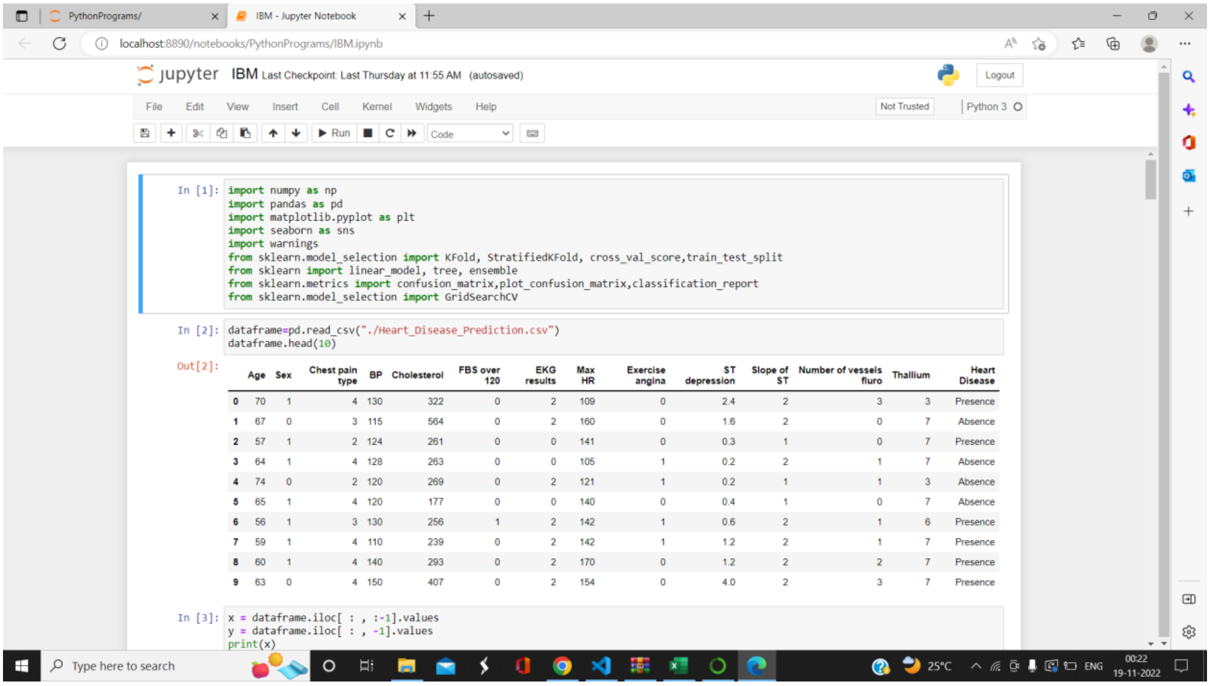
DELIVERY SPRINT 3

ML Model and Backend

Date	7 November 2022
Team ID	PNT2022TMID53144
Project Name	Visualizing and Predicting Heart Diseases with an Interactive Dash Board

Loading dataset:


PythonPrograms/
IBM - Jupyter Notebook
localhost:8890/notebooks/PythonPrograms/IBM.ipynb
jupyter IBM Last Checkpoint: Last Thursday at 11:55 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Type here to search
25°C
ENG
00:22
19-11-2022

