```
        Q1: Linear Regression           from sklearn.datasets import
fetch_california_housing from sklearn.model_selection import
train_test_split from sklearn.linear_model import
LinearRegression from sklearn.metrics import
mean_squared_error, r2_score import matplotlib.pyplot as plt
import pandas as pd

        data = fetch_california_housing()
        X = pd.DataFrame(data.data, columns=data.feature_names) y
        = data.target
        # (California dataset has no missing values, normalization optional)

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

        model = LinearRegression() model.fit(X_train,
        y_train)
        print("Regression Coefficients:", model.coef_)
        print("Intercept:", model.intercept_)


        y_pred = model.predict(X_test) mse =
        mean_squared_error(y_test, y_pred) r2 =
        r2_score(y_test, y_pred) print(f"MSE:
        {mse:.3f}") print(f"R² Score: {r2:.3f}")


        plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
        plt.xlabel("Actual Prices") plt.ylabel("Predicted
        Prices") plt.title("Predicted vs Actual House
        Prices") plt.show()
```

Q2: Logistic Regression

```
from sklearn.datasets import load_iris from
sklearn.model_selection import train_test_split from
sklearn.preprocessing import StandardScaler from
sklearn.linear_model import LogisticRegression from
sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns import matplotlib.pyplot as plt

iris = load_iris()
X = iris.data y
= iris.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=0.2,
random_state=42)

clf = LogisticRegression(max_iter=1000) clf.fit(X_train,
y_train)

print("Model Coefficients:\n", clf.coef_)


y_pred = clf.predict(X_test) print("\nConfusion Matrix:\n",
confusion_matrix(y_test, y_pred)) print("\nClassification Report:\n",
classification_report(y_test, y_pred))

plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d",
cmap="Blues") plt.xlabel("Predicted") plt.ylabel("Actual")
plt.title("Confusion Matrix - Iris Logistic Regression") plt.show()
```

Q3: Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier,
plot_tree from sklearn.metrics import accuracy_score iris
= load_iris() X = iris.data y = iris.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=0.2,
random_state=42)

clf = LogisticRegression(max_iter=1000) clf.fit(X_train,
y_train)

print("Model Coefficients:\n", clf.coef_)

for depth in [3, 5, None]:     tree =
DecisionTreeClassifier(max_depth=depth, random_state=42)
tree.fit(X_train, y_train)

    y_train_pred = tree.predict(X_train)
y_test_pred = tree.predict(X_test)

    print(f"\nDecision Tree (max_depth={depth})")     print("Training
Accuracy:", accuracy_score(y_train, y_train_pred))     print("Testing
Accuracy:", accuracy_score(y_test, y_test_pred))
plt.figure(figsize=(12,8)) plot_tree(tree,
feature_names=iris.feature_names,
class_names=iris.target_names, filled=True)
plt.show()
```

Q4: K-Means Clustering

```
from sklearn.cluster import KMeans from sklearn.preprocessing
import StandardScaler from sklearn.metrics import
silhouette_score, adjusted_rand_score import numpy as np

X = iris.data y_true
= iris.target

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

kmeans = KMeans(n_clusters=3, random_state=42) kmeans.fit(X_scaled)
```

```
print("Inertia:", kmeans.inertia_)

inertia = []
K = range(1, 10) for
k in K:
   km = KMeans(n_clusters=k, random_state=42).fit(X_scaled)
inertia.append(km.inertia_)

plt.plot(K, inertia, marker='o')
plt.xlabel('k') plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()

labels = kmeans.labels_ ari =
adjusted_rand_score(y_true, labels) silhouette
= silhouette_score(X_scaled, labels)
print(f"Adjusted Rand Index: {ari:.3f}")
print(f"Silhouette Score: {silhouette:.3f}")

plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels, cmap='viridis', alpha=0.6)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
color='red', marker='X', s=200, label='Centroids') plt.title("K-Means
Clustering on Iris") plt.legend() plt.show()
```

Q5. Cross Validation & Ensemble

```
from sklearn.datasets import load_breast_cancer
from sklearn.svm import SVC from sklearn.ensemble import
RandomForestClassifier from sklearn.model_selection import
cross_val_score, train_test_split from sklearn.metrics import
confusion_matrix, classification_report import numpy as np

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

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

```
svm = SVC(kernel='rbf', gamma='scale') scores =
cross_val_score(svm, X_train, y_train, cv=5) print("SVM
Cross-Validation Accuracies:", scores) print(f"Average:
{scores.mean():.3f}, Std: {scores.std():.3f}")
 t
svm.fit(X_train, y_train) print("SVM Test Accuracy:",
svm.score(X_test, y_test))

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train) y_pred = rf.predict(X_test)

print("\nRandom Forest Test Accuracy:", rf.score(X_test, y_test))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

rf_scores = cross_val_score(rf, X_train, y_train, cv=5) print("\nRandom
Forest Cross-Validation Accuracy:", rf_scores) print(f"Average:
{rf_scores.mean():.3f}, Std: {rf_scores.std():.3f}")


importances = rf.feature_importances_ indices
= np.argsort(importances)[::-1]

plt.figure(figsize=(10,6)) plt.bar(range(10),
importances[indices[:10]], align="center")
plt.xticks(range(10), [data.feature_names[i] for i in
indices[:10]], rotation=45) plt.title("Top 10 Feature
Importances (Random Forest)") plt.show()
```