1. Implement Principal Component Analysis (PCA) and Singular Value Decomposition (sVD) using NumPy.

CODE:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

data = load_iris()
X = data.data
y = data.target

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

svd = TruncatedSVD(n_components=2)
X_train_svd = svd.fit_transform(X_train)
X_test_svd = svd.transform(X_test)

def evaluate_classifier(X_train, X_test, y_train, y_test):
    clf = LogisticRegression(max_iter=1000)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    return accuracy_score(y_test, y_pred)

pca_accuracy = evaluate_classifier(X_train_pca, X_test_pca, y_train, y_test)


svd_accuracy = evaluate_classifier(X_train_svd, X_test_svd, y_train, y_test)

print(f"PCA Classification Accuracy: {pca_accuracy}")
print(f"SVD Classification Accuracy: {svd_accuracy}")
```

2.Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samnples. Read the training data from a .CSV file.

CODE:

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

```
data = pd.read_csv("trainingdata.csv")
concepts = np.array(data)[:, :-1]
target = np.array(data)[:, -1]

def train(concepts, target):

    for i, val in enumerate(target):
        if val == "Yes":
            specific = concepts[i].copy()
            print("Specific hypothesis initialized as:", specific)
            break

    if specific is not None:
        for i, val in enumerate(concepts):
            if target[i] == "Yes":
                for j in range(len(specific)):
                    if val[j] != specific[j]:
                        specific[j] = "?"
                    else:
                        pass
                print(specific)

    return specific
specific_hypothesis = train(concepts, target)
print("Final specific hypothesis:", specific_hypothesis)
```

3.For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a
description of the set of all hypotheses consistent with the training
examples.

CODE:

```
import numpy as np
import pandas as pd

data = pd.read_csv("trainingdata.csv")
concepts = np.array(data.iloc[:, :-1])
target = np.array(data.iloc[:, -1])

specific_h = concepts[0].copy()
general_h = [["?" for _ in range(len(specific_h))] for _ in range(len(specific_h))]

for i, h in enumerate(concepts):
    if target[i] == "Yes":
        for j in range(len(specific_h)):
            if h[j] != specific_h[j]:
                specific_h[j] = "?"
                general_h[j][j] = "?"
```

```
    elif target[i] == "No":
        for j in range(len(specific_h)):
            if h[j] != specific_h[j]:
                general_h[j][j] = specific_h[j]
            else:
                general_h[j][j] = "?"

general_h = [h for h in general_h if h != ["?" for _ in range(len(specific_h))]]

print("Specific Hypothesis is:", specific_h)
print("General Hypotheses are:", general_h)
```

---

4. Write a programn to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

CODE:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, export_text
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

df = pd.read_csv('tennisdata.csv')
df_encoded = df.copy()
for column in df_encoded.columns:
    df_encoded[column] = df_encoded[column].astype('category').cat.codes

X = df_encoded.drop('PlayTennis', axis=1)
y = df_encoded['PlayTennis']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(criterion='entropy', random_state=42)
clf.fit(X_train, y_train)
tree_rules = export_text(clf, feature_names=list(X.columns))
print("Decision Tree:\n", tree_rules)

new_sample = pd.DataFrame({'Outlook': [0], 'Temperature': [1], 'Humidity': [0], 'Windy': [0]})
prediction = clf.predict(new_sample)
result = 'Yes' if prediction[0] == 1 else 'No'
print(f"Prediction for the new sample: {result}")

y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on the test set: {accuracy * 100:.2f}%")
```

5.Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.

CODE:

```
import numpy as np
```

```
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)

X = X / np.amax(X, axis=0)
y = y / 100

inputSize = 2
hiddenSize = 3
outputSize = 1
learning_rate = 0.01

W1 = np.random.randn(inputSize, hiddenSize) * 0.1
W2 = np.random.randn(hiddenSize, outputSize) * 0.1

def sigmoid(s):
    return 1 / (1 + np.exp(-s))

def sigmoidPrime(s):
    return s * (1 - s)

def forward(X, W1, W2):
    z = np.dot(X, W1)
    z2 = sigmoid(z)
    z3 = np.dot(z2, W2)
    o = sigmoid(z3)
    return o

def backward(X, y, W1, W2, o):
    o_error = y - o
    o_delta = o_error * sigmoidPrime(o)
    z2_error = o_delta.dot(W2.T)
    z2_delta = z2_error * sigmoidPrime(np.dot(X, W1))

    W1 += learning_rate * X.T.dot(z2_delta)
    W2 += learning_rate * np.dot(sigmoid(np.dot(X, W1)).T, o_delta)

    return W1, W2

for i in range(1000):
    o = forward(X, W1, W2)
    W1, W2 = backward(X, y, W1, W2, o)

# Results
print("\nInput: \n" + str(X))
print("\nActual Output: \n" + str(y))
print("\nPredicted Output: \n" + str(forward(X, W1, W2)))
print("\nLoss: \n" + str(np.mean(np.square(y - forward(X, W1, W2)))))
```

6.rite a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the
classifier, considering few test data sets.

CODE:

```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
data=pd.read_csv("tennisdata.csv")
print(data.head())
x=data.iloc[:,:-1]
y=data.iloc[:,-1]

le_outlook=LabelEncoder()
x.Outlook=le_outlook.fit_transform(x.Outlook)

le_temperature=LabelEncoder()
x.Temperature=le_outlook.fit_transform(x.Temperature)

le_humidity=LabelEncoder()
x.Humidity=le_outlook.fit_transform(x.Humidity)

le_windy=LabelEncoder()
x.Windy=le_outlook.fit_transform(x.Windy)

le_playtennis=LabelEncoder()
y=le_playtennis.fit_transform(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
classifier=GaussianNB()
classifier.fit(x_train,y_train)
from sklearn.metrics import accuracy_score
print("Accuarcy :",accuracy_score(classifier.predict(x_test),y_test))
```