"Data Science Applications Lab"

Lab report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(UGC AUTONOMOUS)

(Affiliated to AU, Approved by AICTE and Accredited by NBA&NAAC)

Sangivalasa- 531162, Bheemunipatnam Mandal, Visakhapatnam Dist (A. P).

(2023–2024)

Submitted by

Dabburi Pushpa Latha

(320126511014)

Under the guidance of

Mrs. G. Vijaya Lakshmi (Assistant Professor)

Dr. A. Anupama (Associate Professor)

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)
(Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)
Sangivalasa, Bheemunipatnam mandal, Visakhapatnam dist.(A.P)

CERTIFICATE

This is to certify that **DABBURI PUSHPA LATHA** studying **IV/IV B.Tech** register no **320126511014** branch **Information Technology** has done **10** no. of experiments during the year **2023-24** in the subject **DATA SCIENCE APPLICATIONS LAB.**

Lecture-in-charge

Head of the Department

INDEX

SL NO:	Date	Name of the experiment	Mapping of the course outcome	Grade	Signature
1.		Introduction to Python LibrariesNumPy, Pandas, Matplotlib, Scikit, Bokeh Data Visualization using Bar Graph, Pie Chart, Box Plot, Histogram, Line Plots, scatter plots Case Study: Analyse sports data and answer the following questions: 1.Which country played the most matches. 2.Top 3 countries who won the most matches. 3.which country played most matches in home ground 4.How was the performance of Sri Lanka Team 5.which toured most foreign country Month in which matches are played	CO-2		
2.		Perform Data exploration and preprocessing in Python. Write a program to compute summary statistics such as mean, median, mode, standard deviation and variance of the given different types of data.	CO-2		
3.		Write a program to demonstrate Regression analysis with residual plots on a given data set.	CO-3		
4.		Write a program to implement the Naive Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.	CO-3		
5.		Implement regularized Linear regression	CO-3		
6.		Build Machine Learning models using Ensembling techniques: Bagging, Stacking and Boosting	CO-1		
7.		Write a program to demonstrate the working of the decision tree-based ID3algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.	CO-3		

SL NO:	Date	Name of the experiment	Mapping of the course outcome	Grade	Signature
8.		Write a Program to implement Logistic Regression on SMS Spam and Ham dataset.	CO-3		
9.		Write a program to implement k-Nearest Neighbour algorithm to classify the diabetes data set. Print both correct and wrong predictions using Python ML library classes.	Co-3		
10.		Write a program to implement k-Means clustering algorithm to cluster the set of data stored in .CV file. Compare the results of various "k" values for the quality of clustering. Determine the value of K using Elbow method.	CO-1		

Program No.01:

Introduction to Python Libraries- NumPy, Pandas, Matplotlib, Scikit, Bokeh Data Visualization using Bar Graph, Pie Chart, Box Plot, Histogram, Line Plots, scatter plots .

Aim: to write a program to Analyse sports data and answer the following questions:
1.Which country played the most matches.
2.Top 3 countries who won the most matches.
3.which country played most matches in home ground
4.How was the performance of Sri Lanka Team
5.which toured most foreign country.

Description:
Pandas is a Python library used for working with data sets. It has functions for analysing, cleaning, exploring, and manipulating data.
Libraries Used:

Pandas:
read_csv(): We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files). In our example, we will be using a CSV file called 'Sportdata.csv'.

data.head(): This function returns the first n rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it.

data.describe(): Generate descriptive statistics. Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.

Pandas Index.value_counts(): function returns object containing counts of unique values. Excludes NA values by default. Note: In Pandas, we can simply access the data similar to arrays like data[0]. Slicing is also possible in Pandas.

Program:

```
import pandas as pd
```

data=pd.read_csv('Sportdata.csv')
data.head(5) data.describe()
data.shape

#country played maximun number of matches
var1=data['Team 1'].value_counts()
var2=data['Team 2'].value_counts()
var1+var2
#Top 3 winners
winner=data['Winner'].value_counts()
print(winner[:3])

#country that played most matches in home ground
filter=(data['Team1']==data.Host_Country)|(data['Team2']==data.Host_County)
home_ground=data[filter].Host_Country.value_counts()
print(home_ground.index.tolist()[0])

#Performance of SriLanka
srilanka=data[(data['Team 1']=="Sri Lanka") | (data['Team 2']=="SriLanka")]
srilanka.Winner.value_counts().plot(kind='pie')

#Team played Most in Foreign Countries
f1=data['Team 1'] != data.Host_Country f2=data['Team 2'] !=
data.Host_Country
df1=data[f1]['Team1'].value_counts()
df2=data[f2]['Team 2'].value_counts()
df1.add(df2).index.tolist()[0]

Output:

	Unnamed: 0	Scorecard	Team 1	Team 2	Margin	Ground	Match Date	Winner	Host_Country	Venue_Team1	Venue_Team2	Innings_Team1	Innings_Team2
0	0	ODI # 1	Australia	England	Winner2ndInning	Melbourne	Jan 5, 1971	Australia	Australia	Home	Away	Second	First
1	1	ODI # 2	England	Australia	Winner2ndInning	Manchester	Aug 24, 1972	England	England	Home	Away	Second	First
2	2	ODI # 3	England	Australia	Winner2ndInning	Lord's	Aug 26, 1972	Australia	England	Home	Away	First	Second
3	3	ODI # 4	England	Australia	Winner2ndInning	Birmingham	Aug 28, 1972	England	England	Home	Away	Second	First
4	4	ODI # 5	New Zealand	Pakistan	Winner1stInning	Christchurch	Feb 11, 1973	New Zealand	New Zealand	Home	Away	First	Second

	Unnamed: 0
count	7494.000000
mean	1877.024019
std	1082.905280
min	0.000000
25%	939.250000
50%	1878.000000
75%	2814.750000
max	3750.000000

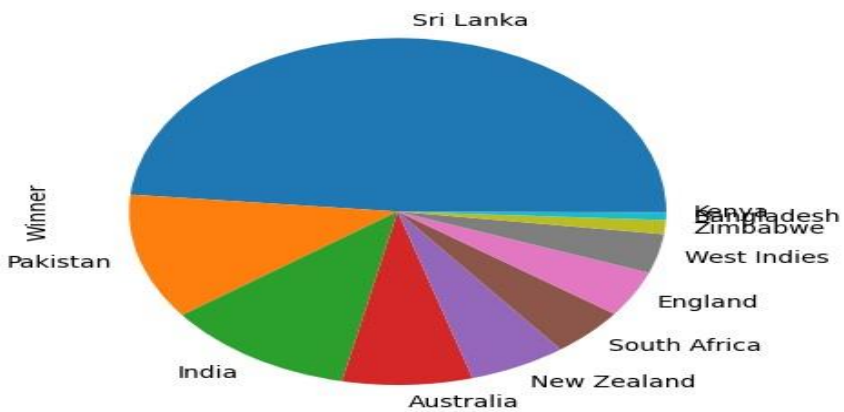
```
(7494, 13)
```

```
India           1760
Australia       1718
Pakistan        1708
Sri Lanka       1532
West Indies     1462
New Zealand     1372
England         1328
South Africa    1122
Zimbabwe         948
Bangladesh       656
Kenya            298
Ireland          226
Scotland         174
Afghanistan      162
Canada           150
Netherlands      144
Bermuda           72
U.A.E.            70
Hong Kong         36
P.N.G.            20
Namibia           16
East Africa        8
U.S.A.             6
dtype: int64
```

```
Australia    1104
India         950
Pakistan      932
Name: Winner, dtype: int64
```

```
Australia
```

```
<Axes: ylabel='Winner'>
```



Program No.02:

Aim: To write a program to Perform Data exploration and pre-processing in Python. Write a program to compute summary statistics such as mean, median, mode, standard deviation and variance of the given different types of data.

Description:

Data exploration is the first step of data analysis used to explore and visualize data to uncover insights from the start or identify areas or patterns to dig into more.

Data preprocessing describes any type of processing performed on raw data to prepare it for another data processing procedure. It has traditionally been an important preliminary step for the data mining process.

Libraries Used:

Pandas: read_csv(): We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files). In our example, we will be using a CSV file called 'test.csv'.

data.describe(): Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.

Numpy: np.mean() ,np.nanmedian() ,np.std() ,np.var():

Statistics: st.mode()

Program:

```
import pandas as pd
import numpy as np
import statistics as st
data=pd.read_csv('test.csv')
b=data.head(5) print(b) data.shape data.describe()
data.drop(['blue','fc','m_dep','pc','sc_h','sc_w'],axis='columns',inplace=True)
target=data.mobile_wt
print(target)
memory=data['int_memory']
mean=np.mean(memory)
median=np.nanmedian(memory)
mode=st.mode(memory)
std=np.std(memory)
var=np.var(memory)
print('Mean is:',mean)
print('Median is:',median)
```

```
print('Mode is:',mode)
print('Standard Deviation:',std)
print('Variance:',var)
data.info()
```

Output:

```
   id  battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
0   1           1043     1          1.8         1  14       0           5
1   2            841     1          0.5         1   4       1          61
2   3           1807     1          2.8         0   1       0          27
3   4           1546     0          0.5         1  18       1          25
4   5           1434     0          1.4         0  11       1          49

   m_dep  mobile_wt  ...  pc  px_height  px_width   ram  sc_h  sc_w  \
0    0.1        193  ...  16        226      1412  3476    12     7
1    0.8        191  ...  12        746       857  3895     6     0
2    0.9        186  ...   4       1270      1366  2396    17    10
3    0.5         96  ...  20        295      1752  3893    10     0
4    0.5        108  ...  18        749       810  1773    15     8

   talk_time  three_g  touch_screen  wifi
0          2        0             1     0
1          7        1             0     0
2         10        0             1     1
3          7        1             1     0
4          7        1             0     1

[5 rows x 21 columns]
0      193
1      191
2      186
3       96
4      108
      ...
995    170
996    186
997     80
998    171
999    140
Name: mobile_wt, Length: 1000, dtype: int64
```

```
Mean is: 33.652
Median is: 34.5
Mode is: 56
Standard Deviation: 18.119627369236923
Variance: 328.32089599999983
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             1000 non-null   int64
 1   battery_power  1000 non-null   int64
 2   clock_speed    1000 non-null   float64
 3   dual_sim       1000 non-null   int64
 4   four_g         1000 non-null   int64
 5   int_memory     1000 non-null   int64
 6   mobile_wt      1000 non-null   int64
 7   n_cores        1000 non-null   int64
 8   px_height      1000 non-null   int64
 9   px_width       1000 non-null   int64
 10  ram            1000 non-null   int64
 11  talk_time      1000 non-null   int64
 12  three_g        1000 non-null   int64
 13  touch_screen   1000 non-null   int64
 14  wifi           1000 non-null   int64
dtypes: float64(1), int64(14)
memory usage: 117.3 KB
```

