**INFORMATION TECHNOLOGY DEPARTMENT**

**A**

**Laboratory Manual**

****

**PYTHON FOR MACHINE LEARNING**

**(Course Code MLP190910)**

**Semester : V**

# Vision /Mission Statements

## Institute Vision

**SBM Polytechnic aspires to be the lead institute in providing need based technical education.**

## Institute Mission

* **To provide state-of-the-art infrastructure and latest equipments for providing a stimulating learning environment.**
* **To prepare students to meet the dynamic needs of the industry by periodic reviewing and upgradation of curriculum through an interactive process with industry.**
* **To inculcate a spirit of excellence in terms of academic performance, research and innovation in faculty by providing appropriate support and incentive systems.**
* **To promote and support Co-Curricular, extra-curricular activities and industry interaction to make students socially sensitive and employable.**

## Department Vision

**Information Technology Department**

1. **Vision:**

To create a learning environment that nurtures students and transform them into competent IT Diploma Graduates

1. **Mission:**

M1: To impart technical and managerial skills for pursuing academic excellence through dynamic learning environment.

M2: To foster industry ready graduates by acquiring and utilizing latest technology.

M3: To strengthen holistic development and Professionalism in the Diploma graduates.

1. **Program Educational Objectives(PEO):**

PEO1: Imbibe core knowledge and utilize associated technologies to provide domain related solutions.

PEO2: Be capable of adapting to the rapid pace of technological dynamics through professional competence.

PEO3: Develop a holistic personality equipped with leadership qualities, team skills and humane approach towards society.

**Information Technology Department**

1. **Program Outcomes(PO):**
2. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
3. **Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods
4. **Design/ development of solutions:** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
5. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
6. **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.
7. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
8. **Life-long learning:** Ability to analyse individual needs and engage in updating in the context of technological changes
9. **PROGRAM SPECIFIC OUTCOMES (PSOs)**

**At the end of the programme:**

1. Students will demonstrate fundamental knowledge in core domains of IT such as Software Development, Databases and Information Systems

Students will acquire skills that can provide IT solutions in the field of Networking, IOT, Machine Learning and Cloud Computing

## Program Educational objectives

PEO1-Identify ,frame and solve computing problems by applying knowledge in Computer engineering

PEO2- Promote lifelong learning by integrating academic knowledge and practical applications

PEO3- Depict effective team work and practical skills for holistic development

## Program Outcomes

1. Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
2. Problem analysis: Identify and analyse well-defined engineering problems using codified standard methods
3. Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.
6. Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities
7. Life-long learning: Ability to analyse individual needs and engage in updating in the context of technological changes

## Program Specific Outcomes

PSO1: Demonstrate the fundamental knowledge in the areas of Operating system, Web Technology, Microprocessor based system and IOT by applying programming skills and developing applications

PSO2: Administer and manage Open source, Networking, Security and Database domains to enhance student growth

## Course Outcomes

**At the end of the semester student will be able to: -**

1. To implement.
2. To implement
3. To implement memory management algorithms
4. To implement file management algorithms
5. To describe concepts of multiprocessor and distributed operating systems

## CO-PO Mapping

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Outcomes (CO)** | **PO 1** | **PO 2** | **PO 3** | **PO 4** | **PO 5** | **PO 6** | **PO 7** | **PSO 1** | **PSO 2** |
| **CO 1** |  |  |  |  |  |  |  |  |  |
| **CO 2** |  |  |  |  |  |  |  |  |  |
| **CO 3** |  |  |  |  |  |  |  |  |  |
| **CO 4** |  |  |  |  |  |  |  |  |  |
| **CO 5** |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | **3.00** | **1.50** |  | **1.00** |  | **1.00** | **1.00** | **3.00** |  |
|

# Index

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Title of Experiment** | **Pos Covered** | **Page No** |
| 1 | To calculate mean, median, mode and standard deviation of a statistical data using Python. |  |  |
| 2 | To perform the basic array operations with the Numpy Library. |  |  |
| 3 | To practice dataframe and file operations with the pandas library. |  |  |
| 4 | To perform data cleaning and transformation operations using the Numpy and pandas libraries CO2 |  |  |
| 5 | To visualize data using the matplotlib library and seaborn visualization commands. |  |  |
| 6 | To predict housing prices on Boston Housing Price Prediction Dataset using Linear Regression |  |  |
| 7 | To perform grouping of flower varieties on the iris dataset using K-Nearest Neighbors |  |  |
| 8 | To perform grouping of flowers into flower species on the iris dataset using k-means clustering. |  |  |
| 9 | To perform grouping of flowers into flower species on the iris dataset using hierarchical clustering. |  |  |
| 10 | To predict whether a customer will default or not on the credit card dataset using Logistic Regression, Decision Tree and Naive Bayesian Classifier. |  |  |
| 11 | Mini Project |  |  |

# Experiment 1 : Program to calculate mean, median, mode and standard deviation of a statistical data using Python.CO1

**AIM:** To calculate mean, median, mode and Standard deviation of a statistical data using Python.

**THEORY:**

Machine Learning: Machine learning gives the computers the ability to learn without being explicitly programmed.

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if it's performance on T, as measured by P, improves with experience E.

Classification:

1. Supervised Learning

2. Unsupervised Learning

3. Reinforcement Learning

4. Semi- Supervised Learning

Categorization of output:

1. Classification

2. Regression

3. Clustering

Python Machine learning: Python's extensive selection of machine - learning specific libraries and frameworks simplify the development process and cut development time

Python's simple syntax and readability promote rapid testing of complex algorithms, and make the language accessible to non-programmers

Data Set:

A data set is a collection of data. It can be anything from an array / list to a complete database

Example:

list = [2, 4, 6, 8, 9, 21, lists in python

csv files: Comma Seperated Values (csv) is a plain text file that contains data.

3.Excel: Excel is a spreadsheet program used with data like numbers and formulas, texts, etc.

Mean, Median, Mode and Standard Deviation:

Mean: The average value.

Median The mid point value Mode: The most common value.

Standard Deviation. Describes how spread out the values are.

**CODE**:

import numpy

from scipy import stats

Speed = [99, 79, 87, 111, 87, 137, 117, 87]

print ("Mean: " numpy.mean (speed))

print ("Median: ", numpy.median (speed))

print ("Mode: ", stats.mode (speed))

print("Standard Deviation: ", numpy.std (speed))

**CONCLUSION**:

Machine Learning is making the computer learn from studying the data. It enables analysis of massive quantities of data and predict the outcomes. It is a step of AI. Through this experiment we studied about mean, median, mode and standard deviation by python code using python libraries.

**Questions :**

1. Why do we need mean median mode values

2. What statistical data is needed for understanding data

**OUTPUT:**



# Experiment2: To practice dataframe and file operations with the pandas library. Co2

**AIM :** To perform basic array operations with Numpy Library.

**THEORY**: Numpy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and metrices. Numpy was created in 2005 by Travis Oliphant. It is an open source project and you can use

it freely. Numpy stands for Numerical Python.

Lists are used in Python, which are slow as compared to the array object in Numpy called ndarray.

