Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course code and name:	F21DL: Data Mining and Machine Learning
Type of assessment:	**Group**
Coursework Title:	Part 3. Supervised Learning: Generalisation & Overfitting; Decision trees
Student Name:	Arshati Ajay Marchande, Asmitha Krishnakumar, Gauri Revankar, Pooja Sheni Meledath, Prasitha Naidu
Student ID Number:	H00382093, H00376043, H00373987, H00386700, H00379641

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature *(type your name):* *Arshati, Asmitha, Gauri, Pooja, Prasitha*

Date: *12/11/2023*

Copy this page and insert it into your coursework file in front of your title page.
For group assessment each group member must sign a separate form and all forms must be included with the group submission.

Your work will not be marked if a signed copy of this form is not included with your submission.

PART 3: SUPERVISED LEARNING – GENERALISATION AND OVERFITTING

TABLE OF CONTENTS

1. CONTRIBUTIONS

Arshati

- Experimenting with various decision tree parameters
- Report

Asmitha

- Create 30% and 60% Split for Training and Testing Sets, Evaluate Accuracies
- Report

Gauri

- Decision Tree Performance on Training Set using cross validation and building the classifier on Train and Test data
- Report

Pooja

- Exploration of Alternative Decision Tree Algorithms and drawing conclusions based on the performance
- Report

Prasitha

- Experimenting with various decision tree parameters
- Report

2. DECISION TREE PERFORMANCE: CROSS-VALIDATION AND TRAIN-TEST CLASSIFIER

PREPROCESSING

The datasets worked on in this part of the coursework are the X_train_all, y_train_all datasets for training the model and x_test_all and y_test_all datasets for testing the model.

Image Enhancement techniques:

Below are the image enhancement techniques applied in the Train and Test datasets (same as the ones used in part 1 of the coursework):

1. Histogram Equalization
2. Gamma correction

Normalization

The values in the Train and Test datasets were divided by 255 to scale the values between 0 and 1.

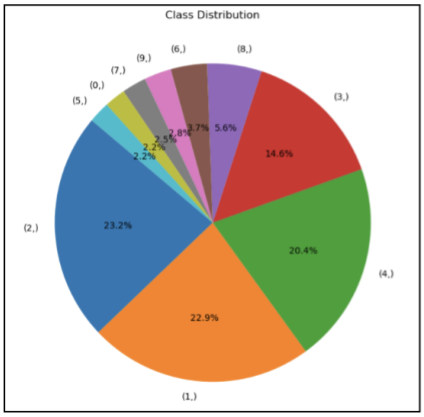
Outlier detection

DBScan was used to detect and remove the outliers in the train dataset.

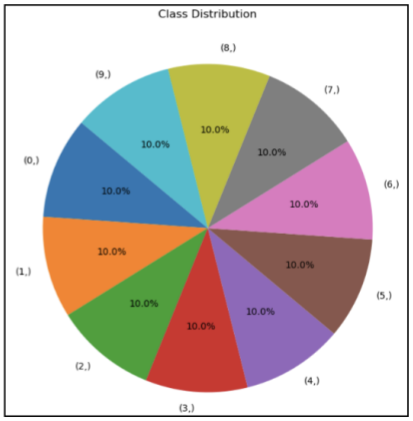
Oversampling

To order the balance the number of samples in each class in the train dataset, oversampling was used.

Before oversampling:



After oversampling:



DECISION TREE ALGORITHM

(70:30 split)

A 93.87% accuracy was obtained after running the DT Algorithm on the train dataset.

```
#Evaluating the accuracy of the model
accuracy_score(y_test,y_predict1)
```

```
0.9387838112670327
```

This this the classification report:

```
#Printing Classification Report
print(classification_report(y_test, y_predict1))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       486
           1       0.88      0.84      0.86       492
           2       0.83      0.81      0.82       481
           3       0.90      0.92      0.91       495
           4       0.89      0.87      0.88       510
           5       0.98      1.00      0.99       519
           6       0.97      0.99      0.98       493
           7       0.98      0.99      0.98       485
           8       0.98      0.99      0.98       486
           9       0.98      0.99      0.99       470

    accuracy                           0.94      4917
   macro avg       0.94      0.94      0.94      4917
weighted avg       0.94      0.94      0.94      4917
```

(60:40 split)

A 94.85% accuracy was obtained after running the DT Algorithm on the train dataset.

```
#Evaluating the accuracy of the model
accuracy_score(y_test1,y_predict1_1)
```

```
0.9485967053081147
```

This this the classification report:

```
#Printing Classification Report
print(classification_report(y_test1, y_predict1_1))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       651
           1       0.91      0.88      0.89       638
           2       0.86      0.86      0.86       641
           3       0.92      0.90      0.91       676
           4       0.92      0.90      0.91       687
           5       0.99      1.00      0.99       685
           6       0.97      0.98      0.97       635
           7       0.99      1.00      0.99       652
           8       0.98      0.99      0.98       671
           9       0.97      0.99      0.98       620

    accuracy                           0.95      6556
   macro avg       0.95      0.95      0.95      6556
weighted avg       0.95      0.95      0.95      6556
```

(80:20 split)

A 93.01% accuracy was obtained after running the DT Algorithm on the train dataset.

```
#Evaluating the accuracy of the model
accuracy_score(y_test2,y_predict1_2)
```

```
0.9301403294691886
```

This this the classification report:

```
#Printing Classification Report
print(classification_report(y_test2, y_predict1_2))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       331
           1       0.88      0.85      0.86       315
           2       0.79      0.80      0.80       314
           3       0.88      0.84      0.86       337
           4       0.89      0.87      0.88       334
           5       0.98      0.99      0.99       360
           6       0.95      0.98      0.96       334
           7       0.99      0.99      0.99       315
           8       0.97      0.99      0.98       324
           9       0.97      0.99      0.98       314

    accuracy                           0.93      3278
   macro avg       0.93      0.93      0.93      3278
weighted avg       0.93      0.93      0.93      3278
```

CROSS VALIDATION

The average accuracy obtained after running a 10-fold cross validation on the training set was 92.57%.

```
# average cross validation on train dataset
cross_validation(clf_model_cv, X_data, y_data, 10)
```

```
Average cross-validation score: 0.9257
```

EVALUATION METRICS

The Training set was split using TrainTestSplit module in a 70:30, 60:40 and 80:20 ratio (train:test) and the decision tree model was trained on the split data.

The following metrics were used for evaluation:

1. Model Accuracy
2. Precision
3. Recall
4. Mean Absolute Error
5. F1 Score
6. Sensitivity
7. Specificity
8. FP, FN, TP, TN rates

The scores obtained are as follows:

70:30 split train data

```
evaluationMetrics(clf_model, X_train, y_train, X_test, y_test, y_predict1)
```

```
Training set score: 1.0000
Test set score: 0.9388

Model Accuracy Score : 0.9388
Precision Score : 0.9388
Recall Score : 0.9388
Mean Absolute Error : 0.1296
F1 Score : 0.9388


Sensitivity (true positive rate), for each class: [0.98971193 0.83739837 0.80665281 0.92323232 0.87058824 0.99807322
 0.99188641 0.99175258 0.98559671 0.99361702]

Sensitivity (Avg): 0.9388509607197184

----------------------------------------------------------------
Specificity (true negative rate), for each class: [0.99796886 0.98757062 0.98264202 0.98824062 0.98797368 0.99818099
 0.9966094  0.99751805 0.99774317 0.99752642]

Specificity (Avg): 0.9931973831002825

----------------------------------------------------------------
FP rate, for each class: [0.00203114 0.01242938 0.01735798 0.01175938 0.01202632 0.00181901
 0.0033906  0.00248195 0.00225683 0.00247358]

FP rate (Avg): 0.0068026168997174325

----------------------------------------------------------------
FN rate, for each class: [0.01028807 0.16260163 0.19334719 0.07676768 0.12941176 0.00192678
 0.00811359 0.00824742 0.01440329 0.00638298]

FN rate (Avg): 0.061149039280281536

----------------------------------------------------------------
TP rate, for each class: [0.98971193 0.83739837 0.80665281 0.92323232 0.87058824 0.99807322
 0.99188641 0.99175258 0.98559671 0.99361702]

TP rate (Avg): 0.9388509607197184
```

```
------------------------------------------------------------
TN rate, for each class: [0.99796886 0.98757062 0.98264202 0.98824062 0.98797368 0.99818099
 0.9966094  0.99751805 0.99774317 0.99752642]

TN rate (Avg): 0.9931973831002825
```

