Spam data analysis and prediction

Pierre Le Lay, DIA 2

Origin of the dataset

- The dataset comes from the UCI (University of California Irvine) Machine Learning Repository. It was developed by researchers at Hewlett Packard Labs in 1999 and donated the same year to the UCI Repository.

- These sources, recognised in the field of machine learning, suggest the data is likely to be reliable and of quality.

- The spam emails came from individuals who had filed spam while the non-spam emails came from work and personal emails of the research team.

The dataset

- 57 attributes are used to predict whether an email is spam.
- 48 are word frequencies named word_freq_WORD, percentage of words in the email matching WORD: decimal numbers between 0 and 100 (float).
- 6 are character frequencies named char_freq_CHAR, percentage of characters in the email matching CHAR: decimal numbers between 0 and 100 (float).

- The last three attributes are the average, longest and total (sum) lengths of sequences of uppercase letters in the email.
- capital_run_length_average is a decimal number (float).
- capital_run_length_longest and capital_run_length_total are integers (int).
- The 58th variable is the target class represented by an integer (int) taking the value 1 (spam) or 0 (not spam).
- The whole dataset contains data on 4601 emails. 1813 of those are spam emails. Our dataset is not evenly distributed.

- Before we start, we need to define spam. In the case of emails, a short definition would be “unwanted or undesirable emails”.

- The aim of a spam filter is to identify emails that are not wanted by the recipient.

- Our goal here is to analyse the dataset and use it as both training and testing data, find links between the features and the target class (spam / not spam) and develop an accurate prediction model for this data.

Validity of the data

- Because of the origin of the non-spam data, using it to predict whether a new email is spam would likely not bring very satisfying results.
- Indeed, some words used in the features – ‘george’, ‘650’, ‘hpl’, ‘hp’, ‘857’ for instance – are specific to the research team emails, like phone numbers, names, company and lab names.
- Also, as the dataset dates back to 1999 other features complete this list: ‘1999’ and ‘telnet’ are two.

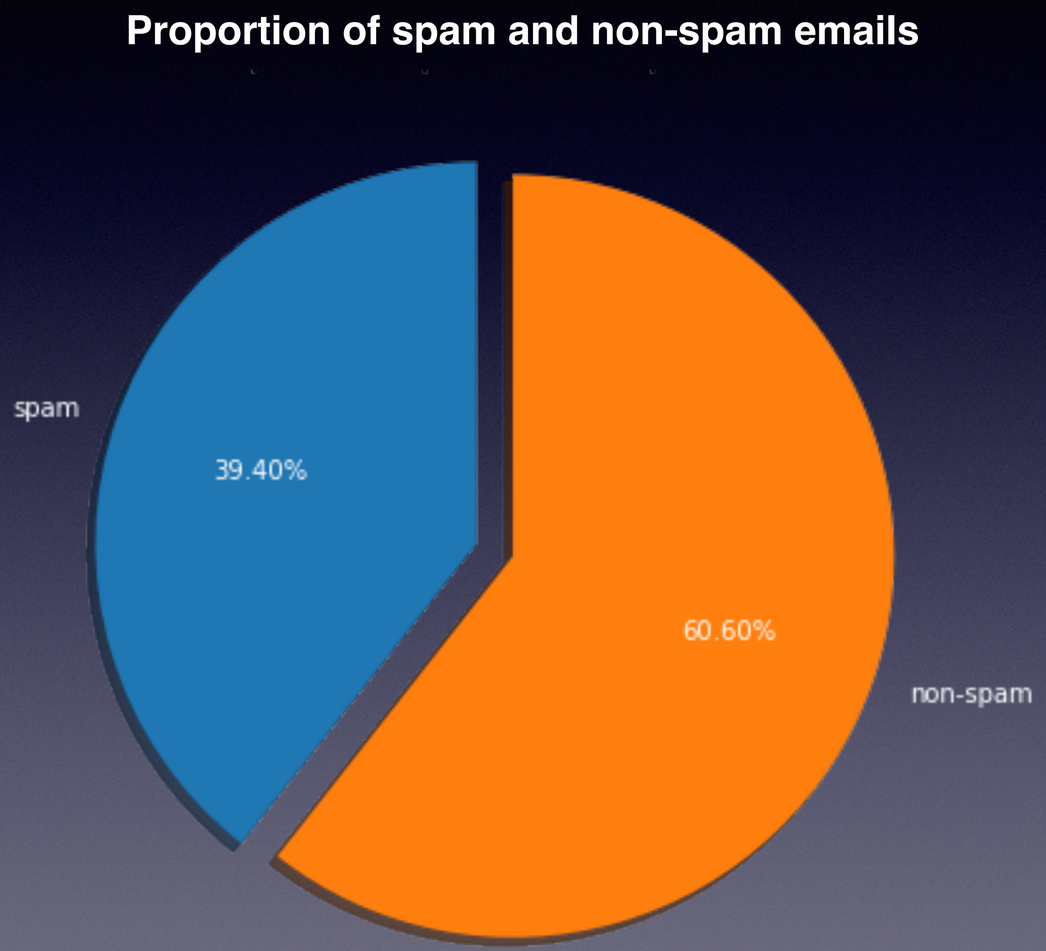
- Using this dataset to predict the spam nature of your personal emails or any other emails not related to that research team or lab would most likely output results of limited accuracy.