Installation:

pip install numpy

1. Creating arrays:

import numpy as np

 array1= np.array ([(1, 2, 3, 7), (4, 5, 6, 9)])

2. Indexing: Array indexing is same as accessing an array element.

 print (array 1 [1] [3]) →9

3. Array Dimensions: Numpy provides the ndim attribute that returns an integer that tells how many dimensions the array have.

print (array1.ndim)   → 2

4. Array slicing: Slicing in python means taking elements from one given index to another index. Syntax: [start:end]

               [start: end: step]

print (array1 [0,2])    → 3

print (array 1 [0:,3])    → [7,9]

5. Datatypes: The numpy array object has property called dtype that returns data type of the array.

print(array1.dtype) →int 32

6.  Shape: Numpy arrays have an attribute called shape that returns tuple with each index having the number of corresponding elements.

print (array1. shape) → (2,4).

7.  Itemsize: Numpy arrays have an attribute called itemsize that returns the length of each element of array in bytes.

print (array1. itemsize) →4

8. Array Size: Numpy arrays have an attribute called size that returns the size of an array.

print (array1. size) →8

9. Reshape: Reshaping means changing the shape of an array.

print (array1. reshape (4,2))

10. Copy / View Copy or View the original array.

 print (array1.copy ())

print (array1. view())

Methods:

1.arange: Returns an ndarray object containing evenly spaced values with given range.

np.arange (start, stop, step, dtype)

2. linespace similar to arange(), instead of stepsize,

the number of evenly spaced values with given range.

np.linespace (start, stop, number, endpoint, retstep, dtype)

3. zeros: Returns new array of specified size with zeros

np.zeros(shape, dtype, order)

4. ones: Returns new array of specified size, with ones.

np.ones(shape, dtype, order)

5. empty: Uninitialized array of specified shape and dtype.

 np.empty (shape, dtype, order)

6. concatenate : Joining Numpy arrays.

np. concatenate()

7. split: Splitting Numpy arrays, pass the array to split and number of splits.

8. search: searching for a certain value, and return. the indexs that get a match.

 np.where ()

9. Sort: The Numpy ndarray object has a function called sort(), that sorts the array.

np. sort()

10. Filter: In Numpy, filter an array using a boolean index list

**PROBLEM STATEMENT:** WAP to perform basic operations. with Numpy libraries.

**SOURCE CODE:**

import numpy as np

a= np. array([(1,2,3,4), (3,4,5,6), (7,8,9,10)])

print(a)

print ("dimension:", a.ndim)

print("itemsize: “,a.itemsize)

print("Datatype:", a.datatype)

print("shape", a.shape)

print("slicing: ", a[0, 2])

print ("slicing: ", a[0:,2])

print ("Slicing: ", a[0: 2,2])

b= np. array ([1,2,3,4), (3,4,5,6)])

print (b)

b= b.reshape (4,2)

print ("Reshape: ", b)

p.np.arange (15)

print (p)

print (np. arange(10, 15))

print (np. zeros((3, 3))

print (empty (3,4))

a = np. ones ((2,3), dtype = np. int16)

print (a)

print (np.random. random ((2,3)))

**CONCLUSION**:

Numpy is a python library used for working with arrays. By this experiment, we learnt and implemented basic operations with Numpy library.

**Sample Questions**

1. Describe the methodology for installing jupyter notebook.

2. Why to use?

**OUTPUT:**

[[1 2 3 4]

[3 4 5 6]

[7 8 9 10]]

dimension: 2

Itemsize: 4

Datatype: int32

shape: (3,4)

Slicing: 3

slicing [3 5 9]

Slicing: [3 5]

[[1 2 3 4]

 [3 4 5 6]]

Reshape: [[1 2]

[3 4]

[3 4]

[5 6]]

[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]

[10 11 12 13 14]

[[0. 0. 0.]

[0. 0. 0.]

[0. 0. 0.]]

[[6.230420-367   3.560430530-301   1.0602196e-306 2.44763750-307]

[1.691193e-306   1.780215e-306   1.0512165e-307   5.2156110e-306]

[9.3462151e-307  1.7801235e-306   8.34451120-308  1.11262153e-306]]]

[[1 1 1]

[ 1 1 1]]

# Experiment 3 : To practice dataframe and file operations with the pandas library.

**AIM:** To practice dataframe and file operations with the pandas library.

**THEORY :**

‘Pandas’ is a Python library for data analysis started by Wes Mckinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas has grown into one of the most popular Python libraries.

It an extremely active community of contributors. It is used for working with data sets. It has functions for analysing, cleaning, exploring, and manipulating data.

Pandas allows us to analyze big data and make conclusions based on statistical theories.

It can clean messy data sets, and make them readable and relevant.

* Installation of Pandas:

pip install pandas

* Import Pandas:

import pandas

* Pandas is usually imported under the pd alias.

import pandas as pd

Pandas introduced two new types of objects for storing data that make analytical tasks easier and eliminate the need to switch tools : Series, which have a list-like structure and DataFrames which have a tabular structure.

Pandas Series : Pandas series is a one-dimensional labelled array capable of holding data of any type. This axis labels are collectively called index. Pandas series is nothing but a column in an excel sheet. A series can be created using various inputs like Array, Dict, scalar value or constant.

eg. data = np.array (['a', 'b', 'c','d'])

S = pd. Series(data)

As we did not pass index, so by default, it assigned the indexes ranging from 0 to len(data)-1.

Pandas DataFrames : Pandas DataFrame is a two dimensional size mutable, potentially heterogeneous tabular data structure with labelled axis. A DataFrame is two dimensional data structure, ie data assigned in a tabular fashion in rows and Columns. Pandas Dataframes consists of three principles components, the data, rows and columns.

eg. data-[[‘Alex’, 10], ['Bob', 12], ['clarke', 13]]

df=pd. DataFrame (data, columns = ['Name', 'Age']]

Kaggle:

kaggle is an online community of data scientists. and machine learning practitioners. It allows users to find and publish data sets, explore and build models in a web-based data science environment. To download any dataset, visit the site: https://www.kaggle.com/datasets and search for the required dataset (csv file). Download it.

File Operations:

1. Reading .csv file which was downloaded from kaggle.

→ import pandas as pd.

data = pd. read.csv ("nba.csv")

2.head(): Returns top n (5 default) rows of a dataframe.

→ data.head()

3 describe(): View basic statistical details like percentile, mean, std, etc. of data frame of numerical values.

→ data.describe().

4.column Deletion: drop() method deletes columns by dropping columns with column names.

→ data.drop ( ["College"), axis = 1, inplece=True)

data.head()

5. Row Selection: loc[ ] function takes only index lables and returns row or data frame if index lable exists in the caller data frame.

→ data=pd. Read\_csv ("nba.csv", index\_col="Name")

data.loc [R.J. Hunter]

6.Row Addition: concat() method is used to add new row in Pandas Dataframe.

