Week 6

Implement Project/Module Depending on Week1-5 Training

Task 1: SVM Classification on Iris Dataset

Step 1: Load the Iris Dataset

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Load dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Target (species)

# Display feature names and target labels
print("Features:", iris.feature_names)
print("Target Labels:", iris.target_names)
```

Step 2: Split Data into Training and Testing Sets

```python
# Split data (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 3: Train an SVM Classifier

```python
# Initialize SVM with default parameters
model = SVC(kernel='linear', random_state=42)
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)
```

Step 4: Evaluate Model Accuracy

```python
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Classification report
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

Output:

```
Copy
Accuracy: 1.00
              precision  recall  f1-score   support
      setosa       1.00    1.00      1.00        10
  versicolor       1.00    1.00      1.00         9
   virginica       1.00    1.00      1.00        11
```

Insight:

- SVM achieves **100% accuracy** on the Iris dataset with a linear kernel.

Task 2: K-Means Clustering on Iris Dataset

Step 1: Load the Iris Dataset

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Use the same Iris dataset
X = iris.data  # Features (no labels for clustering)
```

Step 2: Apply K-Means Clustering

```python
# Initialize K-Means (3 clusters for 3 species)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)

# Add clusters to original data for comparison
iris_df = pd.DataFrame(X, columns=iris.feature_names)
iris_df['Cluster'] = clusters
iris_df['Actual_Species'] = y  # Actual labels
```

Step 3: Visualize Clusters

```python
# Plot clusters vs actual species
plt.figure(figsize=(12, 5))

# Cluster Assignments
plt.subplot(1, 2, 1)
sns.scatterplot(x='petal length (cm)', y='petal width (cm)', hue='Cluster', data=iris_df, palette='viridis')
plt.title('K-Means Clusters')

# Actual Species
plt.subplot(1, 2, 2)
sns.scatterplot(x='petal length (cm)', y='petal width (cm)', hue='Actual_Species', data=iris_df, palette='viridis')
```

```
plt.title('Actual Species')

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

Step 4: Compare Clusters with Actual Labels

```
Python
# Cross-tabulation
comparison = pd.crosstab(iris_df['Actual_Species'], iris_df['Cluster'])
print("\nCluster vs Actual Species:\n", comparison)
```

Output:

```
Cluster vs Actual Species:
Cluster       0   1   2
Actual_Species
0            50   0   0
1             0  39  11
2             0  14  36
```

Insights:

- **Cluster 0** perfectly matches **Setosa**.
- **Versicolor (1)** and **Virginica (2)** overlap slightly due to similar petal measurements.
- **Accuracy:** ~89% (adjusted Rand score can quantify this).