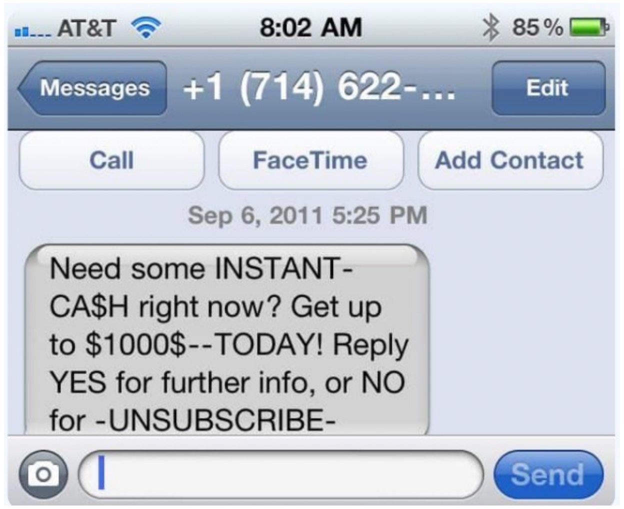
Capstone Project #2 : Machine Learning/ NLP
SMS spam detection models



Sep 6, 2011 5:25 PM

Need some INSTANT-CA$H right now? Get up to 1000--TODAY! Reply YES for further info, or NO for -UNSUBSCRIBE-

Built a binary classification model to detect whether a text message is spam or not and used Natural language processing (NLP) to get valuable insights. **NLP is the field concerned with the ability of a computer to understand, analyze, manipulate, and potentially generate human language**.

February 18, 2022

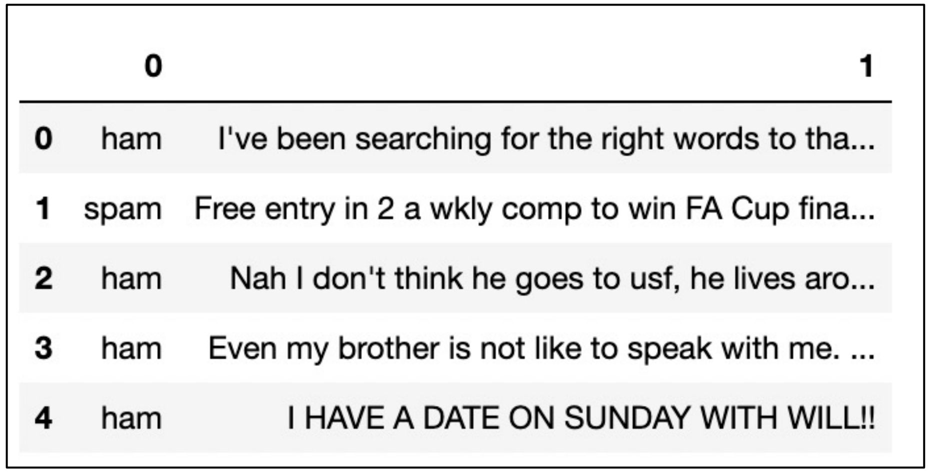
Manika Midha

Machine Learning Pipeline

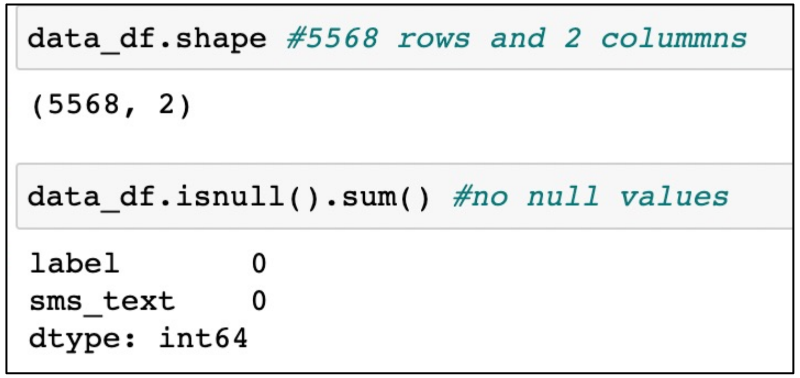
1. Read in raw text
2. Data analysis
3. Clean text and tokenize
4. Feature Engineering
5. Fit base models, evaluate based on precision, recall, AUC and select top two models
6. Tune hyperparameters and evaluate with GridsearchCV
7. Final model selection

Data Exploration

The SMS Spam Collection Data Set is obtained from UCI Machine Learning Repository. It is a set of 5568 tagged English messages- ham(legitimate) or spam that have been collected for SMS Spam research.



	0	1
0	ham	I've been searching for the right words to tha...
1	spam	Free entry in 2 a wkly comp to win FA Cup fina...
2	ham	Nah I don't think he goes to usf, he lives aro...
3	ham	Even my brother is not like to speak with me. ...
4	ham	I HAVE A DATE ON SUNDAY WITH WILL!!



