INDEX

Name : Pratik Pradeep Bagdi Class Roll No.: 29

Branch : Computer Science and Engineering Year / Semester : 3rd Year / 6th Sem

Name of Subject : Professional Skills - II

S.No.	Name of Experiment	Page No.	Date	Remarks
1.	Descriptive Statistics	2-11	10/1/24	A 18/1/24
2.	Reading and Writing different types of datasets	12-17	18/1/24	A 24/1/24
3.	Visualizations	18-25	24/1/24	A 31/1/24
4.	Correlation and Covariance	26-33	31/1/24	A 7/2/24
5.	Regression Model	34-39	7/2/24	A 14/2/24
6.	Multiple Regression Model	40-44	14/2/24	B 13/3/24
7.	Classification Model	46-51	13/3/24	B 20/3/24
8.	Clustering Model	52-57	20/3/24	B 27/3/24
9.	K-Nearest Neighbour Algorithm	58-65	27/3/24	A+ 3/4/24
10.	K-Means Algorithm	66-71	3/4/24	A+ 12/4/24

Practical No - 1

Aim :- Descriptive Statistics

a. Write an program to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets.

b. Write an program to find subset of dataset by using subset(), aggregate() functions on iris dataset.

Theory :-

Descriptive Statistics is about describing and summarizing data. It uses two main approaches:

1. The quantitative approach describes and summarizes data numerically.
2. The visual approach illustrates data with charts, plots, histograms, and other graphs.

You can apply descriptive statistics to one or many datasets or variables. When you describe and summarize a single variable, you're performing univariate analysis. When you search for statistical relationships among a pair of variables, you're doing a bivariate analysis. Similarly, a multivariate analysis is concerned with multiple variable at once.

1. describe() :- The describe() method returns description of the data in the Data Frame. If the Dataframe contains numerical data, the description contains these information of each column:

count - The number of non-empty values.
mean - The average (mean) value.
std - The standard deviation.
min - the minimum value.
25% - The 25% percentile*.

50% - The 50% percentile*.
75% - The 75% percentile*.
max - The maximum value.

2. info() - The info() method prints information about the DataFrame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

3. quantile() - The quantile() method calculates the quantile of the values in a given axis. Default axis is row. By specifying the column axis (axis='columns'), the quantile() method calculates the quantile column-wise and returns the mean value for each row.

4. subset() - The subset function is used to extract a specific subset of the dataset based on certain conditions.

5. groupby() - The 'groupby()' function is a powerful tool for grouping data based on one or more criteria and then applying a function to each group independently.

6. agg() - The 'agg()' function is used to aggregate data in a DataFrame or a Series. The function is used in conjugation with 'groupby()' for more complex data aggregation tasks.

Input :-
Program a :-
Usage :- mtcars.csv

The 'mtcars' dataset is a well-known dataset in the field of statistics data analysis. The dataset provides information about various aspects of different car models, particularly focusing on fuel efficiency and performance.

Format :- A data frame with 32 observations (rows) on 11 numeric variables (columns)

	Variables	Description
1.	mpg	Miles per Gallon
2.	cyl	Number of cylinders
3.	disp	Displacement (cu-in.)
4.	hp	Gross horsepower
5.	drat	Rear axle ratio
6.	wt	Weight (1000 lbs)
7.	qsec	Quatre-mile time
8.	vs	Engine type (0 = V-shaped, 1 = straight)
9.	am	Transmission (0 = automatic, 1 = manual)
10.	gear	Number of Forward Gears
11.	carb	Number of Carburetors

Program 6:-

Usage:- iris.csv

The Iris dataset is a classic dataset in the field of machine learning and statistics. The dataset consists of measurements of various features of three different species of iris flowers.

Format:- A dataframe with 150 observations (rows) on 5 variables (columns).

Output a:-

```
Summary Statistics for mtcars:
              mpg        cyl        disp  ...         am       gear      carb
count   32.000000  32.000000   32.000000  ...  32.000000  32.000000  32.0000
mean    20.090625   6.187500  230.721875  ...   0.406250   3.687500   2.8125
std      6.026948   1.785922  123.938694  ...   0.498991   0.737804   1.6152
min     10.400000   4.000000   71.100000  ...   0.000000   3.000000   1.0000
25%     15.425000   4.000000  120.825000  ...   0.000000   3.000000   2.0000
50%     19.200000   6.000000  196.300000  ...   0.000000   4.000000   2.0000
75%     22.800000   8.000000  326.000000  ...   1.000000   4.000000   4.0000
max     33.900000   8.000000  472.000000  ...   1.000000   5.000000   8.0000

[8 rows x 11 columns]
```

```
General Information for mtcars:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   model   32 non-null     object
 1   mpg     32 non-null     float64
 2   cyl     32 non-null     int64
 3   disp    32 non-null     float64
 4   hp      32 non-null     int64
 5   drat    32 non-null     float64
 6   wt      32 non-null     float64
 7   qsec    32 non-null     float64
 8   vs      32 non-null     int64
 9   am      32 non-null     int64
 10  gear    32 non-null     int64
 11  carb    32 non-null     int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
None
```

```
Quartile Information for mtcars:
      mpg  cyl     disp     hp   drat  ...     qsec   vs   am  gear  carb
0.25  15.425  4.0  120.825   96.5  3.080  ...  16.8925  0.0  0.0   3.0   2.0
0.50  19.200  6.0  196.300  123.0  3.695  ...  17.7100  0.0  0.0   4.0   2.0
0.75  22.800  8.0  326.000  180.0  3.920  ...  18.9000  1.0  1.0   4.0   4.0

[3 rows x 11 columns]
```

Variables	Description
1. sepal_length	Represent length of the iris flowers sepal (outermost whorl of a flower) in cm.
2. sepal_width	Represent width of the sepal of iris flower in cm.
3. petal_length	Represent length of the petal (inner whorl) of the iris flower in cm.
4. petal_width	Represent width of the petal of the iris flower in cm.
5. species	Categorical column represent the species of flower among Setosa, Versicolor and Virginica.

Program a:-

```
import pandas as pd
mtcars = pd.read_csv('mtcars.csv')
print ("Summary Statistics for mtcars:")
print (mtcars.describe())
print ("\n General Information for mtcars:")
print (mtcars.info())
print ("\n Quartile Information for mtcars:")
numeric_columns = mtcars.select_dtypes (include=['number'])
.columns
print (mtcars[numeric_columns].quartile ([0.25, 0.5, 0.75]))
```

Program b:-

Output b:-

Original Iris dataset:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
Subset of Iris Dataset:
0      True
1     False
2     False
3     False
4     False
      ...
145   False
146   False
147   False
148    True
149   False
Length: 150, dtype: bool
```

```
Aggregate result - Mean petal length for each species:
            petal_length
species
setosa             1.464
versicolor         4.260
virginica          5.552
```

Program b:-

```
import pandas as pd
iris = pd.read_csv('iris.csv')
print("Original Iris dataset:")
print(iris.head(1)
subset_condition = (iris['sepal_length'] > 5.0) & (iris['sepal_width'] > 3.0)
print("\n\Subset of Iris Dataset:")
print(subset_condition)
aggregate_result = iris.groupby('species').agg({'petal_length': 'mean'})
print("\n\Aggregate result - Mean petal length for each species:")
print(aggregate_result)
```

Conclusion :- The program to implement descriptive statistics to perform relevant given operations on iris and mtcars dataset is executed successfully.

18/1/24

Practical No :- 2

Aim :- Reading and Writing different types of datasets.

a. Reading different types of data sets (.txt .csv) from Web and desk and writing in file in specific disk location.

b. Reading Excel data sheet in Python.

Theory :-

Data Representation and Storage :- The practical explores various data types (text, CSV, Excel) and their formats and structures. Text files (.txt) store data as plain text, while CSV files (.csv) use commas to seperate values in a tabular format. Excel is a spreadsheet application that stores data in cells organized into rows and columns. Understanding these files formats is crucial for working with different datasets.

Data Reading and Writing in Python :- Python offers libraries like pandas and requests for efficient data access and manipulation. pandas is well-suited for tabular data like CSV and Excel, while requests helps fetch data from websites.

File Handling and Path Management :- The practical delves into opening, reading, writing and processing data files locally or from the web. It's essential to know how to navigate file paths and directories correctly.

Input :-

Program a :-

i) perimeter.txt - It is a simple text document.

ii) cars.csv - It is a dataset consisting of 32 observations (rows) on 11 variables (columns)

Output:-

a)

Txt File from Disk:

This a sample txt doc.

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Odit repellat, ad totam quas consectetur cum hic eaque error perferendis perspiciatis.

CSV File from Disk:

	Unnamed: 0	mpg	cyl	disp	hp	...	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	...	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	...	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	...	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	...	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	...	17.02	0	0	3	2

[5 rows x 12 columns]

Txt File from Web:

The following are the graphical (non-control) characters defined by
ISO 8859-1 (1987). Descriptions in words aren't all that helpful,
but they're the best we can do in text. A graphics file illustrating
the character set should be available from the same archive as this
file.

Hex	Description	Hex	Description
20	SPACE		
21	EXCLAMATION MARK	A1	INVERTED EXCLAMATION MARK
22	QUOTATION MARK	A2	CENT SIGN
23	NUMBER SIGN	A3	POUND SIGN
24	DOLLAR SIGN	A4	CURRENCY SIGN
25	PERCENT SIGN	A5	YEN SIGN
26	AMPERSAND	A6	BROKEN BAR
27	APOSTROPHE	A7	SECTION SIGN
28	LEFT PARENTHESIS	A8	DIAERESIS
29	RIGHT PARENTHESIS	A9	COPYRIGHT SIGN
2A	ASTERISK	AA	FEMININE ORDINAL INDICATOR
2B	PLUS SIGN	AB	LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
2C	COMMA	AC	NOT SIGN

CSV File from Web:

	first_name	last_name	date
0	Alice	Smith	2023-01-05
1	Bobby	Hadz	2023-03-25
2	Carl	Lemon	2021-01-24

iii) txt_url - The given link is of the text document consisting the graphical (non-control) characters defined by ISO 8859-1 (1987)

iv) csv_url - The given link is of the csv dataset consisting of 3 observations (rows) on 3 variables (columns)

Program a:-

```
import pandas as pd
import requests

txt_path = 'perimeter.txt'
csv_path = 'cars.csv'
txt_url = 'https://www.w3.org/TR/2003/REC-PNG-20031110/iso_8859-1.txt'
csv_url = 'https://gist.githubusercontent.com/bobbyhadz/9061dd50a9c0d9
          628592b15632625ff/raw/381229ffc3a72c04066/employees.csv'

perimeter = open(txt_path, 'r')
print("Txt File from Disk: \n\n")
print(perimeter.read())
print('\n')

cars = pd.read_csv(csv_path)
print("CSV File from Disk: \n\n")
print(cars.head())
print('\n')

response = requests.get(txt_url)
peri = response.text
```

b)

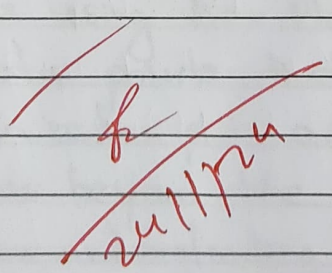
	Sr. No.	Name	DOB
0	1	Rohit	2002-01-01
1	2	Pratham	2001-04-23
2	3	Pratik	2002-10-20
3	4	Shubham	2001-01-07
4	5	Samay	2001-09-08
5	6	Rugved	2000-05-21
6	7	Ajay	2003-06-05
7	8	Amar	2001-02-04

```
print("Txt File from Web: \n\n")
print(pesr)
print('\n')

data = pd.read_csv(csv_url, sep=',', encoding='utf-8',)
print("CSV File from Web: \n\n")
print(data.head())
print('\n')
```

Program 6:-

```
import pandas as pd
excel_file_path = r'C:\Users\prati\OneDrive\Desktop\Datafoorprac.xlsx'
df = pd.read_excel(excel_file_path)
print(df)
```

Conclusion:- The program to read and write different types of datasets is successfully executed.



Practical No :- 3

Aim:- Visualizations
a. Find the data distributions using box and scatter plot.
b. Find the outliers using plot.
c. Plot the histogram, bar chart and pie chart on sample data.

Theory :-

* Data Distributions using Box and Scatter Plots :

a) Box Plot :

A box plot (box-and-whisker) plot is a graphical representation of the distribution of a dataset. It displays the summary of a set of data values, including the minimum, first quartile (Q1), median, third quartile (Q3), and maximum.

b) Scatter Plot:

A scatter plot is a two-dimensional data visualization that uses points to represent individual observations. Each point on the plot represents the values of two variables.

* Finding Outliers using Box-Plot:

Outliers are data points that significantly differ from the rest of the data. In a box plot, outliers are often represented as individual points beyond the whiskers. Identifying outliers is important for understanding the data's overall distribution and detecting any anomalies or errors.

* Histogram, Bar Chart and Pie Chart

a) Histogram:

A histogram is a graphical representation of the distribution of a dataset. It divides the data into bins and displays the frequency of observations in each bin. Histograms are useful

for understanding the underlying probability distribution of a continuous variable.

b) Bar Chart :
A bar chart (or bar graph) presents categorical data with rectangular bars. The lengths of the bars are proportional to the values they represent. Bar charts are commonly used to compare the values of different categories.

c) Pie Chart:
A pie chart is a circular statistical graphic divided into slices to illustrate numerical proportions. Each slice represents a proportionate part of the whole. Pie charts are useful for displaying the composition of a categorical variable as a part of a whole.

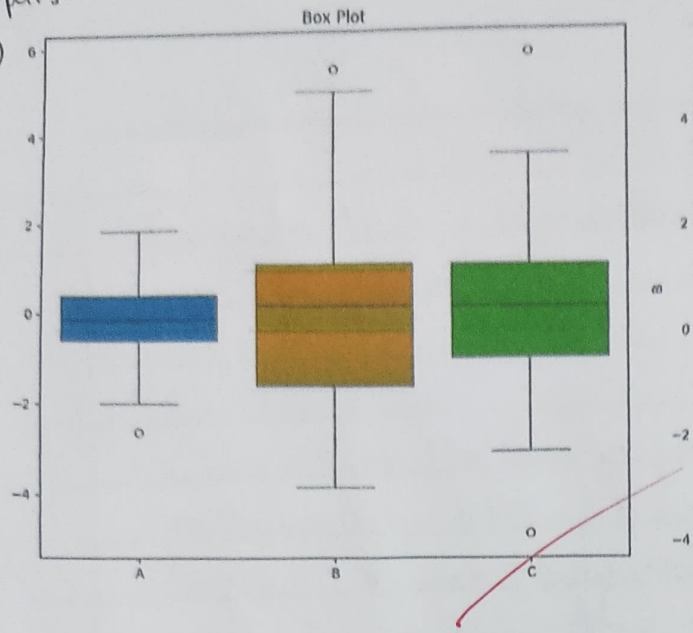
Program :-

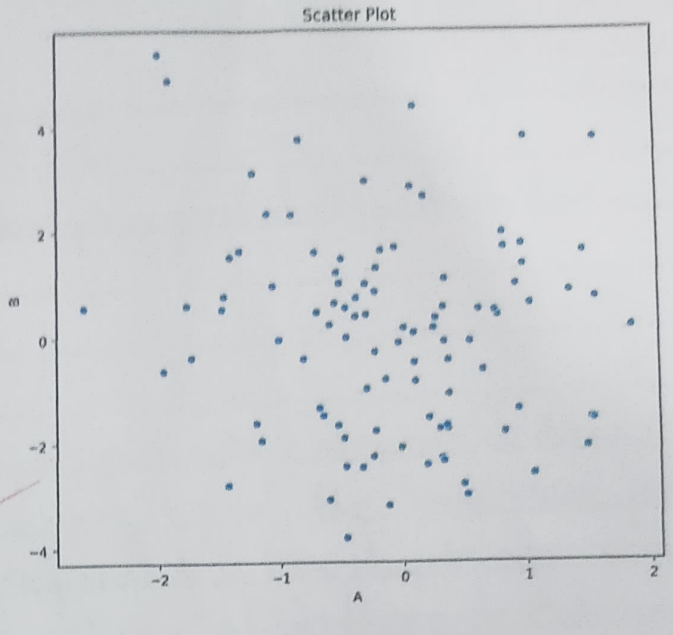
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Generate sample data
np.random.seed(42)
data = {'A': np.random.normal(0, 1, 100),
        'B': np.random.normal(0, 2, 100),
        'C': np.random.normal(0, 1.5, 100)}
df = pd.DataFrame(data)

# a. Find the data distributions using box and scatter plot.
```

```
plt.figure(figsize=(12,6))
plt.subplot(1, 2, 1)
sns.boxplot(data=df)
plt.title('Box Plot')

plt.subplot(1, 2, 2)
sns.scatterplot(x='A', y='B', data=df)
plt.title('Scatter Plot')
plt.tight_layout()
plt.show()

#b. Find the outliers using plot.
plt.figure(figsize=(8,6))
sns.boxplot(data=df, showfliers=True)
plt.title('Outliers Detection')
plt.show()

#c. Plot the histogram, bar chart, and pie chart on sample data
plt.figure(figsize=(15,5))

#Histogram
plt.subplot(1, 3, 1)
sns.histplot(df['A'], bins=20, kde=True)
plt.title('Histogram')

#Bar Chart
plt.subplot(1, 3, 2)
df.sum().plot(kind='bar')
plt.title('Bar Chart')
```

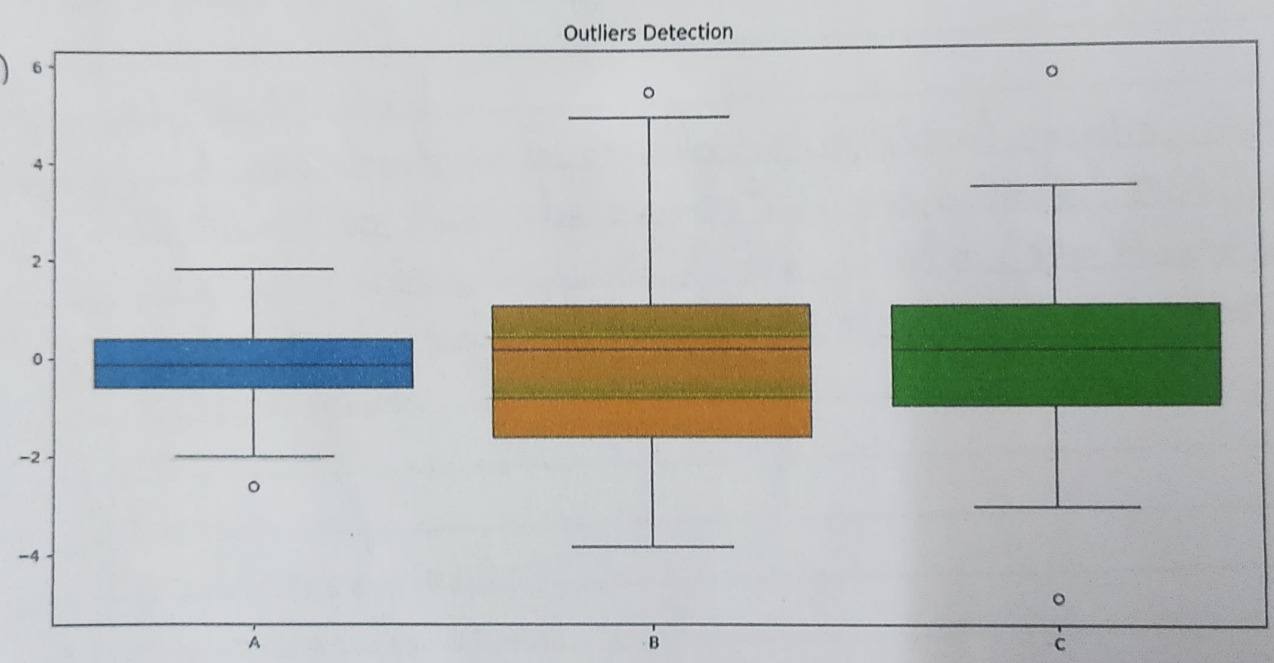
Outputs:-

a)

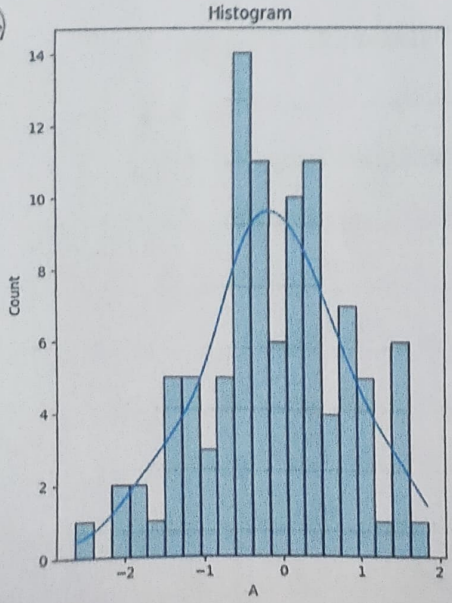

Box Plot
A
B
C

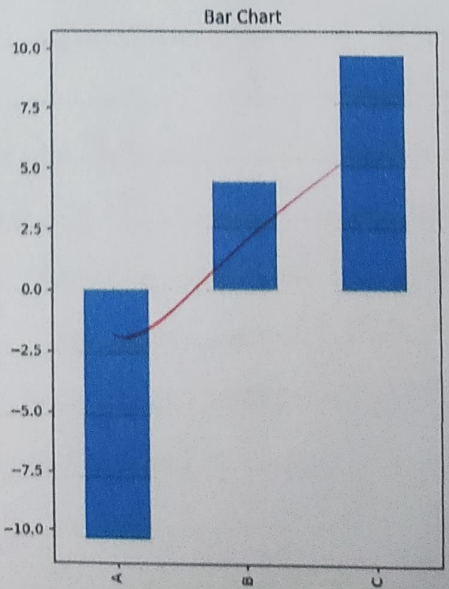

Scatter Plot
A
B

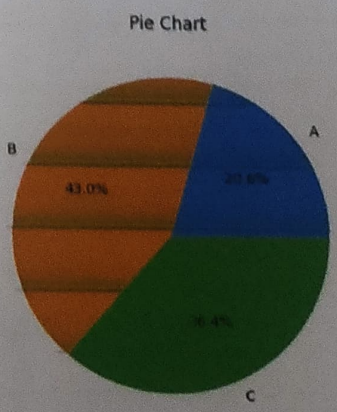
b)


Outliers Detection
A
B
C

c)


Histogram
Count
A


Bar Chart


Pie Chart
A
B
C
43.0%

```
# Pie Chart (using absolute values)
plt.subplot(1, 3, 3)
df.abs().sum().plot(kind='pie', autopct='%.1f%%')
plt.title('Pie chart')

plt.tight_layout()
plt.show()
```

Conclusion :- The program to implement visualizations to plot some of types of plots on sample Dataset performed successfully.

31/1/24

Practical No :- 4

Aim :- Correlations and Covariance
a. Find the correlation matrix
b. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data.
c. Analysis of covariance : variance (ANOVA), if data have categorical variables on iris data.

Theory :-

Correlation :- Correlation measures the statistical association between two variables. It quantifies the strength and direction of a linear relationship between them. The correlation coefficient ranges from -1 to 1. A value of 1 indicates a perfect positive correlation, -1 indicates a pretect negative correlation, and 0 indicates no correlation.

The correlation matrix is calculated using the 'corr()' method in pandas.

Covariance :- Covariance measures how much two variables change together. A positive covariance indicates a positive relationship, while a negative covariance indicates a negative relationship. However, covariance's value is not standardized, making it difficult to interpret.

The covariance matrix is not explicitly calculated in the program, but it can be derived from the correlation matrix.

ANOVA (Analysis of Variance) :- ANOVA is a statistical method used to determine if there are any statistically significant differences between the means of three or more independent groups. In the context

ANOVA is performed to analyze the covariance between each feature of the iris dataset and the categorical variable 'species' (setosa, versicolor, virginica). The F-statistic and p-value obtained from ANOVA help determine whether there are significant differences in the means of the groups.

Input:-
iris.csv - The iris dataset is a well-known dataset in the field of machine learning and statistics. The dataset consists of 150 samples from three different species of iris flowers (setosa, versicolor and virginica). For each species, four features were measured : sepal length, sepal width, petal length, and petal width.

Conclusion :- The program to implement Correlation and covariance and perform ANOVA Test on results implemented successfully.

Program :-

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from scipy.stats import f_oneway

# Load the iris dataset
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target_names[iris.target]

# a. Find the correlation matrix
correlation_matrix = iris_df.corr(numeric_only=True)
print("Correlation Matrix:")
print(correlation_matrix)

# b. Plot the correlation plot
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Plot of Iris Dataset')
plt.show()

# c. Analysis of covariance (ANOVA) for categorical variable 'species'
feature_columns = iris_df.columns[:-1]
for feature in feature_columns:
    anova_results = f_oneway(
        iris_df[feature][iris_df['species'] == 'setosa'],
        iris_df[feature][iris_df['species'] == 'versicolor'],
        iris_df[feature][iris_df['species'] == 'virginica']
    )
    print(f"\nANOVA results for '{feature}':")
    print("F-statistic:", anova_results.statistic)
    print("P-value:", anova_results.pvalue)

# Visualize boxplots for each feature based on species
plt.figure(figsize=(15, 8))
for i, feature in enumerate(feature_columns, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x='species', y=feature, data=iris_df)
    plt.title(f'{feature} vs Species')

plt.tight_layout()
plt.show()
```

Output :-

a. Correlation Matrix:

	sepal length (cm)	...	petal width (cm)
sepal length (cm)	1.000000	...	0.817941
sepal width (cm)	-0.117570	...	-0.366126
petal length (cm)	0.871754	...	0.962865
petal width (cm)	0.817941	...	1.000000

[4 rows x 4 columns]

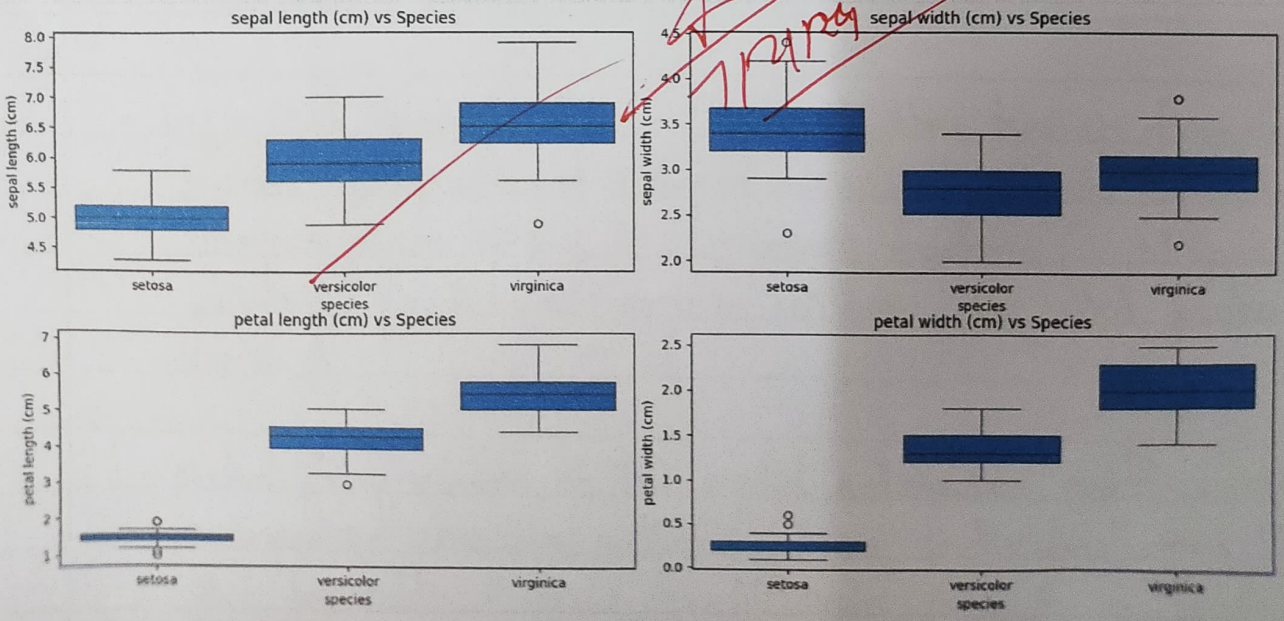
c. ANOVA results for 'sepal length (cm)':
F-statistic: 119.26450218450468
P-value: 1.669669190769383e-31

ANOVA results for 'sepal width (cm)':
F-statistic: 49.160040089612075
P-value: 4.492017133309l1e-17

ANOVA results for 'petal length (cm)':
F-statistic: 1180.161182252981
P-value: 2.856776610961539e-91

ANOVA results for 'petal width (cm)':
F-statistic: 960.007146801809
P-value: 4.1694458394430593e-85

b.



Practical No :- 5

Aim :- Regression Model

Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in a institute based on his or her GRE score, GPA obtained and rank of the student, Also check the model is fit or not. Require (foreign), require (MASS).

Theory :-

Logistic regression is a powerful statistical technique widely used in machine learning for binary classification tasks. It aims to model the relationship between a set of independent variables (e.g., GRE score, GPA, rank) and a binary dependent variable (e.g., admission status) by estimating the probability of one class (in the case, admission) occurring given specific values of the independent variables.

- Binary Classification – Logistic regression is suitable for predicting outcomes with two possible classes (e.g. admitted / not admitted), unlike linear regression, which handles continuous numerical targets.

- Log- Odd Transformation – The core idea lies in transforming the linear combination of independent variables into log-odds of the target class using the sigmoid function (S-shaped curve). This ensures predictions remain within the valid probability range (0 to 1).

- Model Coefficients – The model estimates coefficients for each independent variable, reflecting their individual contributions to the log-odds. Positive coefficients indicate a positive association

Program :-

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import statsmodels.api as sm
# Import data from web storage and name the dataset
url = 'https://raw.githubusercontent.com/pratikbagdi/PS-2-Practical/main/admission.csv'
df = pd.read_csv(url)
df.name = "Admissions_Dataset"
# Checking the first few rows of the dataset and Separating features and target variable
print(df.head())
X = df[['gre', 'gpa', 'rank']]
y = df['admit']
# Adding constant term for logistic regression
X = sm.add_constant(X)
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Fitting logistic regression model
log_reg = sm.Logit(y_train, X_train)
result = log_reg.fit()
# Checking model summary and Predicting on the test set
print(result.summary())
y_pred = result.predict(X_test)
# Converting probabilities to binary predictions
y_pred_binary = [1 if p >= 0.5 else 0 for p in y_pred]
# Calculating accuracy
accuracy = accuracy_score(y_test, y_pred_binary)
print("Accuracy of the logistic regression model:", accuracy)
```

Output :-

```
   admit    gre   gpa  rank
0    0.0  380.0  3.61   3.0
1    1.0  660.0  3.67   3.0
2    1.0  800.0  4.00   1.0
3    1.0  640.0  3.19   4.0
4    0.0  520.0  2.93   4.0
Optimization terminated successfully.
         Current function value: 0.565088
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                  admit   No. Observations:                  320
Model:                          Logit   Df Residuals:                      316
Method:                           MLE   Df Model:                            3
Date:                Thu, 08 Feb 2024   Pseudo R-squ.:                 0.09016
Time:                        11:58:21   Log-Likelihood:                -180.83
converged:                       True   LL-Null:                       -198.75
Covariance Type:            nonrobust   LLR p-value:                 8.100e-08
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -3.4005      1.288     -2.639      0.008      -5.926      -0.875
gre            0.0017      0.001      1.411      0.158      -0.001       0.004
gpa            0.8911      0.371      2.401      0.016       0.164       1.618
rank          -0.6159      0.143     -4.304      0.000      -0.896      -0.335
==============================================================================
Accuracy of the logistic regression model: 0.6625
```

with admission, while negative coefficients suggest an inverse relationship.

Input:-
admission.csv - The dataset consist of 400 observations (rows) on 4 variables (columns) they are admit, gre, gpa and rank.

Conclusion:- The program to implement Regression Model to perform operations on student result and score dataset implemented successfully.

14/02/2024

Practical No :- 6

Aim :- Multiple Regression model
Apply multiple regressions, if data have a continuous Independent variable. Apply on above dataset.

Theory :-
Multiple linear regression is a statistical technique that helps us understand the relationship between multiple independent variables and a dependent variable. It is an extension of simple linear regression, which only considers one independent variable.

Independent Variables – These are the variables that are believed to influence the dependent variable.
For e.g. Avg. area income, Avg. area house age, Avg. area number of rooms, Avg. area number of bedrooms, Area population

Dependent Variables – This is the variable that we are trying to predict or explain.
For eg. price of a house

Linear Relationship – The core assumption of multiple linear regression is that there exists a linear relationship between the independent variables and the dependent variables. This means that the changes in the dependent variable can be modeled as a straight line in relation to the changes in the independent variables.

Steps involved in building a multiple linear regression model
1. Data Collection :- Collect data that includes the independent variables and the dependent variables.

2. Data Preparation :- Clean and pre-process the data to ensure its suitability for the analysis.

3. Model fitting :- Train the model using a training dataset, which helps the model learn the relationship between the independent and dependent variables.

4. Evaluation :- Evaluate the model's performance on a separate testing dataset to assess its generalizability and effectiveness in predicting the dependent variable.

5. Interpretation :- Analyze the model coefficients and intercept to understand the direction and strength of the relationships between the independent variables and the dependent variable.

Input :-

housing.csv - The dataset consist of 5000 observations on 7 variables they are Avg. Area Income, Avg. Area House Age, Avg. Area Number of Rooms, Avg. Area Number of Bedrooms, Area Population, Price, Address.

Conclusion :- The program for Multiple Regression model implemented successfully.

Program :-

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset
df = pd.read_csv('housing.csv')

# Split data into features (X) and target variable (y)
X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population']]
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("R-squared:", r2_score(y_test, y_pred))

mse = mean_squared_error(y_test, y_pred)
print("Mean squared error (MSE):", mse)

print("Model coefficients:", model.coef_)

print("Intercept:", model.intercept_)
```

Output :-

```
R-squared: 0.917997170698532
Mean squared error (MSE): 10089009299.499416
Model coefficients: [2.16522058e+01 1.64666481e+05 1.19624012e+05 2.44037761e+03
 1.52703134e+01]
Intercept: -2635072.900916781
```

13/3/24

Practical No :- 7

Aim :- Classification model
a. Install relevant package for classification.
b. Choose classifier for classification problem.
c. Evaluate the performance of classifier.

Theory :-

Classification Models -
- Classification is a supervised learning task where the model learns from labeled data. This data consists of features (independent variables) and a target variable (dependent variable).
- The model's objective is to learn the mapping between features and the target variable.
- During prediction, the model takes unseen data with only features and predicts the corresponding category for the target variable.

Random Forest Classifier -
- Random Forest belongs to a family of algorithms called ensemble methods. It combines multiple decision trees, where each tree makes a prediction, and the final prediction is the majority vote of all trees.
- This approach helps reduce variance and improve the model's generalizability.

Installation and Imports -
- scikit-learn is a popular python library for machine learning tasks, including classification. The code checks if it's installed and installs if if needed.
- pandas is used for data manipulation.

- train_test_split helps split the data into training and testing sets.
- RandomForestClassifier is the chosen classifier from scikit-learn.
- accuracy_score and classification_report are metrics used to evaluate the model's performance.

Input:-
climate_change_data.csv - The dataset consist of 10,000 observations on 9 variables, they are date, location, Country, temperature, CO_2 emissions, Sea level rise, precipitation, humidity, Wind speed.

Conclusion:- The program to implement classification model by installing relevant package, choosing classifier for classification problem and evaluating the performance of classifier is executed successfully.

Program:-

```
# Install scikit-learn package if not installed
# pip install scikit-learn
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv("climate_change_data.csv")
data = pd.get_dummies(data, columns=['Country'])
# Binning temperature into categories
temperature_bins = [-float('inf'), 10, 20, float('inf')]
temperature_labels = ['Low', 'Medium', 'High']
data['Temperature_Category'] = pd.cut(data['Temperature'], bins=temperature_bins,
labels=temperature_labels)

# Define features (X) and target variable (y)
X = data.drop(['Date', 'Location', 'Temperature', 'Temperature_Category'], axis=1)
y = data['Temperature_Category']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Choose classifier (Random Forest Classifier)
classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier and Make predictions
classifier.fit(X_train, y_train)
predictions = classifier.predict(X_test)

# Evaluate the performance of classifier
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Other classification metrics
print(classification_report(y_test, predictions))
```

29/3/24

Output :-

```
Accuracy: 0.665
              precision    recall  f1-score   support

        High       0.00      0.00      0.00       341
         Low       0.40      0.01      0.01       327
      Medium       0.67      1.00      0.80      1332

    accuracy                           0.67      2000
   macro avg       0.36      0.33      0.27      2000
weighted avg       0.51      0.67      0.53      2000
```

Practical No :- 8

Aim :- Clustering Model
a. Clustering algorithms for unsupervised classification.
b. Plot the cluster data using matplotlib visualizations.

Theory :-
K-Means Clustering is a fundamental unsupervised machine learning algorithm used for grouping data points into a predefined number of clusters (K). It assumes that data points within a cluster are similar to each other and dissimilar to data points in other clusters.

K-means is an iterative algorithm that aims to partition n observations into k clusters. The algorithm works as follows :-

1. Initialization : Randomly initialize k cluster centroids.
2. Assignment : Assign each data point to the nearest cluster centroid based on a distance metric, typically Euclidean distance.
3. Update : Recalculate the cluster centroids as the mean of all data points assigned to each cluster.
4. Repeat steps 2 and 3 until convergence, i.e, until the centroids no longer change significantly or a specified number of iterations is reached.

Matplotlib Visualization
Matplotlib is a Python library used for creating static, interactive, and animated visualizations. In the provided program, Matplotlib is used to visualize the clustered data points. The scatter plot is used to plot the data points, where each point is colored based on its assigned cluster.

Additionally, black squares are used to represent the centroids of each cluster.

Input :-

The code uses the Iris dataset, commonly used for demonstrating clustering algorithms. This dataset consists of 150 samples from three Iris flower species, each described by four features: sepal length, sepal width, petal length and petal width.

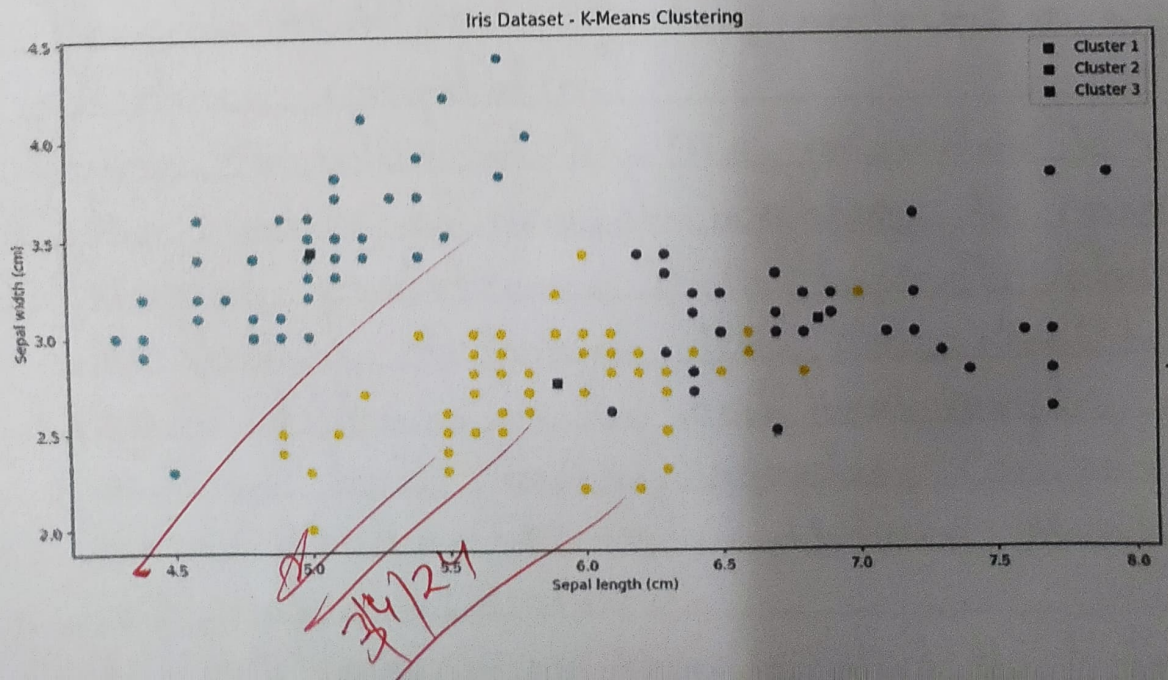
Conclusion :- The program to implement clustering model by using clustering algorithm for unsupervised classification and plot the cluster data using matplotlib visualization is successfully executed.

Program :-

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
iris = load_iris()
data = iris.data
target = iris.target
k = 3
kmeans = KMeans(n_clusters=k, n_init=10)  # Set n_init to 10 explicitly
kmeans.fit(data)
cluster_labels = kmeans.labels_
plt.scatter(data[:, 0], data[:, 1], c=cluster_labels)  # Using first two features for visualization
for i, label in enumerate(kmeans.cluster_centers_):
    plt.scatter(label[0], label[1], color='black', marker='s', label=f'Cluster {i+1}')
plt.title('Iris Dataset - K-Means Clustering')
plt.xlabel('Sepal length (cm)')
plt.ylabel('Sepal width (cm)')
plt.legend()
plt.show()
print("Actual species:")
print(target)
print("Predicted cluster labels:")
print(cluster_labels)
```