Program No.03:

Aim: to Write a program to demonstrate Regression analysis with residual plots on a given data set.

Description: Linear regression is the most basic and commonly used predictive analysis. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

Residual Plot: A residual plot shows the difference between the observed response and the fitted response values.

Libraries Used:

Matplotlib:

plt.plot(): Plot y versus x as lines and/or markers

plt.scatter(): A Scatter Plot y versus x as lines and/or markers

plt.xlim(): Get or set the x limits of the current axes.

plt.ylim(): Get or set the y limits of the current axes.

plt.show(): Displays the Plot.

Seaborn:

seaborn.residplot(): Plot the residuals of a linear regression. This function will regress y on x (possibly as a polynomial regression) and then draw a scatterplot of the residuals.

Pandas:

read_csv(): We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files). In our example, we will be using a CSV file called 'cost_revenue_clean.csv'.

data.head(): This function returns the first n rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it.

data.describe(): Generate descriptive statistics. Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.

Scikit-learn:

From sklearn.linear_model we import LinearRegression() to make our model.

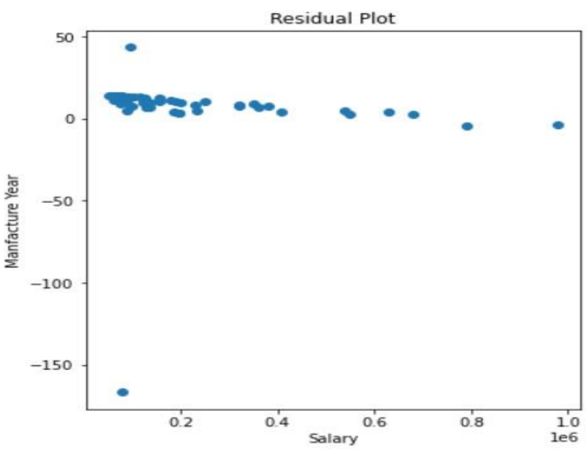
regression.coef_: Estimated coefficients for the linear regression problem. If multiple targets are passed during the fit (y 2D), this is a 2D array of shape (n_targets, n_features), while if only one target is passed, this is a 1D array of length n_features.

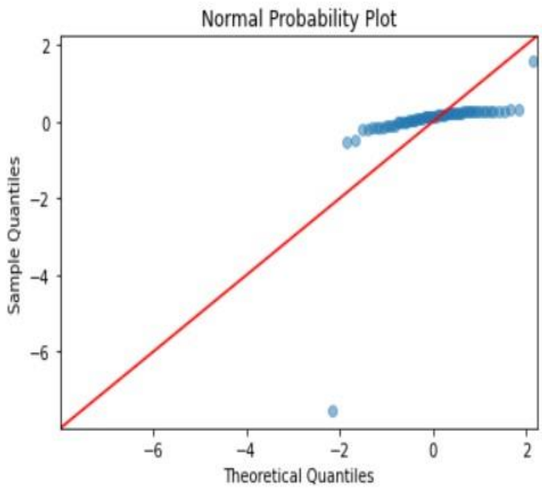
Program:

```
pip install pandas numpy matplotlib seaborn scikit-learn
import pandas as pd import
matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from statsmodels.graphics.gofplots import ProbPlot
# Load the dataset from the Excel file
file_path = "Bike_Price_Prediction.xlsx" # Replace with the actual
file path df = pd.read_excel(file_path)
predictor_column = 'Price'
target_column = 'Manufactured_year'
# Split the dataset into training and testing sets
X = df[[predictor_column]]
y = df[target_column]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_
state=0)
# Fit a linear regression model
model = LinearRegression() model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(X_test, residuals)
plt.xlabel('Salary')
plt.ylabel('Manfacture Year')
plt.title('Residual Plot')
qq = ProbPlot(residuals, fit=True)
plt.subplot(1, 2, 2)
qq.qqplot(line='45', alpha=0.5)
plt.title('Normal Probability Plot')
# Display the plots
plt.show()
X_train_with_const = sm.add_constant(X_train)
model_sm = sm.OLS(y_train, X_train_with_const).fit()
print(model_sm.summary())
```

```
# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Output:






```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Manufactured_year   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                 -0.002
Method:                 Least Squares   F-statistic:                    0.5157
Date:                Thu, 02 Nov 2023   Prob (F-statistic):              0.473
Time:                        17:06:17   Log-Likelihood:                -1519.1
No. Observations:                 246   AIC:                             3042.
Df Residuals:                     244   BIC:                             3049.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2005.0845      9.828    204.027      0.000    1985.727    2024.442
Price       1.705e-05   2.37e-05      0.718      0.473   -2.97e-05    6.38e-05
==============================================================================
Omnibus:                      549.527   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           563706.941
Skew:                         -15.204   Prob(JB):                         0.00
Kurtosis:                     235.532   Cond. No.                     5.46e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.46e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
Mean Squared Error: 578.8272017546644
```

Program No.04:

Aim: to Write a program to implement the Naive Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

Description:

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

Libraries Used:

Scikit-learn:

From Sklearn,we import model_selection to use train_test_split for the purpose of splitting our data set into training and testing data.

From sklearn.preprocessing ,we use StandardScaler.

The Confusion Matrix and accuracy are calculated by importing sklearn.metrics.

Pandas: read_csv(): We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files). In our example, we will be using a CSV file called 'titanic.csv'.

Program:

```
import pandas as pd import numpy as np
data=pd.read_csv('Airlinecsv')
b=data.head(5)
print(b)
data.drop(['First Name','Last Name','Nationality','Airport Name','Airport
Country Code','Country Name','Airport Continent' ,'Continents'
'Departure Date','Arrival Airport','Pilot Name','Flight
Status'],axis='columns',inplace=True)
print(data.head())
#taking out the target column separately
 target = data.Passenger_ID
inputs =data.drop('Passenger_ID',axis='columns')
print(target)
dummies = pd.get_dummies(inputs.Gender)
print(dummies.head(3))
inputs = pd.concat([inputs,dummies],axis='columns')
print(inputs.head(3))
```

```
inputs.drop('Gender',axis='columns',inplace=True)
print(inputs.head(6))
#removing NAN & placing mean value
inputs.columns[inputs.isna().any()]
inputs.Age[:10] inputs.Age =
inputs.Age.fillna(inputs.Age.mean())
print(inputs.head(6))
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(inputs,target,test_size=0.25)
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(x_train,y_train)
print(model.score(x_test,y_test))
#check
print(x_test[:10])
print(y_test[:10])
print(model.predict(x_test[:10]))
```

Output:

```
  Passenger_ID First Name Last Name  Gender  Age Nationality  \
0       ABVWIg     Edithe    Leggis  Female   62       Japan
1       jkXXAX     Elwood      Catt    Male   62   Nicaragua
2       CdUz2g      Darby   Felgate    Male   67      Russia
3       BRS38V   Dominica      Pyle  Female   71       China
4       9kvTLo        Bay   Pencost    Male   21       China

                Airport Name Airport Country Code   Country Name  \
0           Coldfoot Airport                   US  United States
1          Kugluktuk Airport                   CA         Canada
2     Grenoble-Isère Airport                   FR         France
3  Ottawa / Gatineau Airport                   CA         Canada
4            Gillespie Field                   US  United States

  Airport Continent     Continents Departure Date Arrival Airport  \
0               NAM  North America      6/28/2022             CXF
1               NAM  North America     12/26/2022             YCO
2                EU         Europe      1/18/2022             GNB
3               NAM  North America      9/16/2022             YND
4               NAM  North America      2/25/2022             SEE

           Pilot Name Flight Status
0  Fransisco Hazeldine       On Time
1      Marla Parsonage       On Time
2         Rhonda Amber       On Time
3       Kacie Commucci       Delayed
4          Ebonee Tree       On Time
  Passenger_ID  Gender  Age
0       ABVWIg  Female   62
1       jkXXAX    Male   62
2       CdUz2g    Male   67
3       BRS38V  Female   71
4       9kvTLo    Male   21
0        ABVWIg
1        jkXXAX
```

```
   Age  Female  Male
0   62       1     0
1   62       0     1
2   67       0     1
3   71       1     0
4   21       0     1
5   55       1     0
```

```
0.0
       Age  Female  Male
22853   35       0     1
25093   52       1     0
3958    75       0     1
13131   41       0     1
17497   64       0     1
26211   68       1     0
24388   57       1     0
30685   24       1     0
13946   73       0     1
7842    27       1     0
22853    43175
25093    99953
3958     84659
13131    65643
17497    61609
26211    68818
24388    32000
30685    52700
13946    48088
7842     58629
Name: Passenger_ID, dtype: int64
[12926 10069 10845 10493 10676 11648 10345 10459 11630 10192]
```

Program No.05:

Aim: to write a program to Implement regularized Linear regression.

Description: Regression analysis is a set of statistical methods used for the estimation of relationships between a dependent variable and one or more independent variables. It can be utilized to assess the strength of the relationship between variables and for modelling the future relationship between them.

Libraries Used:

Scikit-learn:From sklearn.linear_model we import LinearRegression() to make our model.

sklearn.linear_model.Lasso():Linear Model trained with L1 prior as regularizer (aka the Lasso).

Pandas:

pd.DataFrame: Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure.

read_csv(): We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files). In our example, we will be using a CSV file called 'diabetesds.csv'

df.dropna(): Remove missing values.

df.isna().sum():returns the no.of.missing values in each column.

Program:

```
import pandas as pd
# Load your dataset from a file
file_path = 'your_dataset.csv'
df = pd.read_csv('diabetesds.csv')
# List of attribute names
cols = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
    'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
data = df[cols]
data['SkinThickness'].fillna(data['SkinThickness'].mean(), inplace=True)
data['Insulin'].fillna(data['Insulin'].mean(), inplace=True)
data.isna().sum()
# Drop rows with any remaining missing values
data.dropna(inplace=True)
data.isna().sum()
data=pd.get_dummies(data,drop_first=True)
print(data.head())
```

```
X = data.drop('Outcome', axis=1)
Y = data['Outcome']
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size=0.3,
random_state=2)

from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(train_x, train_y)
linear_reg_score_test = reg.score(test_x, test_y)
linear_reg_score_train = reg.score(train_x, train_y)

from sklearn.linear_model import Lasso
lasso = Lasso(alpha=50, max_iter=100, tol=0.1)
lasso.fit(train_x, train_y)
lasso_score_train = lasso.score(train_x, train_y)
lasso_score_test = lasso.score(test_x, test_y)

print("Linear Regression Score (Test):", linear_reg_score_test)
print("Linear Regression Score (Train):", linear_reg_score_train)
print("Lasso Regression Score (Test):", lasso_score_test)
print("Lasso Regression Score (Train):", lasso_score_train)
```

Output:

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
Linear Regression Score (Test): 0.22466544164835855
Linear Regression Score (Train): 0.3232179537461173
Lasso Regression Score (Test): -0.0036890340363464613
Lasso Regression Score (Train): 0.0
```

Program No.06:

Build Machine Learning models using Ensembling techniques: Bagging, Stacking and Boosting.

Aim: to Build Machine Learning models using Ensembling techniques: Bagging, Stacking and Boosting.

Description:

1)Bagging:

Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset.

2)Boosting:

Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors.

3)Stacking:

Stacking is one of the popular ensemble modeling techniques in machine learning. Various weak learners are ensembled in a parallel manner in such a way that by combining them ,we can predict better predictions for the future.

Libraries Used:

Scikit-learn:

From sklearn, we import model_selection and use Kfold for continuous bootstrapping so that we generate multiple samples with replacement. .

Pandas: We use read_csv() from pandas to read the CSV File.

read_csv(): We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files).

Program:

```
#bagging
import numpy as np
from sklearn.datasets import load_breast_cancer from
sklearn.ensembleimport BaggingClassifier,RandomForestClassifier from
sklearn.model_selection import train_test_split from sklearn.metrics
import accuracy_score # Load the Breast Cancer dataset data =
load_breast_cancer()
X, y = data.data, data.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
base_classifier=RandomForestClassifier(n_estimators=100,random_state
=42)
bagging_classifier = BaggingClassifier(base_estimator=base_classifier,
```

```
n_estimators=10, random_state=42)
 bagging_classifier.fit(X_train, y_train)
 predictions = bagging_classifier.predict(X_test)
 accuracy = accuracy_score(y_test, predictions)
 print('Accuracy:', accuracy)
```

#boosting

```
 import numpy as np
 from sklearn.datasets import load_breast_cancer from sklearn.ensemble
 import BaggingClassifier,
 RandomForestClassifier from sklearn.model_selection import
 train_test_split from sklearn.metrics import accuracy_score data =
 load_breast_cancer()
  X, y = data.data, data.target
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 random_state=42)
 base_classifier=RandomForestClassifier(n_estimators=100,random_state
 =42)
 bagging_classifier =BaggingClassifier(base_estimator=base_classifier,
 n_estimators=10, random_state=42)
  bagging_classifier.fit(X_train, y_train)
  predictions = bagging_classifier.predict(X_test)
  accuracy = accuracy_score(y_test, predictions)
  print('Accuracy:', accuracy)
```

#3)Stacking:

```
 from sklearn import datasets from sklearn.tree
 import DecisionTreeClassifier from sklearn.ensemble import
 RandomForestClassifier from sklearn.neighbors import
 KNeighborsClassifier from sklearn.linear_model import
 LogisticRegression from sklearn.ensemble import StackingClassifier
 import xgboost import pandas as pd from  sklearn.model_selection
 import KFold from sklearn.model_selection import cross_val_score
 df = datasets.load_breast_cancer()
 df.target_names
 X = pd.DataFrame(columns = df.feature_names, data = df.data)
 y = df.target
 X.head()
 X.isnull().sum()
 df.target.shape
 dtc = DecisionTreeClassifier()
 rfc =RandomForestClassifier()
 knn = KNeighborsClassifier()
```

xgb =xgboost.XGBClassifier()
stacking = [dtc,rfc,knn,xgb] for
i in stacking:
score = cross_val_score(i,X,y,cv = 5,scoring = 'accuracy')
print("The accuracy score of {} is:".format(i),score.mean())
dtc = DecisionTreeClassifier() rfc =
RandomForestClassifier()
knn = KNeighborsClassifier()
xgb = xgboost.XGBClassifier()
clf = [('dtc',dtc),('rfc',rfc),('knn',knn),('xgb',xgb)] #list of (str, estimator)
lr = LogisticRegression() stack_model = StackingClassifier(estimators = clf,final_estimator = lr) score = cross_val_score(stack_model,X,y,cv = 5,scoring = 'accuracy')
print("The accuracy score of is:",score.mean())

Output:

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensembl
  warnings.warn(
Accuracy: 0.9649122807017544
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.
  warnings.warn(
Accuracy: 0.9649122807017544
```

```
The accuracy score of DecisionTreeClassifier() is: 0.9138798323241734
The accuracy score of is: 0.9736376339077782
The accuracy score of RandomForestClassifier() is: 0.9631423691973297
The accuracy score of is: 0.9736376339077782
The accuracy score of KNeighborsClassifier() is: 0.9279459711224964
The accuracy score of is: 0.9683589504735288
The accuracy score of XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) is: 0.9701288619779538
The accuracy score of is: 0.9771619313771154
```

Program No.07:

Aim: to Write a program to demonstrate the working of the decision tree based ID3algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Description:

A Decision Tree is a structure that contains nodes (rectangular boxes) and edges(arrows) and is built from a dataset (table of columns representing features/attributes and rows corresponds to records). Each node is either used to make a decision (known as decision node) or represent an outcome .

ID3 uses a top-down greedy approach to build a decision tree. In simple words, the top-down approach means that we start building the tree from the top and the greedy approach means that at each iteration we select the best feature at the present moment to create a node.

Libraries used:

Pandas:

read_csv(): We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files). In our example, we will be using a CSV file called '3- dataset.csv'.

Numpy:

np.unique(): Find the unique elements of an array.Returns the sorted unique elements of an array.

Program:

```
import pandas as pd
import math
import numpy as np
data = pd.read_csv("3-dataset.csv")
features = [feat for feat in data]
features.remove("answer")
class Node:
def __init__(self):
   self.children = [] self.value
   = "" self.isLeaf = False
   self.pred = ""
def entropy(examples):
pos = 0.0 neg = 0.0 for _, row in
examples.iterrows():
   if row["answer"] == "yes":
```

```
        pos += 1 else:
        neg += 1
  if pos == 0.0 or neg == 0.0: return
    0.0
  else:
    p = pos / (pos + neg) n =
    neg / (pos + neg)
    return -(p * math.log(p, 2) + n * math.log(n, 2))
def info_gain(examples, attr):
uniq = np.unique(examples[attr])
#print ("\n",uniq) gain =
entropy(examples) #print
("\n",gain) for u in uniq:
    subdata = examples[examples[attr] == u]
    #print ("\n",subdata) sub_e = entropy(subdata) gain -=
    (float(len(subdata)) / float(len(examples))) * sub_e
    #print ("\n",gain) return
gain
def ID3(examples, attrs):
root = Node() max_gain = 0 max_feat =
"" for feature in attrs: #print
("\n",examples) gain =
info_gain(examples, feature) if gain >
max_gain: max_gain = gain max_feat =
feature root.value = max_feat
#print ("\nMax feature attr",max_feat) uniq =
np.unique(examples[max_feat])
#print ("\n",uniq) for
u in uniq: #print
("\n",u)
    subdata = examples[examples[max_feat] == u]
    #print ("\n",subdata) if entropy(subdata) == 0.0:
    newNode = Node() newNode.isLeaf = True
    newNode.value = u newNode.pred =
    np.unique(subdata["answer"])
    root.children.append(newNode)
    else:
        dummyNode = Node() dummyNode.value =
        u
```

```
        new_attrs = attrs.copy()
        new_attrs.remove(max_feat) child =
        ID3(subdata, new_attrs)
        dummyNode.children.append(child)
        root.children.append(dummyNode)
return root
def printTree(root: Node, depth=0):
for i in range(depth):
   print("\t", end="")
print(root.value, end="") if
root.isLeaf:
   print(" -> ", root.pred)
print() for child in
root.children:
   printTree(child, depth + 1)
root = ID3(data, features)

printTree(root)
```

Output:

```
outlook
        overcast ->  ['yes']

        rain
                wind
                        strong ->  ['no']

                        weak ->  ['yes']

        sunny
                humidity
                        high ->  ['no']