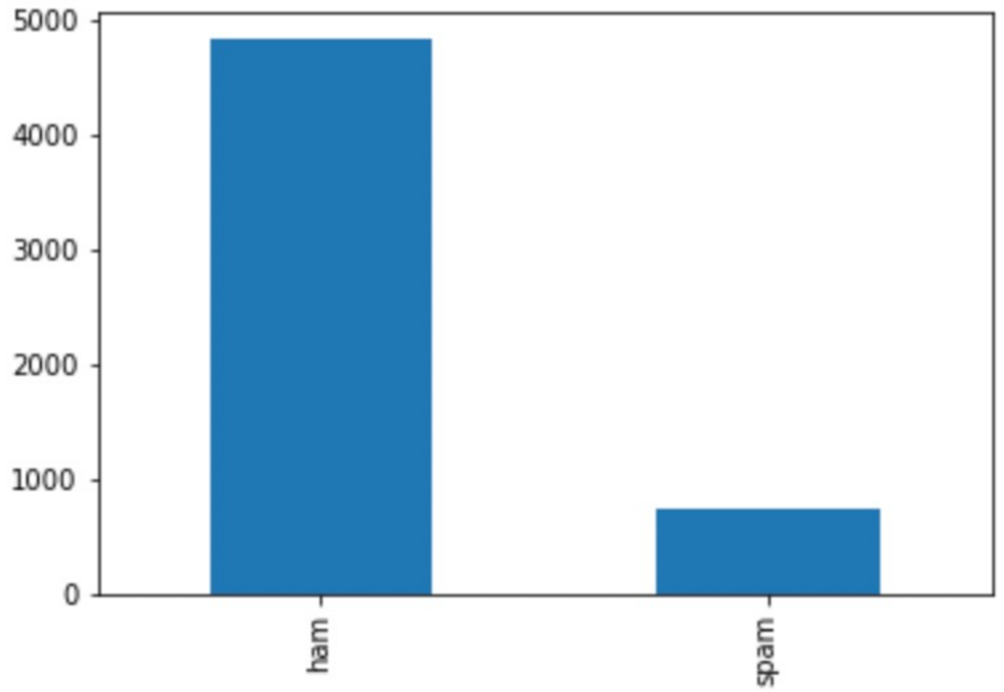
```
data_df.shape #5568 rows and 2 colummns
```

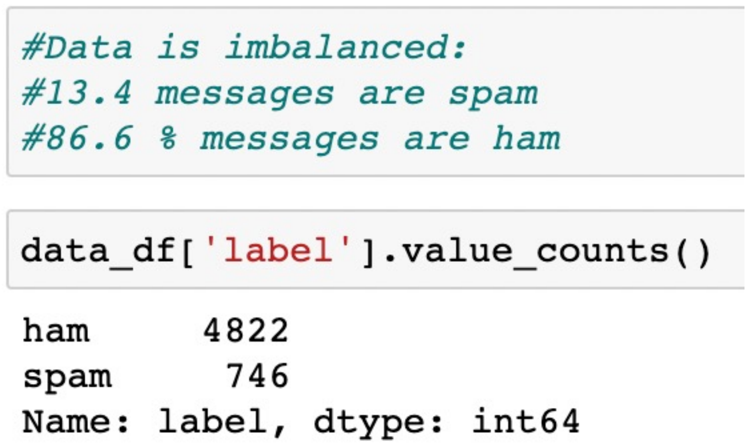
```
(5568, 2)
```

```
data_df.isnull().sum() #no null values
```

```
label       0
sms_text    0
dtype: int64
```

Imbalanced Dataset



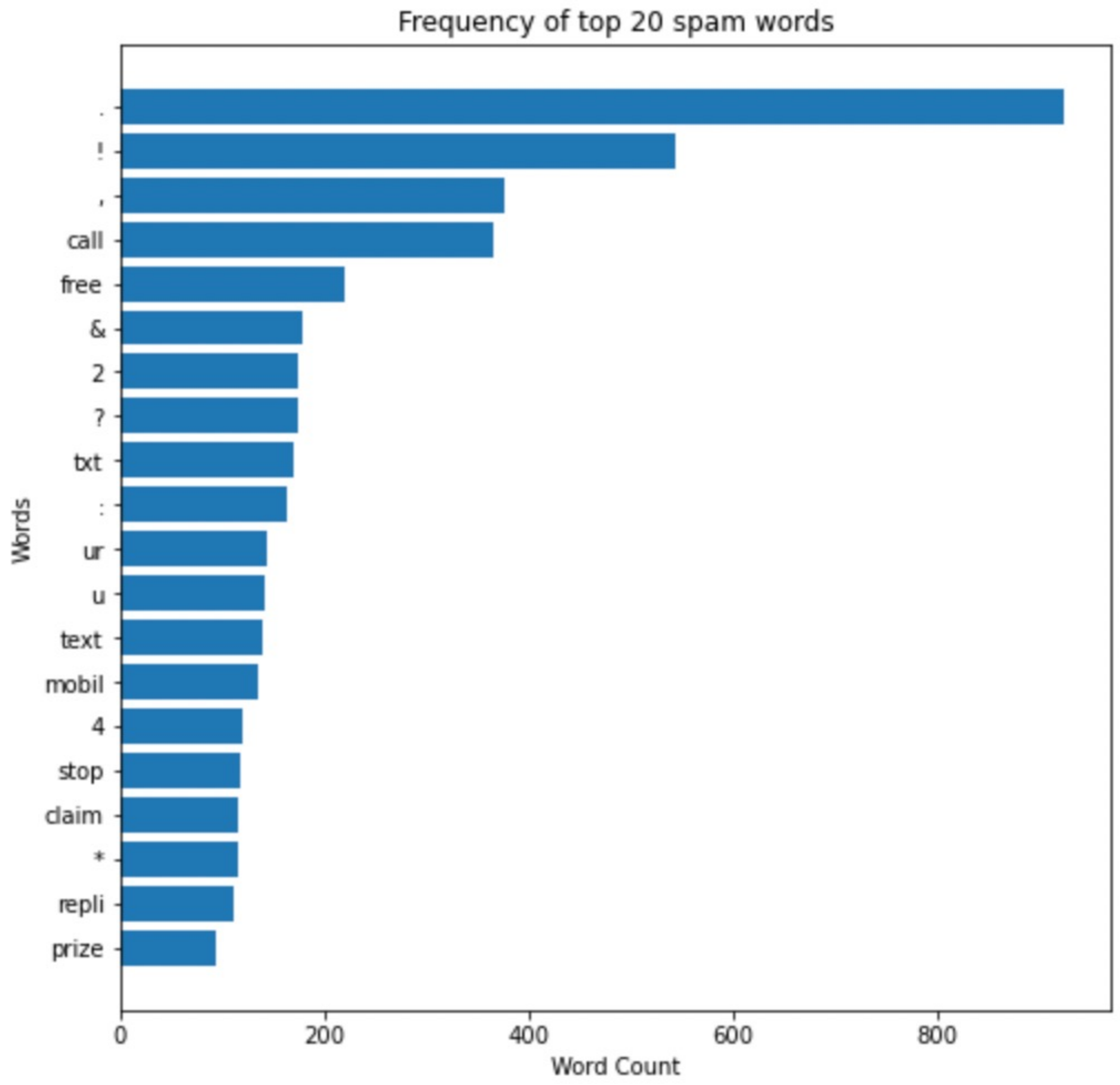


```
#Data is imbalanced:
#13.4 messages are spam
#86.6 % messages are ham
```

```
data_df['label'].value_counts()
```

```
ham     4822
spam     746
Name: label, dtype: int64
```

```
[('.', 924), ('!', 543), (',', 377), ('call', 365), ('free', 219), ('&', 178), ('2', 174), ('?', 174), ('txt', 169),
(':', 163), ('ur', 144), ('u', 142), ('text', 139), ('mobil', 135), ('4', 120), ('stop', 118), ('claim', 115), ('*',
115), ('repli', 112), ('prize', 94)]
```



