Aalto University
School of Science

# Machine Learning in Email Marketing

Muhammad Kamal Memon
February 2019

**CS-E4870 - Research Project in Machine Learning and Data Science**
Neda Barzegar Marvasti

# Table of Contents

# Introduction

Email marketing is an important tool used by almost all established companies today as a communication mechanism to target their customers with specific content. Sending emails is fast, cheap and highly targeted and enables companies to push the content they want to their customer base. One big advantage in this approach is that related statistics can be gathered directly concerning user responses to these emails such as how many opened and followed through to a link, which users remained idle and who opted to quit the mailing list. These statistics are important as they reveal quantifiable metrics such as conversion rate and unsubscription ratios. Email marketing is prevalent across all industries and is not confined to any particular market hence there has always been a need to improve its efficiency.

The goal of this research project is to focus on various Machine learning methods which can help improve the efficacy of email marketing. We strive to find an optimal solution using the gathered data of user interaction or lack of it to classify an email response in order to increase the conversion rate.

In this report, work is done using an email marketing campaign dataset for a small and medium sized enterprise (SME). The problem is approached by first examining the data, performing Exploratory Data Analysis to find the relations and pattern between the features. Next step is to then do the processing and modeling classifiers around it. Lastly, the optimization and evaluation of the results is performed with a comparison of various models which are employed for classification purposes.

# Literature Review

Silver, David et al (2004) studied online customer interactions and applied a concurrent reinforcement learning framework to model these interactions by using a variant of temporal-difference (TD) learning algorithm to learn from online partial interaction sequences. It serves the purpose in a sense that information acquired from one customer is efficiently assimilated and applied in subsequent interactions with other customers and is a better way to accommodate the information. In the study it is established that any business entity or a company has objective functions, such as maximising revenue, customer satisfaction and/or customer loyalty. These depends primarily on the sequence of interactions between company and customers. A key aspect of this setting is that interactions with different customers which occur in parallel. The difference in this approach from traditional reinforcement learning is that the agent interacts with many customers concurrently. Using a simulator, comparison of the concurrent TD method with Monte Carlo, traditional TD and Contextual Bandit algorithms is done and its and demonstrated that TD methods works better. [1]

For modelling consumer responses to direct marketing, Cui, Geng et al (2006) present another interesting proposal which is to use Bayesian Networks (BN) learned by Evolutionary Programming (EP). The company under the study sells multiple product lines of general merchandise and sends regular emails to its list of customers. The data set used

contains the records of 106,284 consumers with 361 variables, including purchase data from recent promotions and the customer's purchase history over a 12-year period. Cui, Geng et al (2006) have applied BN learned by the EP algorithm which automatically finds the directed edges between the nodes to identify a network model that can best describe the relationships between variables. Once the best network structure has been identified, the conditional probabilities are calculated based on the data to describe these relationships. This network is then used to generate a probability score for each example which is used for predictive modeling. The results suggest that Bayesian networks have distinct advantages over other benchmark methods in accuracy of prediction, transparency of procedures, interpretability of results, and explanatory insight. [2]

In the study of dynamically managing profitable email marketing campaigns Zhang, Xi et al (2017) find that contrary to conventional wisdom, email-active customers are not necessarily active in purchases. Moreover the number of emails sent by the campaigner has a nonlinear effect on both the short- and long-term profitability. The study is done using a dataset from a home improvement retailer in the United States which consists of information about customer purchase transactions, the number of emails the firm sent to these customers, and the customers' email open histories. Using a Hidden Markov Model (HMM) the relationship between purchases and email opens is explored and an optimal marketing contact strategy is presented. Using what-if simulation and robustness checks the strategy is validate and the result suggests that sending the optimal number of emails is critical for the profitability of email marketing program. [3]

In the related works reviewed above, approaches such as Reinforcement Learning and its variants, Bayesian Networks and Hidden Markov models were employed by researchers to explore and determine patterns in the available data for improvement in the processes of marketing through emails and customer interactions. In this study, the aim is to pursue further in this research by devising a classification approach based on the relationships in the data. To do so, the focus is on modern Machine Learning classifiers and their comparison for the best outcomes using empirical evaluation metrics.

# Dataset Review

In order to propose machine learning approaches for email marketing problems, various available datasets are investigated and examined. Following are the details of the datasets which are considered:

**Email Marketing Stats (Infographic):** This is a blog which has summary statistics showcasing the impact of email marketing by concerte numbers. However since it's an infographic not enough data can be extracted to work on the problem at hand. [4]

**Enron Email Dataset:** The Enron email dataset contains text of the emails sent by employees of the Enron Corporation. It was obtained during investigation of systemic financial fraud in Enron. This dataset is also not suitable for us as it does not pertain to any kind of marketing but would rather be appropriate for some NLP problem. [5]

**Email Campaign Management for SME:** This is the most interesting dataset as it has data from Small and medium-sized enterprises (SME) who use email marketing to target their customers. The dataset includes different aspects of emails to characterize it and also tracks if the mail is ignored, read or acknowledged by the recipient. This is of particular interest as we are striving to work on a similar solution. [6]

Of the above, we settled upon the Email Campaign Management for SME dataset. There is potential to apply supervised machine learning approaches such as multi-class classification and/or regression analysis to establish relations between email attributes and the recipient's response.

# Exploratory Data Analysis

The data is aggregated for a single user and doesn't have any time series. However there is 'Time Email Sent' feature. There are ~68k unique users with 11 features in this dataset.

## Summary statistics

There are 11 features with following descriptive statistics summarizing the central tendency, dispersion and shape of their distribution:

| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Email_Type | 68,353.00 | NaN | NaN | NaN | 1.29 | 0.45 | 1.00 | 1.00 | 1.00 | 2.00 | 2.00 |
| Subject_Hotness_Score | 68,353.00 | NaN | NaN | NaN | 1.10 | 1.00 | 0.00 | 0.20 | 0.80 | 1.80 | 5.00 |
| Email_Source_Type | 68,353.00 | NaN | NaN | NaN | 1.46 | 0.50 | 1.00 | 1.00 | 1.00 | 2.00 | 2.00 |
| Customer_Location | 56758 | 7 | G | 23173 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| Email_Campaign_Type | 68,353.00 | NaN | NaN | NaN | 2.27 | 0.47 | 1.00 | 2.00 | 2.00 | 3.00 | 3.00 |
| Total_Past_Communications | 61,528.00 | NaN | NaN | NaN | 28.93 | 12.54 | 0.00 | 20.00 | 28.00 | 38.00 | 67.00 |
| Time_Email_sent_Category | 68,353.00 | NaN | NaN | NaN | 2.00 | 0.63 | 1.00 | 2.00 | 2.00 | 2.00 | 3.00 |
| Word_Count | 68,353.00 | NaN | NaN | NaN | 699.93 | 271.72 | 40.00 | 521.00 | 694.00 | 880.00 | 1,316.00 |
| Total_Links | 66,152.00 | NaN | NaN | NaN | 10.43 | 6.38 | 1.00 | 6.00 | 9.00 | 14.00 | 49.00 |
| Total_Images | 66,676.00 | NaN | NaN | NaN | 3.55 | 5.60 | 0.00 | 0.00 | 0.00 | 5.00 | 45.00 |
| Email_Status | 68,353.00 | NaN | NaN | NaN | 0.23 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | 2.00 |

```
Email_Type                    68353 non-null int64
Subject_Hotness_Score         68353 non-null float64
Email_Source_Type             68353 non-null int64
Customer_Location             56758 non-null object
Email_Campaign_Type           68353 non-null int64
Total_Past_Communications     61528 non-null float64
Time_Email_sent_Category      68353 non-null int64
Word_Count                    68353 non-null int64
Total_Links                   66152 non-null float64
Total_Images                  66676 non-null float64
Email_Status                  68353 non-null int64
dtypes: float64(4), int64(6), object(1)
```

It can be observed that there are missing values for *Customer_Location*, *Total_Past_Communications*, *Total_Links* and *Total_Images*.

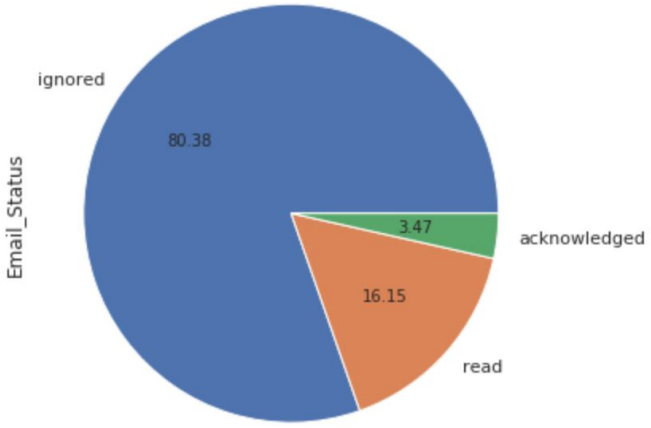
# Distributions

## Email Status

This is the feature of interest as the dataset considers its the label feature. It has three possible values describing the outcome of the email sent to a particular user:

- 0 - Ignored
- 1 - Read
- 2 - Acknowledged

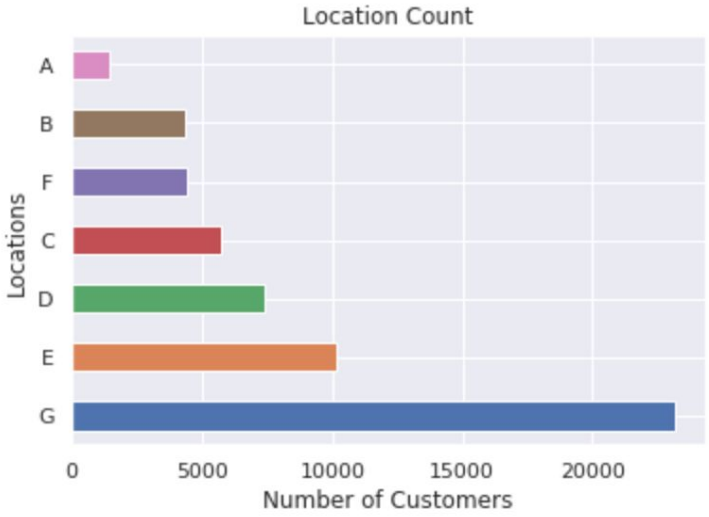
Following are the proportions of these values in the dataset:




The pie chart shows that there is very high ratio of ignored emails in the dataset. Hence for performing classification we will face the class imbalance problem.

## Customer Location

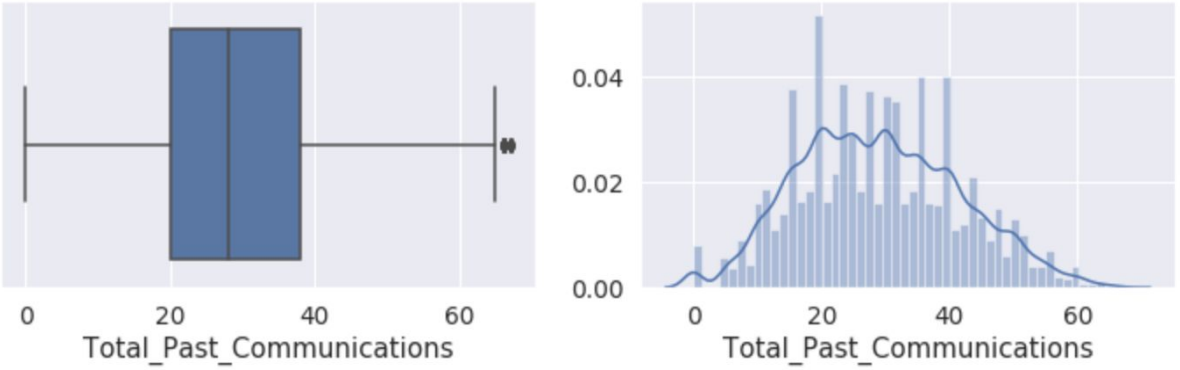
In the dataset, the customer locations are masked by alphabetical categories. Below is a visualization of their distribution and how many users account for each:




The value count in the plot above shows that majority of the users are from the same *G* location but the other half is relatively evenly divided in other locations. The *Customer_Location* feature is categorical and will be hot-encoded in the processing phase.

## Total Past Communications

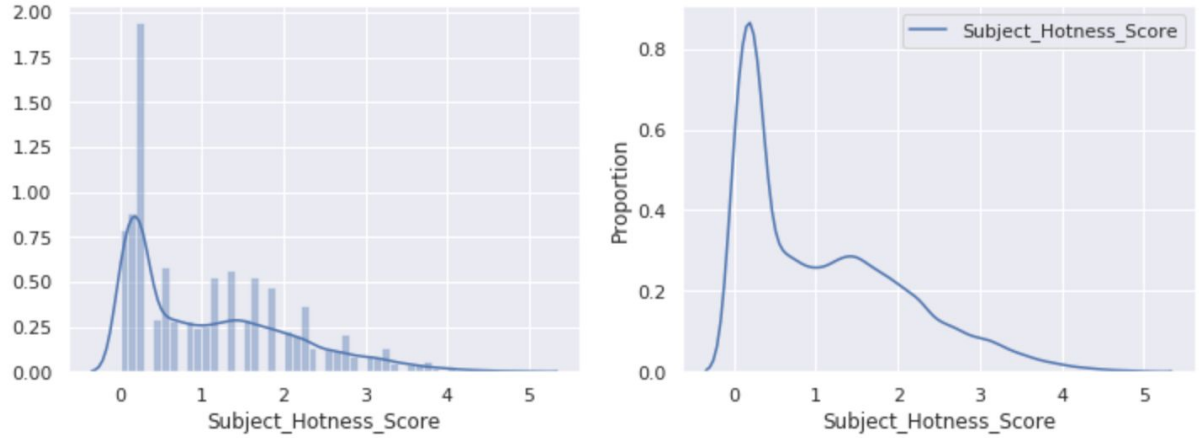
This feature describes how many times a user has been communicated with in the past. Below it is plotted to visualize the distribution of observations and to check for outliers:




It can observed that the distribution is mostly centered around mean and there are very few outliers. This means that there isn't any major irregularity and the figures from total past communications are quite random since they center at mean. Also this helps the classifier as normally distributed features with less outliers helps avoiding exploding/vanishing gradients.

## Subject Hotness Score

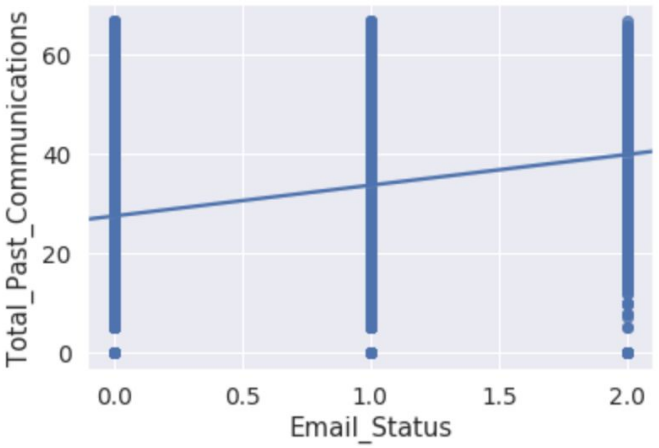
Another feature of interest in the dataset is about the emails subject hotness score. The score indicate a value for 'hotness' or how attractive the subject of the email is. However, in the data it doesn't specifically indicate what the continuous range of values from 0-5 represent. When plotted, the distribution of this feature has a sharp hike around 0.3:



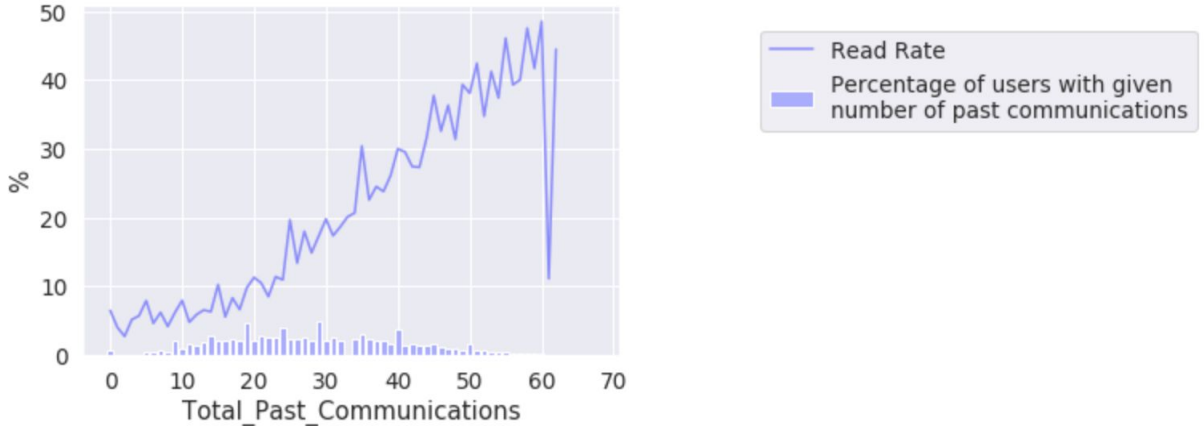

This shows that most emails sent had the hotness score around 0.3 hence it would make sense to standardize this feature as standardizing tends to make the training process well behaved because the numerical condition of the optimization problems gets improved. [7]

## Email Status vs Past Communications

The plot below shows the relation between past communications and the email status:



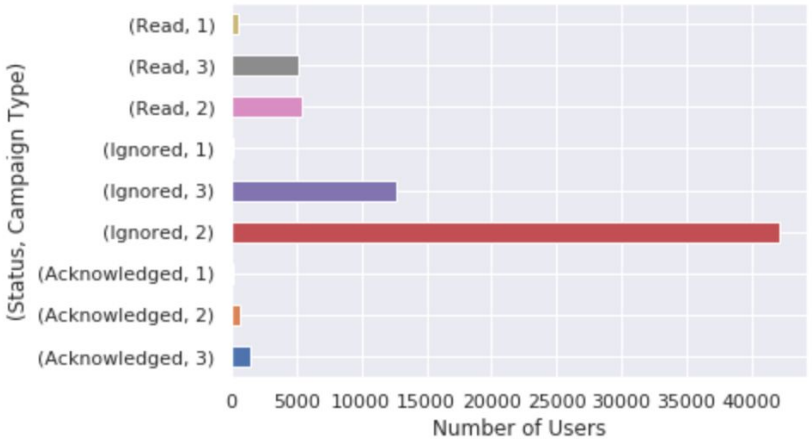

This shows a steady ascent when the status is 2 meaning when the email is read. Hence a simple hypothesis is that the more users have communicated the more they might tend to read the email. Lets explore this more by comparing Past communications with only the values when the Email was actually read:




The plot above depicts that the relation is not exactly proportional as thought of because there is sharp descent around 60 communications and then it goes back up again. However, this effect still has significant impact in a sense that number of past communications are directly correlated with read rate of the email. Since emails being read is one of our target classes hence it can be concluded that keeping this feature will help the classifier benefit from this correlation.

## Email Status VS Email Campaign Type

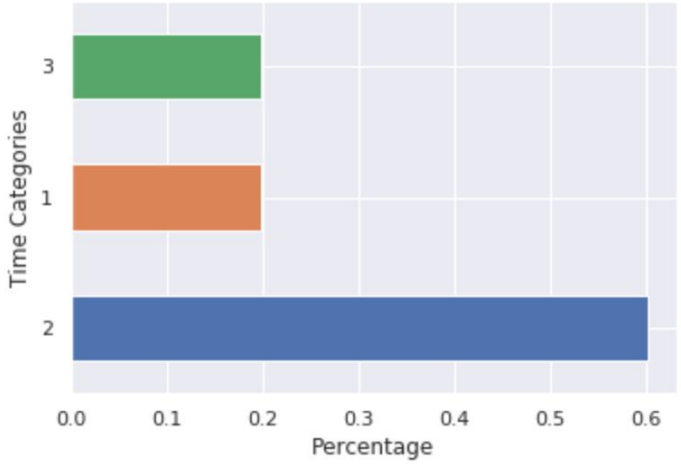
There are three campaign categories and it would be beneficial to learn which one has impacted the conversions most. The types are grouped together and plotted to visualize the results:




Here it can observed that Campaign 3 has resulted in most READ and ACKNOWLEDGED emails. This result helps us in understanding how the features are related to target classes and in turn will help us in feature engineering for the classifier.

## Email Status vs Time Email Sent Category

The data itself doesn't describe exactly what the time categories (1, 2, 3) represent hence they are kept as simple categorical feature. To explore how many time categories there are and how this feature is distributed it is plotted as below:




It shows that 1 and 3 are each 20% while category 2 covers 60% of the records. Let's check what relation they have with email response (*Email_status*).


The plot actually follows the time email sent category distribution. The more email sent at a time the more responses were hence there isn't any anomaly to notice here.

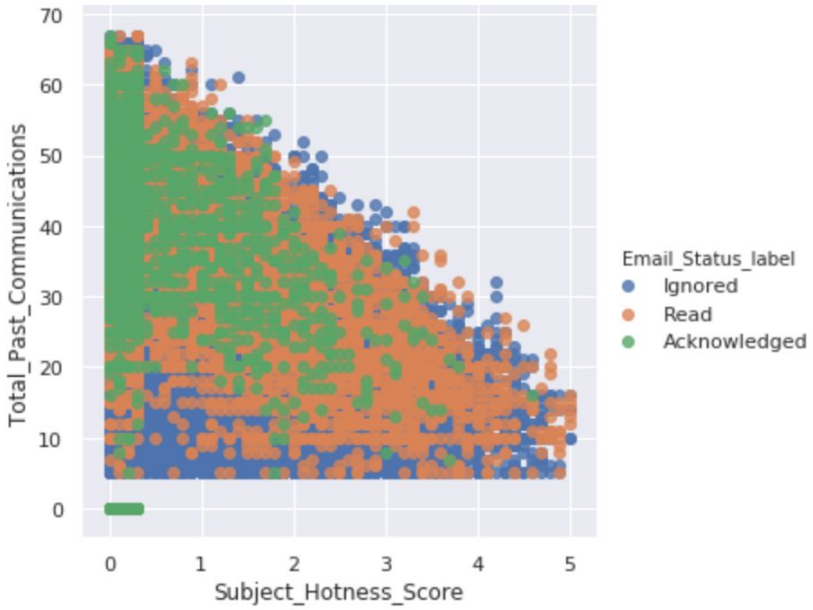
## Email Status: Time Sent vs Past Communications

To consider how past communications and the time the email was sent effect email response in the data, it is plotted as below:


The plot shows that the most favourable responses (read or acknowledged) are with the time category 2 with a high number of past communications. Hence both past communications and the time when email was sent has an impact on email response.

## Email Status: Subject Hotness Score vs Past Communications

Similarly to do the same factor impact analysis of Subject hotness score feature along with past communication on favorable email responses.




It can be seen that with lower subject hotness score and higher past communications, the email's response of Read and Acknowledge is high. However, with high subject score and even high communications there are not many responses, this can be due to the distribution of score as very few emails have high subject hotness score therefore also less responses.

## Random Forest feature importance

Random Forests [8] are among the most popular machine learning methods thanks to their relatively good accuracy, robustness and ease of use. A random forest is made up of a number of decision trees. Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set. The measure based on which the (locally) optimal condition is chosen is called impurity. For classification, it is typically Gini impurity. [9]

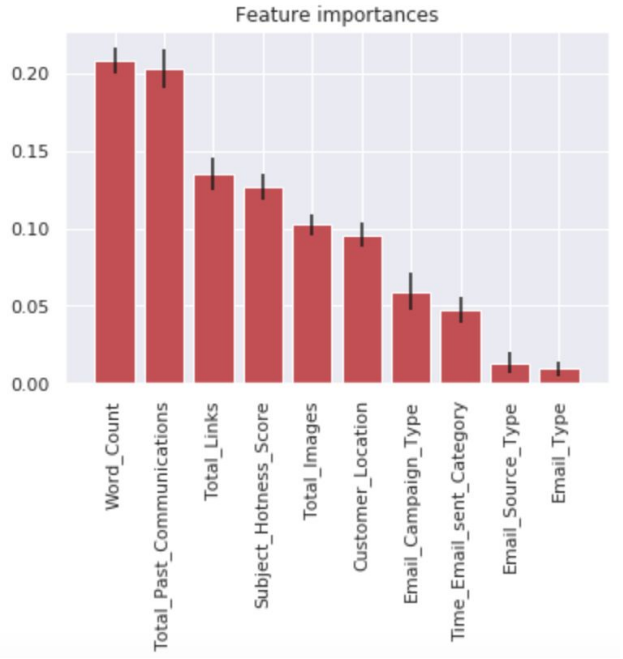
Gini impurity [10] is a measurement of the likelihood of an incorrect classification of a new instance of a random variable, if that new instance were randomly classified according to the distribution of class labels from the data set. Gini impurity of a feature is calculated as follows:

$$G(k) = \sum_{i}^{J} P(i) * (1 - P(i))$$

Where, $P(i)$ is the probability of a certain classification $i$, per the training data set.

Thus when training a tree, Gini impurity enables to compute how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features are ranked according to this measure.

Accordingly, feature importances when measured for the dataset in study resulted into following:




As observed earlier in exploratory analysis, past communications and word count has relatively higher impact on the outcome of email response.

## Analysis Conclusion

By exploring and analysing the features in the data it can be deduced that we can model a multi-class classification solution around the Email_Status variable treating it as target label and taking into account all other features. We settle on not removing any feature from the set for now as training, optimizing and comparing various models will yield the best possible solution.

# Implementation

## Methodology

For classification, optimization and training the models Python's *Scikit-learn* [11] library is used. *Pandas* is used for cleaning and building the dataset and for visualizations and exploratory analysis *matplotlib* and *seaborn* are utilized.

Beginning with preprocessing the data by normalizing the continuous feature space and hot-encoding categorical features to numeric ones, the methodology to find the best fit classifier for the dataset in study involves experimenting with four different algorithms. For comparison of these different classifiers, key methods are implemented and evaluated on two empirical metrics, namely Accuracy and F1 Score.

The ***accuracy*** [12] here represents the fraction of samples predicted correctly and it is calculated as:

$$accuracy\ (y,\ \widehat{y}\ )\ =\ \frac{1}{n}\ \sum_{i=0}^{n-1}\ 1\ (\widehat{y_i} = y_i\ )$$

where, $n$ is the number of samples, $\widehat{y_i}$ is the predicted value of the *i*-th sample and $y_i$ is the corresponding true value.

Whereas, the ***F1 score*** [13] can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is

$$F1\ =\ 2\ *\ \frac{(precision * recall)}{(precision + recall)}$$

Moreover, in the dataset there is a huge class imbalance due to the nature of email responses. Hence in order to overcome this imbalance, an oversampling technique is employed to have a viable distribution of labels for classification. Finally to optimize these models Bayesian optimization technique is utilized using *Hyperopt* is which is a Python library for serial and parallel optimization. The models are trained on a 70/30 split training and testing dataset with 3 fold cross-validation.

## SMOTE Oversampling

The biggest challenge at hand is the distribution of label class which represent the email responses. In the dataset the three classes (Ignored, Read and Acknowledged) are distributed as such:

| Class | Proportion |
|---|---|
| Ignored | 80% |
| Read | 16% |
| Acknowledged | 4% |

It can be observed clearly that there is a huge class imbalance and regardless of the type of classifying model it won't be able to fit properly. Hence to tackle this problem we proceed by oversampling the data from the same distribution from where it comes. Oversampling is a well-known way to potentially improve models trained on imbalanced data. To proceed with oversampling we used **SMOTE - Synthetic Minority Over-sampling Technique** [14].

At a high level, SMOTE creates synthetic observations of the minority class(es) by:

- Finding 4 k-nearest-neighbors for minority class observations (finding similar observations)
- Randomly choosing one of the k-nearest-neighbors and using it to create a similar, but randomly tweaked, new observation.

For the minority class of *Acknowledged* email, about six thousand synthetic data points are generated in order to balance it with the *Read* class. After oversampling the class balance is much better:

| Class | Over sampled Proportion |
|---|---|
| Ignored | 71% |
| Read | 14% |
| Acknowledged | 14% |

# Classifiers

Following are the supervised machine learning methods used:

## Naive Bayes Classifier

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features value of the class variable. In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations. [15]

In this study, Multinomial Naive Bayes algorithm is used which implements the naive Bayes algorithm for multinomially distributed data. The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class $y$, where $n$ is the number of features and $\theta_{yi}$ is the probability $P(x_i \mid y)$ of feature $i$ appearing in a sample belonging to class $y$. The parameters $\theta_y$ is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Where, $\alpha$ is the smoothing parameter and $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature $i$ appears in a sample of class $y$ in the training set $T$ and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class $y$.[16]

## Support Vector Classification (SVC)

Support vector machines [17] are a set of supervised learning methods. SVM differs from the other classification algorithms in the way that it chooses the decision boundary that maximizes the distance from the nearest data points of all the classes. An SVM strives to find the most optimal decision boundary. The most optimal decision boundary is the one which has maximum margin from the nearest points of all the classes. The nearest points

from the decision boundary that maximize the distance between the decision boundary and the points are called support vectors.

Given training vectors $x_i \in \mathbb{R}^p$, i=1,...,n, in two classes and a vector $y \in \{1, -1\}^n$, SVC solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

Its dual is

$$\min_{\alpha,\alpha^*} \frac{1}{2}(\alpha - \alpha^*)^T Q(\alpha - \alpha^*) + \varepsilon e^T(\alpha + \alpha^*) - y^T(\alpha - \alpha^*)$$
$$\text{subject to } e^T(\alpha - \alpha^*) = 0$$
$$0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, \ldots, n$$

Where $e$ is the vector of all ones, $C > 0$, $Q$ is an $n$ by $n$ positive semidefinite matrix, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function $\phi$ [18]. For multi- class classification the “one-against-one” approach [19] is implemented. If *nclass* is the number of classes (which is three in this case), then *nclass * (nclass - 1) / 2* classifiers are constructed and each one trains data from two classes.

In the implementation of SVC in this study following hyperparameters are tuned to fit the data:

| **Hyperparameter** | **Description** |
|---|---|
| C | C is the penalty parameter of the error term. It controls the trade off between smooth decision boundary and classifying the training points correctly. |
| kernel: rbf | Radial basis function kernel; $\exp(-\gamma\|x - x'\|^2)$ |
| gamma ($\gamma$) | gamma is the inverse of the standard deviation of the rbf kernel. The higher the gamma value, svm tries to exactly fit the training data. |

## Random Forest Classifier

Random forest method [8] as described above is based on an ensemble of decision trees. It works as a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. In the implementation of Random Forest classifier in this study, following hyperparameters are tuned:

| **Hyperparameter** | **Description** |
|---|---|
| n_estimators | This the number of trees in the forest, the larger it is the better but consequently takes more time to compute and results stop getting significantly better beyond a certain number. |
| criterion | The function to measure the quality of a split. Gini impurity is found to be the best and is described above in the *Random Forest feature importance* section. |
| max_depth | The maximum depth of the tree. |
| max_features | The number of features to consider when looking for the best split. The lower it is the greater the reduction of variance, but also the greater the increase in bias. |
| class_weight: balanced | These are the weights associated with classes, when 'balanced' values of *y* automatically adjust weights inversely proportional to class frequencies in the input data. |

## KNeighborsClassifier

This classifier implements the k-nearest neighbors [20] vote mechanism. In the generic k-NN model, each time a prediction is to be made for a data point, first this data point's distance from all other points is to be calculated and then only nearest k-points can be discovered for voting.

In k-nearest neighbor algorithm, the example data is first plotted on an n-dimensional space where n is the number of data-attributes. Each point in n-dimensional space is labeled with its class value. To discover classification of an unclassified data, the point is plotted on this n-dimensional space and class labels of nearest k data points are noted. Generally k is an odd number. That class which occurs for the maximum number of times among the k nearest data points is taken as the class of the new data-point. That is, decision is by voting of k neighboring points. [21]

## Hyperparameter Optimization

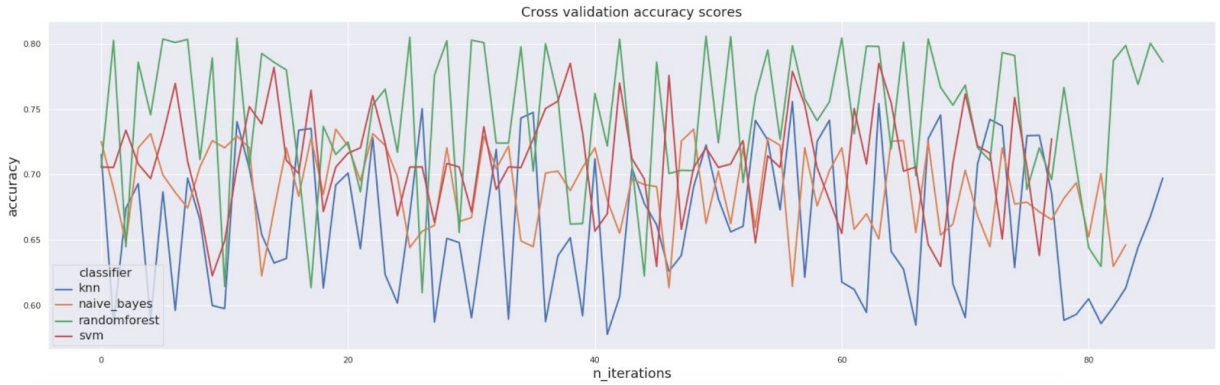
In this study, Random Search optimization methods are employed to optimize hyperparameters of the classification models. Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. Studies [22] [23] have shown empirically and theoretically that randomly chosen trials over a parameter space are more efficient for hyper-parameter optimization than trials on a grid.

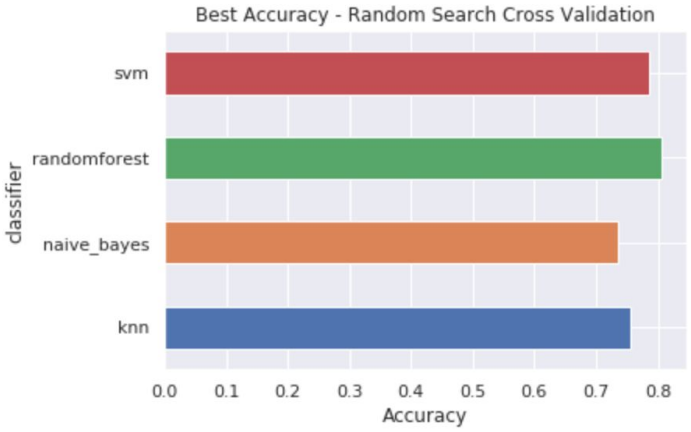
```
# Classifiers
clfs = {
    'naive_bayes': MultinomialNB(),
    'svm': SVC(),
    'randomforest': RandomForestClassifier(),
    'knn': KNeighborsClassifier()
}
#Hyperparamter search space for classifiers
param_space = {
    'naive_bayes': {
        'alpha': uniform(0.0, 2.0)
    },
   'svm': {
        'C': uniform(0, 10.0),
        'kernel': ['linear', 'rbf'],
        'gamma': uniform(0, 20.0)
    },
   'randomforest': {
        'max_depth':range(1,20),
        'max_features': range(1,5),
        'n_estimators': range(1,20),
        'criterion': ["gini", "entropy"],
        'class_weight': ['balanced']
    },
    'knn': {
        'n_neighbors': range(1,30)
    }
}
```

Hyperparameter search space for classifiers

In our implementation, to optimize the classification models Random search is used with accuracy from 3 fold cross validation as evaluation metric. The optimization run for max 90 iterations with a choice of hyperparameters for each model in a well defined search space, the history of the accuracy (which is our evaluation metric) achieved with each random combination of the hyperparameters is tracked and best performing combination is saved. Once the optimization process completed we use the best found classifier and hyperparameters to train the model.

Following is a visualization of the accuracy values over iterations for each classifier whilst performing random search for best hyperparamters:




# Results

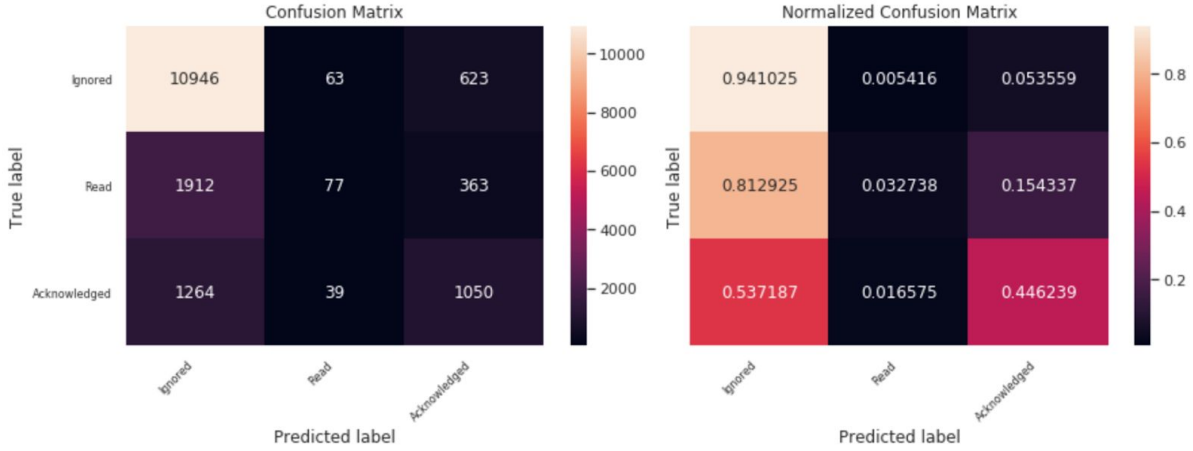
For each of the four classifiers in consideration, the most optimal hyperparameters are found by optimization which gives us the best accuracy. Here below are our findings:

| **Classifier** | **Best parameters** | **Highest accuracy** | **Weighted F1 Score** |
|---|---|---|---|
| KNN | *n_neighbors*: 27 | 0.739 | 0.68 |
| SVC | C = 9.8679,<br>gamma = 19.9878,<br>kernel = rbf | 0.7297 | 0.63 |
| Random Forest | 'max_depth': 19,<br>'max_features': 4,<br>'n_estimators': 19,<br>'criterion': gini,<br>'class_weight': 'balanced' | 0.8064 | 0.80 |
| Naive Bayes | *alpha:* 1.9881 | 0.7206 | 0.61 |

Following are the classification reports for each method displaying the main classification metrics along side confusion matrices:

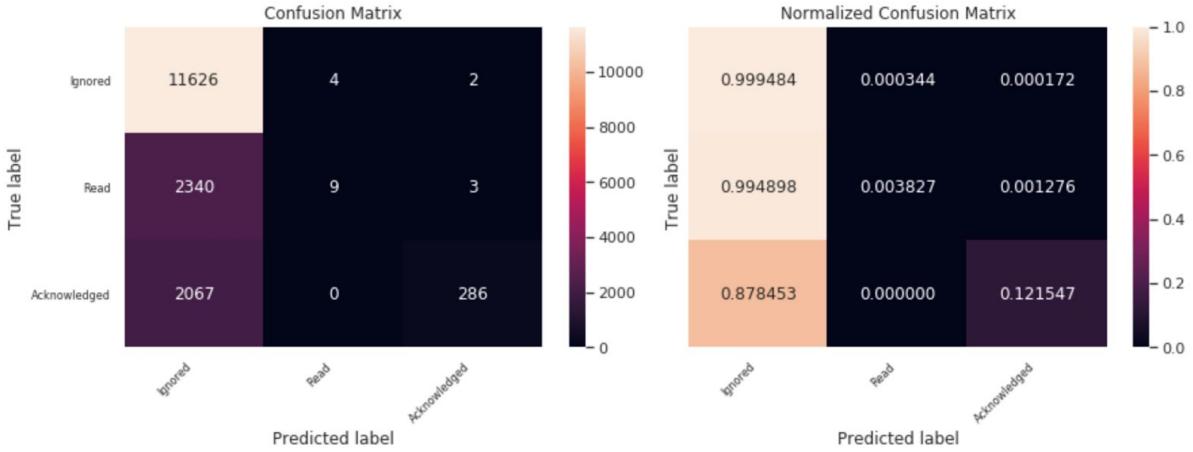
## KNN

```
KNeighborsClassifier
Accuracy:  0.739
Classification Report:
                precision    recall  f1-score   support

           0       0.78      0.94      0.85     11632
           1       0.43      0.03      0.06      2352
           2       0.52      0.45      0.48      2353

   micro avg       0.74      0.74      0.74     16337
   macro avg       0.57      0.47      0.46     16337
weighted avg       0.69      0.74      0.68     16337
```



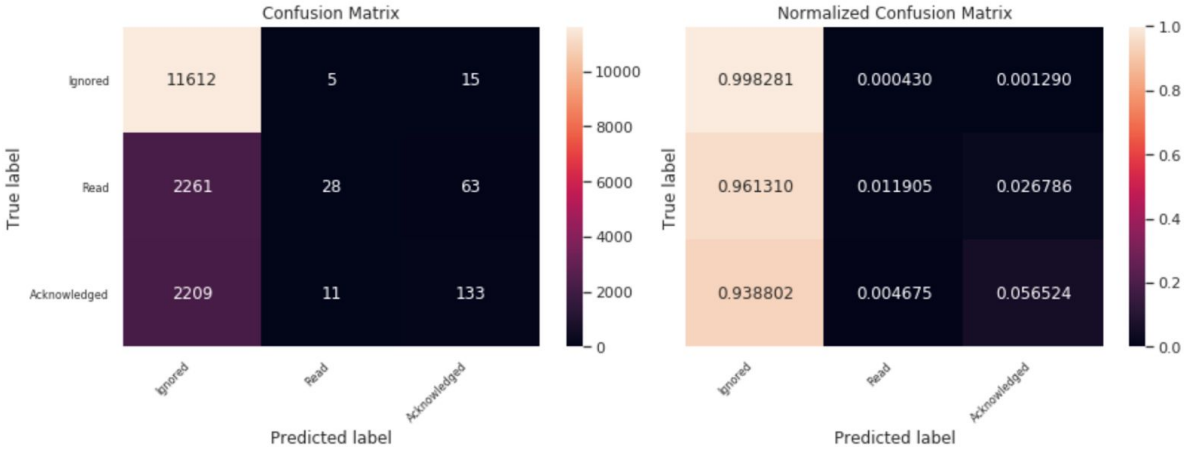

## SVC

```
SVC
Accuracy:  0.7297
Classification Report:
                precision    recall  f1-score   support

           0       0.73      1.00      0.84     11632
           1       0.69      0.00      0.01      2352
           2       0.98      0.12      0.22      2353

   micro avg       0.73      0.73      0.73     16337
   macro avg       0.80      0.37      0.35     16337
weighted avg       0.76      0.73      0.63     16337
```



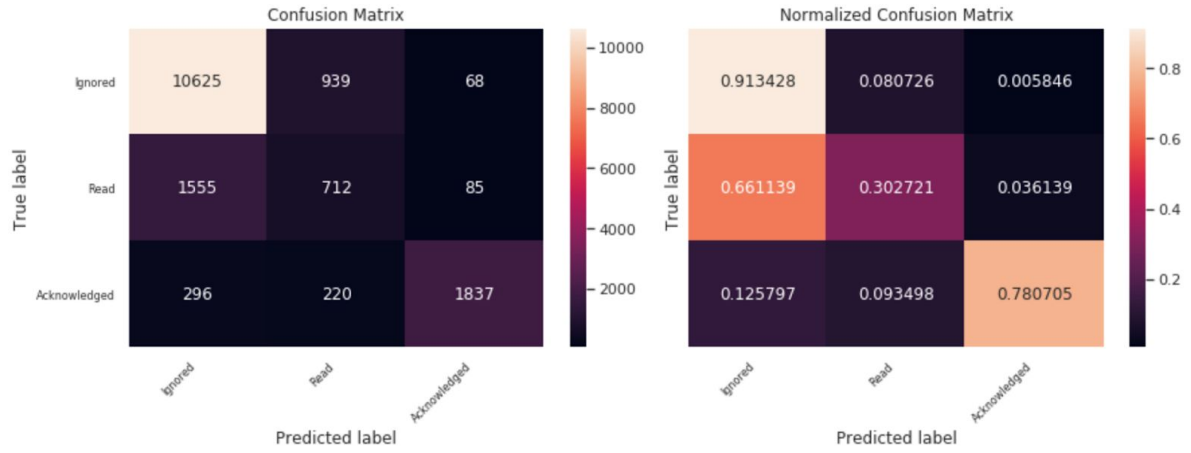

## Naive Bayes

```
Naive Bayes
Accuracy:  0.7206
Classification Report:
              precision    recall  f1-score   support

           0       0.72      1.00      0.84     11632
           1       0.64      0.01      0.02      2352
           2       0.63      0.06      0.10      2353

   micro avg       0.72      0.72      0.72     16337
   macro avg       0.66      0.36      0.32     16337
weighted avg       0.70      0.72      0.61     16337
```




### Random Forest

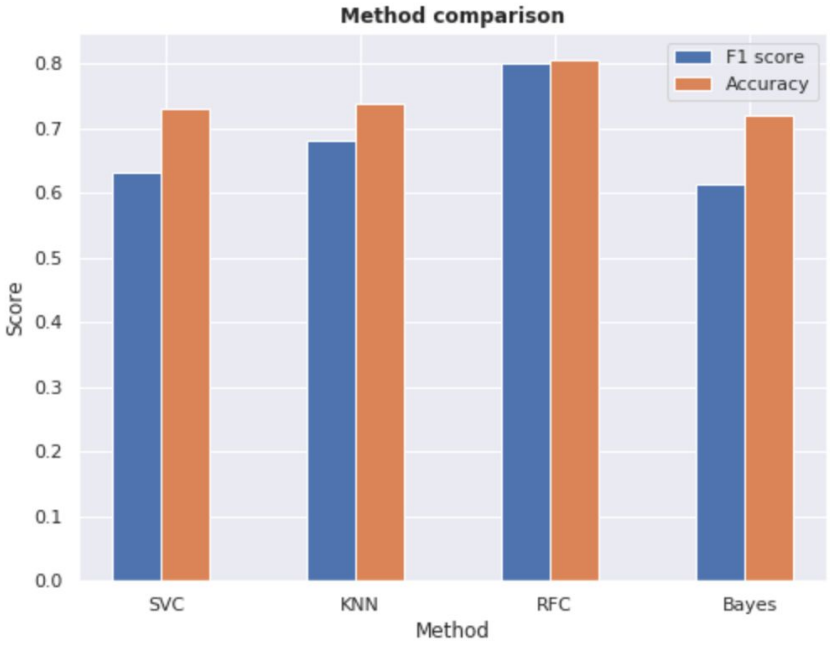
```
Random Forest Classifier
Accuracy:  0.8064
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.91      0.88     11632
           1       0.38      0.30      0.34      2352
           2       0.92      0.78      0.85      2353

   micro avg       0.81      0.81      0.81     16337
   macro avg       0.72      0.67      0.69     16337
weighted avg       0.79      0.81      0.80     16337
```




# Comparison

Based on the results of our experimentation we can assert that the Random Forest method has performed better than other approaches in both Accuracy and F1 Score metrics. In general we can observe that methods other than Random Forest are on a similar level with

each other and KNN has the lowest contrast in F1 score and accuracy. This comparison however is on two metrics and doesn't fully reflect the nuances of the multi-class classification problem at hand but still however provide and empirical baseline for evaluation of methods. The comparison is shown in the figure below:




## Code

The code of this implementation can be found here:
https://github.com/kamalmemon/MLResearch/blob/master/src

# Conclusion

Through this work, we employed learning models for predicting the response of targeted marketing emails in terms of Ignored, Read or Acknowledged response. The dataset we used is based on the features extracted from the emails, email recipients' profiles and the engagement characteristics of the recipients. Extensive exploratory analysis and visualizations is done in order to gauge the relationships of features among themselves and with the target label. After getting a good grasp on the dataset, four classification methods are applied and compared for this multi-class classification problem. The results show that Random Forest Classifier performs better among all, however more experimentation with different evaluation metrics and methods is left for future work.

# Future Work

Based on the literature review, there can be various approaches to solve the problem of the efficiency of marketing emails through various machine learning techniques. We have attempted one in this study however the approaches of online Reinforcement Learning and

Bayesian Networks also holds immense potential. In the future, an extensive study can be done using different datasets with various approaches to determine which method work better under which circumstances. On the other hand more features can be extracted for the dataset from the emails and user profiles.

# References


[1] David Silver, Leonard Newnham, David Barker, Suzanne Weller, Jason McFall ; Proceedings of the 30th International Conference on Machine Learning, PMLR 28(3):924-932, 2013.

[2] Cui, G., Wong, M. L., & Lui, H.-K. (2006). Machine learning for direct marketing response models: Bayesian networks with evolutionary programming. Management Science, 52(4), 597-612. doi: 10.1287/mnsc.1060.0514

[3] Zhang, Xi (Alan), V. Kumar, and Koray Cosguner. “Dynamically Managing a Profitable Email Marketing Program.” Journal of Marketing Research 54, no. 6 (December 2017): 851–66. doi:10.1509/jmr.16.0210.

[4] Mohsin, Maryam. "10 Email Marketing Stats You Need to Know in 2019". Oberlo (blog). February 19, 2019. Accessed March 23, 2019. https://www.oberlo.com/blog/email-marketing-statistics.

[5] Cohen, William. "Enron Email Dataset". May 8, 2015. https://www.cs.cmu.edu/~./enron/

[6] Corefactors, Email Campaign Management for SME, March 2017. https://www.kaggle.com/loveall/email-campaign-management-for-sme

[7] DeFilippi, Robert. "Standardize or Normalize?". Medium (blog). Apr 29, 2018. Accessed March 23, 2019. https://medium.com/@rrfd/standardize-or-normalize-examples-in-python-e3f174b65dfc

[8] Leo Breiman, Random Forests, Machine Learning, v.45 n.1, p.5-32, October 1 2001

[9] Saabas, Ando. "Selecting good features – Part III: random forests". Datadive (blog). Dec 01, 2014. Accessed March 23, 2019. https://blog.datadive.net/selecting-good-features-part-iii-random-forests/

[10] Breiman, Friedman, "Classification and regression trees". Chapman & Halucrc, 1984.

[11] Pedregosa et al. Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.

[12] Accuracy score, "Model evaluation: quantifying the quality of predictions". Scikit-learn Org. https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score

[13] Nancy Chinchor, MUC-4 Evaluation Metrics, in Proc. of the Fourth Message Understanding Conference, pp. 22–29, 1992.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique". Journal Of Artificial Intelligence Research, V. 16, pp. 321-357, 2002. doi:10.1613/jair.953

[15] Zhang, Harry. "The Optimality of Naive Bayes". In FLAIRS2004 conference, 2004.

[16] Multinomial Naive Bayes, "Naive Bayes". Scikit-learn Org. https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes

[17] Hearst, M. A. "Support vector machines". *IEEE Intelligent Systems* 1998*, 13*(4), p. 18.

[18] Mathematical formulation, "Support Vector Machines". Scikit-learn Org. https://scikit-learn.org/stable/modules/svm.html#mathematical-formulation

[19] Pal, Mahesh. “Multiclass Approaches for Support Vector Machine Based Land Cover Classification.” CoRR abs/0802.2411 (2008)

[21] T. Cover, P. Hart. "Nearest neighbor pattern classification". IEEE Transactions on Information Theory (Volume: 13 , Issue: 1 , January 1967)

[20] Jaimini Pandya, Vishwam. "Comparing Handwritten Character Recognition by AdaBoostClassifier and KNeighborsClassifier". 271-274 (2016). 10.1109/CICN.2016.59.

[21] J. Bergstra, D. Yamins, and D. D. Cox. 2013. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28 (ICML'13), Sanjoy Dasgupta and David McAllester (Eds.), Vol. 28. JMLR.org I-115-I-123

[22] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13. 281-305 (February 2012)

[23] Horia Mania, Aurelia Guy, Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning" (Mar 2018)