→ new-row = apd. DataFrame ( {"Name': 'Stefan Gilbert', 'Team':

'Boston', 'Number’: 9, 'Position': 'PG', 'Age’: 33, 'Height': '6-2', 'weight' 189, 'college': 'MIT', 'Salary: 9999 }, index = [0])

data = pd. concat ( [new\_row, df]) .reset\_index (drop=True)

data.head()

7. Row Deletion :drop() method deletes row by dropping rows by index label.

→ data=pd.read\_csv ("nba.csv", index\_cd= "Name")

data.drop (["A very Bradley", "John Holland", "R.J Hunter"],inplace= True)

8. shape: Returns the number of rows and columns in a tuple.

→ data. shape

9.iloc[ ]: Used when index label of dataframe is something other than Numeric or in case when the user doesn't know the index label.

→ data. iloc [3]

10.value\_counts(): Counts unique values in the particular column.

→ data [' college']. value\_counts()

11.Columns: Returns names of the columns in the dataset.

→ data columns.

12.Values Returns rows with all the values in the dataset.

→ data. values.

13. info( ): Returns detailed information about the dataset

→ data. info()

**CONCLUSION**:

By this experiment, we learnt about pandas, dataframes and series, downloading datasets from kaggle and performed operations with pandas library.

**SAMPLE QUESTIONS :**

1. Describe the context switching practice dataframe and file operations with the pandas

2. Identify

3. Calculate

**OUTPUT:**

# Experiment 4: To perform data cleaning and transformation operations using the Numpy and pandas

**AIM :** To perform data cleaning and transformation operations using the Numpy and pandas libraries

**THEORY :**

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate or incomplete data within a dataset.

1. Dropping Columns in a Dataframe: Pandas provide a handy way of removing unwanted columns or rows from a DataFrame with the drop() function

eg: data.drop (['College' ], axis = 1, inplace = True)

2. Changing the index of a Dataframe: A Pandas Index extends the functionality of Numpy arrays to allow for more versatile slicing and labelling. In many Cases, it is helpful to use uniquely valued identifying field of the data as its index.

eg: data. set\_index ('Number')

3. Tidying up Fields in the data: Cleaning specific columns and get them to a uniform format uniform format to get a better understanding of the dataset and enforce consistency.

4. Cleaning Entire Dataset: In certain situations, the inconsistency is not localized to one column but is spread out. There are some instances where to apply a customised function to each cell or element of a DataFrame is more advisable. Pandas applymap() method is similar to the built-in map() function and simply applies a function to all elements in the dataframe. It takes only one parameter, i.e function that should be applied to each element.

5. Renaming columns and skipping nows: Often the datasets will not have either easy columns names to understand, or unimportant information in first few / last rows such

as definations or footnotes

new = {'Number': 'JNumber')

eg: data.rename (columns-new, inplace=True)

eg: data= pd. Read\_csv ("Dataset", header=1)

**CODE:**

import pandas as pd

import numpy as np.

data=pd. Read\_csv (‘clympics.csv')

data.head ()

data1= pd. read\_csv (‘olympics.csv), header = 1)

data. head()

new = {'0': 'ID', '1': 'Country', '2': 'Summer Olympics', '3': 'Gold', '4': 'Silver', '5': 'Bronze', '6': 'Total'}

data rename (Columns = new, inplace=True)

data head ()

data.set\_index ('ID')

data.drop (['Summer Olympics' ], axis = 1, inplace=True)

data. head()

**CONCLUSION:**

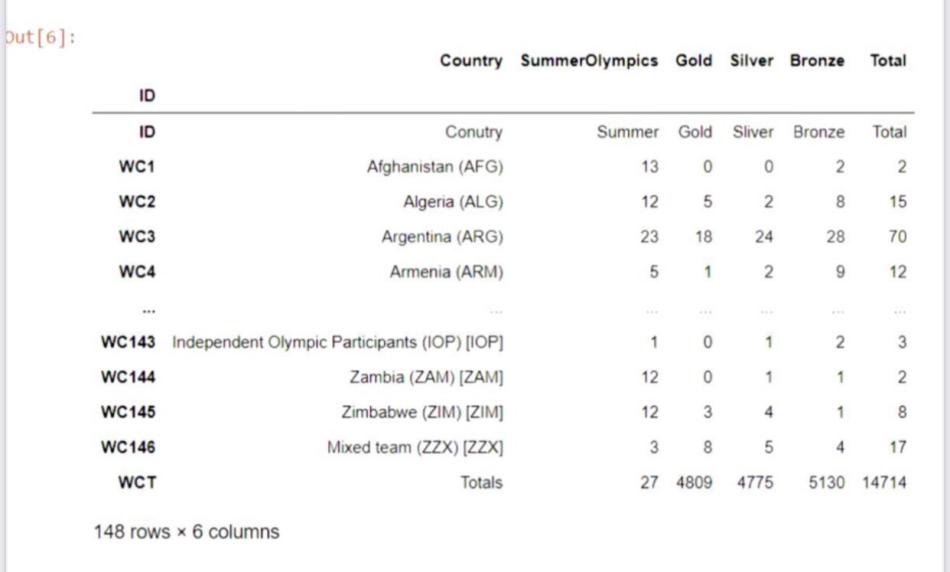
Data cleaning helps in fixing and removing incorrect or duplicate data within dataset. There are many ways to clean and transform dataset. Some of them are dropping columns, cleaning datasets using applymap(), skipping rows, renaming columns, etc.

**SAMPLE QUESTIONS :**

1. Describe the context switching practice dataframe and file operations with the pandas

Identify

**OUTPUT**:



# Experiment 5: To visualize data using the matplotlib library and seaborn visualization commands.

**AIM:** To visualize data using the matplotlib library and seaborn visualization commands.

**THEORY:**

Matplotlib: Matplotlib is a visualization library in Python for 2D plotes of arrays. Matplotlib is a multi-platform data visualization library built on Numpy arrays and designed to work with the broader SciPy stack. It consists several plots line, bar, scatter, histogram, etc...

Seaborn: Seaborn is a visualization library for statistical graphics plotting in Python. It provides attractive default styles, color palettes and statistical plots. It is built on the top of matplotlib library and also closely integrated to the data structures from Pandas.

Matplotlib v/a Seaborn:

Matplotlib is a python library used for plotting graphes with the help of other libraries like NumPy and Pandas. It is powerful for data visualization. It is used to create statistical interfaces and plotting 2D graphs or arrays. Uses PyPlot for providing MATLAB like interface free and open-source. It is capable of dealing with various OS and graphical backends.

Seaborn is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and NumPy. It is built on the root of Matplotlib and is considered as a Superset of Matplotlib. It helps in visualization of bivariate and univariate data. It uses attractive themes for enhancing Matplotlib graphics. Important tool in picturing linear Regression Models. It makes graphs of statistical Time-series data. It Eliminates overlapping of graphs.