Punctuations like . ! , & ? : * are common
Words like call, free, text, claim, prize are common

Text converted to lower case, stopwords removed, and words stemmed

Big Picture view

- Classification Algorithms : Logistic Regression, K Nearest Neighbours, Decision Trees, Random Forest , and Gradient Boosting
- The dataset was divided into 80% training and 20% test data
- Cross Validation score, AUC, Precision, Recall,F1 score, Fit and Prediction time were used for evaluation of performance
- Libraries used – Scikit-Learn, NLTK, Pandas, Matplotlib
- Code was written in Jupyter Notebook

Data Preprocessing

Hypotheses for feature engineering :

- Spam messages tend to be longer
- They tend to contain more digits, punctuations and capital letters
- They tend to contain urls

Steps :

1. Punctuation removed
2. Data converted to lower case and tokenized
3. Stopwords removed
4. Stemming applied (Porter Stemmer from nltk)
5. Feature Engineering
6. Vectorisation (Tfid Vectorizer)

Tokenization :
Split a string into a list of words.

Stopwords :
Commonly used words like the, it, but that do not contribute much to the meaning of a sentence.

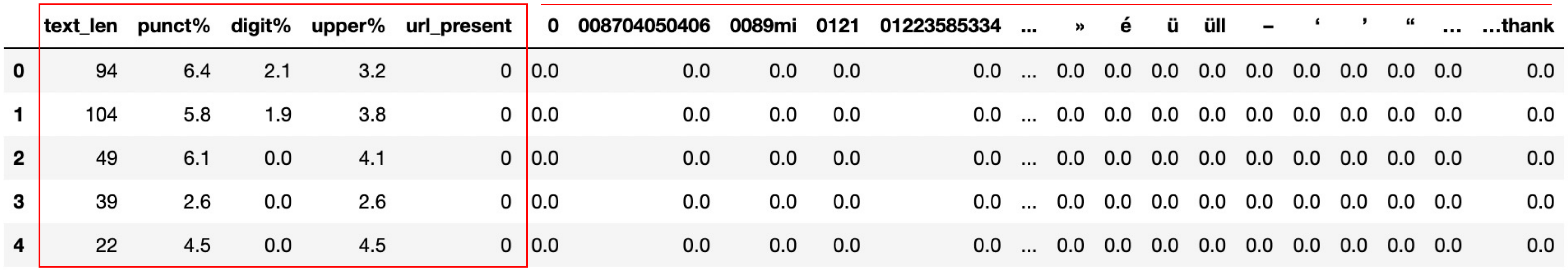
Stemming :
Process of reducing derived words to their word root. Stemming reduces variations of the same word. Eg: Stemming and stemmed will be reduced to word : stem.

Vectorizing :
Process of encoding text as numbers to create feature vectors. Feature vector is an n-dimensional vector of numerical features that Represent some object. Raw test needs to be converted to numbers for Python/ Machine learning to understand.

Inverse document frequency weighting : TF – IDF
Creates a document matrix where each row represents an sms and each column represents a single unique word. Each cell represents a weighting which identifies how important a word is to an individual
sms. The rarer a word is, higher is the weight. If a word occurs frequently within an sms but infrequently elsewhere a higher weight will be assigned. And this word will be important in distinguishing this sms from others. This method helps pull out important but seldom used words.

Feature engineering

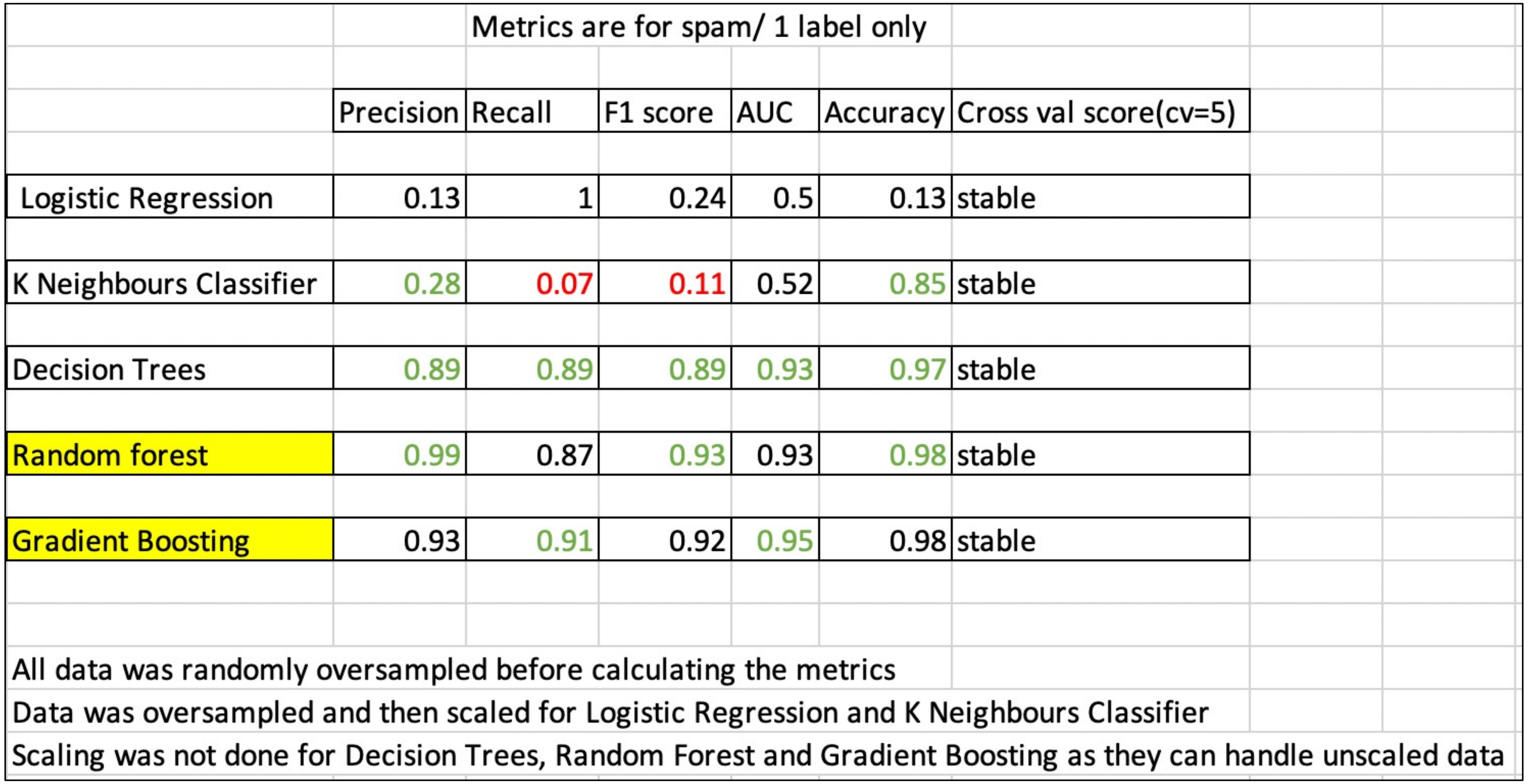
Vectorizing



	text_len	punct%	digit%	upper%	url_present	0	008704050406	0089mi	0121	01223585334	...	»	é	ü	üll	–	‘	’	“	…	…thank
0	94	6.4	2.1	3.2	0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	104	5.8	1.9	3.8	0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	49	6.1	0.0	4.1	0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	39	2.6	0.0	2.6	0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	22	4.5	0.0	4.5	0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

7285 features/ columns

Classification Results of Classifiers



Metrics are for spam/ 1 label only

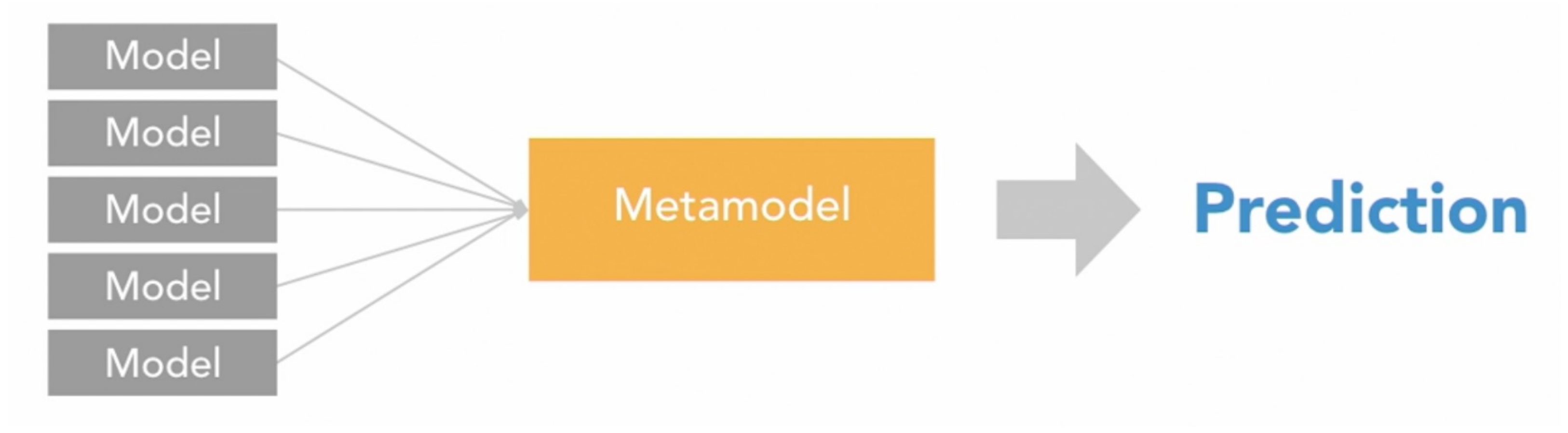
	Precision	Recall	F1 score	AUC	Accuracy	Cross val score(cv=5)
Logistic Regression	0.13	1	0.24	0.5	0.13	stable
K Neighbours Classifier	0.28	0.07	0.11	0.52	0.85	stable
Decision Trees	0.89	0.89	0.89	0.93	0.97	stable
Random forest	0.99	0.87	0.93	0.93	0.98	stable
Gradient Boosting	0.93	0.91	0.92	0.95	0.98	stable

All data was randomly oversampled before calculating the metrics

Data was oversampled and then scaled for Logistic Regression and K Neighbours Classifier

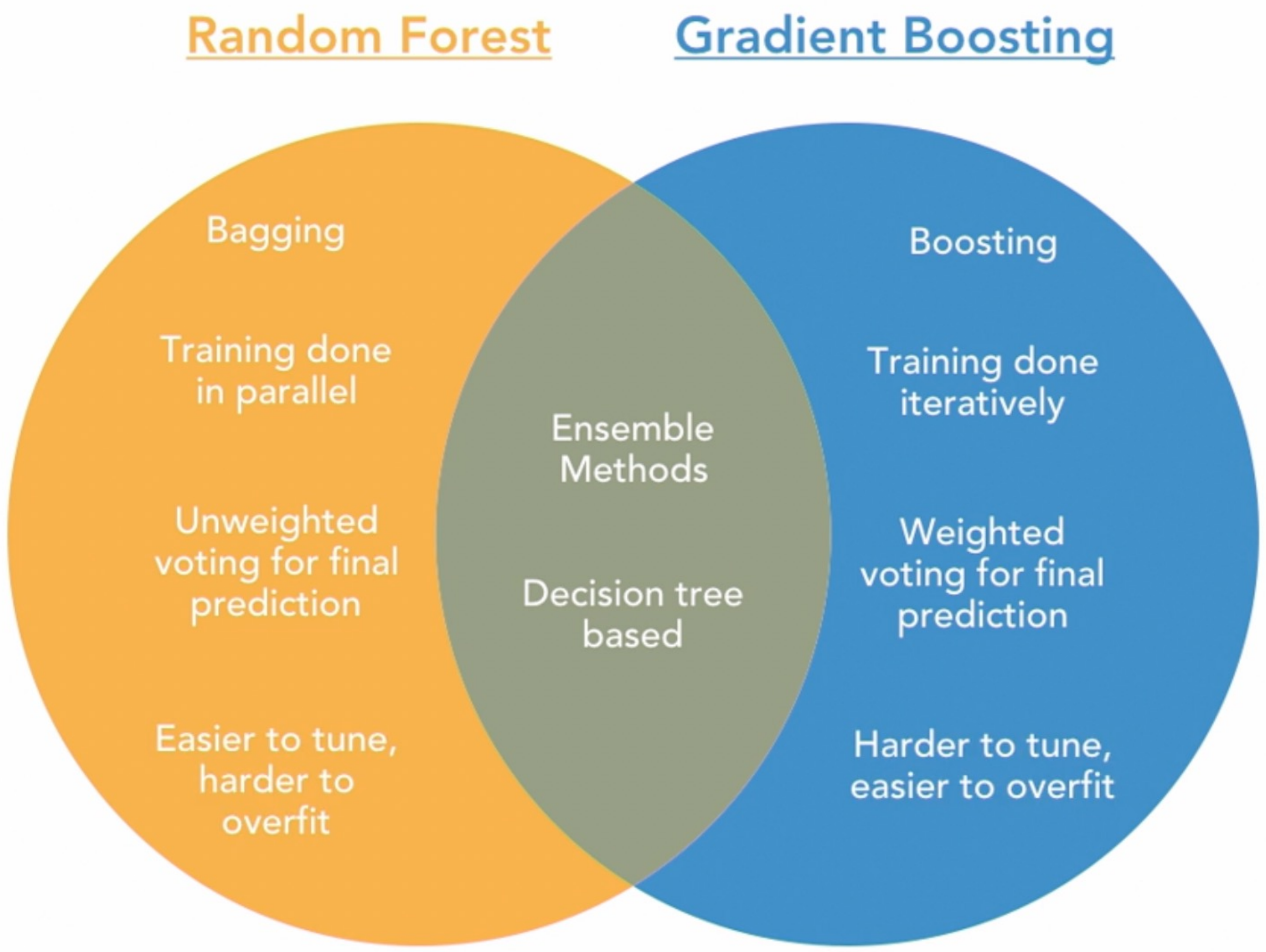
Scaling was not done for Decision Trees, Random Forest and Gradient Boosting as they can handle unscaled data

Ensemble Method : Technique that creates multiple models and then combines them to produce better results than any of the single models individually. Thus, we use the aggregated opinion of many over the isolated opinion of one model. Both, Random Forest and Gradient Boosting are ensemble methods.



Random Forest constructs a collection of decision trees (weak individual trees) simultaneously and then aggregates the predictions of each tree to determine the final prediction using a voting method. RF can handle outliers, missing values and unscaled data.

Gradient Boosting takes an iterative approach to combining weak learners to create a strong learner by focusing on mistakes of prior Iterations. It uses decision trees as well. It evaluates what it got right or wrong on the first tree. In the next iteration, t places a heavier weight on what it got wrong. It does this over and over again, focusing on examples it doesn’t understand yet till it minimizes the error as much as possible.


Random Forest
Gradient Boosting
Bagging
Training done in parallel
Unweighted voting for final prediction
Easier to tune, harder to overfit
Ensemble Methods
Decision tree based
Boosting
Training done iteratively
Weighted voting for final prediction
Harder to tune, easier to overfit

Similarities :

- Both are ensemble methods that use decision trees

Differences :

- RF uses bagging while GB uses boosting. Bagging samples randomly while boosting samples with an increased weight on ones it got wrong previously.

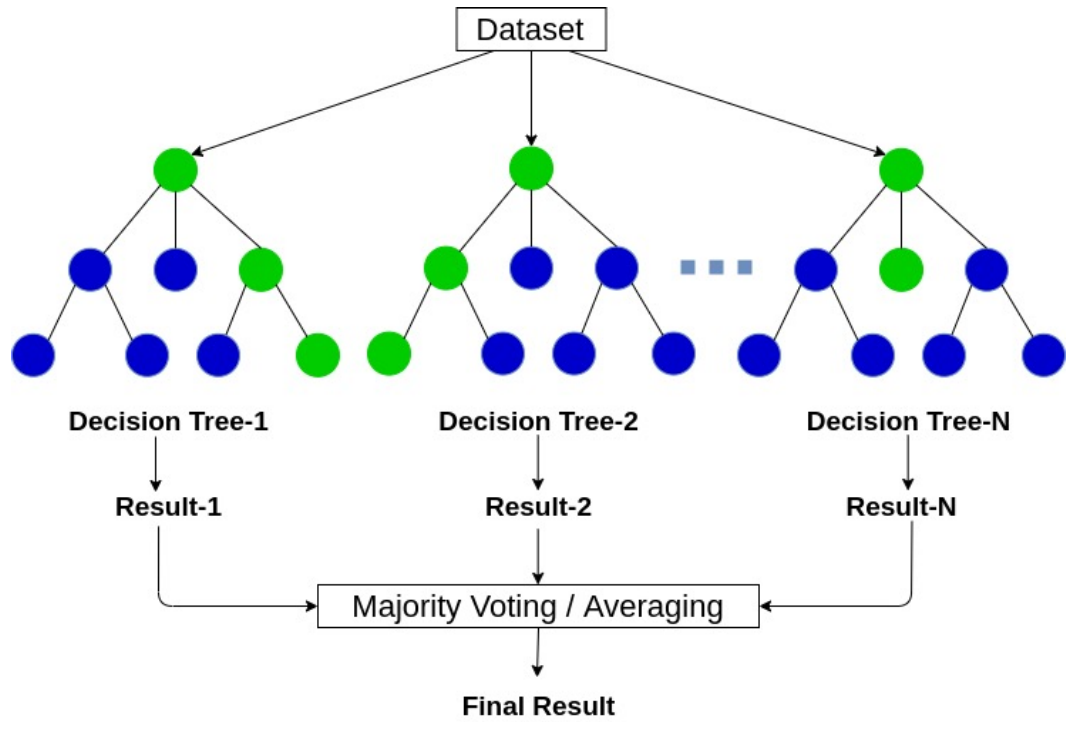
- In RF, as trees are built without any consideration to other trees , it is easy to parallelize. Thus, can train quickly. GB is iterative. It relies on the results of the tree before it to apply a higher weight to the ones that the previous trees got incorrect. Thus, it cannot be parallelized and takes longer time to train.

- For RF, the final predictions are an unweighted/ average voting while for GB, they are weighted.

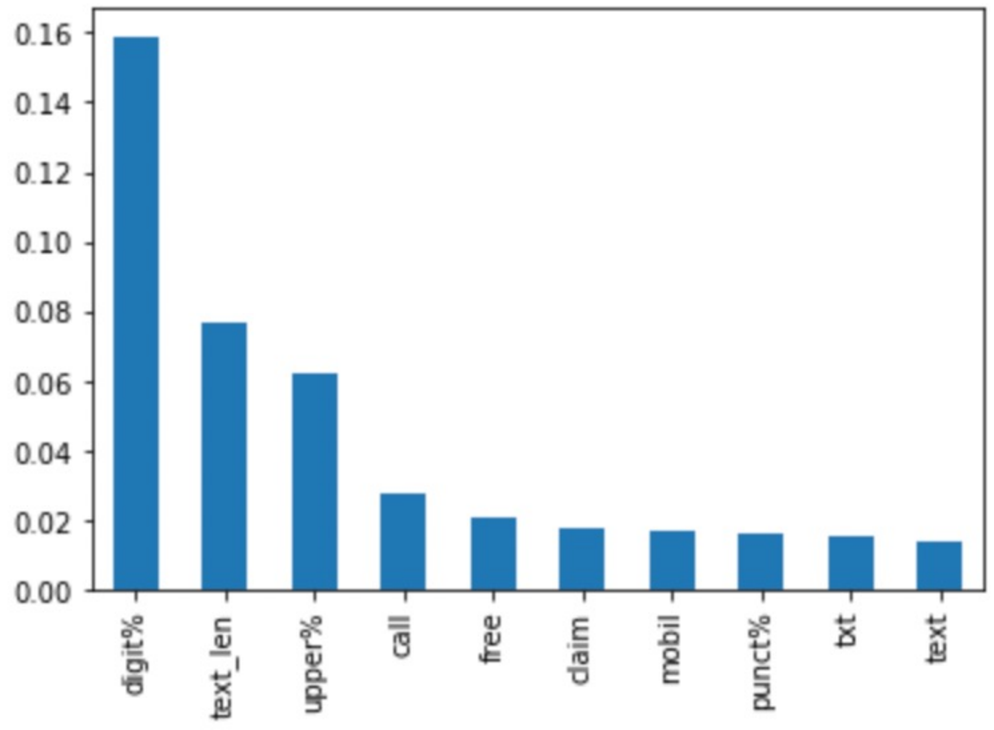
- GF is harder to tune as it has more parameters.

GB is typically more powerful and better performing, if tuned properly.

Random Forest Classifier - Evaluation

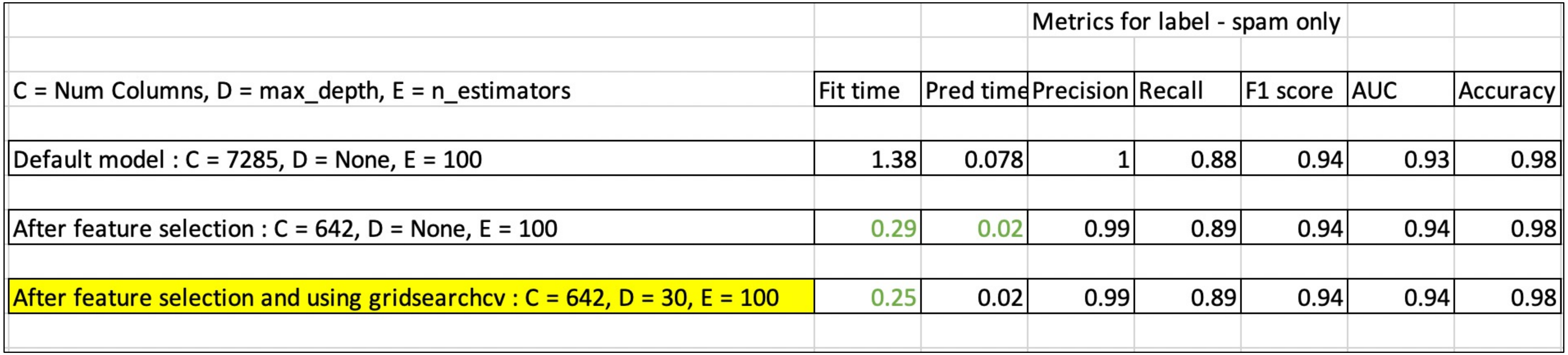


Feature Importance – top ten features



These features are most important when distinguishing between spam and ham. As a business owner, I do not want my well intentioned sms (promotions, response to an existing customer) to be treated as spam. So, I should be careful to keep above results in mind when drafting messages. Number of digits, uppercase letters, punctuations, and text length should be moderate. I should avoid using words like ‘free’, ‘call’, ‘claim’ in my messages.

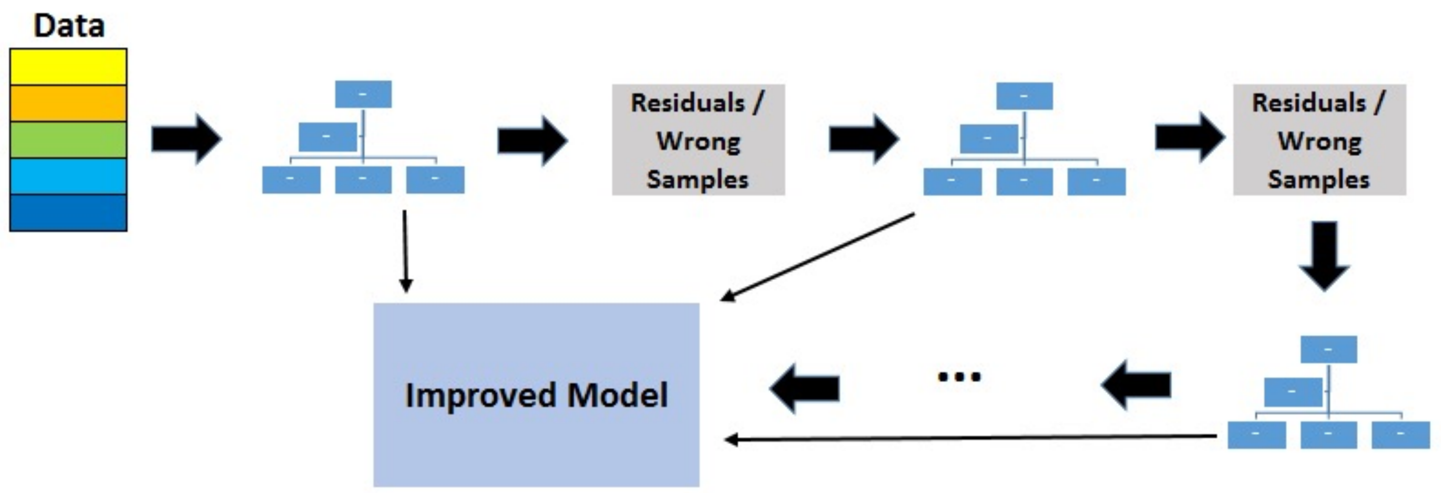
Evaluation of different Random Forest Classifier models



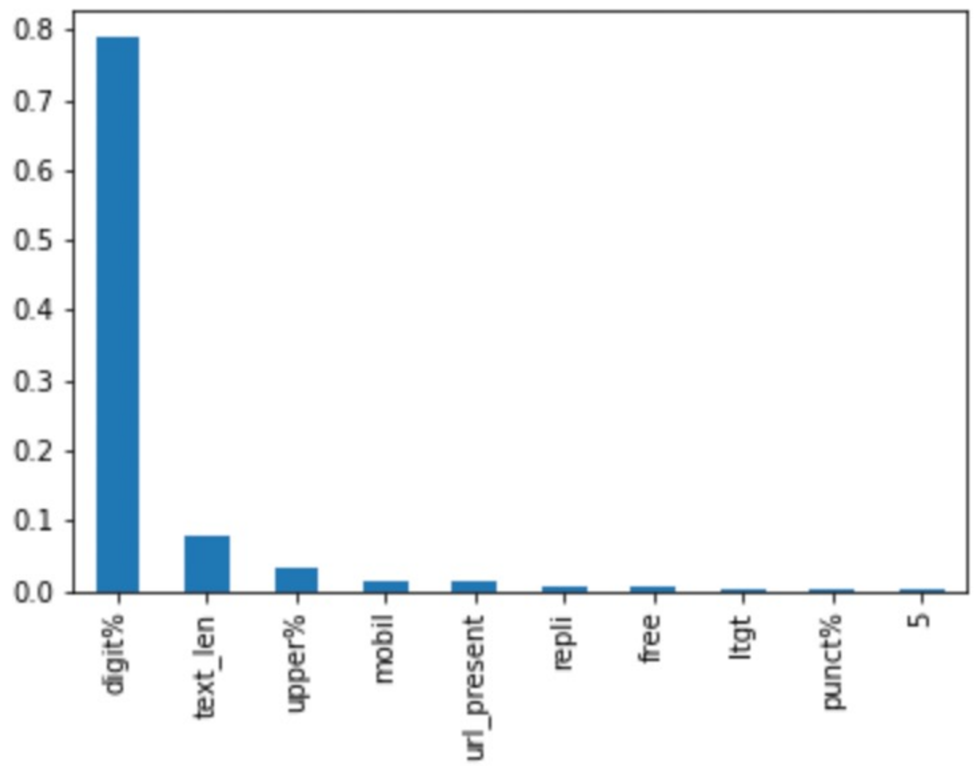
			Metrics for label - spam only				
C = Num Columns, D = max_depth, E = n_estimators	Fit time	Pred time	Precision	Recall	F1 score	AUC	Accuracy
Default model : C = 7285, D = None, E = 100	1.38	0.078	1	0.88	0.94	0.93	0.98
After feature selection : C = 642, D = None, E = 100	0.29	0.02	0.99	0.89	0.94	0.94	0.98
After feature selection and using gridsearchcv : C = 642, D = 30, E = 100	0.25	0.02	0.99	0.89	0.94	0.94	0.98

- Best Features : Out of 7285 columns, 642 columns whose feature importance was greater than or equal to the mean of feature importance of all columns, were selected.
- Number of False Positives were 0,1,1 for all above models respectively. A correct message being misclassified as spam is a False Positive. They should be as less as possible and Precision as high as possible.
- All models do a great job of distinguishing between classes correctly (AUC ~ .94)

Gradient Boosting Classifier - Evaluation



Feature Importance – top ten features



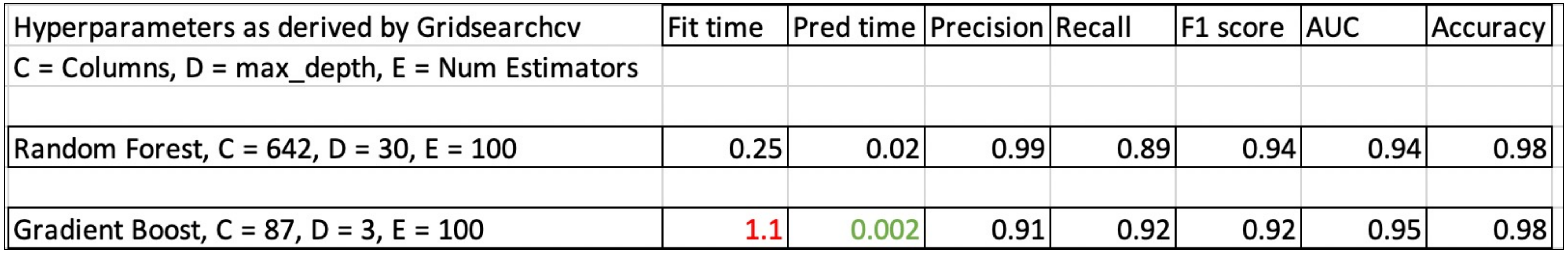
Top three features – digit%, text_len and upper% are same as that for Random Forest Classifier

Evaluation of different Gradient Boosting Classifier models

			Metrics for label - spam only				
C = Num Columns, D = max_depth, E = n_estimators	Fit time	Pred time	Precision	Recall	F1 score	AUC	Accuracy
Default model : C = 7285, D = 3, E = 100	29	0.12	0.91	0.91	0.91	0.94	0.98
After feature selection : C = 87, D = 3, E = 100	0.75	0.002	0.92	0.92	0.92	0.95	0.98
After feature selection and using gridsearchcv : C = 87, D = 3, E = 150	1.1	0.002	0.91	0.92	0.92	0.95	0.98

- Best Features : Out of 7285 features, 87 features whose feature importance was greater than or equal to the mean of feature importance of all columns, were selected.
- Number of False Positives were 12,12,12 for all above models respectively. A correct message being misclassified as spam is a False Positive. It should be as less as possible and Precision as high as possible.
- All models do a great job of distinguishing between classes correctly (AUC ~ .95)

Final Model Evaluation



Hyperparameters as derived by Gridsearchcv	Fit time	Pred time	Precision	Recall	F1 score	AUC	Accuracy
C = Columns, D = max_depth, E = Num Estimators							
Random Forest, C = 642, D = 30, E = 100	0.25	0.02	0.99	0.89	0.94	0.94	0.98
Gradient Boost, C = 87, D = 3, E = 100	1.1	0.002	0.91	0.92	0.92	0.95	0.98

Both models have similar performance metrics. If prediction time is a deal breaker, Gradient Boost is better. If time is not an issue, go with Random Forest as precision is higher. We don’t want a higher number of False positives as we do not want our correct sms to be classified as spam. We are ok if few spam messages make their way to the inbox (FN can be a little bit high, less recall).