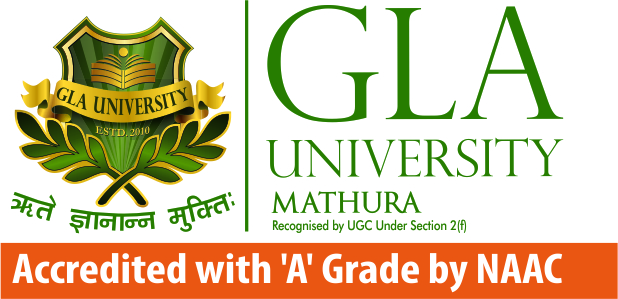
GLA UNIVERSITY, MATHURA

**LAB RECORD**

**SUBJECT: Machine Learning LAB**

**SUBJECT CODE: BCSE0133**

**SESSION: 2023-2024**

**SUBMITTED TO:**

Dr. Sachin Upadhyay

**SUBMITTED BY:**

**Siddhant Sahay**

**B.TECH III YEAR**

**Branch- B.Tech. (CSE)**

**Class Roll No-61**

**University Roll No -2115000992**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.**  **No.** | **Experiment** | **Date of Perform** | **Date of Submission** | **Signature** |
| **1** | **Introduction to Pandas, Upload data and Preprocessing**  **Introduction to Numpy and Matplotlib in Python** |  |  |  |
| **2** | **Implement Linear Regression with one variable in Python**  **(Salary Prediction data)** |  |  |  |
| **3** | **Implement Linear Regression with multi variables in Python**  **(Housing Price dataset)** |  |  |  |
| **4** | **Implement binary classification using Logistic Regression in Python**  **(Bank Customer Churn Dataset)** |  |  |  |
| **5** | **Implement Principal Component Analysis (PCA) in Python** |  |  |  |
| **6** | **Implement Support Vector Machine (SVM) classifier in Python** |  |  |  |
| **7** | **Implement k Nearest Neighbor (KNN) in Python** |  |  |  |
| **8** | **Implement Random Forest in Python** |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **9** | **Implement Naïve Bayes (NB) in Python** |  |  |  |
| **10** | **Implement ANN Classifier** |  |  |  |
| **11** |  |  |  |  |
| **12** |  |  |  |  |
| **13** |  |  |  |  |
| **14** |  |  |  |  |
| **15** |  |  |  |  |
| **16** |  |  |  |  |
| **17** |  |  |  |  |
| **18** |  |  |  |  |
| **19** |  |  |  |  |
| **20** |  |  |  |  |

Experiment-1

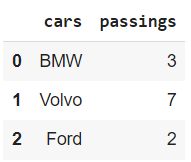
Objective-Introduction to pandas, upload data and preprocessing

Pandas

->Pandas is a open-source library that is built on top of NumPy library. It offer various package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier.

import pandas as pd

v=pd.DataFrame(mydataset)



Upload data

**Data** is a important component in the field of Machine Learning.

It refers to the set of observations or measurements that can be used to train a machine-learning model. The quality and quantity of data available for training and testing plays an important role in determining the performance of a ML model.

Data is divided into two types:

1. Labeled data-It includes a label or target variable that the model is trying to predict.
2. Unlabeled data-It doesn’t include a label or target variable.

**Data :** It can be any unprocessed fact, value, text, sound, or picture that is not being interpreted and analyzed.

**Information:** Data that has been interpreted and manipulated and has now some meaningful inference for the users.

Knowledge**:** Combination of inferred information, experiences, learning, and insights. Results in awareness or concept building for an individual or organization.

Data is split into three types in Machine Learning

1.Training Data

2.Validation data

3.Testing data

Preprocessing

Data preprocessing sin ML refers to a technique of preparing(cleaning and organizing) the raw data to make it suitable for a building and training a Machine Learning Models.