**SOURCE CODE:**

* Matplotlib

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

df= pd.read\_csv(‘Admission\_Predict\_Ver1.1.csv’)

plt. figure (1, figsize- (14,4))

for i in range (1,6):

plt.subplot (1,5,i)

plt.boxplot (df [df. columns [i]))

plt.title (df columns [i])

plt.show()

df.describe()

df [['gre', 'toefl, "rating", "gpa", "research", 'chance ]]. hist ( figsize (14,10), bins = 10, linewidth="1", color= "blue", edgecolor= "white", grid= False)

plt. show ()

correlation =df.corr( )[‘chance’]

plt.bar (df..columns, correlation, color="red", edgecolors="white)

plt.show()

print (correlation)

df. plot (kind="scatter", x = 'gre', y ='toefl', color="green", edge color= 'white')

plt.xlabel ("GRE)

plt.ylabel("TOEFL")

plt.show()

df. plot (kind="scatter", x = 'gpa', y ='gre', color="green", edge color= 'white')

plt.xlabel ("GPA)

plt.ylabel("GRE")

plt.show()

* Seaborn

import numpy as np

import pandas as pd

import matplotlib. pyplot as plt

import seaborn as sns

sns. set ( style="white grid")

a= sns.load\_dataset (“flights”)

sns. relplot (x= "passengers", y:"month" data= flights, color="red)

sns.relplot (x= "passengers", y = "month", hue="year", data = flights)

data =pd. Dataframe ({ 'Day' : [1,2,3,4,5,6.7], “Grossary”: [30, 9.0, 45, 23, 51, 46, 16], Clothes [13,40,34, 23, 54, 61, 98], Utensils [12,32, 27, 56, 87, 54,34]}, index= [1,2,3,4,5,6,1])

sns.relplot (n= “Day”, y= "Clothes", kind= line, data=data, color="red")

data = sns. load\_dataset ("tips")

sns\_catplot (x="day", y="totalbill", data=data)

sns. catplot (x= “day”, y="total-bill", kind="violen", data = data)

sns. boxplot (x= "day", y="total\_bill", data=data)

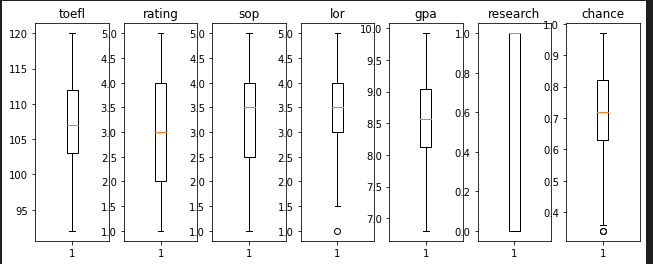
sns. boxplot (x="day", y = "total-bill, data= data)

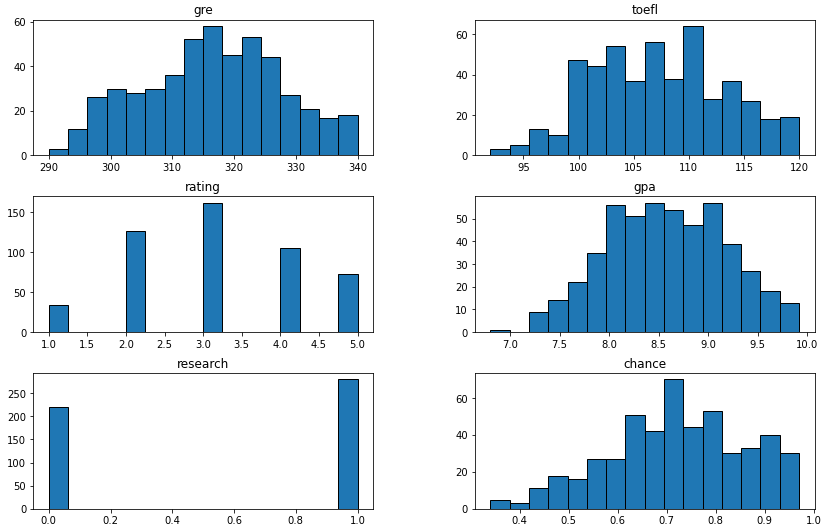
sns. PairGrid (“flights”)

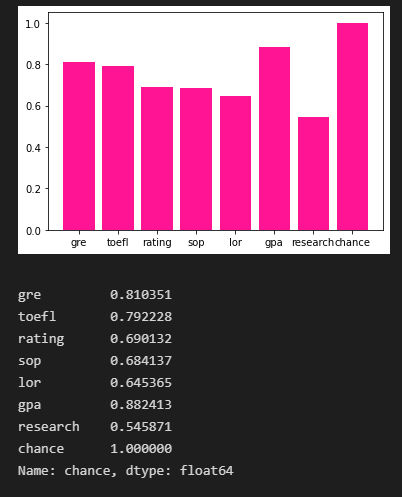
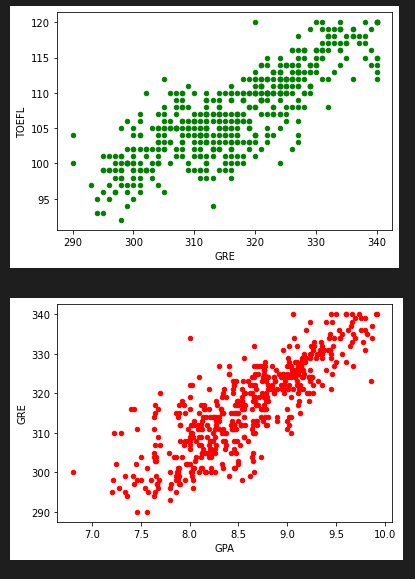
map (plt. scatter, color="blue")

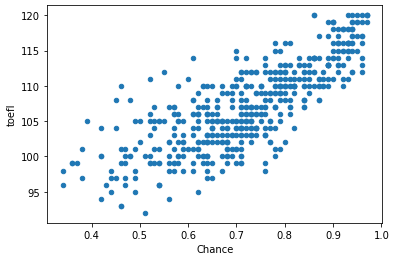
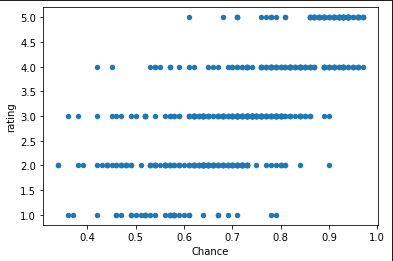
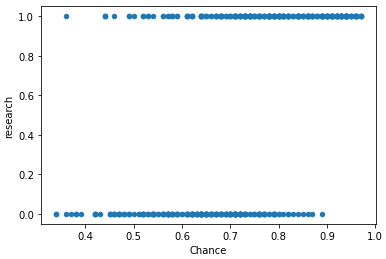
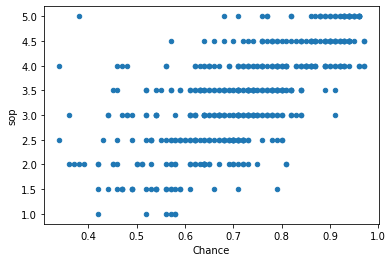
**CONCLUSION:** Matplotlib and Seaborn both are data visualization libraries. Seaborn is built on the top of Matplotlib and is more attractive and advanced. Through this experiment we have learnt about date visualization tools and implemented various graphs.

