Foundation of Machine Learning – Assignment 2

Multi-Class Email Classification

Team name - '**PlanetFood**'

Pallav SAHU
pallav.sahu@student-cs.fr

Akshay SHASTRI
akshay.shastri@student-cs.fr

Anmol KATIYAR
anmol.katiyar@student-cs.fr

Arun JEGATHESH
arun.jegathesh@student-cs.fr

Problem Definition

We often face the problem of searching meaningful emails among thousands of promotional emails. This project focuses on creating a multi-class classifier that can classify an email into different classes namely, Updates, Personal, Promotions, Forums, Purchases, Travel, Spam, and Social based on the metadata features extracted from the email.

Dataset Description

The metadata extracted from the dataset of emails contains information on the date received, sender's organization, top level domain of the organization, number of ids in cc, flag denoting if this mail is part of bcc list. The mail type denotes if a mail has just plain text or any further additional image/hyperlinks. The count of images and count of URLs in the mail is given. Finally flags denoting if there is a salutation in the mail and if the designation of the sender is mentioned have been captured.

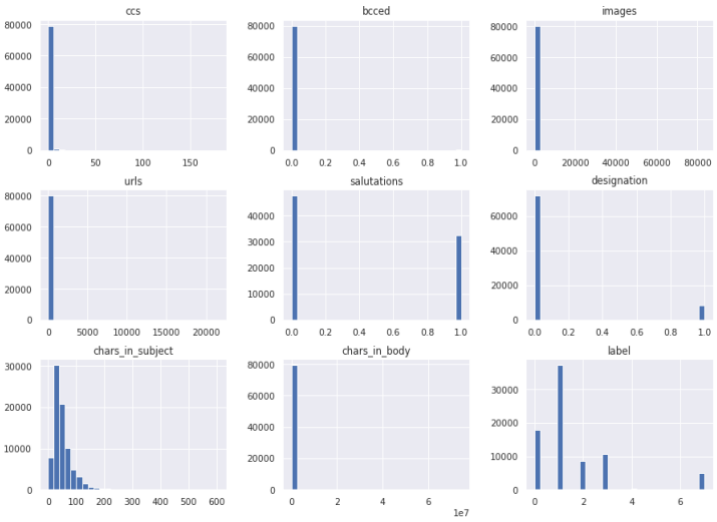

Figure 1.1 frequency distribution for all variables

Process Overview

As the first step to any model building starts with understanding of data and wrangling, we explored the dataset to observe a pattern and its characteristics, including the distribution of the classes, and correlation between the features. This gives us leverage to process the data efficiently with feature engineering and selection of variables to the model.

- To begin with, we checked for missing values across columns and imputed them (mean for numeric and mode for categorical data). Cleaning of columns with string datatype included standardization, trimming white spaces and removing unwanted characters.

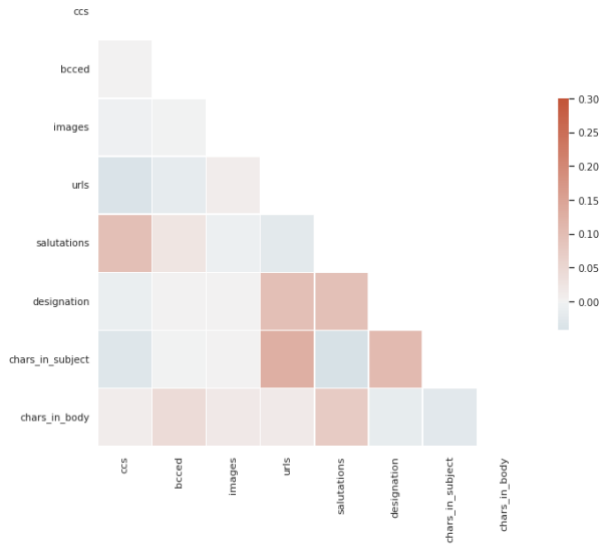

Figure 1.2 Correlation heatmap of all the independent variables

- Feature engineering was performed for date, org and the tld columns for standardizing the data columns across the required characteristics. New features were added, and the importance of all features was checked.
- We used the "LazyClassifier" method from the "*lazypredict*" package to see which model performs well for this data. We found that boosting algorithms worked best so we tried two – XGBoost, and Catboost (since most of the data is categorical). Perceptron provided a not-so-great F1-score, so we tried multi-layer neural network model as well.
- Label encoder was used to transform the categorical data as needed to fit the model for XGBoost and neural network. Catboost works well with unencoded data, so no encoding has been done for that model, but data is passed using the special "Pool" datatype which works much better for categorical data.
- For tuning the parameters in Catboost, the inbuilt "randomized search" method is used, while for XGBoost GridsearchCV method is used.
- Finally, the 3 different models are trained to predict the class of the emails (on the training set – 80% of total training set provided). We then calculate the respective F1 scores (on the test set – remaining 20% of the training set).