Data Preprocessing Steps:

1. Data Cleaning -In this we locates and fixes errors or discrepancies in the data.
2. Data Integration-It integrates information extracted from multiple sources to outline and create a single dataset.
3. Data Transformation-this process entails putting the data in a format that will allow for analysis.
4. Data Reduction-It a process of lowering the data the datasets size while maintain crucial information.

Objective- Introduction to NumPy and MatPlotlib in Python:

Numpy

->NumPy is a Python library used for working with arrays.

->It also has functions for working in domain of linear algebra, fourier transform, and matrices.

->NumPy stands for Numerical Python.

import numpy

a=numpy.array([[0,1],[2,3]])

print(a)

[[0 1]

[2 3]]

c=numpy.ones(2)

print(c)

b=numpy.zeros((2,4))

[1. 1.]

Array ([[0., 0., 0., 0.],

[0., 0., 0., 0.]])

1.5.3

MatplotLib

->Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

->Matplotlib is open source and we can use it freely.

->Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

import matplotlib.pyplot as plt

df = pd.DataFrame({

    'Task': ['Task 1', 'Task 2', 'Task 3'],

    'Start': ['2023-01-01', '2023-02-01', '2023-03-01'],

    'Finish': ['2023-02-01', '2023-03-01', '2023-04-01']

})

df['Start'] = pd.to\_datetime(df['Start'])

df['Finish'] = pd.to\_datetime(df['Finish'])

plt.figure(figsize=(10, 5))

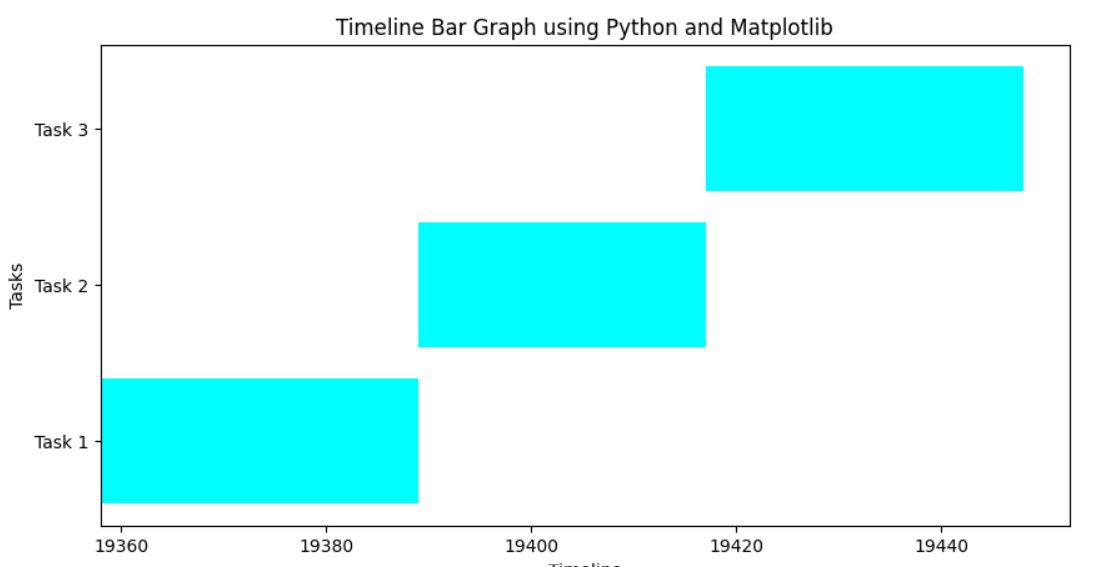
plt.barh(df['Task'], df['Finish'] - df['Start'], left=df['Start'], color='cyan')

plt.xlabel('Timeline')

plt.ylabel('Tasks')

plt.title('Timeline Bar Graph using Python and Matplotlib')

plt.show()



Experiment-2

Objective-Implement Linear Regression with one variable in Python

(Salary Prediction data)

Dataset link-<https://www.kaggle.com/datasets/krishnaraj30/salary-prediction-data-simple-linear-regression>

Linear Regression

Linear regression is used to predict the relationship between two variables by applying a linear equation to observed data. There are two types of [variable](https://www.vedantu.com/maths/variable), one variable is called an independent variable, and the other is a dependent variable. Linear regression is commonly used for predictive analysis.