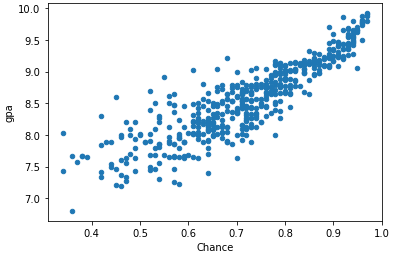
**OUTPUT:**

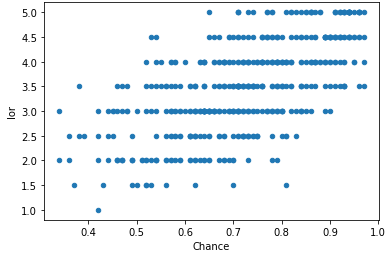










# Experiment 6: To predict housing prices on Boston Housing Price Prediction Dataset

**AIM** : To predict housing prices on Boston Housing Price Prediction Dataset

**THEORY:**

Linear Regression: Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression tack Regression models a target prediction value based on independent variables. It is mostly word for finding out the relationship between variables.

**SOURCE CODE:**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyPlot as plt

%/matplotlib inline

from Sklearn.datasets import load\_boston

boston= load \_boston()

boston.keys()

boston.feature\_name

data= pd.DataFrame(boston \_data)

data.columns=boston.feature\_names

boston .target.shape

data [PRICE] = boston.target

Sns.distplot (data [PRICE])

sns .Implot (x =”RM”, y= "PRICE" data=data),

x = data.drop ("PRICE", axis=1)

y= data [“PRICE”]

from sklearn.modelselection import train\_test\_split

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

random-state-42)

from sklearn.linear\_model import LinearRegression

lm=LinearRegression()

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

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

print (lm intercept)

print (lm. coef)

plt. scatter (Y-test, 4-pered)

sns histplot (Y-test- y pued), bins= 50)

from sklean import metrics

print ('MAE :', metrics .mean\_absolute\_error (y\_test, y\_pred)

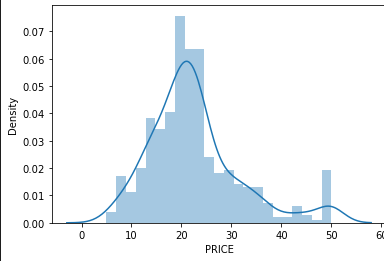
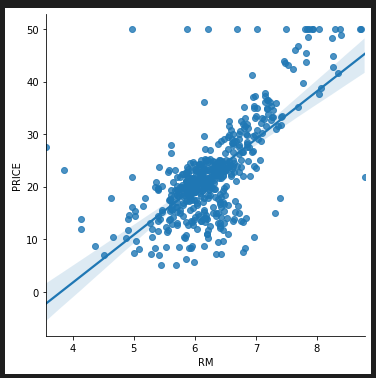
print ("MSE :', metrics. mean\_squared\_error (y\_test, y\_pred))

print ('RSME:', np. sqrt (metrics. mean \_squared\_error (y\_test, y\_pred))

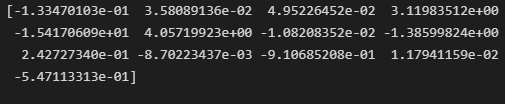
**CONCLUSION**:

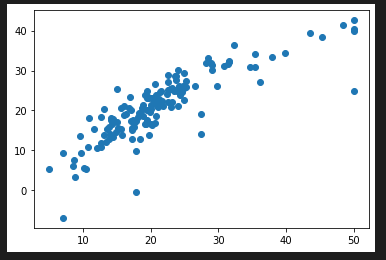
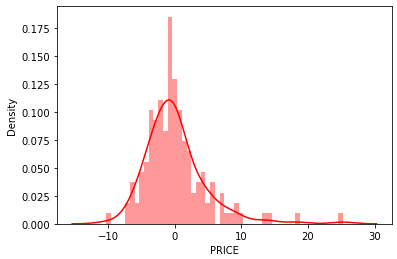
Linear Regression is a supervised machine learning algorithm. Through this experiment we have used Boston Housing Price Prediction Dataset and predicted housing prices using linear regression model.

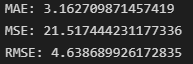
**OUTPUT**:











# Experiment 7: To perform grouping of flower varieties on the iris dataset using K-Nearest Neighbors

**AIM :** To perform grouping of flower varieties on the iris dataset using K-Nearest Neighbors

**THEORY :**

K- Nearest Neighbour: KNN is a machine learning algorithmn based on Supervised learning technique. It 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 It stores all the available data and classifies a new data point based on similarity. KNN can be used for Regression as well as classification. KNN is non-parametric algorithmn. It is also called 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 action on the dataset.

Steps

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

**SOURCE CODE:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

from sklearn import preprocessing

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

iris=pd.read\_csv('Iris.csv')

X = preprocessing.StandardScaler().fit\_transform(X)

X[0:4]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1)

knnmodel=KNeighborsClassifier(n\_neighbors=3)

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

y\_predict1=knnmodel.predict(X\_test)

from sklearn.metrics import accuracy\_score

acc=accuracy\_score(y\_test,y\_predict1)

acc

from sklearn.metrics import confusion\_matrix

cm= confusion\_matrix(y\_test.values,y\_predict1)

prediction\_output=pd.DataFrame(data=[y\_test.values,y\_predict1],index=['y\_test','y\_predict1'])

Ks=21

mean\_acc=np.zeros((Ks-1))

for n in range(1,Ks):

    neigh=KNeighborsClassifier(n\_neighbors=n).fit(X\_train,y\_train)

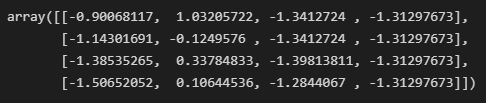
    yhat=neigh.predict(X\_test)

    mean\_acc[n-1]=accuracy\_score(y\_test,yhat)

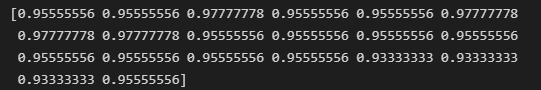
print("The best accuracy was with", mean\_acc.max(),"with k=", mean\_acc.argmax()+1)

**CONCLUSION:** KNN is a supervised machine learning algorithm mostly used for classification problems. Through this experiment, we have implemented KNN model to predict classes of the iris dataset.

**OUTPUT:**









# Experiment 8: To perform grouping of flowers into flower species on the iris dataset using k-means clustering

**AIM:**  To perform grouping of flowers into flower species on the iris dataset using k-means clustering

**THEORY:**

K-means Clustering Algorithm

K-means clustering is an unsupervised learning algorithm. that is used to solve the clustering problem in machine learning and, Here K defines the number of pre-defined cluster that need to be created in the process because as if K= 2, there will be two clusters and for k = 3 , there will be three clusters and so on.

It is an iterative algorithm. that divides the unlabelled. dataset into K. different clusters in such a way that each dataset belongs only one group that has similar properties. It allows us to cluster the data into different groups in the unlabelled dataset on its own without need. It is a centroid based algorithm, where each cluster in associated with centroid. The main aim of this algorithm is to minimize the sum of distance, between the data paint and corresponding clusters.

