B.Sc.(Computer Science) V Sem

Machine Learning

Practical Programs

1. **Write a Python program using Scikit-learn to split the Iris dataset consisting of 150 records into 70% train data and 30% test data. Also split data set into the training set will contain 120 records and the test set contains 30 of those records. Print both datasets.**

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import pandas as pd
iris = load_iris()                    # Load the iris dataset
X = iris.data                         # Features
Y = iris.target                       # Labels

# Split the data into 70% training and 30% testing
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,random_state=42)

# Convert the training and testing data to pandas DataFrames for better readability
train_data = pd.DataFrame(X_train, columns=iris.feature_names)
train_data['species'] = y_train

test_data = pd.DataFrame(X_test, columns=iris.feature_names)
test_data['species'] = y_test

print("Training Data (70%):")                # Print both datasets
print(train_data)
print("\nTest Data (30%):")
print(test_data)

# Split the data into 120 records training and 30 records testing data.
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=30,random_state=42)

# Convert the training and testing data to pandas DataFrames for better readability
train_data = pd.DataFrame(X_train, columns=iris.feature_names)
train_data['species'] = y_train

test_data = pd.DataFrame(X_test, columns=iris.feature_names)
test_data['species'] = y_test

print("Training Data (120 records):")        # Print both datasets
print(train_data)
print("\nTest Data (30 records):")
print(test_data)
```

2. **Write Python program to use sklearn's Decision Tree Classifier to build a decision tree for the sklearn's datasets. Find the importance of a split (entropy, gini measure, accuracy)**

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import tree
import math
import pandas as pd
iris=load_iris()
x=pd.DataFrame(iris.data,columns=iris.feature_names)
y=pd.DataFrame(iris.target)
print(x)
print(y)
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size=0.2,random_state=42)
model=DecisionTreeClassifier(random_state=13)
model.fit(x_train,y_train)
tree.plot_tree(model)
predicted=model.predict(x_test)
print(predicted)
accuracy=accuracy_score(predicted,y_test)
print('Accuracy=',accuracy)

entropy = -sum(p * math.log2(p) for p in y.value_counts(normalize=True) if p > 0)
print("Entropy:",entropy)
gini = 1 - sum(p**2 for p in y.value_counts(normalize=True))
print("Gini:",gini)
```

3. **Write a Python program to implement your own version of the K-means algorithm.**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.metrics import silhouette_score

X = load_iris().data                    # Load Iris dataset
centroids = X[:3]                       # Initialize centroids
for i in range(10):                     # K-Means Clustering
    labels = np.argmin(np.linalg.norm(X[:, None] - centroids, axis=2), axis=1)
    centroids = np.array([X[labels == j].mean(axis=0) for j in range(3)])
print("Silhouette Score =", silhouette_score(X, labels))  # Evaluate performance
```

```
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.scatter(centroids[:, 0], centroids[:, 1], marker="X", s=200)
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("K-means Clustering - Iris")
plt.show()
```

4. Design a Perceptron classifier to classify handwritten numerical digits (0-9). Implement using scikit

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

digits = load_digits()                    # Load handwritten digit dataset
X = digits.data
y = digits.target

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

model = Perceptron(max_iter=1000, random_state=42)      # Create Perceptron classifier

model.fit(X_train, y_train)                    # Train the model

predicted = model.predict(X_test)              # Predict digits

for i in range(10):
   print("Actual:", y_test[i], "Predicted:", predicted[i])
   plt.imshow(X_test[i].reshape(8, 8), cmap="gray")
   plt.show()

print("Accuracy =", accuracy_score(y_test, predicted))              # Calculate accuracy

plt.imshow(digits.images[0], cmap="gray")                  # Display a handwritten digit
plt.title("Digit: " + str(digits.target[0]))
plt.show()
```

5. **Write a Python program to classify text as spam or not spam using the Naïve Bayes Classifier**

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
# Multinomial Distribution and Discrete Data
messages = [
   "Free money now",
   "Hello, how are you?",
   "Claim your prize today",
   "Are we meeting tomorrow?"
]                                                    # Training data
labels = ["spam", "ham", "spam", "ham"]              # ham means not a spam message

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(messages)               #Convert text into number counts

model = MultinomialNB()
model.fit(X, labels)                                 #Train the Naive Bayes model

new_message = ["Are we meeting tomorrow?"]           #Test a new message
X_new = vectorizer.transform(new_message)

prediction = model.predict(X)
accuracy = accuracy_score(labels, prediction)        # Evaluate the model
print(f"Accuracy: {accuracy:}")

prediction = model.predict(X_new)
print(f"Message: {new_message[0]}")
print(f"Prediction: {prediction[0]}")

new_message = ["Free money now"]                     #Test another message
X_new = vectorizer.transform(new_message)

prediction = model.predict(X_new)
print(f"Message: {new_message[0]}")
print(f"Prediction: {prediction[0]}")
```

output:

```
Accuracy: 1.0
Message: Are we meeting tomorrow?
Prediction: ham
Message: Free money now
Prediction: spam
```