ARVY - Foundations of Machine Learning

Yasmina Hobeika
Master's in DSBA
B00804284
yasmina.hobeika@student-cs.fr

Victor Hong
Master's in DSBA
B00802726
victor.hong@student-cs.fr

Ali Najem
Master's in DSBA
B00806315
ali.najem@student-cs.fr

Arindam Roy
Master's in DSBA
B00802759
arindam.roy@student-cs.fr

Section 1:
Part 1: Pre-Processing

Before doing feature engineering, we cleaned the data and formatted the columns as follows:

1. The 'org', 'tld', 'mail_type' and 'chars_in_subject' columns were processed for NA values. Placeholder values were used for null types in org, tld and mail_type. Median values of the column were used to replace NA values for the 'chars_in_subject' column.
2. Categorical data such as 'org' and 'tld' were identified as features to be numerically label encoded.
3. One hot encoding was employed to identify and extract the 'mail_types' (multipart/alternative/text/html/etc.) into distinct categories containing binary values ('1' if present, '0' otherwise).
4. Top 20 most common occurrences in 'tld' column were identified and one hot encoded as features (.com/ .in/ .ac.com/etc.)
5. Other remaining columns/features whose data were already in numerical values were converted to 'integer' format where necessary and scaled using Standard Scaler to normalise the range of independent variables for later algorithms to be run effectively.

Part 2: Feature Engineering

After the preprocessing, we started with the feature engineering steps:

- **"Date":** To better analyse the date column, we split it into several: 'Year', 'Month', 'Day', 'Weekday', 'Hour', 'Minute', and 'Timezone' (GMT). We had to deal with exceptions, such as a row which had dashes and capitalised date inputs. We also disregarded the partially given weekdays of each date, and then appended a 'Weekday' column using a calendar function. We also adopted dropping all timezone names and only sticking with the code instead (Ex: +0000).
- **"Org" and "tld":** Label encoding was used on the mail type. The encoder was fit and transformed on the union of train and test sets.
- **"Mail _type"**: All the mail types have been Onehotencoded into different columns.
- **"Prob_labels"**: Each organisation's probability distributed by label was calculated and joined to the train and test data. This was done by grouping on the org column and label column and dividing each row by the sum of the row in the resultant dataframe.

- **Scaling and dealing with outliers:** All fields with outliers were standard scaled. This brought a normal distribution to the data and outliers were given significantly higher weights. No outliers were discarded.
- **Columns that we created:**

Name of Column	Distinct Values
Period of the Month	Start, Middle, End
Period of Week	Weekday, Weekend
Time of Day	Morning, Working hours, Evening, Night
Time period of the Day	AM, PM

- We assumed that spam emails could be identified by detecting patterns from the proportionality of characters. To test this, new features involving the proportions of the following were engineered:

Ratio	Formula
Characters in Subject proportion	$\frac{\text{Characters in Subject}}{\text{Characters in Subject} + \text{Characters in Body}}$
Characters in Body proportion	$\frac{\text{Characters in Body}}{\text{Characters in Subject} + \text{Characters in Body}}$
URLs + Images proportion in body	$\frac{\text{Number of URLs} + \text{Number of Images}}{\text{Number of URLs} + \text{Number of Images} + \text{Characters in Body}}$
URLs/Characters in body ratio	$\frac{\text{Number of URLs}}{1 + \text{Characters in Body}}$
Images/Characters in body ratio	$\frac{\text{Number of Images}}{1 + \text{Characters in Body}}$

Part 3: Dimensionality Reduction

PCA and **LDA** (Unsupervised vs. supervised learning algorithms) were important tools that helped us reduce the number of features we had based on their parameters (variance percentage and number of components). We tested our models with PCA and LDA to check if our accuracy improved in case our models were overfitting;
PCA: Used with n_components= 0.7-0.85. This means our model maintains 70-85% of its variance.
LDA: Used with n_components = 7. Then only 7 components are retained after dimensionality reduction.
Since LDA finds the direction of maximum class separability, we initially thought that it would outperform both the standard model and the one with PCA reduction. After much testing, we concluded that our full model performed better on its own, since PCA and LDA applied made

us lose valuable information from features. This in turn proved that our final dataset had very few correlated features.

Section 2: Model Tuning and Comparison

In the first iteration, the team employed an array of classification models to observe which model returned the highest degree of accuracy. This included *decision tree classification, k nearest neighbours, naive_bayes* and *random forest* models. Our initial stage of pre-processing and feature engineering, arrived at ~73 features.

'Based on these number of features, that team experimented with a range of parameters between 16 to 20. This range was selected as the team's initial hypothesis was that spam emails could be explained from features related to *date and time*, *relative proportion of characters in body and subject of email, presence of URLs and images*, as well as *type of class of email.* Preliminary results were most promising for decision tree classification and k-nearest neighbours, which scored an accuracy between 40-55%.