**CODE:**

* from sklearn.datasets import load\_iris

• iris = load\_iris()

• iris.data

• iris.target

• from sklearn.cluster import KMeans

• kmeans = KMeans(n\_clusters = 3)

• kmeans

• KMmodel = kmeans.fit(iris.data)

• KMmodel.labels\_

• KMmodel.cluster\_centers\_

• import pandas as pd

• pd.crosstab(iris.target,KMmodel.labels\_)

• data = pd.DataFrame(iris.data)

• data.describe()

• data.head()

• iris.target

**CONCLUSION:**

K-Means Clustering is an unsupervised machine learning algorithm. It is a centroid-based algorithm. Through this experiment, we have performed grouping of flower species using iris dataset.

**OUTPUT:**

Text

Description automatically generated

Calendar

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

A picture containing text

Description automatically generated

# Experiment 9: To perform grouping of flower varieties on the iris dataset using hierarchical clustering

**AIM :** To perform grouping of flower varieties on the iris dataset using hierarchical clustering

**THEORY :**

Hierarchical Clustering: It is a method that works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data points as a seperate cluster. In the Hierarchical clustering, the aim is to produce a hierarchical series of nested clusters. A diagram is called Dendogram (a tree are diagram that statistics the Sequence of merge or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or clusters are break up (top-down)

Types of Hierarchical Clustering

•Agglomerative clustering

Initially every data point as an individual cluster and at every step, merge the nearest pairs of the cluster. It is a bottom-up method. A structure that is more informative than the unstructured set of clusters returned by the flat clustering. This clustering algorithm does not require us to prespecify the number of clusters.

2. Divisive clustering

Also known as top-down approach. This algorithm also does not require to prespecify the number of clusters. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.

**SOURCE CODE:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import AgglomerativeClustering

iris=pd.read\_csv('Iris.csv')

iris.head()

iris.drop(['Species'],axis=1,inplace=True)

sns.pairplot(iris)

iris=iris[['PetalLengthCm','PetalWidthCm']]

sns.scatterplot(x='PetalLengthCm',y='PetalWidthCm', data=iris)

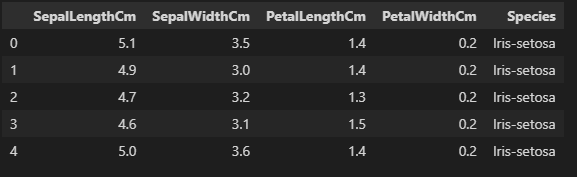
groups=AgglomerativeClustering(n\_clusters=3, affinity='euclidean',linkage='ward')

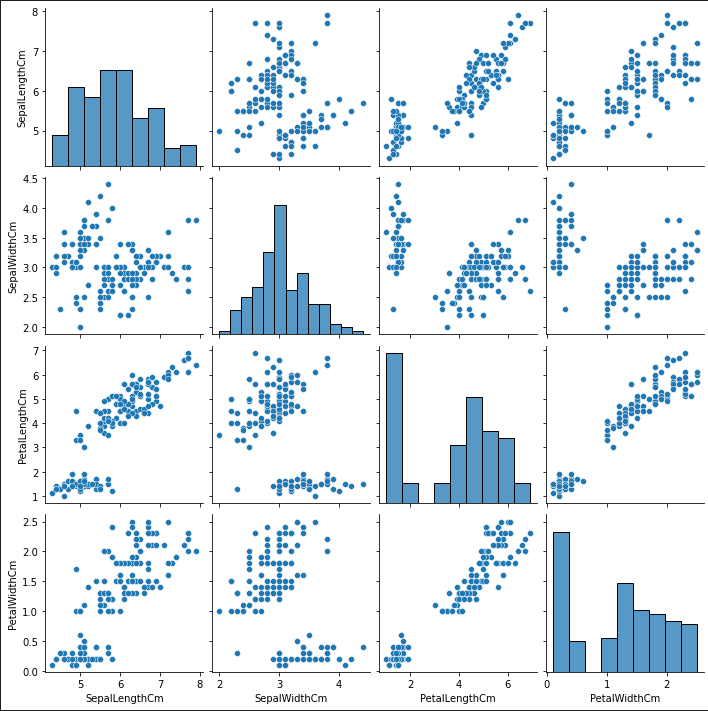
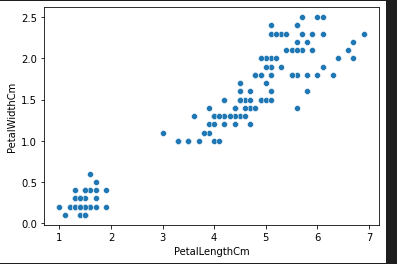
groups.fit\_predict(iris)

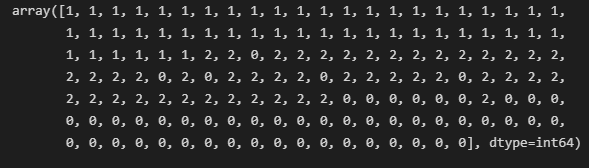
plt.scatter(iris['PetalLengthCm'], iris['PetalWidthCm'],c=groups.labels\_,cmap='autumn')

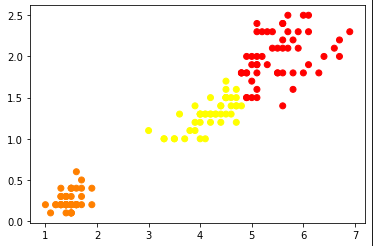
**CONCLUSION:** Hierarchical groups data into tree of clusters. A Dendrogram graphically represents this hierarchy. Agglomerative and Divisive are the two types of Hierarchical clustering. Through this experiment we have grouped flowers species on the iris data using hierarchical clustering.

**OUTPUT**:









# Experiment 10: To predict whether a customer will default or not on the credit card dataset using Logistic Regression, Decision Trees and Naive Bayes Classifier.

**AIM :** To predict whether a customer will default or not on the credit card dataset using logistic Regression, Decision Trees and Naive Bayes Classifier

**THEORY**

**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 used for solving the classification problems.**

**Decision Trees:** Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

**Naïve** **Bayes**: Naïve 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. Naïve 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.

**SOURCE CODE:**

import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
%matplotlib inline

data = pd.read\_csv('Credit\_Card.csv', index\_col = "ID")  
data.rename(columns = lambda x: x.lower(),inplace = True)  
data['grad\_school'] = (data['education'] == 1).astype('int')  
data['university'] = (data['education'] == 2).astype('int')  
data['high\_school'] = (data['education'] == 3).astype('int')  
data.drop('education', axis = 1, inplace = True)  
  
data['male'] = (data['sex'] == 1).astype('int')  
data.drop('sex', axis=1, inplace=True)  
  
data['married'] = (data['marriage'] == 1).astype('int')  
data.drop('marriage', axis = 1, inplace = True)  
  
pay\_features = ['pay\_0','pay\_2','pay\_3','pay\_4','pay\_5','pay\_6']  
for p in pay\_features:  
    data.loc[data[p] <= 0, p] = 0  
  
data.rename(columns = {'default.payment.next.month':'default'}, inplace = True)

from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix,precision\_recall\_curve  
from sklearn.preprocessing import RobustScaler

target\_name  = data['default']  
X = data.drop(columns = ['default'], axis = 1)  
robust\_scaler = RobustScaler()  
X = robust\_scaler.fit\_transform(X)  
y = target\_name  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.15, random\_state = 123, stratify = y)

def CMatrix(CM, labels = ['pay', 'default']):  
    df = pd.DataFrame(data = CM, index = labels, columns = labels)  
    df.index.name = 'TRUE'  
    df.columns.name = 'PREDICTION'  
    df.loc['Total'] = df.sum()  
    df['Total'] = df.sum(axis = 1)  
    return df

metrics = pd.DataFrame(index = ['accuracy', 'precision', 'recall'],  
                      columns = ['NULL', 'Logistic', 'ClassTree', 'NaiveBayes'])

y\_pred\_test  = np.repeat(y\_train.value\_counts().idxmax(), y\_test.size)  
metrics.loc['accuracy', 'NULL'] = accuracy\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
metrics.loc['precision', 'NULL'] = precision\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
metrics.loc['recall', 'NULL'] = recall\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
  
