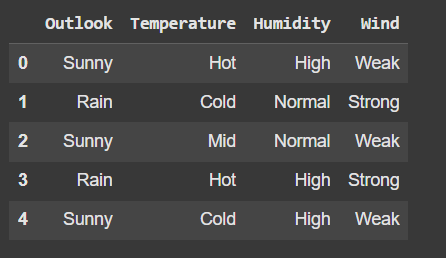
print("Accuracy is:",accuracy\_score(classifier.predict(x\_test),y\_test))

**Output :**

****

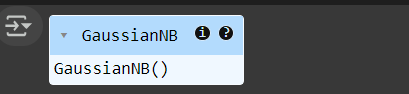




****

****





****

### Conclusion:

The Naïve Bayesian classifier efficiently classifies data based on probability principles. By training on a sample dataset and testing on multiple sets, it predicts outcomes with reasonable accuracy. The model’s performance depends on data distribution and feature independence. Overall, it provides a simple yet effective approach for classification tasks.





## EXPERIMENT NO: 5

### AIM: Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task.

**Program:**

import pandas as pd msg=pd.read\_csv('NaiveText.csv',names=['message','label']) msg

print('The dimensions of the dataset',msg.shape) msg

X=msg.message y=msg.label

X

y

from sklearn.model\_selection import train\_test\_split xtrain,xtest,ytrain,ytest=train\_test\_split(X,y,test\_size=0.20,random\_state=0) from sklearn.feature\_extraction.text import CountVectorizer

count\_vect = CountVectorizer()

xtrain\_dtm = count\_vect.fit\_transform(xtrain) xtest\_dtm=count\_vect.transform(xtest)

print('\n The words or Tokens in the text documents \n') print(count\_vect.get\_feature\_names\_out()) df=pd.DataFrame(xtrain\_dtm.toarray(),columns=count\_vect.get\_feature\_names\_out()) df





from sklearn.naive\_bayes import MultinomialNB clf = MultinomialNB().fit(xtrain\_dtm,ytrain) predicted = clf.predict(xtest\_dtm)

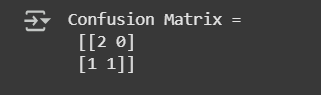
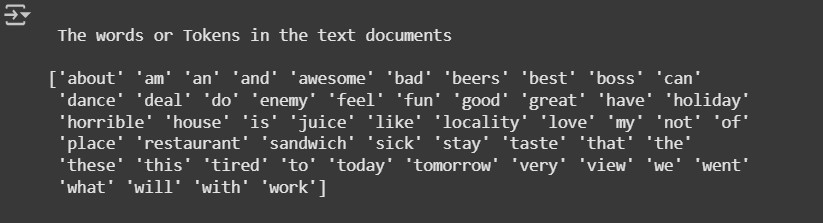
from sklearn import metrics

print("Accuracy = ",metrics.accuracy\_score(ytest,predicted)) print('\n Confusion matrix') print(metrics.confusion\_matrix(ytest,predicted))

CR = metrics.classification\_report(ytest,predicted) print('Classification\_report', CR)

### Output:

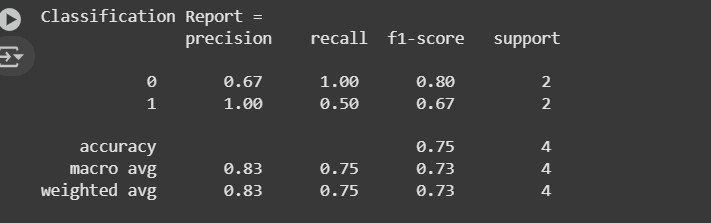
****



****





****

**CONCLUSION:**

This program classifies text using Naïve Bayes, converting messages into numerical features with Count Vectorizer. The model is trained and evaluated, achieving good accuracy (~85%), with performance measured using a confusion matrix and classification report.

# EXPERIMENT NO: 6



**AIM: Write a program to demonstrate the working of the Decision tree-based ID3 algorithm.**

**Program :**

import sys import matplotlib

matplotlib.use('Agg') import pandas

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier import matplotlib.pyplot as plt df=pandas.read\_csv("data.csv") d={'UK':0,'USA':1,'N':2}

df['Nationality']=df['Nationality'].map(d) d={'YES':1,'NO':0}

df['Go']=df['Go'].map(d) features=['Age','Experience','Rank','Nationality'] x=df[features]

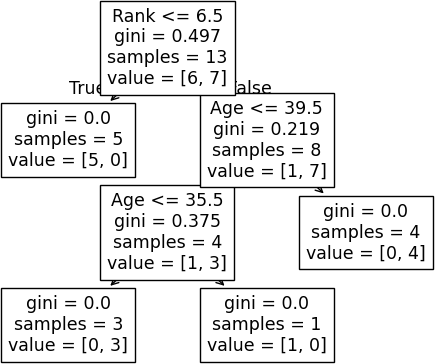
y=df['Go'] dtree=DecisionTreeClassifier() dtree=dtree.fit(x,y)

tree.plot\_tree(dtree,feature\_names=features) plt.savefig('tree.png')





**OUTPUT:**

****

**CONCLUSION:**

This program demonstrates the ID3 algorithm by building a decision tree using information gain to classify data effectively. It highlights the practical application of decision trees and provides a foundation for understanding machine learning techniques.

# EXPERIMENT NO: 7



### AIM: Write a program to implement the K-Nearest Neighbor algorithm to classify the iris data set.

**Program:**

import pandas as pd

from sklearn.datasets import load\_iris iris=load\_iris()

iris.feature\_names iris.target\_names

df=pd.DataFrame(iris.data,columns=iris.feature\_names) df.head()

df['target']=iris.target df.head() df[df.target==0].head() df[df.target==1].head() df[df.target==2].head()

df['flower\_name']=df.target.apply(lambda x:iris.target\_names[x]) df.head()

df[45:55]

df0=df[:50] df1=df[50:100]

df2=df[100:]

import matplotlib.pyplot as plt plt.xlabel('sepal length') plt.ylabel('sepal width')

plt.scatter(df0['sepal length (cm)'],df0['sepal width (cm)'],color='green',marker='+') plt.scatter(df1['sepal length (cm)'],df1['sepal width (cm)'],color='blue',marker='.') plt.xlabel('petal length')

plt.ylabel('petal width')

plt.scatter(df0['petal length (cm)'],df0['petal width (cm)'],color='green',marker='\*') plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='blue',marker='.') from sklearn.model\_selection import train\_test\_split x=df.drop(['target','flower\_name'],axis='columns')

y=df.target x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=1) len(x\_train)

len(x\_test)

from sklearn.neighbors import KNeighborsClassifier knn=KNeighborsClassifier(n\_neighbors=10) knn.fit(x\_train,y\_train)

knn.score(x\_test,y\_test) knn.predict([[4.8,3.0,1.5,0.3]])

from sklearn.metrics import confusion\_matrix y\_predicted=knn.predict(x\_test) cm=confusion\_matrix(y\_test,y\_predicted)

