INTRODUCTION

Algorithms:

- Logistic Regression: A linear model for binary classification.
- Decision Trees: Tree-like models that make decisions based on features.
- K-Nearest Neighbors (KNN): A non-parametric method that classifies based on the majority class of its nearest neighbors.

LINEAR REGRESSION

In linear regression with two variables (X and Y), there are two regression lines: y on x (to predict Y from X) and x on y (to predict X from Y).

```python
# Linear_Regression.py > ...
1    from sklearn.linear_model import LinearRegression
2    X = [[1],[2],[3],[4],[5]]
3    y = [40,50,65,75,90]
4    model = LinearRegression()
5    model.fit(X, y)
6    hours = float(input('Enter study hours: '))
7
8    prdicted_marks = model.predict([[hours]])
9    print(f'Based on your study hour:{hours}, you may score around : {prdicted_marks}')
```

LOGISTIC REGRESSION

the probability of a binary outcome (like Yes/No) is related to predictor variables via a linear equation, and second, this probability is then transformed by a sigmoid (or logit) function to produce an S-shaped curve that models the relationship and outputs a probability between 0 and 1.

```python
Logistic Regression.py > ...
1   from sklearn.linear_model import LogisticRegression
2   X = [[1],[2],[3],[4],[5]]
3   y = [0,0,1,1,1]
4
5   model = LogisticRegression()
6   model.fit(X,y)
7   hours = float(input('enter study hours:'))
8   prdicted_value = model.predict([[hours]])[0]
9   if prdicted_value == 0:
10      print(f'Based on your study hour:{hours}, you may not pass the exam')
11  else:
12      print(f'Based on your study hour:{hours}, you may pass the exam')
```

DECISION TREE

A branch connecting a decision node to a subsequent node (either another decision node or a leaf node).

```python
from sklearn.tree import DecisionTreeClassifier
X = [
    [7,2],
    [8,3],
    [9,8],
    [10,9]
]

y = [0,0,1,1]
model = DecisionTreeClassifier()
model.fit(X,y)
size = float(input('Enter size of fruit:'))
shade = float(input('enter color shade 1-10;'))
value = model.predict([[size,shade]])[0]
if value == 0:
    print(  f'Based on size:{size} and shade:{shade}, the fruit is apple')
else:
    print(f'Based on size:{size} and shade:{shade}, the fruit is orange')
```

APR-SCORE

Accuracy is overall correctness, precision is the correctness of positive predictions, and recall is the model's ability to find all positive instances in the data

```python
APR F1.py > ...
1   from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
2
3   y_true = [1,0,1,1,0,1,0]
4   y_pred =[1,0,1,0,0,1,1,]
5   print('Accuracy score:', accuracy_score(y_true,y_pred))
6   print('precision score:', precision_score(y_true,y_pred))
7   print('recall score:', recall_score(y_true,y_pred))
8   print('f1 score:', f1_score(y_true,y_pred))
```

CONFUSION MATRIX

A confusion matrix is a table that evaluates the performance of a classification algorithm. The matrix uses target values to compare with machine learning (ML) predicted values. Each row in the matrix represents instances in a predicted class and each column represents instances in an actual class or vice versa

```python
confusion
1    from       Configure global settings and get information about the working environment.
2    from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
3
4    y_true =[1,0,1,1,0,1,0,0,1,0]
5    y_pred = [1,0,1,0,0, 1,1,0,1,0]
6    cm = confusion_matrix(y_true, y_pred)
7    print("Confusion Matrix:")
8    print(cm)
9    print('Accuracy score:', accuracy_score(y_true,y_pred))
10   print('precision score:', precision_score(y_true,y_pred))
11   print('recall score:', recall_score(y_true,y_pred))
12   print('f1 score:', f1_score(y_true,y_pred))
```

MAE, MSE, RMSE

Mean Absolute Error (MAE), which measures the average magnitude of the errors.
Mean Squared Error (MSE), which measures the average squared magnitude of the errors.
Root Mean Squared Error (RMSE), which measures the square root of the average squared magnitude of the errors.

```python
MAE MSE RMSE.py > ...
1   from sklearn.metrics import mean_absolute_error, mean_squared_error
2   import numpy as np
3
4   real_score = [90,60,80,100]
5
6   predicted_score = [85,70,70,95]
7   mae = mean_absolute_error(real_score, predicted_score)
8   mse = mean_squared_error(real_score, predicted_score)
9   rmse = np.sqrt(mse)
10  print("Mean Absolute Error (MAE):", mae)
11  print("Mean Squared Error (MSE):", mse)
12  print("Root Mean Squared Error (RMSE):", rmse)
```

THANK YOU