CM = confusion\_matrix(y\_pred = y\_pred\_test, y\_true = y\_test)  
CMatrix(CM)

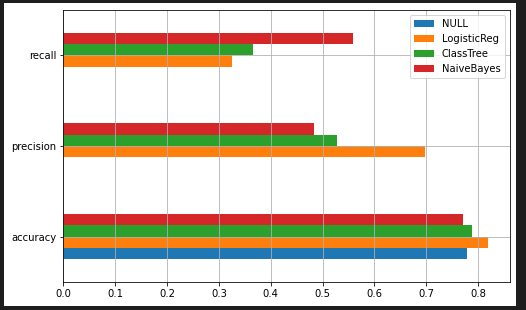
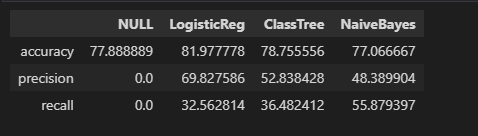
from sklearn.linear\_model import LogisticRegression  
logistic\_regression = LogisticRegression(n\_jobs = -1, random\_state = 15)  
logistic\_regression.fit(X\_train, y\_train)  
  
y\_pred\_test = logistic\_regression.predict(X\_test)  
metrics.loc['accuracy','Logistic'] = accuracy\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
metrics.loc['precision','Logistic'] = precision\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
metrics.loc['recall','Logistic'] = recall\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
  
CM = confusion\_matrix(y\_pred = y\_pred\_test, y\_true = y\_test)  
CMatrix(CM)

from sklearn.tree import DecisionTreeClassifier  
  
class\_tree = DecisionTreeClassifier(min\_samples\_split = 30, min\_samples\_leaf = 10, random\_state = 10)  
class\_tree.fit(X\_train, y\_train)  
  
y\_pred\_test = class\_tree.predict(X\_test)  
metrics.loc['accuracy','ClassTree'] = accuracy\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
metrics.loc['precision','ClassTree'] = precision\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
metrics.loc['recall','ClassTree'] = recall\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
  
CM = confusion\_matrix(y\_pred = y\_pred\_test, y\_true = y\_test)  
CMatrix(CM)

from sklearn.naive\_bayes import GaussianNB  
NBC = GaussianNB()  
NBC.fit(X\_train, y\_train)  
  
y\_pred\_test = NBC.predict(X\_test)  
metrics.loc['accuracy','NaiveBayes'] = accuracy\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
metrics.loc['precision','NaiveBayes'] = precision\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
metrics.loc['recall','NaiveBayes'] = recall\_score(y\_pred = y\_pred\_test, y\_true = y\_test)  
  
CM = confusion\_matrix(y\_pred = y\_pred\_test, y\_true = y\_test)  
CMatrix(CM)

fig, ax = plt.subplots(figsize = (8,5))  
metrics.plot(kind = 'barh', ax = ax, color = ['red','gold','lawngreen','dodgerblue'])  
ax.grid()

**CONCLUSION:** Through this experiment we have studied about three supervised learning algorithms, i.e. Logistic Regression, Decision Tree Classifier and Naive Bayes Classifier and implemented them to predict whether a customer will default or not on the credit card and performed comparison for the same.

 **OUTPUT:**

# MINI PROJECT:

**Topic: Detecting Phishing websites**

1. import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

1. data = pd.read\_csv("phishing.csv")
2. data.columns

Index(['UsingIP','LongURL','ShortURL','Symbol@','Redirection//', 'PrefixSuffix-','SubDomains','HTTPS','DomainRegLen','Favicon',

'NonStdPort','HTTPSDomainURL','RequestURL','AnchorURL',

'LinksInScriptTags','ServerFormHandler','EmailInformation',

'AbnormalURL','WebsiteForwards','StatusBarCust',

'DisableRightClick','UsingPopupWindow','IframeRedirection',

'AgeofDomain','DNSRecord','WebsiteTraffic','PageRank',

'GoogleIndex','LinksPointingToPage','StatsReport','Class'],

dtype='object')

1. data.shape

(11054, 31)

1. data.isnull().sum()

UsingIP 0

LongURL 0

ShortURL 0

Symbol@ 0

Redirection// 0

PrefixSuffix- 0

SubDomains 0

HTTPS 0

DomainRegLen 0

Favicon 0

NonStdPort 0

HTTPSDomainURL 0

RequestURL 0

AnchorURL 0

LinksInScriptTags 0

ServerFormHandler 0

EmailInformation 0

AbnormalURL 0

WebsiteForwards 0

StatusBarCust 0

DisableRightClick 0

UsingPopupWindow 0

IframeRedirection 0

AgeofDomain 0

DNSRecord 0

WebsiteTraffic 0

PageRank 0

GoogleIndex 0

LinksPointingToPage 0

StatsReport 0

Class 0

dtype: int64

1. data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 11054 entries, 0 to 11053

Data columns (total 31 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 UsingIP 11054 non-null int64

1 LongURL 11054 non-null int64

2 ShortURL 11054 non-null int64

3 Symbol@ 11054 non-null int64

4 Redirection// 11054 non-null int64

5 PrefixSuffix- 11054 non-null int64

6 SubDomains 11054 non-null int64

7 HTTPS 11054 non-null int64

8 DomainRegLen 11054 non-null int64

9 Favicon 11054 non-null int64

10 NonStdPort 11054 non-null int64

11 HTTPSDomainURL 11054 non-null int64

12 RequestURL 11054 non-null int64

13 AnchorURL 11054 non-null int64

14 LinksInScriptTags 11054 non-null int64

15 ServerFormHandler 11054 non-null int64

16 EmailInformation 11054 non-null int64

17 AbnormalURL 11054 non-null int64

18 WebsiteForwards 11054 non-null int64

19 StatusBarCust 11054 non-null int64

20 DisableRightClick 11054 non-null int64

21 UsingPopupWindow 11054 non-null int64

22 IframeRedirection 11054 non-null int64

23 AgeofDomain 11054 non-null int64

24 DNSRecord 11054 non-null int64

25 WebsiteTraffic 11054 non-null int64

26 PageRank 11054 non-null int64

27 GoogleIndex 11054 non-null int64

28 LinksPointingToPage 11054 non-null int64

29 StatsReport 11054 non-null int64

30 Class 11054 non-null int64

dtypes: int64(31)

memory usage: 2.6 MB

1. data.describe()
2. X = data.drop(columns = 'Class')