60:40 split train data

```
Training set score: 1.0000
Test set score: 0.9242

Model Accuracy Score : 0.9486
Precision Score : 0.9486
Recall Score : 0.9486
Mean Absolute Error : 0.1117
F1 Score : 0.9486
```

```
Sensitivity (true positive rate), for each class: [0.99078341 0.87931034 0.86115445 0.90088757 0.89810771 0.99708029
 0.98110236 0.99539877 0.98956781 0.99354839]

Sensitivity (Avg): 0.9486941113328158

------------------------------------------------------------
Specificity (true negative rate), for each class: [0.99796782 0.99070632 0.98444632 0.99081633 0.99062873 0.99846704
 0.9964533  0.99898374 0.99728122 0.99713612]

Specificity (Avg): 0.9942886945312152

------------------------------------------------------------
FP rate, for each class: [0.00203218 0.00929368 0.01555368 0.00918367 0.00937127 0.00153296
 0.0035467  0.00101626 0.00271878 0.00286388]

FP rate (Avg): 0.0057113054687847765

------------------------------------------------------------
FN rate, for each class: [0.00921659 0.12068966 0.13884555 0.09911243 0.10189229 0.00291971
 0.01889764 0.00460123 0.01043219 0.00645161]

FN rate (Avg): 0.051305888667184274

------------------------------------------------------------
TP rate, for each class: [0.99078341 0.87931034 0.86115445 0.90088757 0.89810771 0.99708029
 0.98110236 0.99539877 0.98956781 0.99354839]

TP rate (Avg): 0.9486941113328158

------------------------------------------------------------
TN rate, for each class: [0.99796782 0.99070632 0.98444632 0.99081633 0.99062873 0.99846704
 0.9964533  0.99898374 0.99728122 0.99713612]

TN rate (Avg): 0.9942886945312152
```

80:20 split train data

```
Training set score: 1.0000
Test set score: 0.9359

Model Accuracy Score : 0.9301
Precision Score : 0.9301
Recall Score : 0.9301
Mean Absolute Error : 0.1507
F1 Score : 0.9301
```

```
Sensitivity (true positive rate), for each class: [0.98791541 0.84761905 0.80254777 0.84272997 0.8742515  0.99444444
 0.9760479  0.99047619 0.99074074 0.98726115]

Sensitivity (Avg): 0.9294034119856874

------------------------------------------------------------
Specificity (true negative rate), for each class: [0.99728537 0.98785015 0.97807018 0.98741925 0.98777174 0.9976011
 0.99456522 0.99865002 0.99661476 0.99662618]

Specificity (Avg): 0.9922453957946697

------------------------------------------------------------
FP rate, for each class: [0.00271463 0.01214985 0.02192982 0.01258075 0.01222826 0.0023989
 0.00543478 0.00134998 0.00338524 0.00337382]

FP rate (Avg): 0.007754604205330301

------------------------------------------------------------
FN rate, for each class: [0.01208459 0.15238095 0.19745223 0.15727003 0.1257485  0.00555556
 0.0239521  0.00952381 0.00925926 0.01273885]

FN rate (Avg): 0.07059658801431254

------------------------------------------------------------
TP rate, for each class: [0.98791541 0.84761905 0.80254777 0.84272997 0.8742515  0.99444444
 0.9760479  0.99047619 0.99074074 0.98726115]

TP rate (Avg): 0.9294034119856874

------------------------------------------------------------
TN rate, for each class: [0.99728537 0.98785015 0.97807018 0.98741925 0.98777174 0.9976011
 0.99456522 0.99865002 0.99661476 0.99662618]

TN rate (Avg): 0.9922453957946697
```

BUILDING THE CLASSIFIER ON TRAIN AND TEST DATA

Here are the evaluation scores of the model after repeating the experiment using train and test data:

```
1  evaluationMetrics(clf_model_test, X_data, y_data, X_test_data, y_test_data, y_predict2)
```

```
Training set score: 1.0000
Test set score: 0.7071

Model Accuracy Score : 0.7071
Precision Score : 0.7071
Recall Score : 0.7071
Mean Absolute Error : 0.6935
F1 Score : 0.7071
```

```
Sensitivity (true positive rate), for each class: [0.21666667 0.69166667 0.82133333 0.76888889 0.75909091 0.4
 0.73333333 0.56666667 0.21333333 0.61111111]

Sensitivity (Avg): 0.5782090909090908

------------------------------------------------------------
Specificity (true negative rate), for each class: [0.99075908 0.9443038  0.88205128 0.95037879 0.93703704 0.99867987
 0.98166667 0.99108911 0.99081633 0.976     ]

Specificity (Avg): 0.9642781950438021

------------------------------------------------------------
FP rate, for each class: [0.00924092 0.0556962  0.11794872 0.04962121 0.06296296 0.00132013
 0.01833333 0.00891089 0.00918367 0.024     ]

FP rate (Avg): 0.035721804956197915

------------------------------------------------------------
FN rate, for each class: [0.78333333 0.30833333 0.17866667 0.23111111 0.24090909 0.6
 0.26666667 0.43333333 0.78666667 0.38888889]

FN rate (Avg): 0.42179090909090905

------------------------------------------------------------
TP rate, for each class: [0.21666667 0.69166667 0.82133333 0.76888889 0.75909091 0.4
 0.73333333 0.56666667 0.21333333 0.61111111]

TP rate (Avg): 0.5782090909090908

------------------------------------------------------------
TN rate, for each class: [0.99075908 0.9443038  0.88205128 0.95037879 0.93703704 0.99867987
 0.98166667 0.99108911 0.99081633 0.976     ]

TN rate (Avg): 0.9642781950438021
```

Below is the classification report summarizing the findings:

	precision	recall	f1-score	support
0	0.32	0.22	0.26	60
1	0.79	0.69	0.74	720
2	0.69	0.82	0.75	750
3	0.73	0.77	0.75	450
4	0.77	0.76	0.76	660
5	0.86	0.40	0.55	60
6	0.55	0.73	0.63	90
7	0.56	0.57	0.56	60
8	0.54	0.21	0.31	150
9	0.43	0.61	0.51	90
accuracy			0.71	3090
macro avg	0.62	0.58	0.58	3090
weighted avg	0.71	0.71	0.70	3090

FINDINGS

To summarize,
The average accuracy obtained during 10-fold cross validation is 92%.
The train and test scores obtained with the train test split (70:30) of training data are 100% and 94% respectively.
The train and test scores obtained with the train test split (60:40) of training data are 100% and 92% respectively.
The train and test scores obtained with the train test split (80:20) of training data are 100% and 93% respectively.
The train and test accuracies obtained with the test data are 100% and 71% respectively.

- The fact that the training accuracy is 100% tells us that the model is overfitting. Overfitting occurs when the model learns the training data too well.
- The test accuracy of the model drops from 93% (**approximate average of the 3 train test split accuracies**) to 71%, when evaluated with a test dataset. This large drop in accuracy suggests that the model may not be generalizing well to new, unseen data.

3. EXPERIMENTING WITH DECISION TREE PARAMETERS

max_depth :

The **max_depth** parameter in a decision tree controls the maximum depth or the maximum number of levels in the tree. It restricts the number of nodes in the tree, which affects the complexity of the model.

Influence: Varies with the dataset and problem. Deeper trees may cause overfitting, but generally, increasing max_depth enables the tree to capture more complex relationships.

min_samples leaf :

The **min_samples_leaf** parameter in a decision tree controls the minimum number of samples that must be present in a leaf node. By setting min_samples_leaf, you can control the size of the leaves in the tree, and is crucial in preventing overfitting.

Influence: Higher values result in simpler trees, preventing overfitting by ensuring a minimum number of samples in each leaf.

criterion :

The **criterion** parameter in a decision tree determines the function used to measure the quality of a split at each node. Gini and entropy are two commonly used criteria and are used to guide the algorithm in choosing the best feature to split the data at each node.

Influence: The choice between 'gini' and 'entropy' influences how the tree measures impurity. 'Gini' is faster but might result in larger trees, while 'entropy' tends to produce more balanced trees.

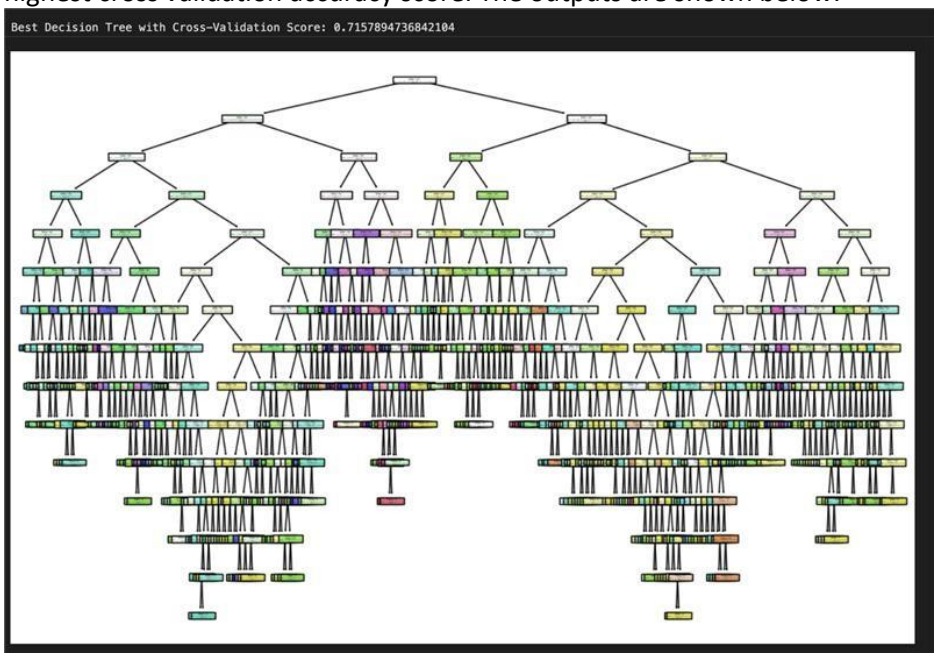
min_impurity_decrease :

The **min_impurity_decrease** parameter in a decision tree specifies the minimum amount by which the impurity must decrease for a split to be considered.

Influence: Higher values result in fewer splits, creating a simpler tree. It can help control overfitting by requiring a minimum impurity decrease for a split to occur.

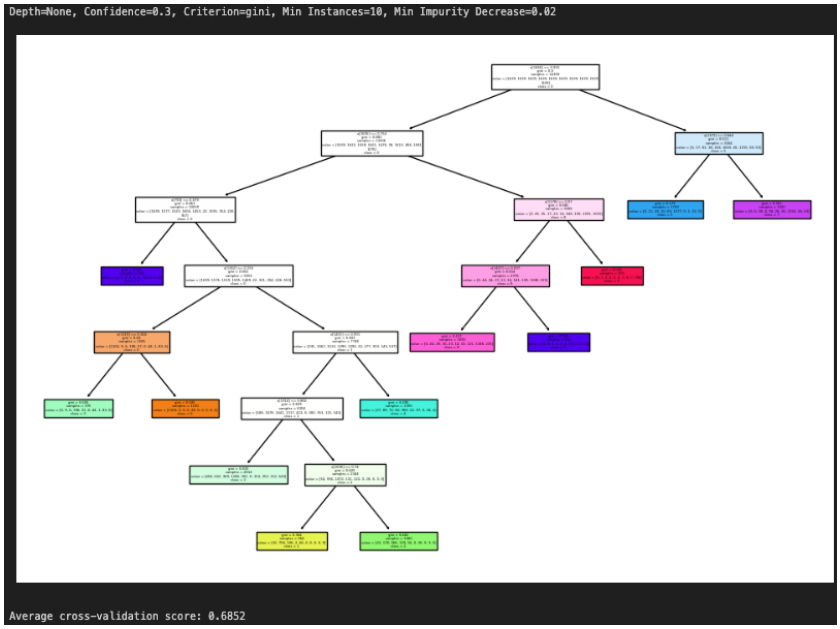
EXPLANATION

The code gives the best performing model based on the highest cross validation score. The code outputs uses the 'plot_tree' function for visualizing the best decision tree after identifying the highest cross validation accuracy score. The outputs are shown below:



The above shows the best decision tree output after comparing with all the parameter combinations and gave a cross validation score of 71.57%.
Below outputs show proof of different decision tree outputs being tried for the various types of parameter combinations before giving the output with the best tree and score.

```
Depth=5, Confidence=0.5, Criterion=gini, Min Instances=5, Min Impurity Decrease=0.01
Depth=5, Confidence=0.5, Criterion=gini, Min Instances=5, Min Impurity Decrease=0.02
Depth=5, Confidence=0.5, Criterion=gini, Min Instances=10, Min Impurity Decrease=0.0
Depth=5, Confidence=0.5, Criterion=gini, Min Instances=10, Min Impurity Decrease=0.01
Depth=5, Confidence=0.5, Criterion=gini, Min Instances=10, Min Impurity Decrease=0.02
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=1, Min Impurity Decrease=0.0
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=1, Min Impurity Decrease=0.01
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=1, Min Impurity Decrease=0.02
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=5, Min Impurity Decrease=0.0
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=5, Min Impurity Decrease=0.01
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=5, Min Impurity Decrease=0.02
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=10, Min Impurity Decrease=0.0
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=10, Min Impurity Decrease=0.01
Depth=5, Confidence=0.5, Criterion=entropy, Min Instances=10, Min Impurity Decrease=0.02
```



This score indicates the performance of the decision tree classifier when evaluated using 10-fold cross validation. The result is an average cross validation score of 0.6852.
Similarly, we changed and experimented with different parameters before obtaining the tree with the best cross-validation scores.

4. SPLIT DATA: 30% AND 60% & EVALUATE ACCURACIES

These are the following results I got when splitting 30% and 60% of the data respectively

```
Splitting the data for 0.3 %
[[ 504    8    9    1   24    0    0    0    0    0]
 [  20  924  145   53   58    1    4    4    1    2]
 [   5   58 1075   55   18    1    7    8    1    3]
 [   8   31   50  814   27    0    6    1    3    5]
 [  11   34   77   39  991    5    3    0    3    7]
 [   0    1    4   10    5  540   11    4    1    3]
 [   0    2    3    3    3    2  559    2    3    6]
 [   0    2    9    3   16    5    2  508    0    0]
 [   3    9   25   16    8    4   13    7  521   30]
 [   1    6    6    5    1    3    6    2   11  519]]
Training set score: 1.0000
Test set score: 0.8686

Model Accuracy Score : 0.8686
Precision Score : 0.8686
Recall Score : 0.8686
Mean Absolute Error : 0.2994
F1 Score : 0.8686
```

```
Splitting the data for 0.6 %
[[ 966   14   14    0   17    0    1    0    0    0]
 [  24 1448  112   49   63    7    6    4    2    2]
 [  12   58 1556   46   27   12    9    6    0    2]
 [   7   29   45 1330   30    1    1    1    2    5]
 [  21   45   50   35 1487   10    3    1    5    8]
 [   0    6    0   13    0 1036   10    3    3    5]
 [   0    3    5    2    3    1 1025    1    3    8]
 [   0    4    8    4    1   18    1  990    0    1]
 [   4    8   19   21    7    8    9   11 1023   27]
 [   1    4   13    4    4    0    5    6    5 1018]]
Training set score: 1.0000
Test set score: 0.9191

Model Accuracy Score : 0.9191
Precision Score : 0.9191
Recall Score : 0.9191
Mean Absolute Error : 0.1937
F1 Score : 0.9191
```

ANALYSIS OF DATA SPLITTING

- For the 30% split:
  - Training set score: 1.0000
  - Test set score: 0.8689
- For the 60% split:
  - Training set score: 1.0000
  - Test set score: 0.9206

CONCLUSION OF DATA SPLITTING

- In both splits, the training set score is 100% so it indicates a potential over fitting problem because the model perfectly fits the training data.
- The test set scores are lower than the training set scores, which suggests some degree of overfitting, especially in the 30% split
- The 60% split performs better on the test set, **suggesting that a larger testing set might mitigate overfitting to some extent**.

5. EXPERIMENTING WITH OTHER MODELS

The below-mentioned experiments were done for two other models **Random Forest Classifier** and **Extra Tree Classifier**. The below table shows the comparison for all the three models:

	Decision Trees Classifier	Random Forest Classifier	Extra Trees Classifier
Accuracy	93.87%	99.36%	99.08%
Cross Validation	92.33%	98.31%	97.74%

Evaluation Metrics	Training set score: 1.0000 Test set score: 0.9388 Model Accuracy Score : 0.9388 Precision Score : 0.9388 Recall Score : 0.9388 Mean Absolute Error : 0.1296 F1 Score : 0.9388	Training set score: 1.0000 Test set score: 0.9937 Model Accuracy Score : 0.9937 Precision Score : 0.9937 Recall Score : 0.9937 Mean Absolute Error : 0.0114 F1 Score : 0.9937	Training set score: 1.0000 Test set score: 0.9908 Model Accuracy Score : 0.9908 Precision Score : 0.9908 Recall Score : 0.9908 Mean Absolute Error : 0.0148 F1 Score : 0.9908
Splitting Dataset 30%	Training set score: 1.0000 Test set score: 0.8719 Model Accuracy Score : 0.8719 Precision Score : 0.8719 Recall Score : 0.8719 Mean Absolute Error : 0.3122 F1 Score : 0.8719	Training set score: 1.0000 Test set score: 0.9379 Model Accuracy Score : 0.9379 Precision Score : 0.9379 Recall Score : 0.9379 Mean Absolute Error : 0.1627 F1 Score : 0.9379	Training set score: 1.0000 Test set score: 0.9133 Model Accuracy Score : 0.9133 Precision Score : 0.9133 Recall Score : 0.9133 Mean Absolute Error : 0.2192 F1 Score : 0.9133
60%	Training set score: 1.0000 Test set score: 0.9183 Model Accuracy Score : 0.9183 Precision Score : 0.9183 Recall Score : 0.9183 Mean Absolute Error : 0.2082 F1 Score : 0.9183	Training set score: 1.0000 Test set score: 0.9610 Model Accuracy Score : 0.9610 Precision Score : 0.9610 Recall Score : 0.9610 Mean Absolute Error : 0.1061 F1 Score : 0.9610	Training set score: 1.0000 Test set score: 0.9479 Model Accuracy Score : 0.9479 Precision Score : 0.9479 Recall Score : 0.9479 Mean Absolute Error : 0.1317 F1 Score : 0.9479

CONCLUSIONS

- While comparing the accuracies of all the three models we get Random Forest Classifiers to achieve the highest accuracy then Extra Tree Classifiers and followed by Decision Tree Classifiers.
- Similarly, in cross validation Random Forest Classifiers **(98.31 %)** to achieve the highest cross validation accuracy then Extra Tree Classifiers **(97.74 %)** and followed by Decision Tree Classifiers **(92.33 %).** Since we are getting extremely high accuracies there might be potential overfitting.
- In the 30 % and 60 % data splitting we find training set for all to be high (1.0) whereas the test set scores are lower when compared to the training set score. This indicates there could be potential overfitting.