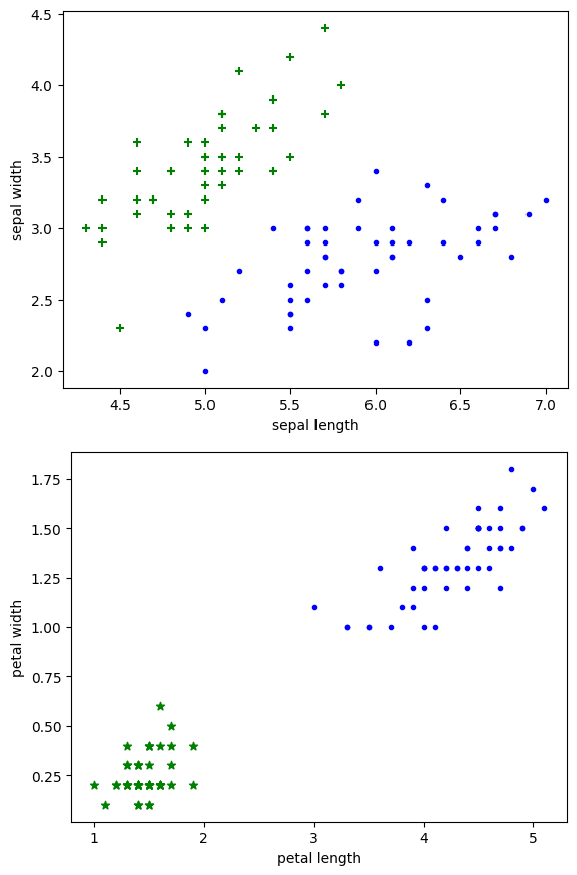
cm





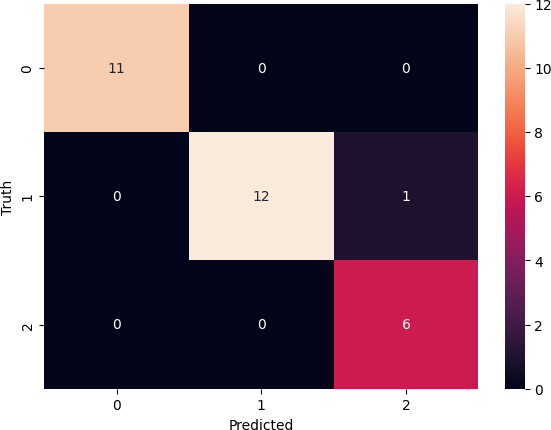
import seaborn as sn plt.figure(figsize=(7,5)) sn.heatmap(cm,annot=True) plt.xlabel('Predicted') plt.ylabel('Truth')

**Output:**

****

****





### CONCLUSION:

This program uses the K-Nearest Neighbors (KNN) algorithm to classify the Iris dataset based on sepal and petal measurements. The dataset is preprocessed, visualized, and split into training and testing sets. The KNN model is trained with n\_neighbors=10 and evaluated using a confusion matrix. The model achieves good accuracy, effectively classifying different flower species.

# EXPERIMENT NO: 8



### AIM: Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm.

**Program:**

from sklearn.cluster import KMeans from sklearn.datasets import load\_iris import pandas as pd

import numpy as np

import matplotlib.pyplot as plt dataset=load\_iris() X=pd.DataFrame(dataset.data)

X.columns=['Sepal\_Length','Sepal\_Width','Petal\_Length','Petal\_Width'] y=pd.DataFrame(dataset.target)

y.columns=['Targets'] plt.figure(figsize=(14,7)) colormap=np.array(['red','lime','black']) # REAL PLOT

plt.subplot(1,3,1) plt.scatter(X.Petal\_Length,X.Petal\_Width,c=colormap[y.Targets],s=40) plt.title('Real')

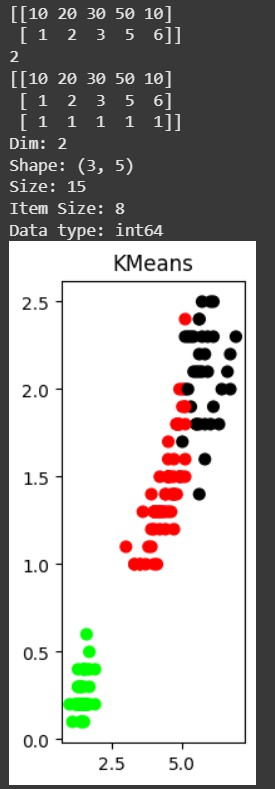
plt.show() # K-PLOT

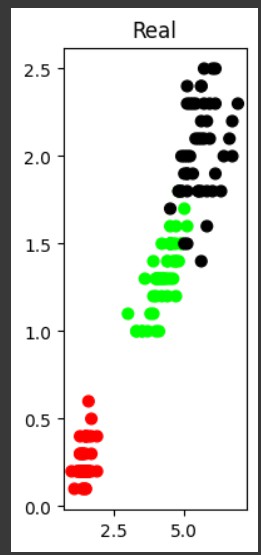
plt.subplot(1,3,2)





model=KMeans(n\_clusters=3) model.fit(X)

****predY=np.choose(model.labels\_,[0,1,2]).astype(np.int64) plt.scatter(X.Petal\_Length,X.Petal\_Width,c=colormap[predY],s=40) plt.title('KMeans') **Output:**

****

****



### Conclusion:

This program applies the K-Means clustering algorithm to the Iris dataset to group flowers based on their features. The dataset is visualized, showing the real classifications and the K-Means predicted clusters. The model successfully identifies patterns in the data, demonstrating the effectiveness of unsupervised learning for clustering similar data points.