**PRACTICAL-13**

**AIM:** Write a program to implement all the functionalities of the Scikit-learn library in Python

**Source Code:**

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

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

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

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

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

from sklearn.feature\_selection import SelectKBest, f\_classif

from sklearn.model\_selection import GridSearchCV

# Load the Iris dataset

iris = load\_iris()

X, y = iris.data, 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=42)

# Standardize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train a Random Forest classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

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

# Predict on the test set

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

# Evaluate the model

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

print(f"Accuracy: {accuracy:.2f}")

report = classification\_report(y\_test, y\_pred)

print("Classification Report:\n", report)

# K-Means clustering

kmeans = KMeans(n\_clusters=3)

kmeans.fit(X)

# Principal Component Analysis (PCA)

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

# SelectKBest for feature selection

selector = SelectKBest(score\_func=f\_classif, k=2)

X\_new = selector.fit\_transform(X, y)

# Grid Search for hyperparameter tuning

param\_grid = {'C': [0.1, 1, 10], 'gamma': [0.001, 0.01, 0.1]}

grid\_search = GridSearchCV(SVC(), param\_grid, cv=3)

grid\_search.fit(X\_train, y\_train)

best\_params = grid\_search.best\_params\_

print("Best Hyperparameters:", best\_params)

**Output:**