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score,train_test_split
        from sklearn import linear_model, tree, ensemble
        from sklearn.metrics import confusion_matrix,plot_confusion_matrix,classification_report
        from sklearn.model_selection import GridSearchCV
```

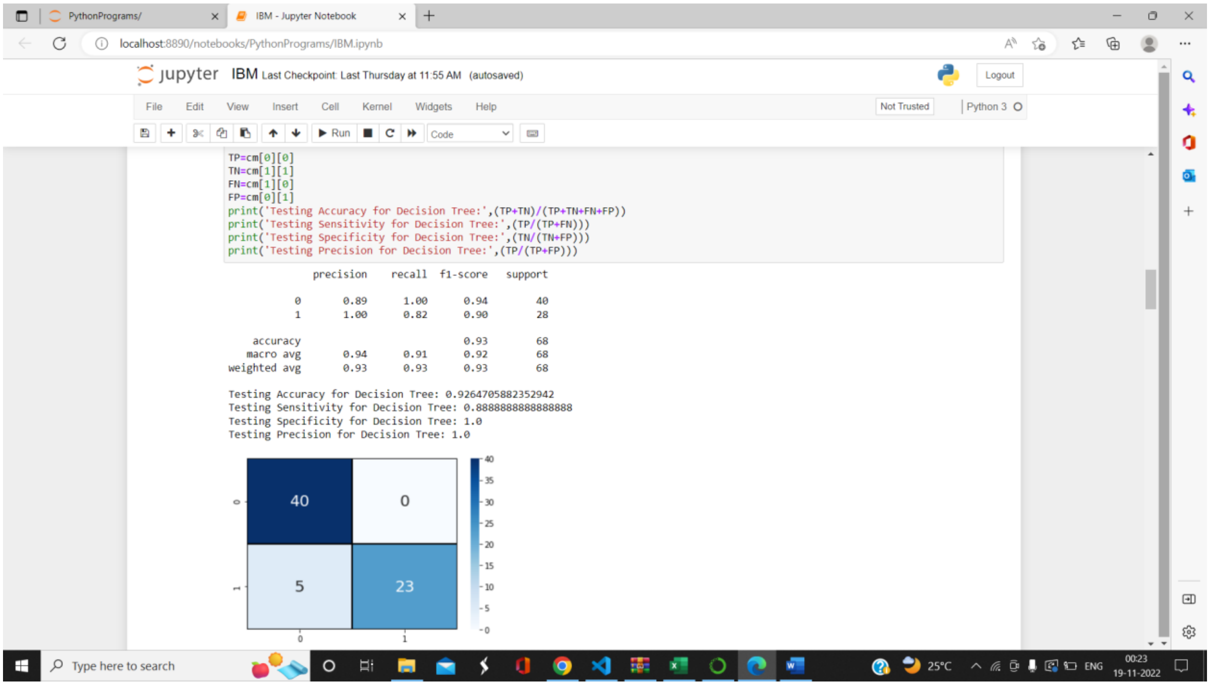
```
In [2]: dataframe=pd.read_csv("./Heart_Disease_Prediction.csv")
        dataframe.head(10)
```

Out[2]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence
5	65	1	4	120	177	0	0	140	0	0.4	1	0	7	Absence
6	56	1	3	130	256	1	2	142	1	0.6	2	1	6	Presence
7	59	1	4	110	239	0	2	142	1	1.2	2	1	7	Presence
8	60	1	4	140	293	0	2	170	0	1.2	2	2	7	Presence
9	63	0	4	150	407	0	2	154	0	4.0	2	3	7	Presence

```
In [3]: x = dataframe.iloc[ : , :-1].values
        y = dataframe.iloc[ : , -1].values
        print(x)
```

Decision Tree Classifier:

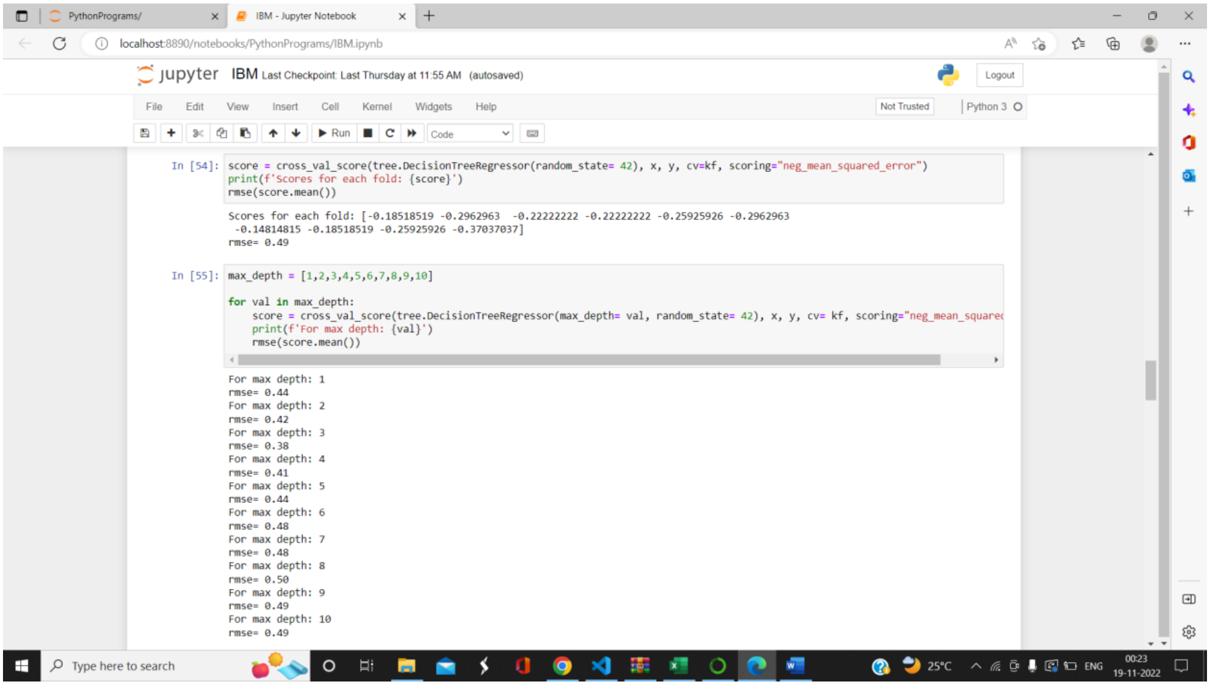