X.head()

1. Y = data['Class']

Y = pd.DataFrame(Y)

Y.head()

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

sns.histplot(data);

Chart

Description automatically generated

1. data.hist(bins=50,figsize=(15,20),color='red')

plt.show()

Calendar

Description automatically generated

Calendar

Description automatically generated

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

sns.kdeplot(data=data,cut=10)

Chart, histogram

Description automatically generated

1. plt.figure(figsize = (20, 20))

sns.heatmap(data.corr(), xticklabels=False,yticklabels=False,cmap="Blues",linewidths = 1, linecolor = "white")

plt.show()

A computer screen capture

Description automatically generated with low confidence

1. def PlotCorrHeatmap(data, idx\_s, idx\_e):

y = data['Class']

temp = data.iloc[:, idx\_s:idx\_e]

temp['Class'] = y

sns.set(rc = {'figure.figsize':(10,5)})

sns.heatmap(temp.corr(), annot=True, fmt='.2f', cmap="Blues", linewidths=1, linecolor="white")

plt.show()

1. PlotCorrHeatmap(data, 0, 10)

Table

Description automatically generated

1. PlotCorrHeatmap(data, 10, 20)

Table

Description automatically generated

1. PlotCorrHeatmap(data, 20, 30)

Table

Description automatically generated

1. sns.set(style = 'whitegrid')

data.boxplot(figsize = (50, 5), color = 'red')

Graphical user interface, application

Description automatically generated

1. from sklearn.model\_selection import train\_test\_split,cross\_val\_score
2. train\_X, test\_X, train\_Y, test\_Y = train\_test\_split(X, Y, test\_size = 0.3)
3. print(train\_X.shape)

print(test\_X.shape)

print(train\_Y.shape)

print(test\_Y.shape)

(7737, 30)

(3317, 30)

(7737, 1)

(3317, 1)

1. from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn import metrics

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

1. ML\_Model = []

Accuracy = []

def storeResults(model, accuracy):

ML\_Model.append(model)

Accuracy.append(round(accuracy,2))

1. def plotMatrix(test\_Y, predict\_y):

C = confusion\_matrix(test\_Y, predict\_y)

plt.figure(figsize=(5,3))

labels = [1,2]

cmap=sns.light\_palette("deepskyblue")

sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)

plt.xlabel('Predicted Class')

plt.ylabel('Original Class')

plt.title("Confusion Matrix")

plt.show()

1. lr = LogisticRegression()

model1 = lr.fit(train\_X, train\_Y)

lrPredict = model1.predict(test\_X)

1. lrAccuracy = accuracy\_score(lrPredict, test\_Y)

lrAccuracy

0.929454326198372

1. print(classification\_report(lrPredict, test\_Y))

Table

Description automatically generated with medium confidence

1. plotMatrix(test\_Y, lrPredict)

Table

Description automatically generated with medium confidence

1. storeResults('LogisticRegression', lrAccuracy)
2. knn = KNeighborsClassifier(n\_neighbors = 3)

model2 = knn.fit(train\_X, train\_Y)

knnPredict = model2.predict(test\_X)

1. knnAccuracy = accuracy\_score(knnPredict,test\_Y)

knnAccuracy

0.9436237564063913

1. print(classification\_report(knnPredict, test\_Y))

A picture containing table

Description automatically generated

1. plotMatrix(test\_Y, knnPredict)

A picture containing table

Description automatically generated

1. storeResults('KNeighborsClassifier', knnAccuracy)
2. tree = DecisionTreeClassifier()

model3 = tree.fit(train\_X,train\_Y)

treePredict = model3.predict(test\_X)

1. treeAccuracy = accuracy\_score(treePredict, test\_Y)

treeAccuracy

0.9586976183298161

1. print(classification\_report(treePredict, test\_Y))

Table

Description automatically generated with low confidence

1. plotMatrix(test\_Y, treePredict)

Table

Description automatically generated with low confidence

1. storeResults('DecisionTreeClassifier', treeAccuracy)
2. rf = RandomForestClassifier()

model4 = rf.fit(train\_X, train\_Y)

rfPredict = model4.predict(test\_X)

1. rfAccuracy = accuracy\_score(rfPredict, test\_Y)

rfAccuracy

0.9686463671992764

1. print(classification\_report(rfPredict, test\_Y))

A picture containing table

Description automatically generated

1. plotMatrix(test\_Y, rfPredict)

A picture containing table

Description automatically generated

1. storeResults('RandomForestClassifier', rfAccuracy)
2. svc = SVC()

model5 = svc.fit(train\_X, train\_Y)

svcPredict = model5.predict(test\_X)

1. svcAccuracy = accuracy\_score(svcPredict, test\_Y)

svcAccuracy

0.9517636418450407

1. print(classification\_report(svcPredict, test\_Y))

Table

Description automatically generated with medium confidence

1. plotMatrix(test\_Y, svcPredict)

Table

Description automatically generated with medium confidence

1. storeResults('SupportVectorMachine', svcAccuracy)
2. print('Logistic Regression Accuracy:',accuracy\_score(lrPredict,test\_Y))

print('K-Nearest Neighbour Accuracy:',accuracy\_score(knnPredict,test\_Y))

print('Decision Tree Classifier Accuracy:',accuracy\_score(treePredict,test\_Y))

print('Random Forest Classifier Accuracy:',accuracy\_score(rfPredict,test\_Y))

print('support Vector Machine Accuracy:',accuracy\_score(svcPredict,test\_Y))

Logistic Regression Accuracy: 0.929454326198372

K-Nearest Neighbour Accuracy: 0.9436237564063913

Decision Tree Classifier Accuracy: 0.9586976183298161

Random Forest Classifier Accuracy: 0.9686463671992764

support Vector Machine Accuracy: 0.9517636418450407

1. results = pd.DataFrame({'Model' : ML\_Model, 'Accuracy' : Accuracy})

results

Table

Description automatically generated

1. results.sort\_values(by=['Accuracy'], ascending=False)

Table

Description automatically generated

1. plt.figure(figsize=(5,3))

labels = ['LR', 'KNN', 'DT', 'RF', 'SVM']

plt.bar(ML\_Model, Accuracy, tick\_label = labels,width = 0.5, color = ['gold'])

plt.xlabel('Models')

plt.ylabel('Accuracy')

plt.ylim((0.7,1.0))

plt.show()

Chart, bar chart

Description automatically generated

**CONCLUSION:**

This project uses various Classification Techniques and Compares the accuracy of the Supervised Learning models. It is not possible to eliminate phishing, however the risk and hazard could be reduced. The research is performed on the phishing websites data set. It contains data about 11054 URLs with 30 attributes and a class label. From all the models developed in the experiment, Random Forest Classifier has highest accuracy of 0.97 and followed by Decision Tree Classifier, Support Vector Machine and K-Nearest Neighbour. Logistic Regression with an accuracy of 0.93 is the model with the lowest.