Output :-

```
Actual species:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
Predicted cluster labels:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
 0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0
 0 2]
```



Practical No :- 9

Aim :- Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and incorrect predictions.

Theory :-

The k-Nearest Neighbors (KNN) is a simple yet effective supervised learning algorithm used for classification and regression tasks. In classification tasks, it assigns a class label to a sample based on the majority class among its k nearest neighbors. The algorithm operates under the assumption that similar data points tend to belong to the same class.

Key steps of the kNN algorithm:

1. Load the dataset - Obtain a dataset with labeled samples.
2. Preprocess the data - This step involves data cleaning, normalization, and splitting the dataset into training and testing sets. Normalization is important to ensure that each feature contributes equally to the distance computation.
3. Choose the value of k - The parameter k represents the number of nearest neighbors to consider when making predictions. It is typically chosen empirically or through technique like cross-validation.
4. Compute distances - For each test sample, calculate the distance to all training samples. Common distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance.
5. Find the k nearest neighbors - Select the k training samples with the shortest distances to the test sample.

6. Classify the test sample - Assign the class label that is most frequent among the k nearest neighbors. In the case of ties, a simple solution is to choose the class label of the closest neighbor.
7. Evaluate the model - Assess the performance of the model using evaluation metrics such as accuracy, precision, recall and F1-score.

Input :-

iris.csv - The iris dataset is the popular bench-mark dataset in machine learning, containing information about 150 iris flower from three different species: Iris setosa, Iris versicolor, and Iris virginica. Each flower is described by four features: Sepal length(cm), Sepal width (cm), petal length (cm), petal width (cm).

Conclusion :- The program to implement K-Nearest Neighbour algorithm to classify the iris dataset is successfully executed.

Program :-

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=45)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define the kNN classifier
k = 3  # Number of neighbors to consider
knn = KNeighborsClassifier(n_neighbors=k)

# Train the classifier
knn.fit(X_train_scaled, y_train)

# Make predictions
y_pred = knn.predict(X_test_scaled)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print correct and incorrect predictions
correct_predictions = []
incorrect_predictions = []
for i in range(len(y_test)):
    if y_test[i] == y_pred[i]:
        correct_predictions.append((X_test[i], y_test[i], y_pred[i]))
    else:
        incorrect_predictions.append((X_test[i], y_test[i], y_pred[i]))


print("\nCorrect predictions:")
for item in correct_predictions:
    print("Input:", item[0], "| True Label:", iris.target_names[item[1]], "| Predicted Label:",
iris.target_names[item[2]])


print("\nIncorrect predictions:")
for item in incorrect_predictions:
    print("Input:", item[0], "| True Label:", iris.target_names[item[1]], "| Predicted Label:",
iris.target_names[item[2]])
```

Output :-

```
Accuracy: 0.9666666666666667

Correct predictions:
Input: [5.1 3.5 1.4 0.2] | True Label: setosa | Predicted Label: setosa
Input: [5.  3.5 1.6 0.6] | True Label: setosa | Predicted Label: setosa
Input: [7.2 3.  5.8 1.6] | True Label: virginica | Predicted Label: virginica
Input: [4.6 3.1 1.5 0.2] | True Label: setosa | Predicted Label: setosa
Input: [4.9 3.1 1.5 0.2] | True Label: setosa | Predicted Label: setosa
Input: [5.1 3.8 1.9 0.4] | True Label: setosa | Predicted Label: setosa
Input: [4.4 3.  1.3 0.2] | True Label: setosa | Predicted Label: setosa
Input: [7.6 3.  6.6 2.1] | True Label: virginica | Predicted Label: virginica
Input: [6.3 2.7 4.9 1.8] | True Label: virginica | Predicted Label: virginica
Input: [6.7 3.1 5.6 2.4] | True Label: virginica | Predicted Label: virginica
Input: [5.2 3.4 1.4 0.2] | True Label: setosa | Predicted Label: setosa
Input: [7.2 3.2 6.  1.8] | True Label: virginica | Predicted Label: virginica
Input: [5.7 2.5 5.  2. ] | True Label: virginica | Predicted Label: virginica
Input: [6.3 2.9 5.6 1.8] | True Label: virginica | Predicted Label: virginica
Input: [5.  3.2 1.2 0.2] | True Label: setosa | Predicted Label: setosa
Input: [6.7 3.  5.2 2.3] | True Label: virginica | Predicted Label: virginica
Input: [5.8 2.7 5.1 1.9] | True Label: virginica | Predicted Label: virginica
Input: [5.  3.5 1.3 0.3] | True Label: setosa | Predicted Label: setosa
Input: [6.3 2.3 4.4 1.3] | True Label: versicolor | Predicted Label: versicolor
Input: [5.4 3.  4.5 1.5] | True Label: versicolor | Predicted Label: versicolor
Input: [6.  3.4 4.5 1.6] | True Label: versicolor | Predicted Label: versicolor
Input: [6.4 3.2 5.3 2.3] | True Label: virginica | Predicted Label: virginica
Input: [6.4 3.2 4.5 1.5] | True Label: versicolor | Predicted Label: versicolor
Input: [5.  3.6 1.4 0.2] | True Label: setosa | Predicted Label: setosa
Input: [6.8 3.  5.5 2.1] | True Label: virginica | Predicted Label: virginica
Input: [5.8 2.6 4.  1.2] | True Label: versicolor | Predicted Label: versicolor
Input: [5.6 2.9 3.6 1.3] | True Label: versicolor | Predicted Label: versicolor
Input: [5.4 3.7 1.5 0.2] | True Label: setosa | Predicted Label: setosa
Input: [6.1 3.  4.6 1.4] | True Label: versicolor | Predicted Label: versicolor

Incorrect predictions:
Input: [6.3 2.8 5.1 1.5] | True Label: virginica | Predicted Label: versicolor
```

27/3/24

Practical No :- 10

Aim :- Implement K-means algorithm to classify the iris dataset

Theory :- K-means clustering is an unsupervised machine learning algorithm used to partition a dataset into k distinct, non-overlapping clusters. The goal is to minimize the sum of squared distances between data points and their corresponding cluster centroids.

Algorithm steps :

1. Initialization : Randomly selected k data points from the dataset as initial cluster centroids.
2. Assignment : Assign each data point to the nearest centroid, forming k clusters.
3. Update Centroids : Calculate the mean of all data points in each cluster and update the centroids.
4. Repeat steps 2 and 3 until convergence, i.e., until the centroids no longer change significantly or a maximum number of iterations is reached.

Applications of K-means to Iris dataset :-

Data Loading - Load the Iris dataset containing feature vectors representing the characteristics of iris flower.

Preprocessing - No significant preprocessing is required for K-means, as it operates on numerical feature vectors.

K-means Clustering - Apply the K-means algorithm to cluster the iris data points into K-clusters.

Evaluation - Assess the quality of clustering using metrics like Silhouette score or visually inspecting cluster separation.

Visualization :- Visualize the clusters and centroids to interpret the results and gain insights into the data distribution.

Input:-

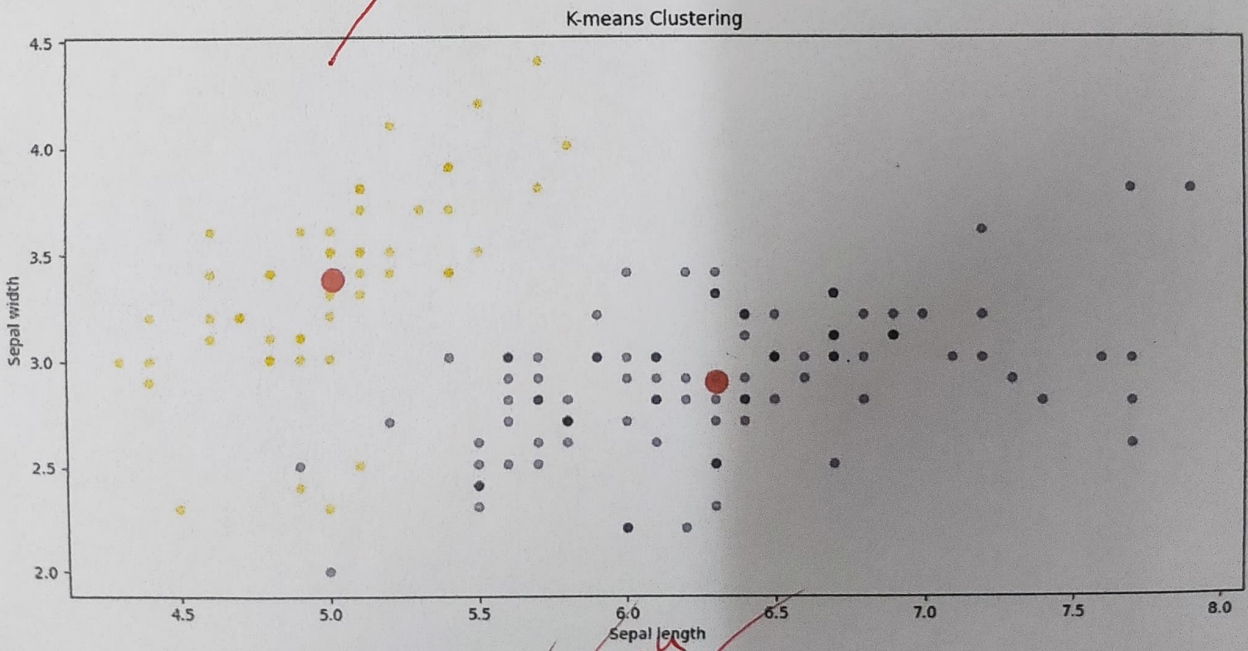
Iris.csv - The Iris dataset is a classic dataset in machine learning and statistics. It consist of 150 samples of iris flowers, each with four features (attributes): sepal length, sepal width, petal length, and petal width. Each sample belongs to one of three species: setosa, versicolor or virginica.

Conclusion:- The program to implement k-means algorithm to classify the iris dataset is successfully executed.

Program :-

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
iris = load_iris()
X = iris.data
def plot_clusters(X, labels, centers):
    plt.scatter(X[:, 0], X[:, 1], c=labels, s=30, cmap='viridis', alpha=0.5)
    plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.8)
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')
    plt.title('K-means Clustering')
    plt.show()
def kmeans_clustering(X, n_clusters):
    kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=42)
    labels = kmeans.fit_predict(X)
    centers = kmeans.cluster_centers_
    return labels, centers
silhouette_scores = []
for n_clusters in range(2, 11):
    labels, _ = kmeans_clustering(X, n_clusters)
    silhouette_scores.append(silhouette_score(X, labels))
optimal_n_clusters = np.argmax(silhouette_scores) + 2
# Perform K-means clustering with the optimal number of clusters
labels, centers = kmeans_clustering(X, optimal_n_clusters)
# Plot the clusters
plot_clusters(X[:, :2], labels, centers)
```

Output :-



12/04/24