```
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print('Testing Accuracy for Decision Tree:',(TP+TN)/(TP+TN+FN+FP))
print('Testing Sensitivity for Decision Tree:',(TP/(TP+FN)))
print('Testing Specificity for Decision Tree:',(TN/(TN+FP)))
print('Testing Precision for Decision Tree:',(TP/(TP+FP)))
```

```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        40
           1       1.00      0.82      0.90        28

    accuracy                           0.93        68
   macro avg       0.94      0.91      0.92        68
weighted avg       0.93      0.93      0.93        68

Testing Accuracy for Decision Tree: 0.9264705882352942
Testing Sensitivity for Decision Tree: 0.8888888888888888
Testing Specificity for Decision Tree: 1.0
Testing Precision for Decision Tree: 1.0
```

localhost:8890/notebooks/PythonPrograms/IBM.ipynb
IBM Last Checkpoint: Last Thursday at 11:55 AM (autosaved)
40
0
5
23
0
1
0
1
40
35
30
25
20
15
10
5
0

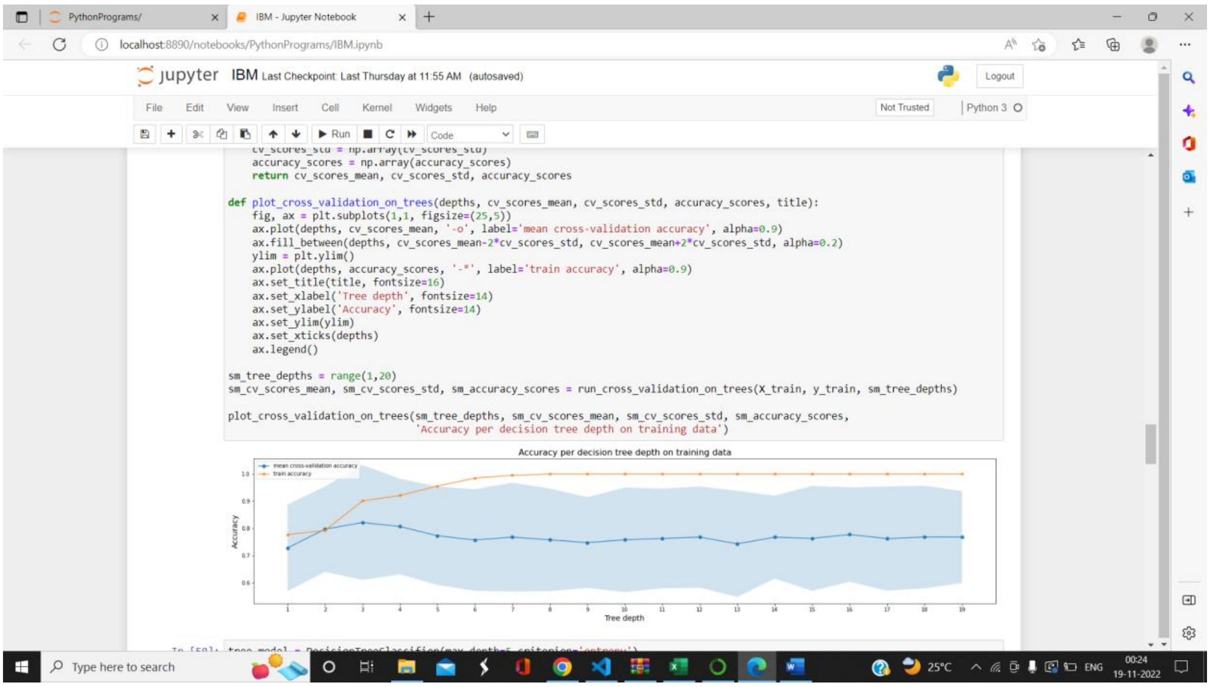
RMSE score:



```
In [54]: score = cross_val_score(tree.DecisionTreeRegressor(random_state= 42), x, y, cv=kf, scoring="neg_mean_squared_error")
         print(f'Scores for each fold: {score}')
         rmse(score.mean())

         Scores for each fold: [-0.18518519 -0.2962963  -0.22222222 -0.22222222 -0.25925926 -0.2962963
          -0.14814815 -0.18518519 -0.25925926 -0.37037037]
         rmse= 0.49

In [55]: max_depth = [1,2,3,4,5,6,7,8,9,10]

         for val in max_depth:
             score = cross_val_score(tree.DecisionTreeRegressor(max_depth= val, random_state= 42), x, y, cv= kf, scoring="neg_mean_squarec
             print(f'For max depth: {val}')
             rmse(score.mean())

         For max depth: 1
         rmse= 0.44
         For max depth: 2
         rmse= 0.42
         For max depth: 3
         rmse= 0.38
         For max depth: 4
         rmse= 0.41
         For max depth: 5
         rmse= 0.44
         For max depth: 6
         rmse= 0.48
         For max depth: 7
         rmse= 0.48
         For max depth: 8
         rmse= 0.50
         For max depth: 9
         rmse= 0.49
         For max depth: 10
         rmse= 0.49
```

localhost:8890/notebooks/PythonPrograms/IBM.ipynb
IBM Last Checkpoint: Last Thursday at 11:55 AM (autosaved)

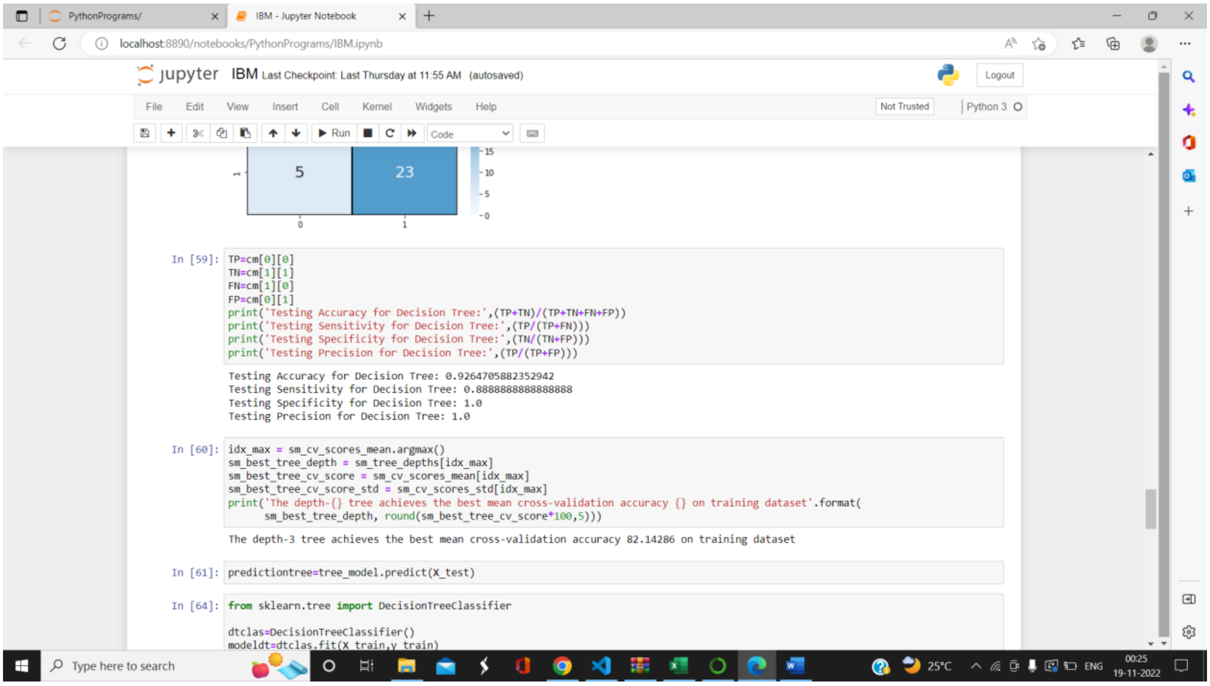
Accuracy Vs Depth graph:



```
    cv_scores_std = np.array(cv_scores_std)
    accuracy_scores = np.array(accuracy_scores)
    return cv_scores_mean, cv_scores_std, accuracy_scores

def plot_cross_validation_on_trees(depths, cv_scores_mean, cv_scores_std, accuracy_scores, title):
    fig, ax = plt.subplots(1,1, figsize=(25,5))
    ax.plot(depths, cv_scores_mean, '-o', label='mean cross-validation accuracy', alpha=0.9)
    ax.fill_between(depths, cv_scores_mean-2*cv_scores_std, cv_scores_mean+2*cv_scores_std, alpha=0.2)
    ylim = plt.ylim()
    ax.plot(depths, accuracy_scores, '-*', label='train accuracy', alpha=0.9)
    ax.set_title(title, fontsize=16)
    ax.set_xlabel('Tree depth', fontsize=14)
    ax.set_ylabel('Accuracy', fontsize=14)
    ax.set_ylim(ylim)
    ax.set_xticks(depths)
    ax.legend()

sm_tree_depths = range(1,20)
sm_cv_scores_mean, sm_cv_scores_std, sm_accuracy_scores = run_cross_validation_on_trees(X_train, y_train, sm_tree_depths)

plot_cross_validation_on_trees(sm_tree_depths, sm_cv_scores_mean, sm_cv_scores_std, sm_accuracy_scores,
                               'Accuracy per decision tree depth on training data')
```

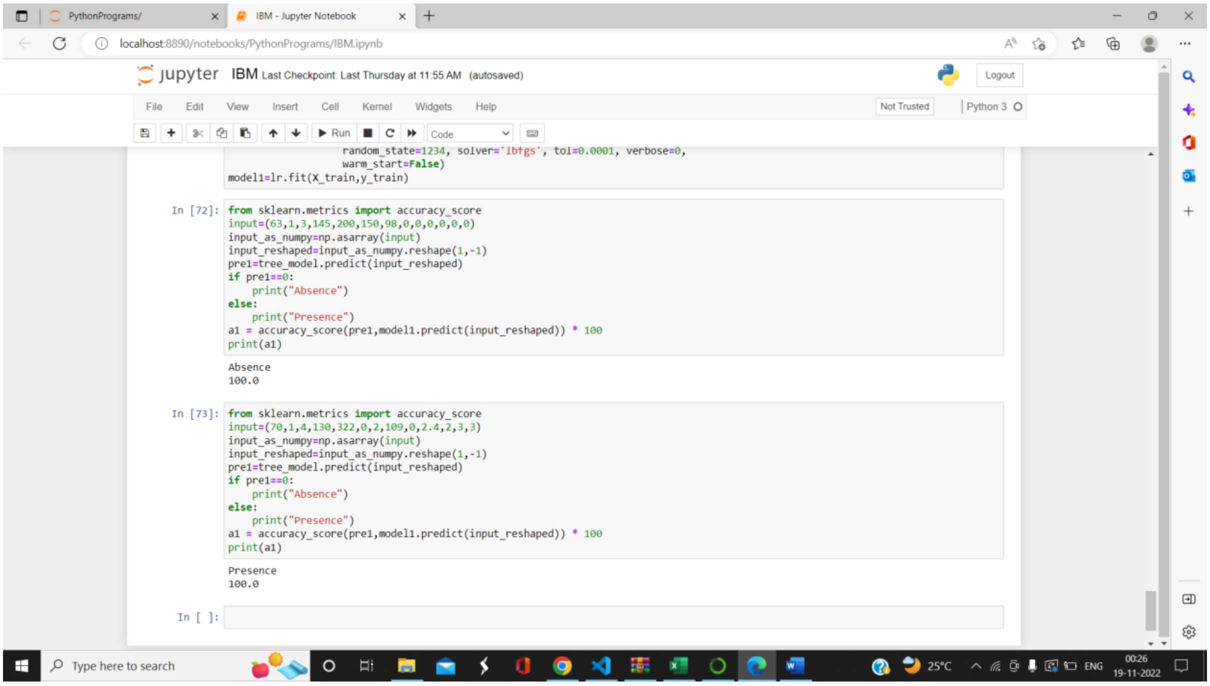
```
In [58]: tree_model = DecisionTreeClassifier(max_depth=5, criterion='entropy')
```

Performance Measure:



```
In [59]: TP=cm[0][0]
         TN=cm[1][1]
         FN=cm[1][0]
         FP=cm[0][1]
         print('Testing Accuracy for Decision Tree:',(TP+TN)/(TP+TN+FN+FP))
         print('Testing Sensitivity for Decision Tree:',(TP/(TP+FN)))
         print('Testing Specificity for Decision Tree:',(TN/(TN+FP)))
         print('Testing Precision for Decision Tree:',(TP/(TP+FP)))

         Testing Accuracy for Decision Tree: 0.9264705882352942
         Testing Sensitivity for Decision Tree: 0.8888888888888888
         Testing Specificity for Decision Tree: 1.0
         Testing Precision for Decision Tree: 1.0

In [60]: idx_max = sm_cv_scores_mean.argmax()
         sm_best_tree_depth = sm_tree_depths[idx_max]
         sm_best_tree_cv_score = sm_cv_scores_mean[idx_max]
         sm_best_tree_cv_score_std = sm_cv_scores_std[idx_max]
         print('The depth-{} tree achieves the best mean cross-validation accuracy {} on training dataset'.format(
               sm_best_tree_depth, round(sm_best_tree_cv_score*100,5)))

         The depth-3 tree achieves the best mean cross-validation accuracy 82.14286 on training dataset

In [61]: predictiontree=tree_model.predict(X_test)

In [64]: from sklearn.tree import DecisionTreeClassifier

         dtclas=DecisionTreeClassifier()
         modeldt=dtclas.fit(X_train,y_train)
```

Prediction & Test:



PythonPrograms/
IBM - Jupyter Notebook
localhost:8890/notebooks/PythonPrograms/IBM.ipynb
jupyter IBM Last Checkpoint: Last Thursday at 11:55 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Type here to search
25°C
ENG
00:26
19-11-2022

```
                     random_state=1234, solver='lbfgs', tol=0.0001, verbose=0,
                     warm_start=False)
model1=lr.fit(X_train,y_train)
```

In [72]:

```
from sklearn.metrics import accuracy_score
input=(63,1,3,145,200,150,98,0,0,0,0,0,0)
input_as_numpy=np.asarray(input)
input_reshaped=input_as_numpy.reshape(1,-1)
pre1=tree_model.predict(input_reshaped)
if pre1==0:
    print("Absence")
else:
    print("Presence")
a1 = accuracy_score(pre1,model1.predict(input_reshaped)) * 100
print(a1)
```

```
Absence
100.0
```

In [73]:

```
from sklearn.metrics import accuracy_score
input=(70,1,4,130,322,0,2,109,0,2.4,2,3,3)
input_as_numpy=np.asarray(input)
input_reshaped=input_as_numpy.reshape(1,-1)
pre1=tree_model.predict(input_reshaped)
if pre1==0:
    print("Absence")
else:
    print("Presence")
a1 = accuracy_score(pre1,model1.predict(input_reshaped)) * 100
print(a1)
```

```
Presence
100.0
```

In [ ]: