Dession Tree:-

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

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

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

```
clf = KNeighborsClassifier(n_neighbors=4)
```

```
clf.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

```
y_pred = clf.predict(X_test)
```

```
print("Predictions for the first 5 samples:", clf.predict(X_test[:5]))
```

```
Predictions for the first 5 samples: [1 0 2 1 1]
```

```
print("Class probabilities for the first 5 samples:", clf.predict_proba(X_test[:5]))
```

```
Class probabilities for the first 5 samples: [[0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]]
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of the model:", accuracy)
```

```
Accuracy of the model: 1.0
```

```
accuracy_alternative = clf.score(X_test, y_test)
print("Accuracy of the model (using score method):", accuracy_alternative)
```

```
Accuracy of the model (using score method): 1.0
```

Linear Regression

```
[ ] import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error, r2_score
```

```
[ ] df = pd.read_csv('/content/drive/MyDrive/Concepts and technologies of AI/housing.csv')
    df.head()
```

	Unnamed: 0	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

```
[ ] X = df.drop('MedInc', axis=1)
    y = df['MedInc']
```

```
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)
```

```
LinearRegression()
```

```
y_pred = model.predict(X_test)
```

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 1.100702371672379
R-squared: 0.6890005434851046
```

Logestic Regression

```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    import pandas as pd
```

```
[ ] df = pd.read_csv('/content/drive/MyDrive/Concepts and technologies of AI/Houseprice.csv')
    df.head()
```

	HouseAge	HouseFloor	HouseArea	HousePrice
0	52	2	112.945574	543917.179841
1	93	1	174.312126	817740.124828
2	15	4	125.219577	387992.503019
3	72	4	121.210124	240840.742388
4	61	4	59.221737	277273.386525

```
[ ]
    X = df.drop('HouseFloor', axis=1)
    y = df['HouseFloor']
```

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

```
model = LogisticRegression(max_iter=1000) # Increased max_iter to ensure convergence
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarni
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression

LogisticRegression(max_iter=1000)

```
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)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Classification Report:\n{class_report}")
```

```
Accuracy: 0.25
Confusion Matrix:
[[4 0 0 3 0]
 [3 0 2 1 0]
 [2 0 0 1 0]
 [1 0 0 1 0]
 [1 0 0 1 0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.36      0.57      0.44         7
           2       0.00      0.00      0.00         6
           3       0.00      0.00      0.00         3
           4       0.14      0.50      0.22         2
           5       0.00      0.00      0.00         2

    accuracy                           0.25        20
   macro avg       0.10      0.21      0.13        20
weighted avg       0.14      0.25      0.18        20
```