From the first iteration of modelling, the team preliminarily concluded that some of the initial set of features had low explanatory power, and more features needed to be engineered that might improve the prediction results. Dimensionality reduction was employed (through Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA)) to reduce the dimensions of the dataset, which marginally improved prediction results, but still not beyond 60% prediction accuracy.In addition to dimensionality reduction, the team explored and employed other models to improve prediction accuracy such as XGBoost, Neural Networks, KNN and many more.

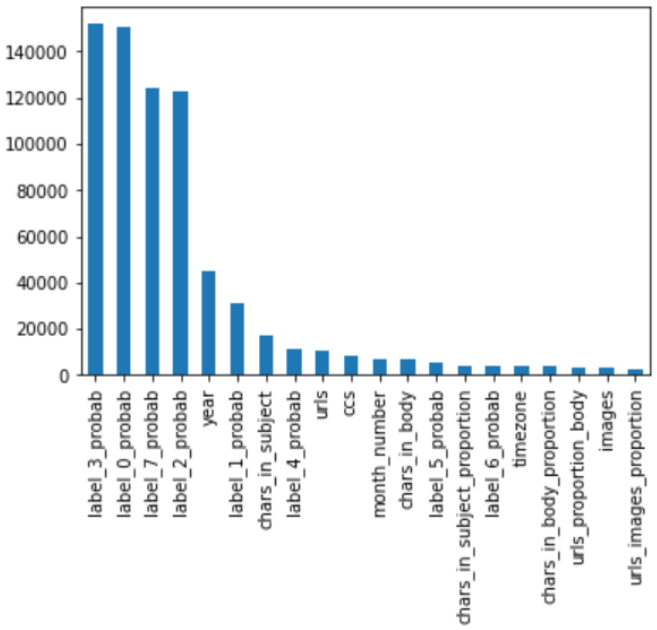

Figure 1: Best 20 features in Light GBM model

Part 1: Final Model

Our final model that performed the best relied on the **LightGBM** Classification with the following tuned parameters: max_depth = -1, num_leaves = 17, max_depth = -1, num_leaves = 17, min_data_in_leaf = 600, boosting_type = 'dart', min_gain_to_split = 0.4, lambda_l1 = 1, lambda_l2 == 1.05, n_estimators = 101, importance_type = 'gain', objective = 'multiclass'. We applied cross-validation testing (20% of the training data for testing accuracy) with every tentative model. We also used the GridSearch algorithm in order to tune the hyperparameters to optimise our model.

Initially, we used all the features that we created and it gave us an accuracy of around 59% on one third of the dataset. Then, we tried using PCA, but the accuracy decreased to 58%. So, we decided to plot the most 20 relevant features as shown in Figure 1. Based on it, we took only these 20 features and re-trained the model, and we achieved our best score in kaggle of 60% of one third of the test dataset.

Part 2: Other Models and Tentatives

K-Nearest Neighbors(KNN)

KNN has a very time-efficient algorithm, and is smooth in application. To best tune two hyperparameters n_neighbours and weights, we used gridsearch to check every possible combination and filter out the best values. 18 nearest neighbours was the final value reflecting our results of 55% on Kaggle.

XGBoost

This model has promising classification potential, and is able to multi-classify the dataset and points using a tree-base model. However, tuning its hyperparameters was heavily demanding, and required extensive running time for testing configurations with cross-validation. We think that XGBoost needs more testing to better classify our data points. For the prediction models, we set objective = “multi-softmax” which is a multiclass classification that returns a predicted class (1 of 8 in our case). The train accuracy and test accuracy ranged between 0.58 and 0.6.

Catboost

An algorithm for gradient boosting on decision trees. Similar to XGBoost, our tuned hyperparameters using Grid Search algorithm were: iterations=14, depth=10, learning_rate=0.1, l2_leaf_reg = 1, one_hot_max_size = 1000. The best accuracy achieved using this model was 0.593. Note that we also tried using LDA and PCA in this model, but was again decreasing our accuracy.

Neural Networks

The neural network model is unique based on its autonomous behaviour. With minimal work, it is able to learn the relationships (Nonlinear and complex) between data inputs and outputs. A three layer neural network was used with one hidden layer. ‘Relu’ activation function was used on the input and hidden layed and softmax and sigmoid activation functions were tested on the output layer. The data was distributed into train and test sets for training the neural network. The train accuracy and test accuracy resulted in 0.59-0.6 range.

SVM

Relies on a hyperplane in an n-dimensional space, which segregates points of data based on their probable class. For multi classification requirements, we selected the kernel sigmoid function. Best test accuracy recorded was 0.56.