                        normal ->  ['yes']
```

Program No.8:

Aim: to write a program to implement Logistic Regression on SMS Spam and Ham Data.

Description:

Logistic Regression is used when the dependent variable(target) is categorical. Logistic Regression is used when the dependent variable(target) is categorical.

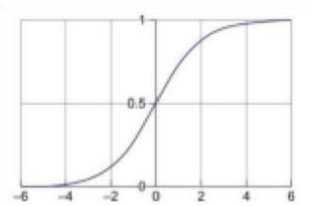
Logistic Regression is used to solve classification problems. Models are trained on historical labelled datasets and aim to predict which category new observations will belong to.

Logistic regression is well suited when we need to predict a binary answer (only 2 possible values like yes or no).

The term logistic regression comes from “Logistic Function,” which is also known as “Sigmoid Function”. The function takes any value and converts it to a number between 0 and 1. The Sigmoid Function is a machine learning activation function that is used to introduce non-linearity to a machine learning model. The formula of Logistic Function is:

$$F(x)= \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

When we plot the above equation, we get S shape curve like below.



The key point from the above graph is that no matter what value of x we use in the logistic or sigmoid function, the output along the vertical axis will always be between 0 and 1. When the result of the sigmoid function is greater than 0.5, we classify the label as class 1 or positive class; if it’s less than 0.5, we can classify it as a negative class or 0.

Libraries Used:

Pandas: We use read_csv() from pandas to read the CSV File. **read_csv():** We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files). In our example, we will be using a CSV file called 'SMSSpamCollection.csv'.

Scikit-learn:

From Sklearn,we import model_selection to use train_test_split for the purpose of splitting our data set into training and testing data.

From sklearn.preprocessing ,we use StandardScaler.

From Scikit-learn.linear_model, we are using LogisticRegression(). The Confusion Matrix ,f1 score and accuracy are calculated by importing sklearn.metrics.

The CountVectorizer method automatically converts all tokenized words to their lower case form so that it does not treat words like 'He' and 'he' differently.

fit_transform( ):

Learn the vocabulary dictionary and return term-document matrix.

transform( ):

Transform documents to document-term matrix.

Program:

```
import pandas as pd
import numpy as np
missing_values = pd.isnull(y_train)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score,
recall_score,f1_score,confusion_matrix data =
pd.read_csv('spam_or_not_spam.csv', sep='\t', names=['label', 'message'])
data.head() data.info() data['label'] = data.label.map({'ham':0, 'spam':1})
data.head()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test=
train_test_split(data['message'],data['label'],test_size =0.25,random_state=1)
print('Number of rows in the total set: {}'.format(data.shape[0]))
print('Number of rows in the training set: {}'.format(X_train.shape[0]))
print('Number of rows in the test set:{}'.format(X_test.shape[0]))
from sklearn.feature_extraction.text import CountVectorizer from
sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
count_vector = CountVectorizer() X_train = [str(item) for
item in X_train] X_test = [str(item) for item in X_test]
```

```
training_data =count_vector.fit_transform(X_train).toarray()
testing_data = count_vector.transform(X_test).toarray()
# Handle missing values in y_train here
LR = LogisticRegression(random_state=0, n_jobs=-1).fit(training_data, y_train)
predictions = LR.predict(testing_data) print('Accuracy score: ',
format(accuracy_score(y_test, predictions)))
print('Precision score: ', format(precision_score(y_test,predictions)))
print('Recall score: ', format(recall_score(y_test, predictions)))
print('F1 score: ', format(f1_score(y_test, predictions)))
print('\nConfusion Matrix :\n', confusion_matrix(y_test,predictions))
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3001 entries, 0 to 3000
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   label    3001 non-null   object
 1   message  0 non-null      float64
dtypes: float64(1), object(1)
memory usage: 47.0+ KB
Number of rows in the total set: 3001
Number of rows in the training set: 2250
Number of rows in the test set: 751
```

```
Accuracy score:  0.9906666666666667
Precision score:  0.9919354838709677
Recall score:  0.9534883720930233
F1 score:  0.9723320158102766

Confusion Matrix :
 [[620   1]
 [  6 123]]
```

Program No.09:

Aim: to Write a program to implement k-Nearest Neighbour algorithm to classify the diabetes data set.

Description:

K-Nearest Neighbours Algorithm is based on Supervised Learning technique. It aims at classifying a new point from the outcomes of earlier points.

Firstly, select the number K of the neighbours. Then, Calculate the Euclidean distance of K number of neighbours. From that, take the K nearest neighbours as per the calculated Euclidean distance. Now, Assign the new data points to that category for which the number of the neighbour is maximum.

Libraries Used:

Scikit-learn:

From Sklearn,we import model_selection to use train_test_split for the purpose of splitting our data set into training and testing data.

From sklearn.preprocessing ,we use StandardScaler.

We import KNeighborsClassifier from skearn.neighbors to apply KNN on our training and test data.

The Confusion Matrix and accuracy are calculated by importing sklearn.metrics.

Pandas: We use read_csv() from pandas to read the CSV File.

read_csv(): We are using read_csv() to read our csv file. A simple way to store big data sets is to use CSV files (comma separated files). In our example, we will be using a CSV file called 'diabetesds.csv'.

Program:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler from
sklearn.neighbors import KNeighborsClassifier from
sklearn.metrics import confusion_matrix from
sklearn.metrics import f1_score from sklearn.metrics
import accuracy_score dataset
=pd.read_csv('diabetesds.csv')
```

```
print(len(dataset)) print(dataset.head()) zero_not_accepted
=['Glucose','BloodPressure','SkinThickness','BMI','Insulin'] for column in
zero_not_accepted:
dataset[column] = dataset[column].replace(0,np.NaN) mean
= int(dataset[column].mean(skipna=True)) dataset[column]
= dataset[column].replace(np.NaN,mean)
#split dataset x = dataset.iloc[:,0:8] y = dataset.iloc[:,8]
print(x,y)
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.25
)
import math
print(len(y_test))
KNN = KNeighborsClassifier(n_neighbors=11, p=2, metric='euclidean')
model=KNN.fit(x_train,y_train) score=KNN.score(x_test,y_test)
print(model,score) #predict the test set results y_pred
=KNN.predict(x_test)
print(y_pred) cm =
confusion_matrix(y_test,y_pred)
print(y_pred)
```

Output:

```
768
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

Program No.10:

Aim: to Write a program to implement k-Means clustering algorithm to cluster the set of data stored in .CV file. Compare the results of various "k" values for the quality of clustering. Determine the value of K using Elbow method.

Description:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process.

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. The k-means clustering algorithm mainly performs two tasks: Determines the best value for K-center points or centroids by an iterative process. Assigns each data point to its closest k center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

Libraries Used:

Scikit-learn:

From sklearn.cluster ,we import KMeans in order to perform KMeans Clustering.

Numpy:

np.random.seed(): Reseed a legacy MT19937 BitGenerator. This is a convenience, legacy function.

np.random.randint(): Return random integers from low (inclusive) to high (exclusive).

np.sqrt(): Computes the square root for input.

Pandas:

pd.DataFrame: Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure.

Matplotlib:

plt.plot(): Plot y versus x as lines and/or markers

plt.scatter(): A Scatter Plot y versus x as lines and/or markers

plt.xlim(): Get or set the x limits of the current axes.

plt.ylim(): Get or set the y limits of the current axes.

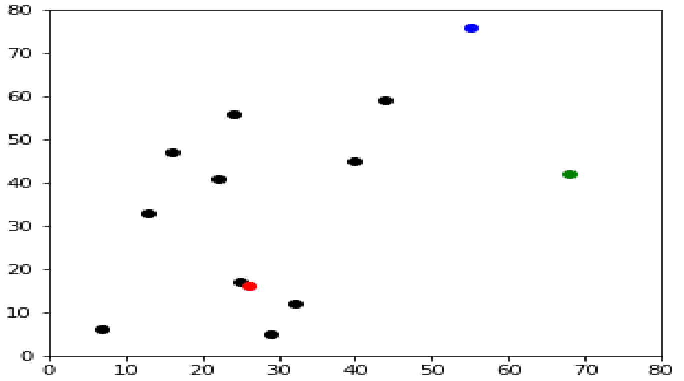
plt.show(): Displays the Plot.

Program:

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
df=pd.DataFrame({'x':[7,44,29,25,13,16,22,40,24,32,16,99],'y':[6,59,5,17,33,47,41,45
,56,12,86,96]})
np.random.seed(200)
k=3
centroids={i+1: [np.random.randint(0,80),np.random.randint(0,80)]for i in range(k)}
fig=plt.figure(figsize=(5,5))
plt.scatter(df['x'],df['y'],color='k')
colmap={1: 'r',2:'g',3:'b'}
for i in centroids.keys():
  plt.scatter(*centroids[i],color=colmap[i])
plt.xlim(0,80)
plt.ylim(0,80)
plt.show()
def assignment(df, centroids):
   for i in centroids.keys():
      df['distance_from_{}'.format(i)]=(np.sqrt((df['x']-centroids[i][0])**2+ (df['y']-
centroids[i][1])**2))
   centroid_distance_cols=['distance_from_{}'.format(i) for i in centroids.keys()]
   df['closest'] = df.loc[:, centroid_distance_cols].idxmin(axis=1)
   df['closest'] = df['closest'].map(lambda x: int(x.lstrip('distance_from_')))
   df['color'] = df['closest'].map(lambda x: colmap[x])
   return df
df = assignment(df,centroids)
print(df.head())
fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
   plt.scatter(*centroids[i], color=colmap[i])
   plt.xlim(0, 80)
   plt.ylim(0, 80)
   plt.show()
df=pd.DataFrame({'x':[7,44,29,25,13,16,22,40,24,32,76,99],'y':[6,59,5,17,33,47,41,45
,56,12,36,96]})
from sklearn.cluster import KMeans
kmeans=KMeans(n_clusters=4)
kmeans.fit(df)
```

OUTPUT:



	x	y	distance_from_1	distance_from_2	distance_from_3	closest	color
0	7	6	21.470911	70.830784	84.876381	1	r
1	44	59	46.615448	29.410882	20.248457	3	b
2	29	5	11.401754	53.758720	75.610846	1	r
3	25	17	1.414214	49.739320	66.189123	1	r
4	13	33	21.400935	55.731499	60.108236	1	r