NEW YORK INSTITUTE OF TECHNOLOGY

PERSONAL PROJECT - 1

College of Engineering & Computing Science

DTSC 620 M01 - Statistics For Data Science

PROJECT ASSIGNMENT - 1

Name:	**AVIVARTTA KRISHNA**
ID:	**1261351**
CLASS:	**DTSC 620 MO1**
DATE:	**11/07/2022**
Professor:	**Dr. Kiran Balgani**

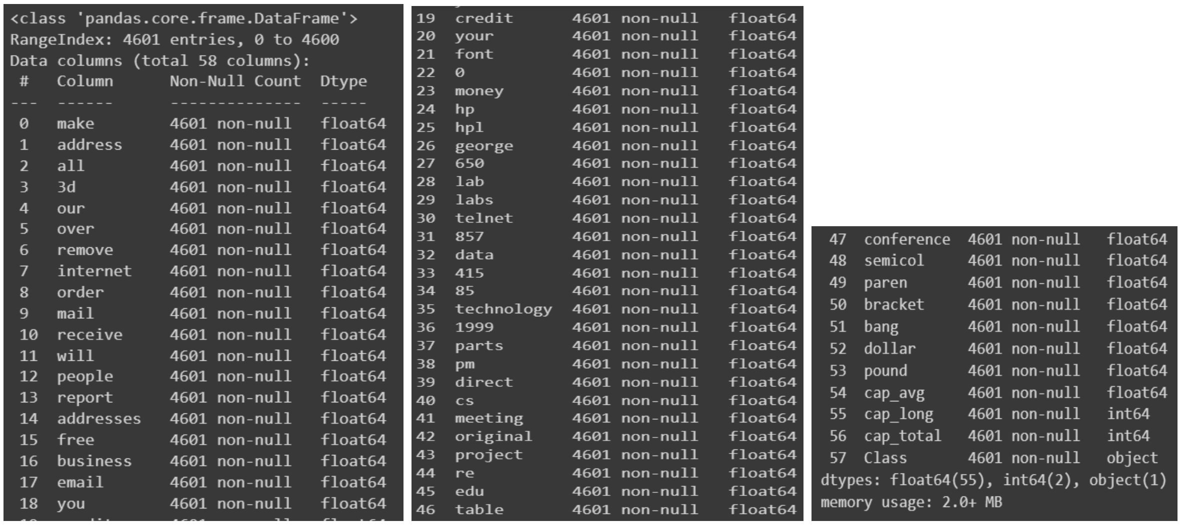
Reporting Tasks:

- **Compare the accuracies of the Random Forest classifier as a function of the number of base learners (e.g., 10, 50, 100, 500, 1000, and 500) and the number of features to consider at each split (e.g., auto or sqrt). Report your observations/conclusions and provide evidence to support your conclusions. [50 points]**

- **Compare the results of all the classifiers (with the best possible parameter setting for each classifier). Use classification accuracy (# of instances correctly classified/total # of instances presented for classification), per class classification accuracy, and confusion matrix to compare the classifiers. [50 points]**

Using spam.info() on the dataset given, we are able to see that there are 58 attributes present with 55 in Float64 data type, 2 in int64 data type and one in Object Data type.

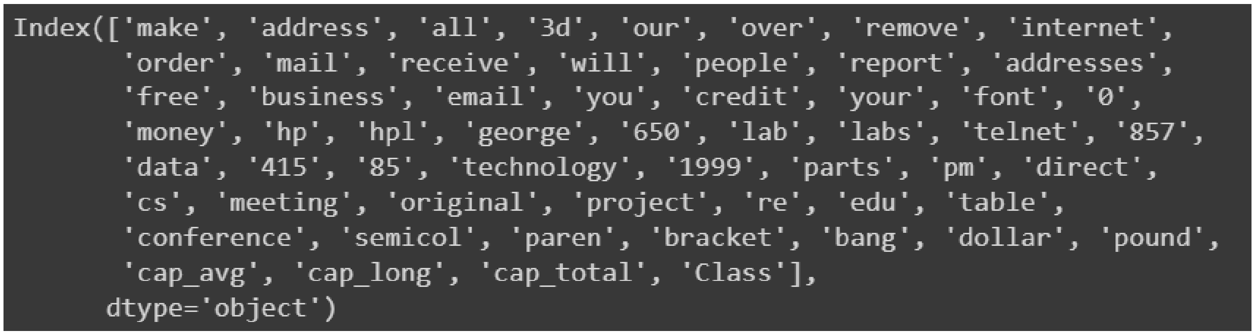
We are also able to see from the figure present below that all attributes are non-null, therefore there is no need for data cleaning.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4601 entries, 0 to 4600
Data columns (total 58 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   make        4601 non-null   float64
 1   address     4601 non-null   float64
 2   all         4601 non-null   float64
 3   3d          4601 non-null   float64
 4   our         4601 non-null   float64
 5   over        4601 non-null   float64
 6   remove      4601 non-null   float64
 7   internet    4601 non-null   float64
 8   order       4601 non-null   float64
 9   mail        4601 non-null   float64
 10  receive     4601 non-null   float64
 11  will        4601 non-null   float64
 12  people      4601 non-null   float64
 13  report      4601 non-null   float64
 14  addresses   4601 non-null   float64
 15  free        4601 non-null   float64
 16  business    4601 non-null   float64
 17  email       4601 non-null   float64
 18  you         4601 non-null   float64
 19  credit      4601 non-null   float64
 20  your        4601 non-null   float64
 21  font        4601 non-null   float64
 22  0           4601 non-null   float64
 23  money       4601 non-null   float64
 24  hp          4601 non-null   float64
 25  hpl         4601 non-null   float64
 26  george      4601 non-null   float64
 27  650         4601 non-null   float64
 28  lab         4601 non-null   float64
 29  labs        4601 non-null   float64
 30  telnet      4601 non-null   float64
 31  857         4601 non-null   float64
 32  data        4601 non-null   float64
 33  415         4601 non-null   float64
 34  85          4601 non-null   float64
 35  technology  4601 non-null   float64
 36  1999        4601 non-null   float64
 37  parts       4601 non-null   float64
 38  pm          4601 non-null   float64
 39  direct      4601 non-null   float64
 40  cs          4601 non-null   float64
 41  meeting     4601 non-null   float64
 42  original    4601 non-null   float64
 43  project     4601 non-null   float64
 44  re          4601 non-null   float64
 45  edu         4601 non-null   float64
 46  table       4601 non-null   float64
 47  conference  4601 non-null   float64
 48  semicol     4601 non-null   float64
 49  paren       4601 non-null   float64
 50  bracket     4601 non-null   float64
 51  bang        4601 non-null   float64
 52  dollar      4601 non-null   float64
 53  pound       4601 non-null   float64
 54  cap_avg     4601 non-null   float64
 55  cap_long    4601 non-null   int64
 56  cap_total   4601 non-null   int64
 57  Class       4601 non-null   object
dtypes: float64(55), int64(2), object(1)
memory usage: 2.0+ MB
```

Figure 1 - Information on each attribute present in dataset

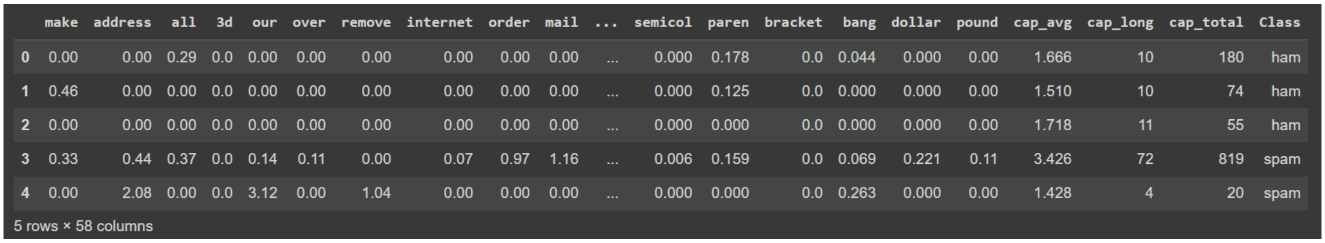
Using print(spam.columns), we are able to see all the column names in an Array index. Visualization of the same is present below in Figure 2.



```
Index(['make', 'address', 'all', '3d', 'our', 'over', 'remove', 'internet',
       'order', 'mail', 'receive', 'will', 'people', 'report', 'addresses',
       'free', 'business', 'email', 'you', 'credit', 'your', 'font', '0',
       'money', 'hp', 'hpl', 'george', '650', 'lab', 'labs', 'telnet', '857',
       'data', '415', '85', 'technology', '1999', 'parts', 'pm', 'direct',
       'cs', 'meeting', 'original', 'project', 're', 'edu', 'table',
       'conference', 'semicol', 'paren', 'bracket', 'bang', 'dollar', 'pound',
       'cap_avg', 'cap_long', 'cap_total', 'Class'],
      dtype='object')
```

Figure 2 - Index array on all column names present in dataset.

Using spam.head(), we are able to see the first five rows of the dataset with all the information present.



	make	address	all	3d	our	over	remove	internet	order	mail	...	semicol	paren	bracket	bang	dollar	pound	cap_avg	cap_long	cap_total	Class
0	0.00	0.00	0.29	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.178	0.0	0.044	0.000	0.00	1.666	10	180	ham
1	0.46	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.125	0.0	0.000	0.000	0.00	1.510	10	74	ham
2	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.000	0.000	0.00	1.718	11	55	ham
3	0.33	0.44	0.37	0.0	0.14	0.11	0.00	0.07	0.97	1.16	...	0.006	0.159	0.0	0.069	0.221	0.11	3.426	72	819	spam
4	0.00	2.08	0.00	0.0	3.12	0.00	1.04	0.00	0.00	0.00	...	0.000	0.000	0.0	0.263	0.000	0.00	1.428	4	20	spam

5 rows × 58 columns

Figure 3 - first five columns of the dataset

REPORT - DECISION TREE

- In the Decision tree, we took the attribute Class as the Target Variable and every other attribute as the feature variable.
- We split the dataset into a training set and Testing Set in the ratio of 78.26% in the testing set which is approximately ~ 3601instanses and the other 1000 or 21.74% instances in the training set.
- We then created the Decision Tree Classifier. And printed the accuracy.
- Accuracy of Decision tree before optimization :

  Accuracy: 0.8850319355734518 ~ 88.5%
- Below I have given the visualization of the decision tree before optimization.

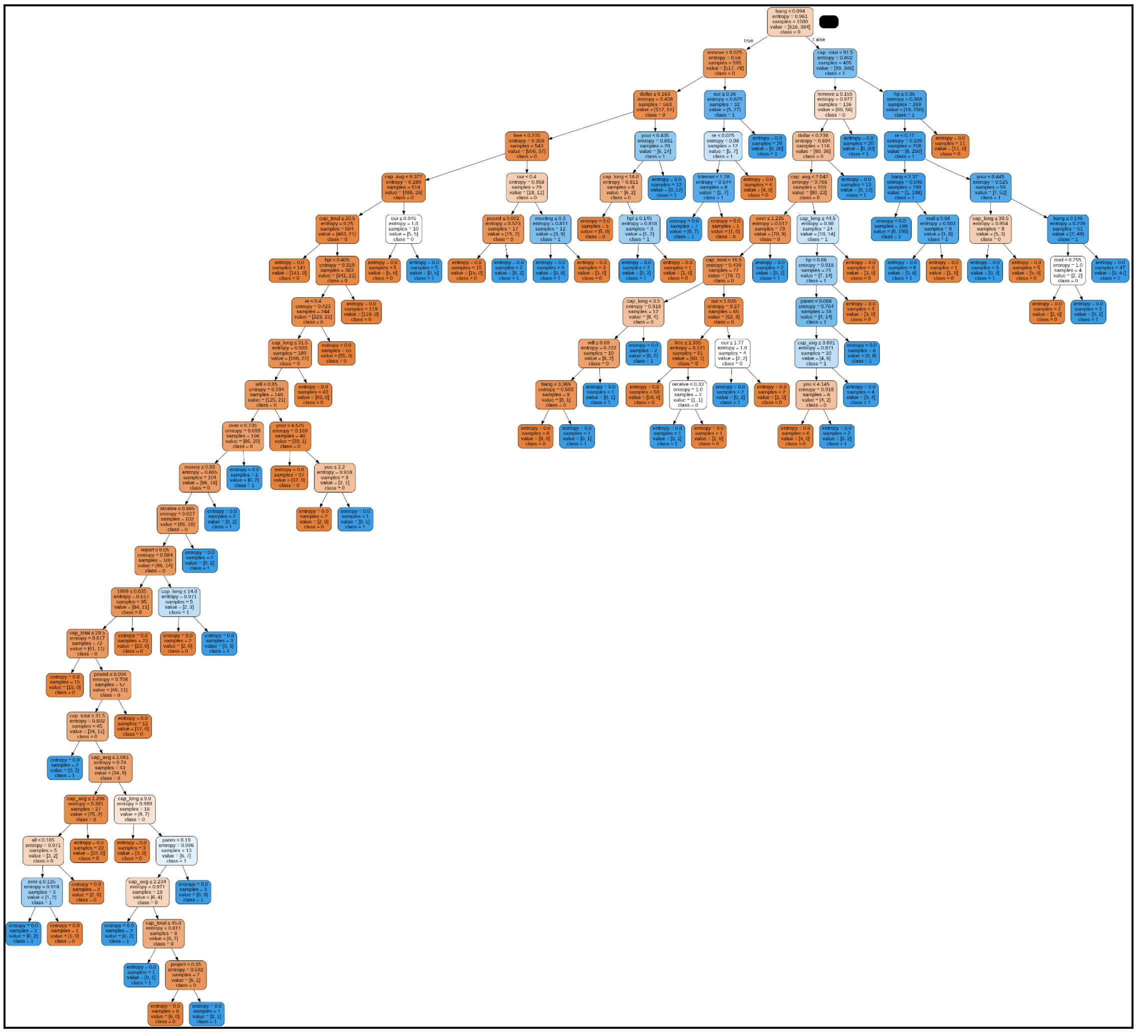


Figure 4 - Decision Tree before Optimization

Optimizing the Decision Tree

- I tried to optimize the decision tree by introducing a new criterion called entropy and trying out all the max_depth in estimation. The best optimized tree was found at max_depth = 30.
- Accuracy: 0.8733685087475701 ~ 87.3%

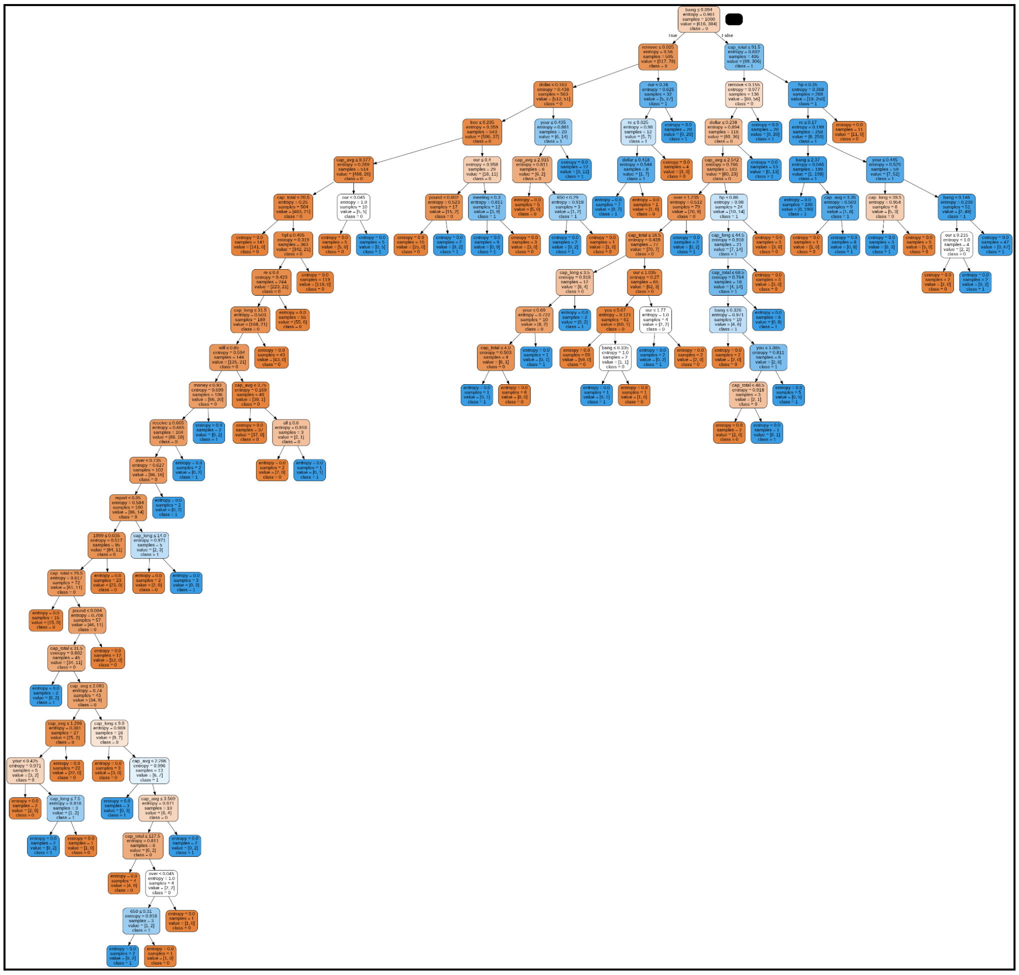
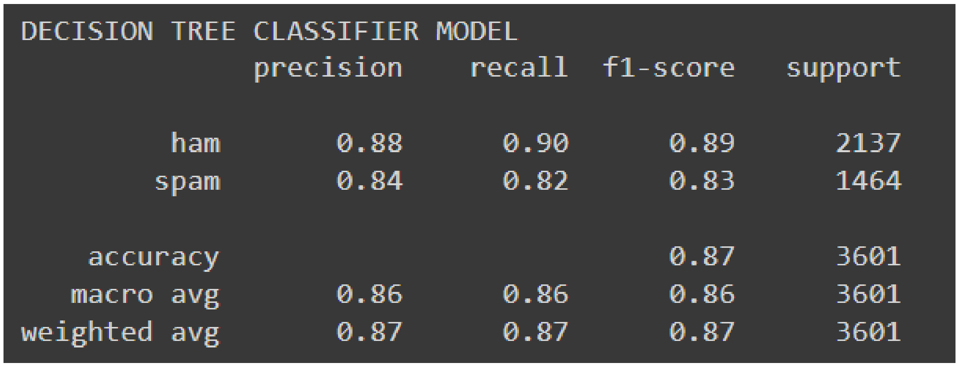


Figure 5 - Decision Tree after Optimization (very minor changes made)

Final Results for Decision Tree



```
DECISION TREE CLASSIFIER MODEL
              precision    recall  f1-score   support

         ham       0.88      0.90      0.89      2137
        spam       0.84      0.82      0.83      1464

    accuracy                           0.87      3601
   macro avg       0.86      0.86      0.86      3601
weighted avg       0.87      0.87      0.87      3601
```

Figure 6 - Confusion Matrix for my Decision Tree

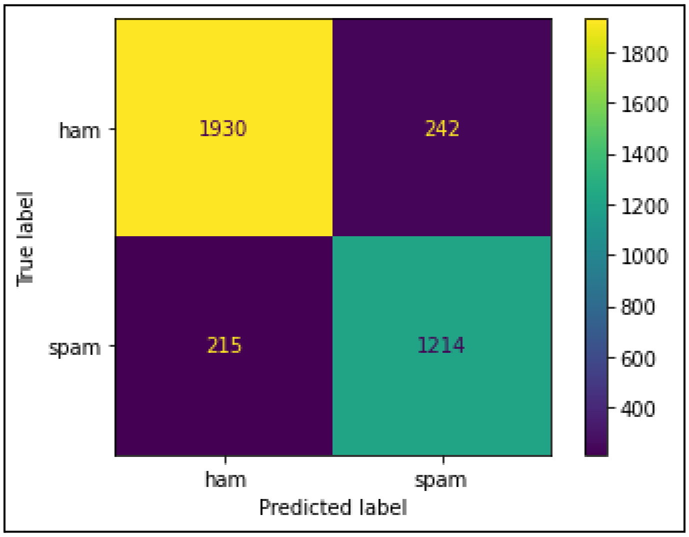


Figure 7 - Predicted Label for Spam Dataset result by Decision Tree

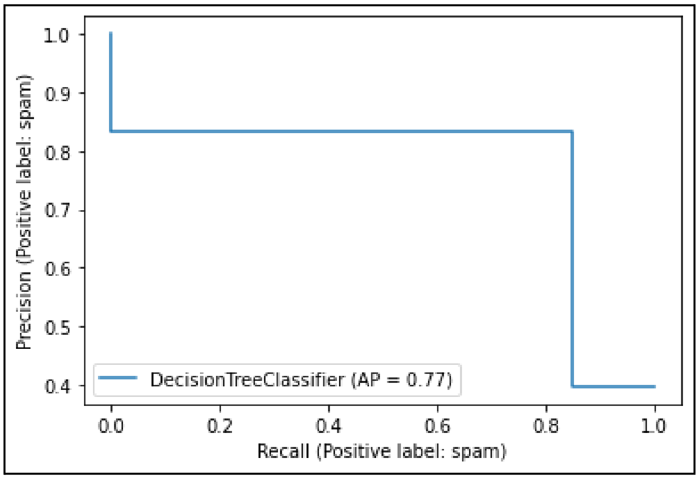


Figure 8 - Precision VS Recall of Decision Tree

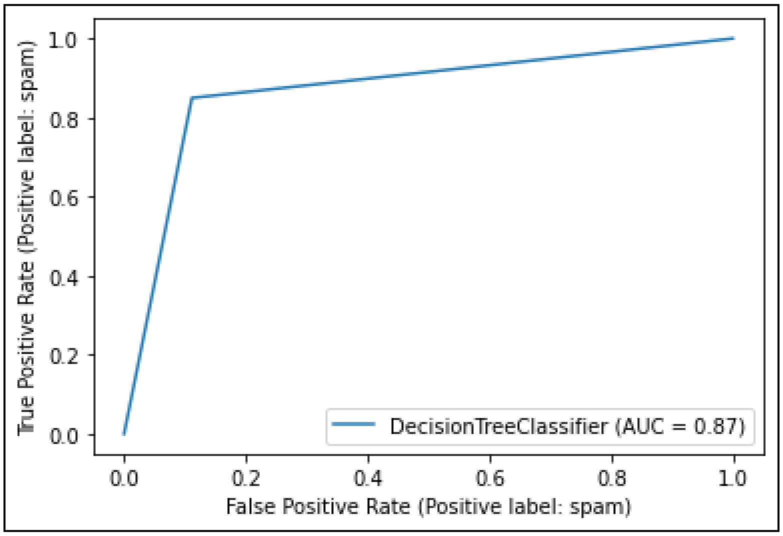


Figure 9 - True positive rate vs False positive rate for Positive Label of Spam

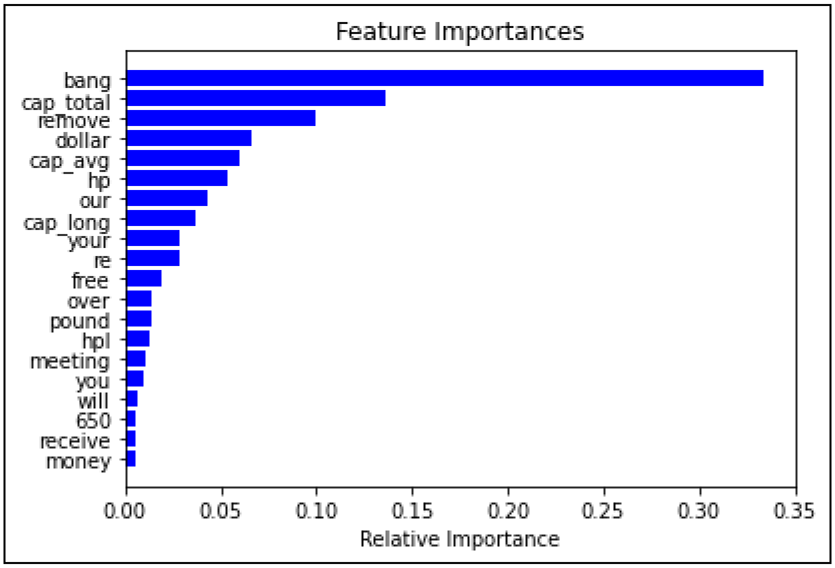


Figure 10 - Feature Importance of top 20 attributes/instances

REPORT - RANDOM FOREST CLASSIFIER

- Selecting a model with the highest accuracy given a list of base learners
  Estimators give are = {10, 50, 100, 500, 1000, 5000}
  Resultant Information:

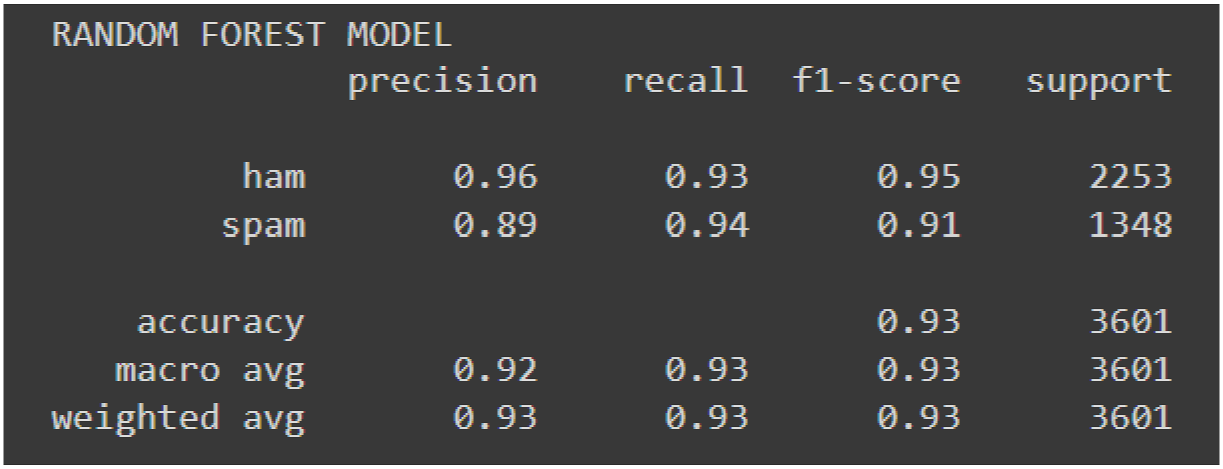
Estimator	**Accuracy**
10	0.9255762288253263~ 92.55%
50	0.927797833935018 ~ 92.77%
100	0.9322410441544016 ~ 93.2%
500	0.9327964454318245 ~ 93.27%
1000	0.932518744793113 ~ 93.25%
5000	*0.9344626492640933 ~ 93.44%*

Highest Accuracy = 5000

- Selecting a model with the highest accuracy given a list of Features
  Features I Chose = {None, Auto, Sqrt, Log2}
  Resultant Information

Feature	**Accuracy**
None	0.9336295473479589 ~ 93.36%
Auto	0.9327964454318245 ~ 93.27%
Sqrt	0.9305748403221328 ~ 93.05%
Log2	*0.932518744793113 ~ 93.75%*

Highest Accuracy = Log2



```
RANDOM FOREST MODEL
              precision    recall  f1-score   support

         ham       0.96      0.93      0.95      2253
        spam       0.89      0.94      0.91      1348

    accuracy                           0.93      3601
   macro avg       0.92      0.93      0.93      3601
weighted avg       0.93      0.93      0.93      3601
```

Figure 11 - Confusion Matrix for Random Forest

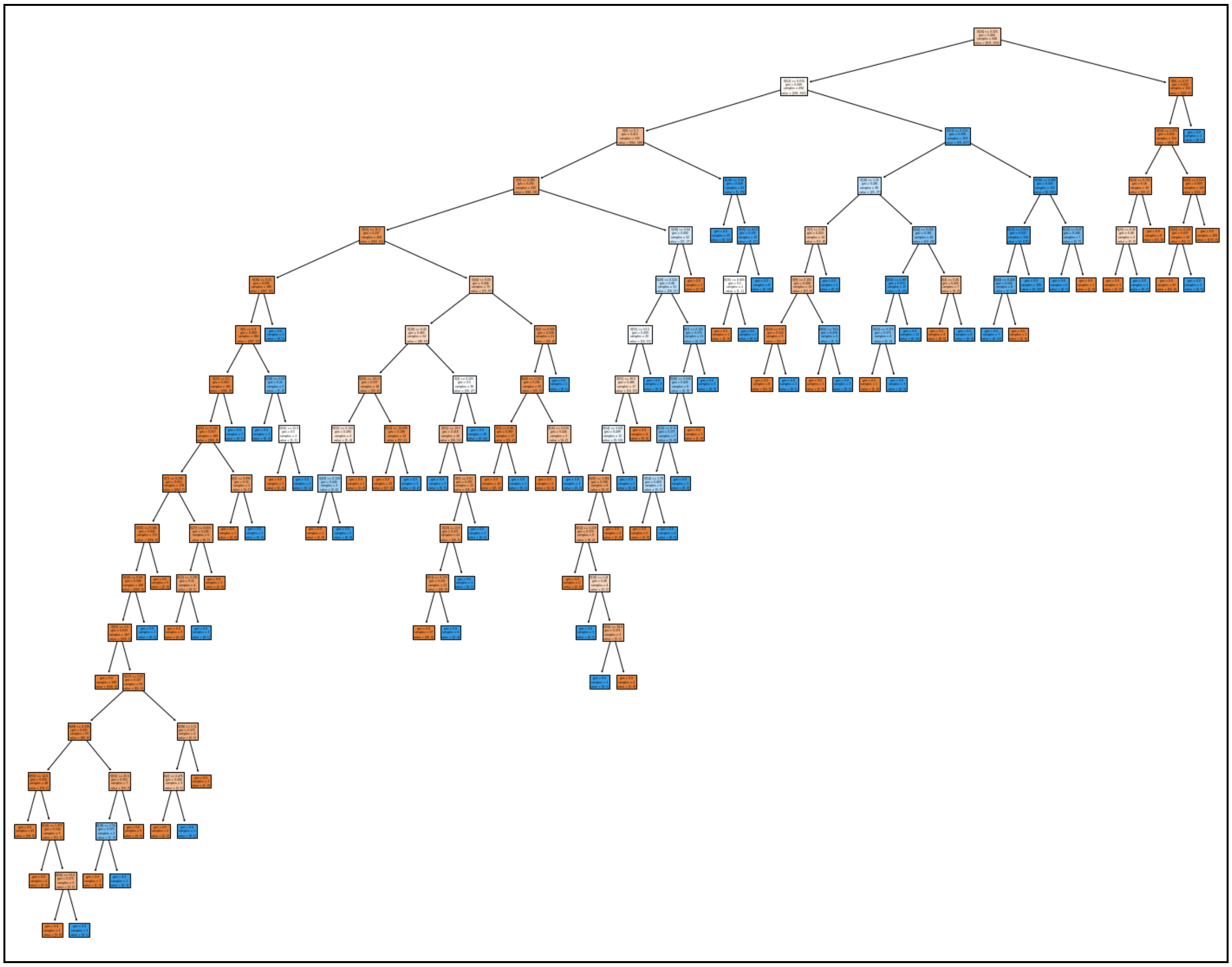


Figure 12 - Random Forest Classifier Visualization

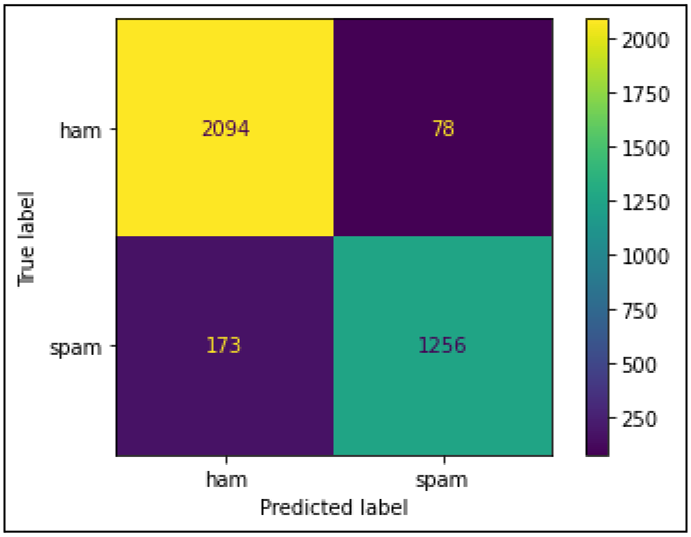


Figure 13 - Predicted Label for Spam Dataset result by Random Forest Classifier

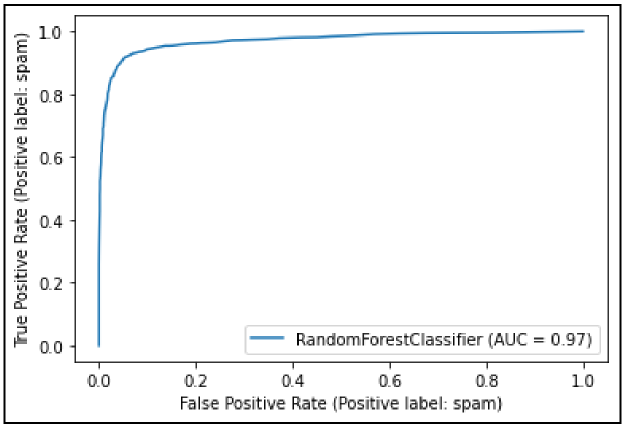


Figure 14 - True Positive VS False Positive for Positive Label of Spam

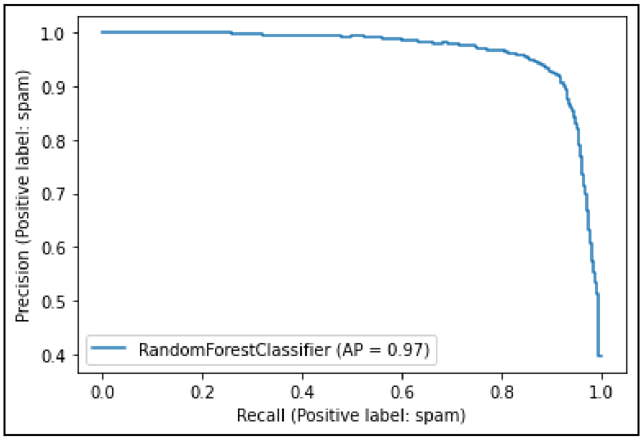


Figure 15 - Precision VS Recall for Positive Label - Spam

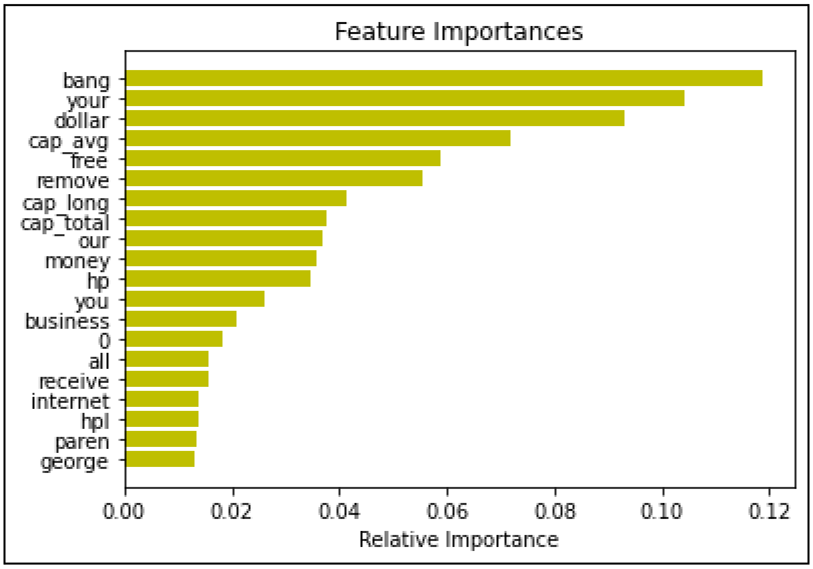


Figure 10 - Feature Importance of top 20 attributes/instances

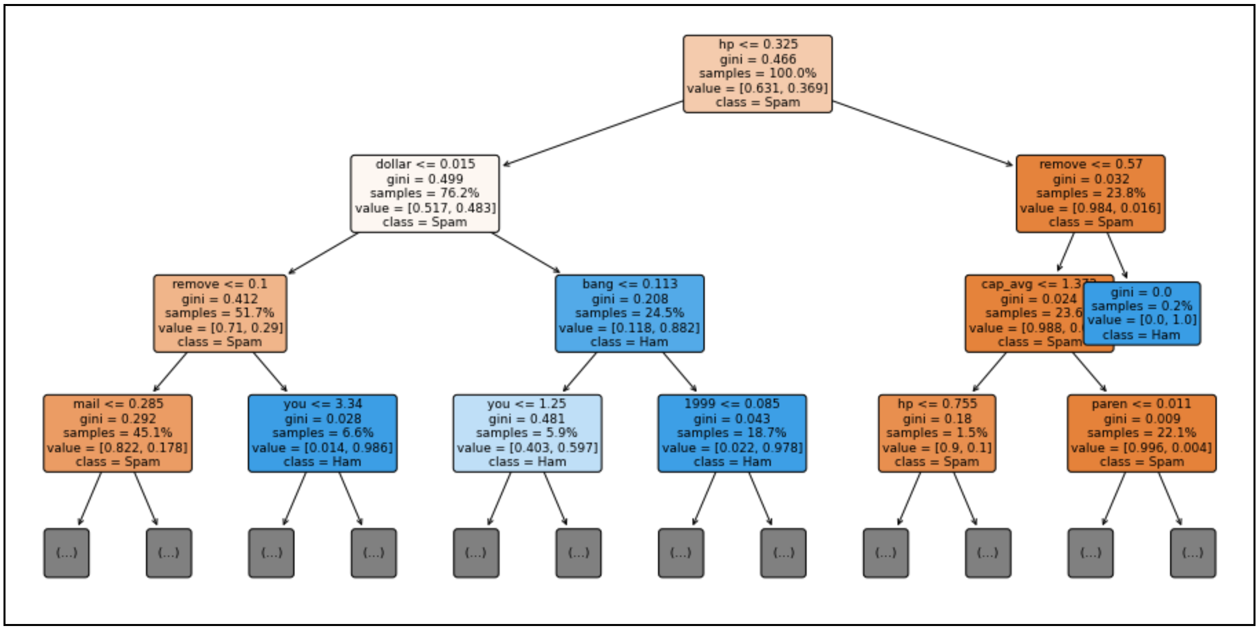


Figure 11 - Optimized Random Forest

FINAL STATEMENT : The Random Forest Classifier has a better performance since it classified 2094 data points correctly as opposed to the decision tree which only classified 1930 data points correctly.

Link to Google Colab CODE:
https://colab.research.google.com/drive/1g0_n5_YsMR2tBWrCS7ZTC_N_fZ2JKLwH
