Practical No:- 1

Aim :-

Descriptive Statistics

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

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

Theory :-

Descriptive statistic is about describing and summarising data.

It use two main approaches :

1. The quantitative approach describes and summarizes that data numerically.

2. The visual approach illustrates data with charts, plots, histograms, and other graphs.

You can apply descriptive statistics to one or more datasets or variable. When you describe the and summarize a single variable, you are performing univariate analysis.

When you search for statistical relationship among pair of variables, you are doing 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 database is the Data Frame. If the Data Frame contains numerical data, the description contains these information of each column :

count - The number of non-empty values

mean - The average (mean) values

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 Data Frame. The information contains the number of the 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 value in the given axis. Default axis is row. By specifying the column axis (axis = ‘columns’), the quantile ( ) method calculate 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 powerful tool for grouping data based on one or more criteria and applying function to each group independently.

6. agg ( ) – The ‘agg ( )’ function is used to aggregate data in a Data Frame or a series. The function is used in conjugation with ‘group by ( )’ for more complex data aggregation tasks.

Input :-

mtcars.csv – This dataset consists of 32 observations (rows) on 11 numeric variables (columns) : mpg, cyl. disp, hp, drat, wt, qsec, vs, am, gear, and carb.

iris.csv - This dataset consists of 150 samples from 3 Iris flower species, each described by four features: sepal length, sepal width, petal length and petal width.

Program :-

```
import pandas as pd


#program for mtcars
mtcars = pd.read_csv('mtcars.csv')


print("Summary Statistics for mtcars:")
print(mtcars.describe())


print("\nGeneral Information for mtcars:")
print(mtcars.info())


print("\nQuartile Information for mtcars:")
numeric_columns = mtcars.select_dtypes(include=['number']).columns
print(mtcars[numeric_columns].quantile([0.25, 0.5, 0.75]))


#program for cars
cars = pd.read_csv('cars.csv')


print("Summary Statistics for cars:")
print(mtcars.describe())


print("\nGeneral Information for cars:")
print(mtcars.info())


print("\nQuartile Information for cars:")
```

```
numeric_columns = mtcars.select_dtypes(include=['number']).columns
print(mtcars[numeric_columns].quantile([0.25, 0.5, 0.75]))


#program for cars
iris = pd.read_csv('iris.csv')


print("Original Iris dataset:")
print(iris.head())


subset_condition = (iris['sepal_length'] > 5.0) & (iris['sepal_width'] > 3.0)
print("\nSubset of Iris Dataset:")
print(subset_condition)


aggregate_result = iris.groupby('species').agg({'petal_length': 'mean'})
print("\nAggregate result - Mean petal length for each species:")
print(aggregate_result)
```

Output :-

Conclusion:-

The program to implement descriptive statistics to perform relevant given operations on iris, mtcars and cars dataset is executed successfully.

Practical No:- 2

Aim :-

Reading and Writing different types of data sets

a. Reading different types of datasets (.txt .csv) from web and disk and writing in a file in a 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 separate values in a tabular format. Excel is a spreadsheet application that stores data in cells organized into rows and columns. Understanding this file format 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 request 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 path and directories correctly.

Input :-

perimeter.txt - It is simple text document.

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

txt_url - The given link is of the text document consisting the graphical (non-control) characteristics defined by ISO8859-1(1987)

csv_url - The given link is of the CSV data set consisting of 3 observations (rows) on 3 variables (columns).

Dataforprac.xlsx – It is an Excel file containing 8 observations on 3 variables.

Program :-

```
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/9061dd50a9c0d9628592b156326251ff/raw/381229ffc3a72c04066397c948cf386e10c98bee/employees.csv'

excel_file_path = r'Dataforprac.xlsx'
```

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

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

# # Opening TXT file from Web
response = requests.get(txt_url)
peri = response.text
print("Txt File from Web:  \n\n")
print(peri)
print('\n')

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

df = pd.read_excel(excel_file_path)
print(df)
```

Output :-

Conclusion:-

The program to read and write different types of data sets 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 Distribution 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 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 bias 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 properties. 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')

# Pie Chart (using absolute values)
plt.subplot(1, 3, 3)
df.abs().sum().plot(kind='pie', autopct='%1.1f%%')
plt.title('Pie Chart')

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

Output :-

Conclusion:-

The program to implement visualizations to plot some of types of plots on sample dataset performed successfully.

Practical No:- 4

Aim :-

Correlation and preview window correlation matrix products correlation plot on the data set and visualise giving an overview of relationships among data on arrest data analysis of covariance variance ANOVA if data have categorical variables on the 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. The value of 1 indicates a perfect positive correlation, -1 indicate perfect negative correlation and zero indicate no correlation. The correlation matrix is calculated using 'corr ( )' method in pandas.

Covariance :- Covariance measure how much variable 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 statistical method used to determine the if there are any statistical significance differences between the means of 3 or more independent group

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

Input :-

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

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 :-

Conclusion:-

The program to implement correlation and covariance and perform ANOVA Test on results implemented successfully.

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 this 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-Odds 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 with admission, while negative coefficients suggest an inverse relationship.

Input :-

Admission.csv – The dataset consists of 400 observations (rows) on 4 variables (columns) they are admit, gre, gpa and rank.

Program :-

