Week-6

```
import numpy as np

import pandas as pd

from sklearn.datasets import load_iris

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn import metrics

import matplotlib.pyplot as plt

from sklearn import tree

iris = load_iris()

X = iris.data

y = iris.target

print(iris.DESCR)

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

model = DecisionTreeClassifier(criterion='entropy', random_state=42)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = metrics.accuracy_score(y_test, y_pred)

print("Accuracy before tuning:", accuracy)

param_grid = {

'criterion': ['gini', 'entropy'],

'max_depth': [3, 5, 7, None],

'min_samples_split': [2, 5, 10]

}

grid_search = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid, cv=5)

grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_

print("Best Parameters:", grid_search.best_params_)
print("Best Model Accuracy after tuning:", grid_search.best_score_)

y_pred_best = best_model.predict(X_test)

final_accuracy = metrics.accuracy_score(y_test, y_pred_best)

print("Final Accuracy with tuned model:", final_accuracy)

plt.figure(figsize=(12, 8))

tree.plot_tree(best_model, filled=True,

feature_names=iris.feature_names, class_names=iris.target_names,

rounded=True)

plt.title("Tuned Decision Tree")

plt.show()
```

week-7

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

from sklearn.datasets import load_iris

from sklearn.metrics import accuracy_score,                  confusion_matrix

iris = load_iris()

X = iris.data[:, :2]

y = iris.target

print(iris.DESCR)

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

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)
```

```
print(f"Accuracy:  {accuracy*100:.2f}%")

print("Confusion Matrix")

print(conf_matrix)

plt.figure(figsize=(8, 6))

plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap='coolwarm', marker='o',s=100,
label='Training Data')

plt.scatter(X_test[:, 0], X_test[:, 1], c='green', marker='x', s=200, label='Test Data')

for test_point in X_test:

    distances, indices = knn.kneighbors([test_point])

    for index in indices[0]:

        neighbor = X_train[index]

        plt.plot([test_point[0],                    neighbor[0]],[test_point[1], neighbor[1]],

                'k--', lw=1)


plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

plt.title('K-Nearest Neighbors Visualization (Iris Dataset) k=3')

plt.legend()

plt.show()
```

week-8

```
import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.datasets import load_iris

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

data = load_iris()

X = data.data

y = data.target
```

```
X = X[y != 2, :2]

y = y[y != 2]

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

model = LogisticRegression()

model.fit(X_train, y_train)

 y_pred = model.predict(X_test)

 accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)

print("Confusion Matrix:\n", conf_matrix)

print("Classification Report:\n", class_report)
```

week-9

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import load_iris

from sklearn.metrics import adjusted_rand_score, silhouette_score

iris = load_iris()

X = iris.data[:, :2]

y = iris.target

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis',

marker='o', edgecolor='k', s=100)

plt.title('Original Iris Dataset')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')
plt.grid()

kmeans = KMeans(n_clusters=3, random_state=42)

kmeans.fit(X)

predictions = kmeans.predict(X)

 inertia = kmeans.inertia_

ari = adjusted_rand_score(y, predictions)

silhouette_avg = silhouette_score(X, predictions)

print(f"Inertia: {inertia:.2f}")

print(f"Adjusted Rand Index (ARI): {ari:.2f}")

print(f"Silhouette Score: {silhouette_avg:.2f}")

plt.subplot(1, 2, 2)

plt.scatter(X[:, 0], X[:, 1], c=predictions, cmap='viridis',

marker='o', edgecolor='k', s=100)

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],

c='red', marker='X', s=200, label='Centroids')

plt.title('K-Means Clustering Result')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.grid()

plt.tight_layout()

plt.show()
```

week-10

```
import numpy as np

import pandas as pd

from sklearn.datasets import load_iris

from sklearn.model_selection import cross_val_score

from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, confusion_matrix

import matplotlib.pyplot as plt

import seaborn as sns

data = load_iris()

X = data.data

y = data.target

accuracy_results = {}

decision_tree = DecisionTreeClassifier()

dt_scores = cross_val_score(decision_tree, X, y, cv=5)

dt_accuracy = dt_scores.mean()

accuracy_results["Decision Tree"] = dt_accuracy

decision_tree.fit(X, y)

dt_y_pred = decision_tree.predict(X)

print("\nDecision Tree Results:")

print(f"Cross-Validation Accuracy: {dt_accuracy:.2f}")

print("Classification Report:\n",

classification_report(y, dt_y_pred, target_names=data.target_names))

dt_conf_matrix = confusion_matrix(y, dt_y_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(dt_conf_matrix, annot=True, fmt="d", cmap="Blues",

xticklabels=data.target_names, yticklabels=data.target_names)

plt.title("Decision Tree- Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

#2. K-Nearest Neighbors Classifier
```

```
knn = KNeighborsClassifier(n_neighbors=3)

knn_scores = cross_val_score(knn, X, y, cv=5)

knn_accuracy = knn_scores.mean()

accuracy_results["K-Nearest Neighbors"] = knn_accuracy

knn.fit(X, y)

knn_y_pred = knn.predict(X)

print("\nK-Nearest Neighbors Results:")

print(f"Cross-Validation Accuracy: {knn_accuracy:.2f}")

print("Classification Report:\n",

classification_report(y, knn_y_pred, target_names=data.target_names))

knn_conf_matrix = confusion_matrix(y, knn_y_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(knn_conf_matrix, annot=True, fmt="d", cmap="Blues",

xticklabels=data.target_names, yticklabels=data.target_names)

plt.title("K-Nearest Neighbors- Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

#3. Logistic Regression Classifier

logistic_regression = LogisticRegression(max_iter=200)

lr_scores = cross_val_score(logistic_regression, X, y, cv=5)

lr_accuracy = lr_scores.mean()

accuracy_results["Logistic Regression"] = lr_accuracy

logistic_regression.fit(X, y)

lr_y_pred =logistic_regression.predict(X)

print("\nLogisticRegressionResults:")

print(f"Cross-ValidationAccuracy:{lr_accuracy:.2f}")

print("ClassificationReport:\n",
classification_report(y,lr_y_pred,target_names=data.target_names))
lr_conf_matrix=confusion_matrix(y,lr_y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(lr_conf_matrix,annot=True,fmt="d",cmap="Blues",
xticklabels=data.target_names,yticklabels=data.target_names)
plt.title("LogisticRegression-ConfusionMatrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
#FinalAccuracyComparison
print("\nFinalComparisonofCross-ValidationAccuracies:")
for name, accuracy in accuracy_results.items():
 print(f"{name}:{accuracy:.2f}")
```