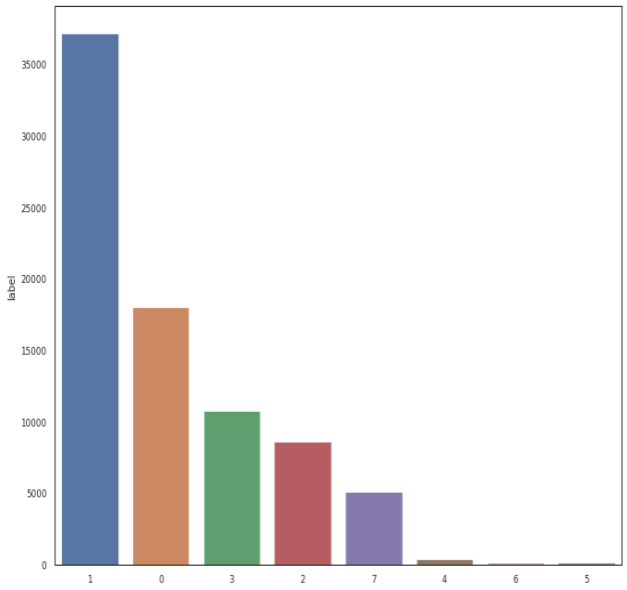

Figure 1.3 Frequency distribution of the dependent variable (the class label of emails)

Feature Engineering

A brief explanation of the processing for various columns are provided below:

Date

Date column is cleaned across both training and test datasets to remove unwanted characters, and some incorrect entries were corrected based on the trend in data. We then converted the date to a standardized time zone (using "*dateparser*" package) for ease of comparison. For XGBoost, we then split the date into date/month/year hour/minute/second and made 6 new columns from this split. We also experimented by encoding the hour into 6 periods instead of taking all 24 values.

Since this data has been generated by the mails received by an Indian user, the general holidays (Indian and French) and weekends are tagged from the date column to find a possible differentiation between emails. In this manner, we created two new columns – "*weekend"* and "*holiday"* which take 1 or 0 values depending on the date of the mail received.

tld and org

tld and org columns, as described in the document, contains information about the organization and trusted top-level domain data. But an EDA of the data showed that the data separation of string values between the two columns are not perfect. So, to get the clean tld and actual org data, we fetched all the available top-level domain data, using the *"publicsuffix2"* package, which compares the old-tld string with all the available domains existing. The function "get_tld(x)" outputs the top-level domain portion from the input string. Thus, the tld values are split based on the input string and it is compared with the master list of valid tlds from the package and retained for further usage. To further optimize the process, we separated our dataset in 2 parts:

A. Having correct org and tld in existing training data

B. Having incorrect tld and org format

So, we only computed the corrections for part B. of the training data. The final output from Part A. and Part B. were the appended properly to get updated columns, i.e., *"new_org"* and *"only_tld"*.

Encoding Categorical Variables

Using OneHot encoding, the categorical variables are transformed into numerical flag of 1 or 0 for usage of the categorical variables into the XGBoost algorithm, but we found that label encoding gave us better results. For the Neural network algorithm, again the function "label_encoder" has been used that provides numerical values to categorical variables but doesn't retain the numerical significance.

Finally, for the CatBoost technique, no encoding was done as the model parameters take care of the categorical variables using the inbuilt "Pool" datatype.

Model Selection, Comparison and Evaluation

Lazypredict is a recent published library which enables us to build multiple basic models (without hyper parameter tuning) using very few lines of code. This allows us to get a basic intuition of which model will work best for our dataset. Results from the "lazyclassifier" method for our training data are as follows:

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	\
NearestCentroid	0.27	0.48	None	0.26	
LGBMClassifier	0.56	0.42	None	0.56	
XGBClassifier	0.55	0.41	None	0.55	
GaussianNB	0.12	0.37	None	0.10	
QuadraticDiscriminantAnalysis	0.13	0.34	None	0.11	
BaggingClassifier	0.43	0.30	None	0.43	
BernoulliNB	0.47	0.30	None	0.45	
RandomForestClassifier	0.44	0.30	None	0.44	
AdaBoostClassifier	0.23	0.29	None	0.23	
KNeighborsClassifier	0.46	0.27	None	0.46	
ExtraTreesClassifier	0.42	0.27	None	0.42	
DecisionTreeClassifier	0.40	0.25	None	0.40	
SVC	0.55	0.25	None	0.51	
ExtraTreeClassifier	0.39	0.24	None	0.38	
LogisticRegression	0.49	0.21	None	0.45	
LinearDiscriminantAnalysis	0.49	0.21	None	0.44	
SGDClassifier	0.48	0.21	None	0.40	
Perceptron	0.41	0.20	None	0.39	
PassiveAggressiveClassifier	0.36	0.19	None	0.35	
LinearSVC	0.49	0.18	None	0.42	
CalibratedClassifierCV	0.49	0.16	None	0.39	
RidgeClassifier	0.48	0.15	None	0.38	
RidgeClassifierCV	0.48	0.15	None	0.38	
DummyClassifier	0.47	0.12	None	0.30	

Figure 1.4 Model performance estimate using the Lazypredict library

Based on these results, we selected the below models to tune according to the data:

1. XGBoost:

XGBoost is one of the most popular implementations of the Gradient Boosted Trees algorithm, a supervised learning method that is based on function approximation by optimizing specific loss functions as well as applying several regularization techniques. To determine the best parameters for fitting the model and tune them, we used

GridsearchCV function from *scikit-learn* API. F1 score achieved was 0.578 on the internal test set (Similar performance in Kaggle). Feature importance (based on "gain") was also calculated as shown below:

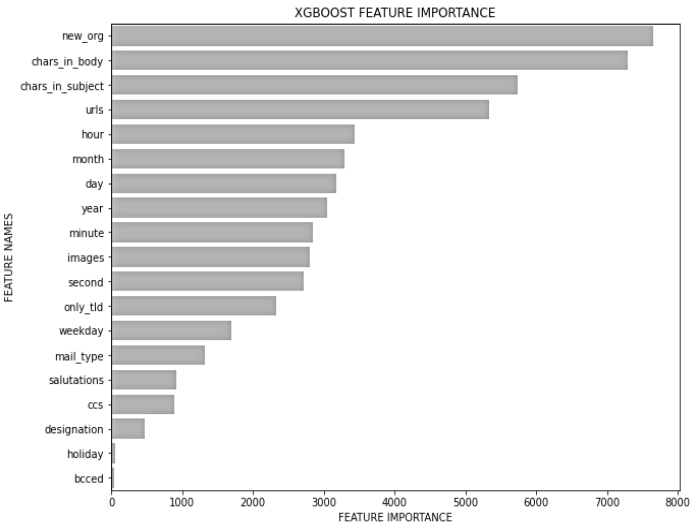

Figure 1.5 Feature importance plot from XGBoost Algorithm

2. Neural Network:

Neural network model tries to find the relationship between features in a data set. It consists of a set of algorithms that mimic the work of the human brain. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. As we need to classify the email across 8 labels, we used the keras neural network that works on top of tensorflow for this project. Since we saw that perceptron did not yield good results, we tried a 3-layer neural network, with an additional output layer. For training this model, we only used features which were significant in the XGBoost model and discarded other features (e.g., bcced etc.) from which the model was not gaining much information, so that our model does not overfit. F1 score achieved was 0.6 on the internal test set (Similar performance in Kaggle). We used EarlyStopping function to stop the model before it starts overfitting on our training data.

3. Catboost:

CatBoost is an algorithm for gradient boosting on decision trees. It is developed by Yandex researchers and engineers, and is used for search, recommendation systems, personal assistant, self-driving cars, weather prediction and can be used in classification problems. Catboost performs best with un-encoded categorical data, so we provided the algorithm with a list of features that are categorical. All data which was given to the algorithm was either "int" or "string". The date was also given directly as a string, but we also tried if providing additionally the day and hour columns could increase the score. Although Catboost works well with the default values, inbuilt "randomized search" method (Catboost) was used to tune the hyper parameters and to perform cross validation, so that the model does not overfit. We used the default value for the learning rate for tuning, as it has been mentioned in documentation that these values set by Catboost are close to the optimal values to save time in the execution. The model fitted using the "randomized search" function achieved a F1 score of 0.77 in the Kaggle dataset, and a little lesser (0.75) in our internal test set.

4.Ensemble:

To determine the best outcome from the 3 different models we tried using an ensemble method to pick mail label class based on a custom weightage across different models. But this method didn't result in a substantial improvement since we couldn't assign the weightage across multiple models because we had only 3 predictions (Giving more weightage to the best performing model was making the other 2 models irrelevant). This ensemble method could work better if we had more finely tuned classifiers.

Conclusion

In our modelling process, we found that the CatBoost methodology was the best predicting model by far, based on the F1 scores in the test set and by the score on 30% of the data in Kaggle. We hope that the model is robust enough to perform well on the other 70% of the test data as well. Hence the final model of choice remains to be CatBoost. We however feel that a better preprocessing of the features for XGBoost would also have yielded a similar result. In the future after tuning even more models to the data, we would like to implement an ensemble method with voting based on the accuracy of the individual models, as this has the potential to be the best performing model and might outperform the current classifier.