The measure of the relationship between two variables is shown by the correlation coefficient. The range of the coefficient lies between -1 to +1. This coefficient shows the strength of the association of the observed data between two variables.

Linear Regression Equation is given below:

Y=a+bX

where X is the independent variable and it is plotted along the x-axis

Y is the dependent variable and it is plotted along the y-axis

Here, the slope of the line is b, and a is the intercept (the value of y when x = 0).

Linear Regression with one variable

import pandas  as pd

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

data={

    'Plot\_size':[100,150,200,250,300,350,400,450,500],

    "Plot\_Price":[200000,250000,300000,350000,400000,450000,500000,550000,600000]

    }

df=pd.DataFrame(data)

print(df.head())

x=df[['Plot\_size']]

y=df['Plot\_Price']

model=LinearRegression()

model.fit(x,y)

ns=[[600],[700]]

p=model.predict(ns)

for size,price in zip(ns,p):

  print(f'Plot size:{size[0]},Plot price:{price:.2f}')

plt.scatter(x,y,color="green",label="Actual price")

plt.plot(x,model.predict(x),color='yellow',linewidth=2,label='LinearRegression')

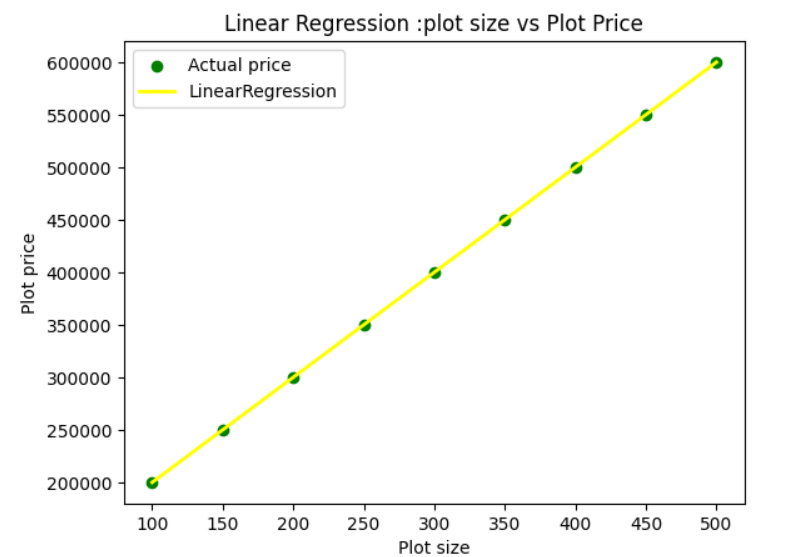
plt.xlabel('Plot size')

plt.ylabel('Plot price')

plt.legend()

plt.title('Linear Regression :plot size vs Plot Price')

plt.show()



Experiment-3

Objective-Implement Linear Regression with multi variables in Python

(Housing Price dataset)

Dataset link-<https://www.kaggle.com/datasets/yasserh/housing-prices-dataset>

MultiLinear Regression

Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.

Each feature variable must model the linear relationship with the dependent variable.

MLR tries to fit a regression line through a multidimensional space of data-points.

Linear Regression with multi variables

import pandas as pd

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

data={

    'Age':[25,30,35,40,45,50,55,60],

    'Years\_of\_experience':[2,5,8,10,12,15,18,20],

    'Salary':[50000,60000,75000,80000,90000,100000,110000,120000]

}

df=pd.DataFrame(data)

x= df[['Age','Years\_of\_experience']]

y=df['Salary']

model=LinearRegression()

#fit the model on the data

model.fit(x,y)

new\_employees\_details=[[30,5]]

predicted\_salary=model.predict(new\_employees\_details)

print("Predicted Salary for new employee details:")

print(f"Age:{new\_employees\_details[0][0]}, Years\_of\_experience:{new\_employees\_details[0][1]},Predicted Salary:{predicted\_salary[0]:.2f}")

Experiment-4

Objective-Implement binary classification using Logistic Regression in Python (Bank Customer churn dataset)

Dataset link-<https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset>

Logistic Regression

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix,accuracy\_score,precision\_score,recall\_score,f1\_score

# Read the CSV file

data = pd.read\_csv("/content/Marketingcampaigns.csv")

# Explore the dataset (optional)

# print(df.head())

data.shape

data=pd.get\_dummies(data,columns=['Location','Gender'])

data.head()

d=data.isnull()

df=pd.DataFrame(d)

df.to\_csv("missing.csv",index=False)

data=data.dropna()

data.shape

x=data.drop('Location',axis=1)

y=data['Location']

x.head()

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

# Create and train a logistic regression model

model = LogisticRegression(random\_state=42)

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

# Make predictions on the test set

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

accuracy=accuracy\_score(y\_test,y\_pred)

f1score=f1\_score(y\_test,y\_pred)

recall=recall\_score(y\_test,y\_pred)

precision=precision\_score(y\_test,y\_pred)

confusion=confusion\_matrix(y\_test,y\_pred)

print(confusion)

print(accuracy)

print(f1score)

print(precision)

print(recall)

ax = sns.heatmap(confusion, annot=True, cmap='Blues')

ax.set\_title('Seaborn Confusion Matrix with labels\n\n');

ax.set\_xlabel('\nPredicted Values')

ax.set\_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order

ax.xaxis.set\_ticklabels(['False','True'])

ax.yaxis.set\_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.

plt.show()

Experiment-5

Objective-Implement Principal Component Analysis (PCA) in Python.

Dataset link-<https://data.world/sdhilip/pizza-datasets>

Principal Component Analysis :

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning.

It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**.

Some real-world applications of PCAareimage processing, movie recommendation system, optimizing the power allocation in various communication channels.

import numpy as np

import matplotlib.pyplot as plt

data=np.array([[2,3],[3,4],[4,5],[5,6],[6,7]])

mean=np.mean(data,axis=0)

sd=np.std(data,axis=0)

ds=(data-mean)/sd

cm=np.cov(ds.T)

eigv ,eigve=np.linalg.eig(cm)

pc=eigve[:,np.argmax(eigv)]

pm=pc.reshape(-1,1)

pca\_data=ds.dot(pm)

plt.figure(figsize=(10,5))

plt.subplot(1,2,1)

plt.scatter(ds[:,0],ds[:,1])

plt.title('Original Data (2D)')

plt.xlabel('Feature1')

plt.ylabel('Feature2')

plt.subplot(1,2,2)

plt.scatter(pca\_data,np.zeros\_like(pca\_data))

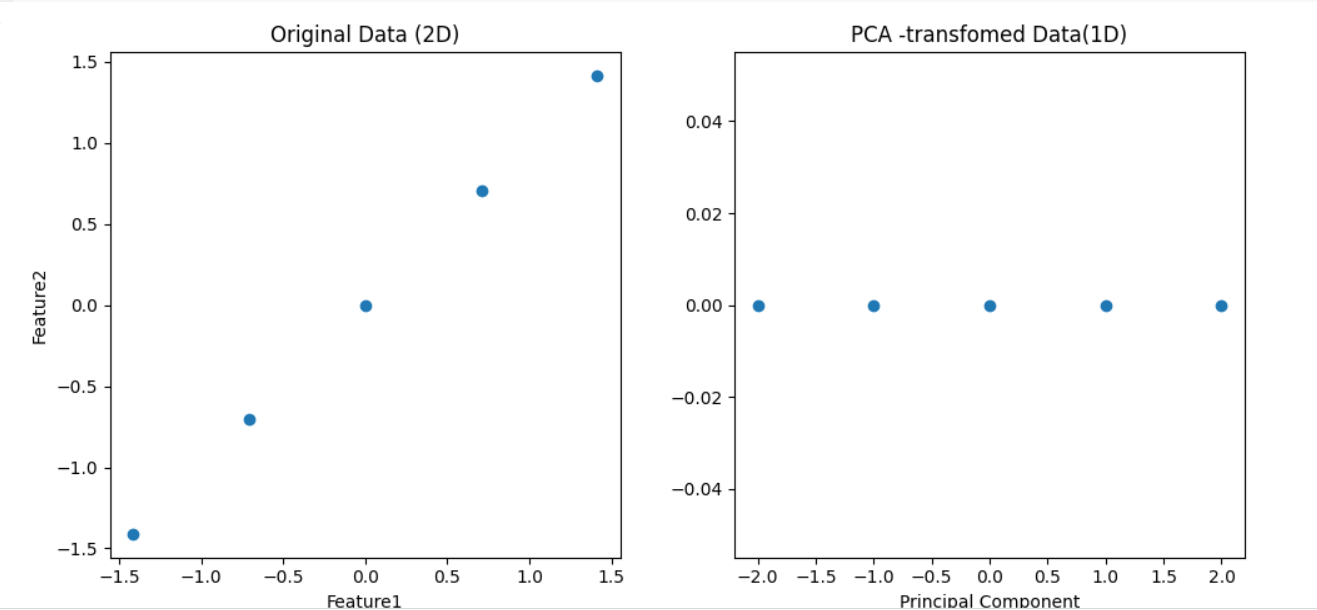
plt.title('PCA -transfomed Data(1D)')

plt.xlabel('Principal Component')

plt.ylabel(' ')

plt.tight\_layout()

plt.show()

**

Experiment-6

Objective- Implement Support Vector Machine (SVM) classifier in Python.

Cell Sample: https://cf-courses-data.s3.us.cloud-object-

storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-

SkillsNetwork/labs/Module%203/data/cell\_samples.csv

Support Vector Machine (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

from  sklearn import datasets

import pandas as pd

import numpy as np

iris=datasets.load\_iris()

iris.target\_names

iris.keys()

iris.data.shape

iris.data

iris=pd.DataFrame(data=np.c\_[iris['data'],iris['target']],columns=iris['feature\_names']+['target'])

iris.head()

species=[]

for i in range(len(iris['target'])):

  if iris['target'][i]==0:

    species.append("setosa")

  elif iris['target'][i]==1:

      species.append('versicolor')

  else:

    species.append('virginica')

iris['species']=species

import matplotlib.pyplot as plt

setosa=iris[iris.species=="setosa"]

versicolor=iris[iris.species=="versicolor"]

virginica=iris[iris.species=="virginica"]

fig,ax=plt.subplots()

fig.set\_size\_inches(10,7)

ax.scatter(setosa['petal length (cm)'],setosa['petal width (cm)'],label='Setosa Petal',facecolor="blue")

ax.scatter(versicolor['petal length (cm)'],versicolor['petal width (cm)'],label='Versicolor',facecolor="green")

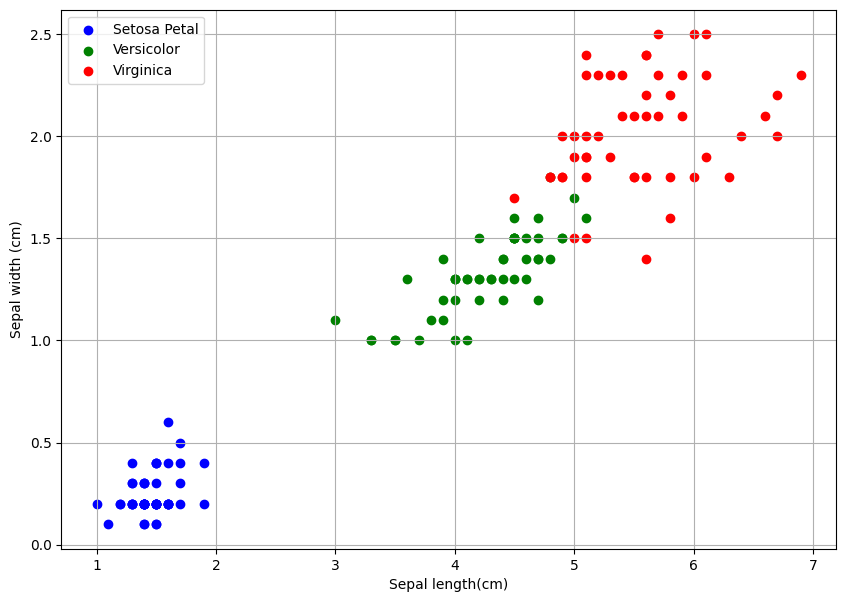
ax.scatter(virginica['petal length (cm)'],virginica['petal width (cm)'],label='Virginica',facecolor="red")

ax.set\_xlabel('Sepal length(cm)')

ax.set\_ylabel('Sepal width (cm)')

ax.grid()

ax.legend()



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

x=iris.drop(['sepal length (cm)','sepal width (cm)','target','species'],axis=1)

y=iris.target

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=82)

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

kernels=['linear','rbf','poly']

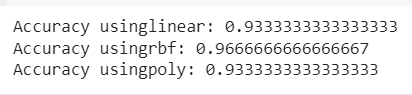
for kernel in kernels:

  model=SVC(kernel=kernel)

  model.fit(x\_train,y\_train)

  pred=model.predict(x\_test)

  print("Accuracy using{}:".format(kernel),accuracy\_score(pred,y\_test))



## Hyperplane and Support Vectors in the SVM algorithm:

**Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

**Support Vectors:** The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Experiment-7

Objective-Implement k Nearest Neighbour (KNN) in Python.

Cell Sample: https://cf-courses-data.s3.us.cloud-object-

storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-

SkillsNetwork/labs/Module%203/data/cell\_samples.csv

K-Nearest Neighbor (KNN)

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

import pandas as pd

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

data=pd.read\_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',names=['sepal\_length1','sepal\_width','petal\_length','petal\_width','species'])

# data.head()

x=data[['sepal\_length1','sepal\_width','petal\_length','petal\_width']]

y=data['species']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.25,random\_state=42)

knn=KNeighborsClassifier(n\_neighbors=5)

knn.fit(x\_train,y\_train)

y\_pred=knn.predict(x\_test)

accuracy=accuracy\_score(y\_test,y\_pred)

print("K-Nearest Neighbors Classifier accuracy",accuracy)



Experiment-8

Objective-Implement Random Forest in Python

Cell Sample: https://cf-courses-data.s3.us.cloud-object-

storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-

SkillsNetwork/labs/Module%203/data/cell\_samples.csv

Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique.

As the name suggests, ***"*Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting**

**import** pandas as pd

**import** matplotlib.pyplot as plt

**import** seaborn as sns

**import** sklearn

**import** warnings

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

**from** sklearn.impute **import** KNNImputer

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

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.metrics **import** f1\_score

**from** sklearn.ensemble **import** RandomForestRegressor

**from** sklearn.ensemble **import** RandomForestRegressor

**from** sklearn.model\_selection **import** cross\_val\_score

warnings.filterwarnings('ignore')

df**=** pd.read\_csv('Salaries.csv')

print(df)

df.info()