```
# Importing required libraries

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

url = 'https://raw.githubusercontent.com/pratikbagdi/PS-2-Practical/main/admission.csv'

df = pd.read_csv(url)


# Name your dataset
```

```
df.name = "Admissions_Dataset"

# Checking the first few rows of the dataset
print(df.head())

# Separating features and target variable
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
print(result.summary())

# Predicting on the test set
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 :-

Conclusion:-

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

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., Average area income, Average area house age, Average area number of rooms, Average area number of bedrooms, Area population

Dependent variable: This is the variable that we are trying to predict or explain.

For e.g., 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 variable. This means that the changes in the dependent variable can be modelled 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 variable.
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 consists 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.

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 :-

Conclusion :-

The program for multiple regression model implemented successfully.

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 labelled 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 is installed and installs if it 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 consists of 10000 observations on 9 variables they are data, location, country, temperature, CO_2 emissions, Sea level rise, precipitation, humidity, Windspeed.

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")

# Preprocessing
# We need to convert the 'Country' feature to numerical values using one-hot encoding
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
classifier.fit(X_train, y_train)

# Make predictions
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))
```

Output :-

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.

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 select 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 coloured based on its assigned cluster. Additionally, blank 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.

Program :-

```
# Import libraries

from sklearn.datasets import load_iris

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt


# Load the Iris dataset
```

```
iris = load_iris()

# Data (features) and target (species)
data = iris.data
target = iris.target

# Define the number of clusters (k)
k = 3

# Create the K-Means model
kmeans = KMeans(n_clusters=k, n_init=10)  # Set n_init to 10 explicitly

# Train the model (fit the data)
kmeans.fit(data)

# Get cluster labels for each data point
cluster_labels = kmeans.labels_

# Plot the data with colors corresponding to clusters
plt.scatter(data[:, 0], data[:, 1], c=cluster_labels)  # Using first two features for visualization

# Add labels for each cluster (optional)
for i, label in enumerate(kmeans.cluster_centers_):
    plt.scatter(label[0], label[1], color='black', marker='s', label=f'Cluster {i+1}')

# Add legend and title
plt.title('Iris Dataset - K-Means Clustering')
plt.xlabel('Sepal length (cm)')
plt.ylabel('Sepal width (cm)')
plt.legend()

# Show the plot
plt.show()
```

```
# Print the target values for reference (optional)
print("Actual species:")
print(target)

# Print the cluster labels for each data point
print("Predicted cluster labels:")
print(cluster_labels)
```

Output :-

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.

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 Neighbours (kNN) algorithm 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 neighbours. 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 labelled 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 neighbours to consider when making predictions. It is typically chosen empirically or through techniques 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 neighbours**: 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 neighbours. In the case of ties, a simple solution is to choose the class label of the closest neighbour.
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 classic dataset in machine learning and statistics. The data set consists 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, virginica.

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

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=45)

# Standardize features by removing the mean and scaling to unit variance
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 :-

Conclusion:-

The program to implement k-Nearest Neighbour algorithm to classify the iris data set is successfully executed.

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 select 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.

Application of K-means to Iris Dataset:

1. **Data Loading**: Load the Iris dataset containing feature vectors representing the characteristics of iris flowers.
2. **Preprocessing**: No significant preprocessing is required for K-means, as it operates on numerical feature vectors.
3. **K-means Clustering**: Apply the K-means algorithm to cluster the iris data points into K clusters.
4. **Evaluation**: Assess the quality of clustering using metrics like silhouette score or visually inspecting cluster separation.
5. **Visualization**: Visualize the clusters and centroids to interpret the results and gain insights into the data distribution.

Input :-

iris.csv – The iris dataset is classic dataset in machine learning and statistics. The data set consists 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, virginica.

Program :-

```
# Importing necessary libraries

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

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Function to plot the clusters
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()

# Function to perform K-means clustering
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

# Determine the optimal number of clusters using silhouette score
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  # Adding 2 because range started from 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 :-

Conclusion :-

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