X **=** df.iloc[:,1:2].values

y **=** df.iloc[:,2].values

**import** pandas as pd

**from** sklearn.ensemble **import** RandomForestRegressor

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

check **for** **and** handle categorical variables

label\_encoder **=** LabelEncoder()

x\_categorical **=** df.select\_dtypes(include**=**['object']).apply(label\_encoder.fit\_transform)

x\_numerical **=** df.select\_dtypes(exclude**=**['object']).values

x **=** pd.concat([pd.DataFrame(x\_numerical), x\_categorical], axis**=**1).values

regressor **=** RandomForestRegressor(n\_estimators**=**10, random\_state**=**0, oob\_score**=**True)

regressor.fit(x, y)

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

oob\_score **=** regressor.oob\_score\_

**print**(f'Out-of-Bag Score: {oob\_score}')

predictions **=** regressor.predict(x)

mse **=** mean\_squared\_error(y, predictions)

print(f'Mean Squared Error: {mse}')

r2 **=** r2\_score(y, predictions)

print(f'R-squared: {r2}')

**import** numpy as np

X\_grid **=** np.arange(min(X),max(X),0.01)

X\_grid **=** X\_grid.reshape(len(X\_grid),1)

plt.scatter(X,y, color**=**'blue')

plt.plot(X\_grid, regressor.predict(X\_grid),color**=**'green')

plt.title("Random Forest Regression Results")

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

**from** sklearn.tree **import** plot\_tree

**import** matplotlib.pyplot as plt

tree\_to\_plot **=** regressor.estimators\_[0]

plt.figure(figsize**=**(20, 10))

plot\_tree(tree\_to\_plot, feature\_names**=**df.columns.tolist(), filled**=**True, rounded**=**True, fontsize**=**10)

plt.title("Decision Tree from Random Forest")

plt.show()

Experiment-9

Objective-Implement Naïve Bayes (NB) in Python

Cell Sample: https://cf-courses-data.s3.us.cloud-object-

storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-

SkillsNetwork/labs/Module%203/data/cell\_samples.csv

Naive Bayes

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

It is mainly used in *text classification* that includes a high-dimensional training dataset.

* Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
* It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
* Some popular examples of Naïve Bayes Algorithm are spam filtration,Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

**Naive**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of colour, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

**Bayes:** It is called Bayes because it depends on the principle of Bayes’s Theorem.

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

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

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score,classification\_report

iris=load\_iris()

x=iris.data

y=iris.target

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

nb=MultinomialNB()

nb.fit(x\_train,y\_train)

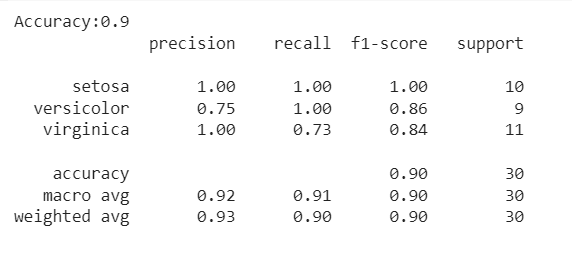
y\_pred=nb.predict(x\_test)

accuracy=accuracy\_score(y\_test,y\_pred)

print(f"Accuracy:{accuracy}")

report=classification\_report(y\_test,y\_pred,target\_names=iris.target\_names)

print(report)



## Bayes' Theorem:

Naïve Bayes Classifier Algorithm

Where ,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

**P(A) is Prior Probability**: Probability of hypothesis before observing the evidence.

**P(B) is Marginal Probability**: Probability of Evidence.

Experiment-10

Objective-Implement ANN classifier.

Dataset link-<https://www.kaggle.com/datasets/hojjatk/mnist-dataset>

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

Artificial Neural Network primarily consists of three layers:

**Input Layer:**

As the name suggests, it accepts inputs in several different formats provided by the programmer.

**Hidden Layer:**

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

**Output Layer:**

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

What is Artificial Neural NetworksPlay Video

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