- Furthermore, emails have evolved since 1999. Today they could contain videos, hyperlinks and content that this spam filter would not detect. Any link, video or image would go undetected.

- And the undesirable nature of an email varies from one user to the next. The words used in this dataset were specific to the researchers and were useful for their personal spam filter. Any attempt to use it outside of that limited context of place and time would bring results of lesser quality.

Data analysis

- After importing the dataset, adding the feature names and checking for missing values (none were found), we start analysing and exploring the data.
- The data we have is unevenly distributed. We will take that into account when building prediction models to make sure it doesn't create a bias in our models.
- The graph shows us that imbalance.



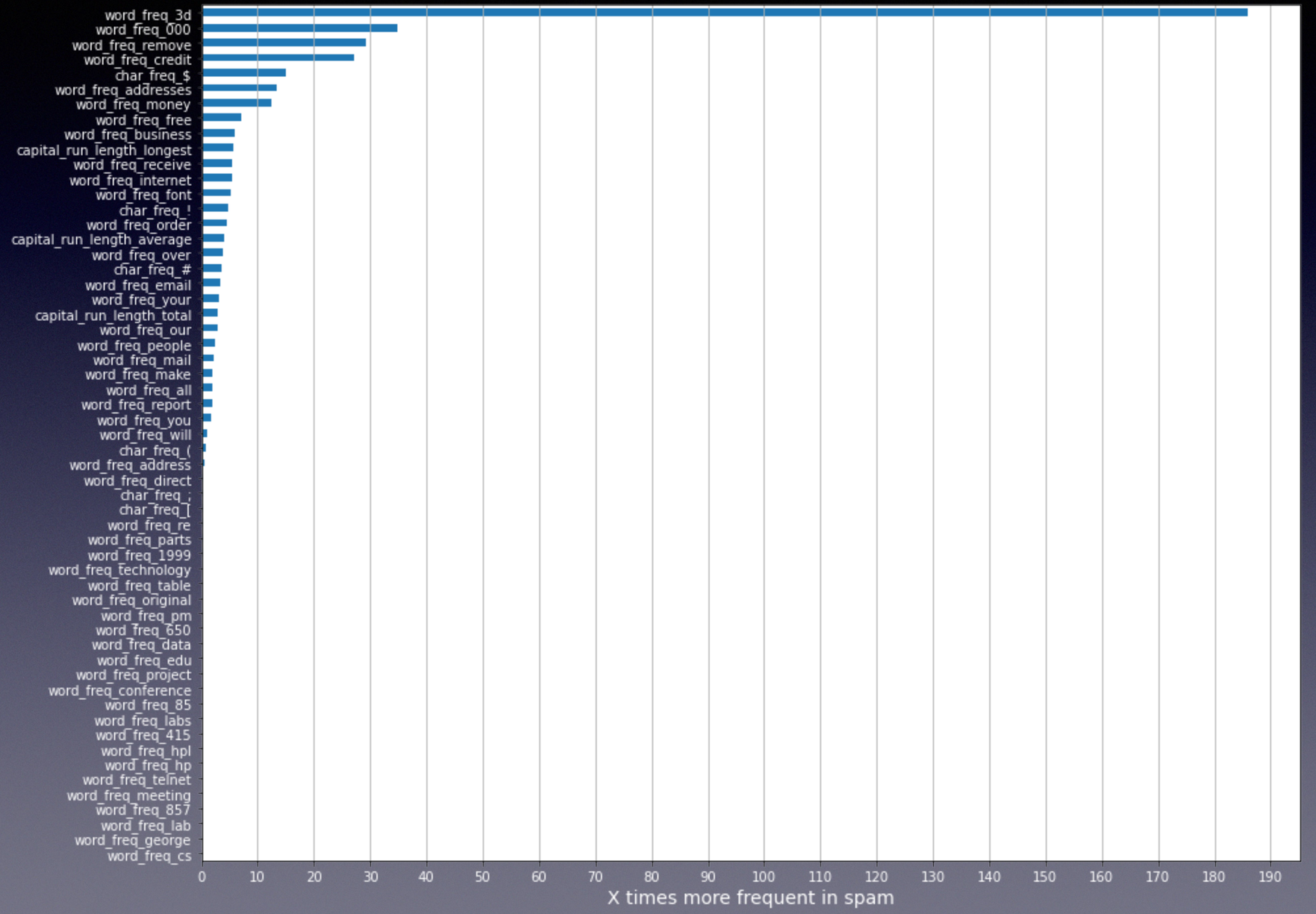
- Understanding which features have the most influence on the target class is useful to build prediction models with less input variables, using only the most significant ones. This makes the models use less resources (of time and memory).

- Although fewer features can mean lower performance, the right selection can make the difference in performance very low.

- To find out the most significant variables, we compare the words' and characters' frequencies in one class versus the other. We use the ratio of one on the other. This tells us how much more frequent a variable (word or character) is in one class than the other.

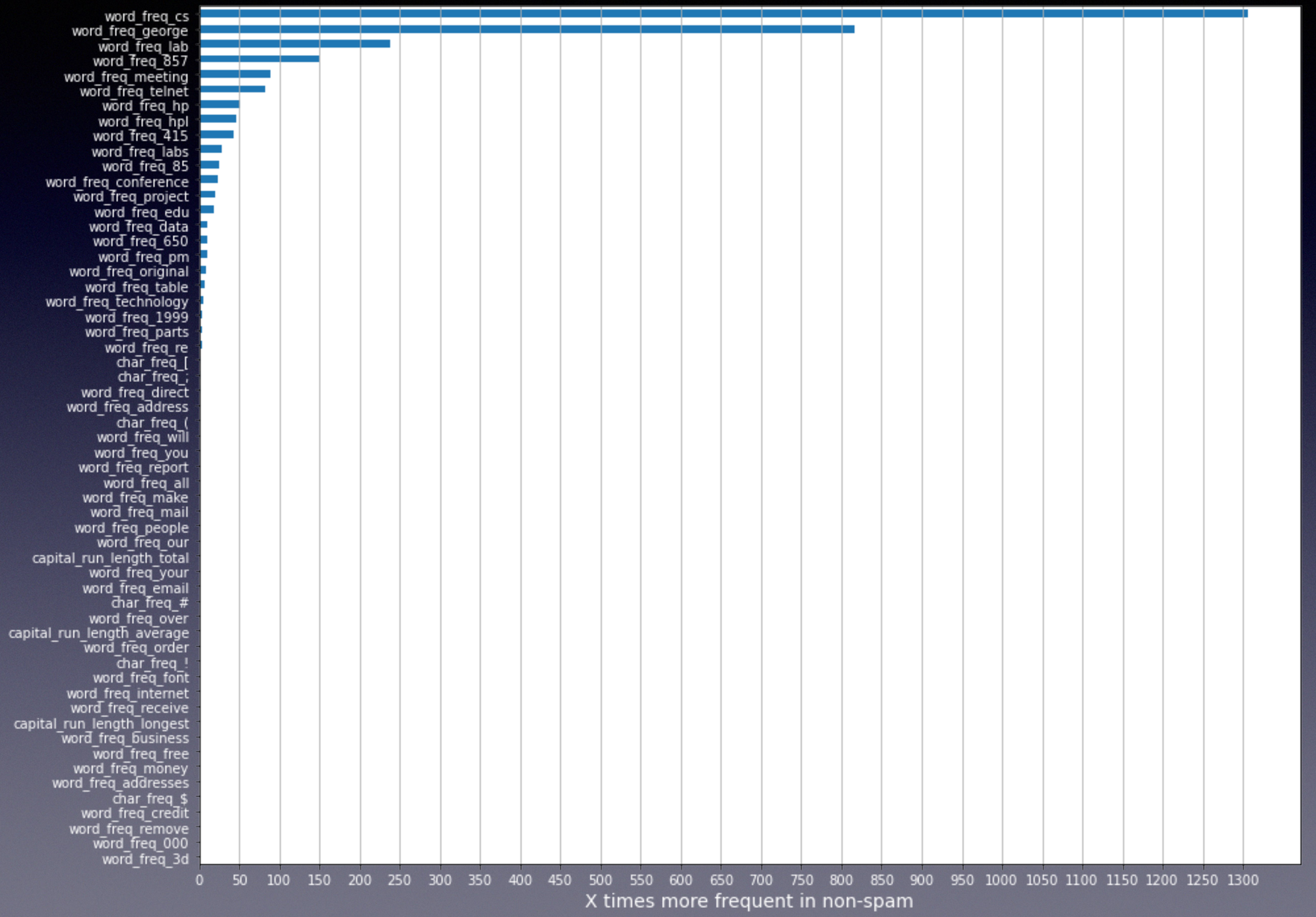
Most frequent words and characters on average in spam emails vs non-spam emails

- This graph shows how much more frequent some words and characters are in spam emails compared to non-spam emails.
- For instance, ‘3d’ is more than 180 times more frequent in spam emails. And ‘000’ about 35 times more frequent in spam emails.

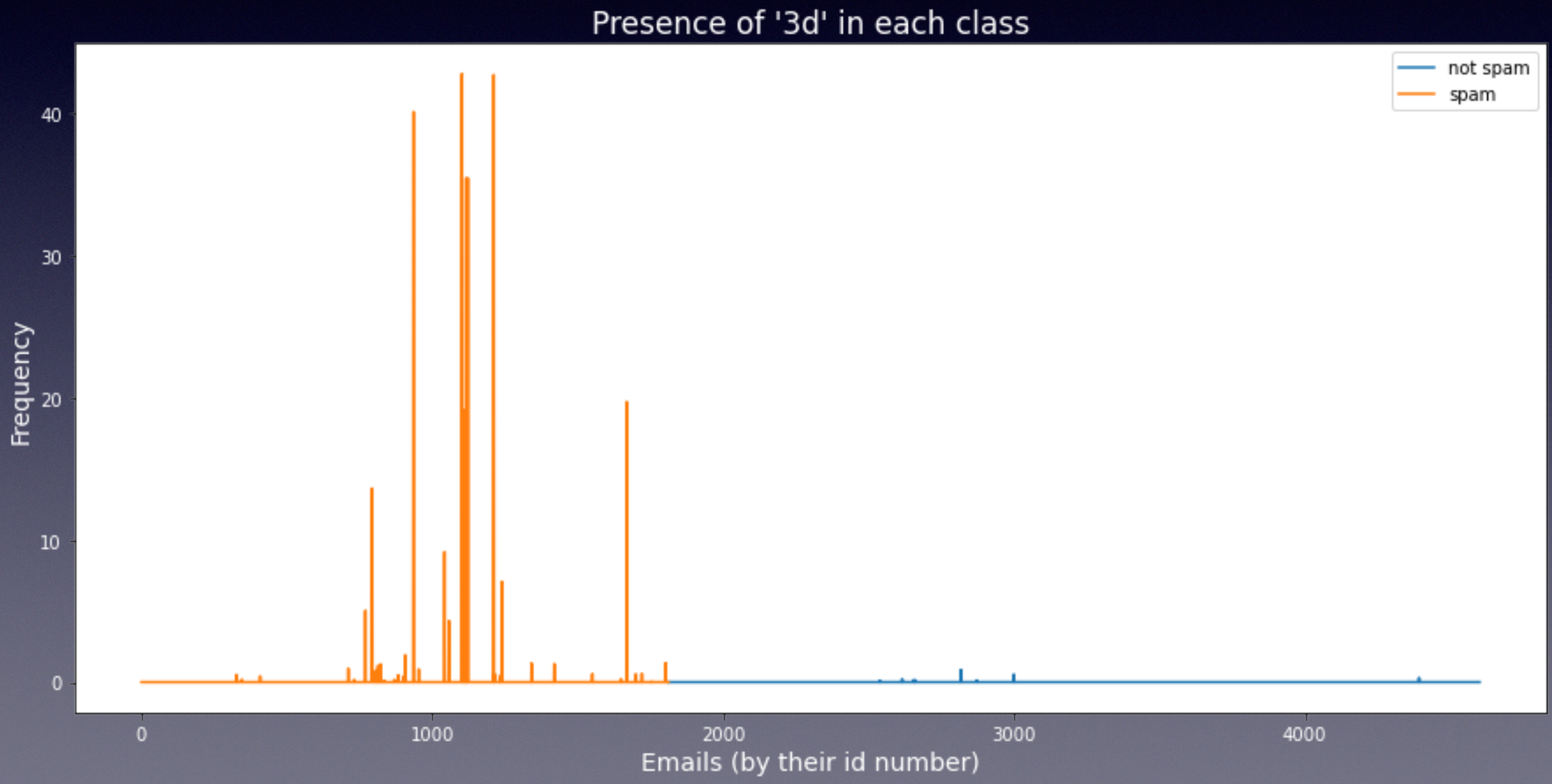


Most frequent words and characters on average in non-spam emails vs spam emails

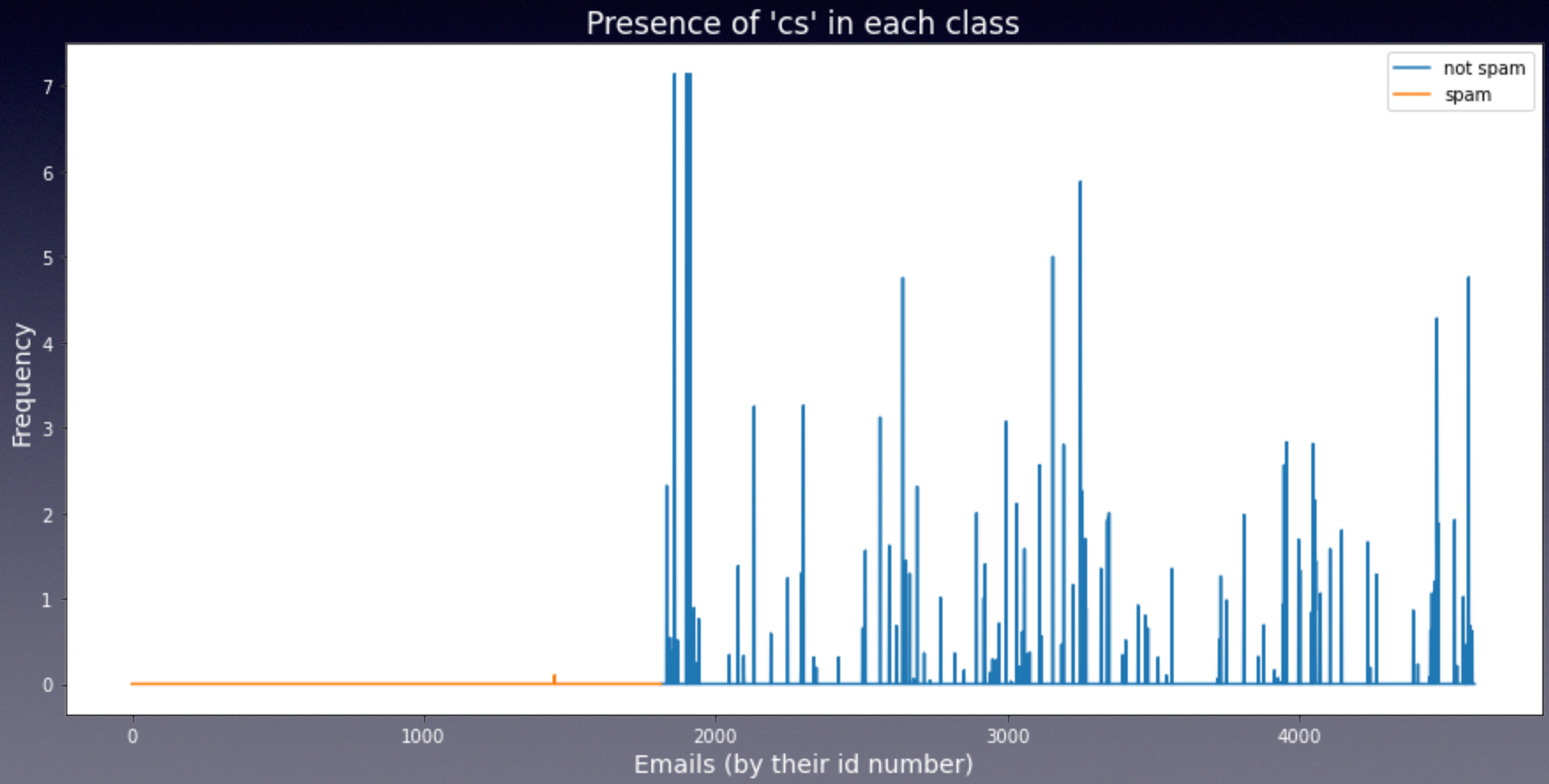
- This graph gives the same information, this time for non-spam emails.
- Examples: ‘cs’ is more than 1300 times more frequent in non-spam emails. And ‘george’ more than 850 times.



Here we show the repartition of the most frequent word in spam: ‘3d’.
It is clear that its frequency is much higher in spam emails than in non-spam.



We do the same with the most frequent word of non-spam emails: ‘cs’.
It is indeed very frequent in emails not classified as spam, much more than in the other class.



Prediction models

- After some analysis of the spam dataset comes the prediction phase.
- We will choose models, train them on some part of the data and test their performance both in training phase (with cross-validation) and in testing phase with some part of the original data not used in training.
- The evaluation of performance at training will allow us to alter our models and choose inputs that will maximise that performance.
- Finally we'll compare the models' performances, all evaluated with balanced accuracy to take into account the imbalance of the data.

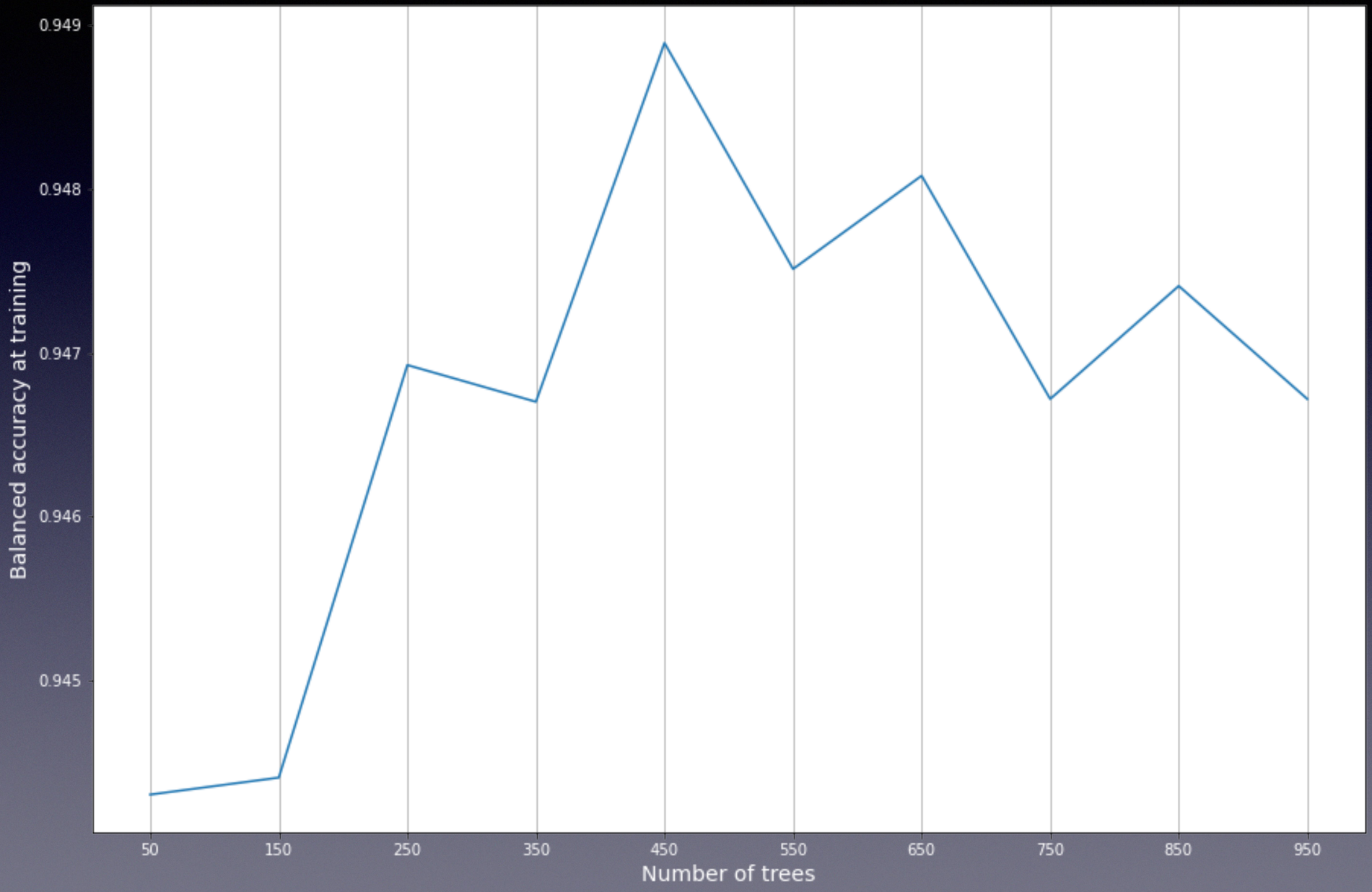
The 4 chosen models

- Random Forest
- K-Nearest Neighbours
- Gaussian Naive Bayes
- K-Means
- All first three models (supervised learning) were trained on 80% of the data. The K-Means model (unsupervised learning) was not trained on any data.

Random Forest

- This model can take several input parameters. We focused only the number of trees, often called n_estimators.
- We created a function to try several values for this hyperparameter and find the one giving the best prediction performance at training.
- We then used that value to make predictions with new, unseen before data and assess the accuracy of those predictions.

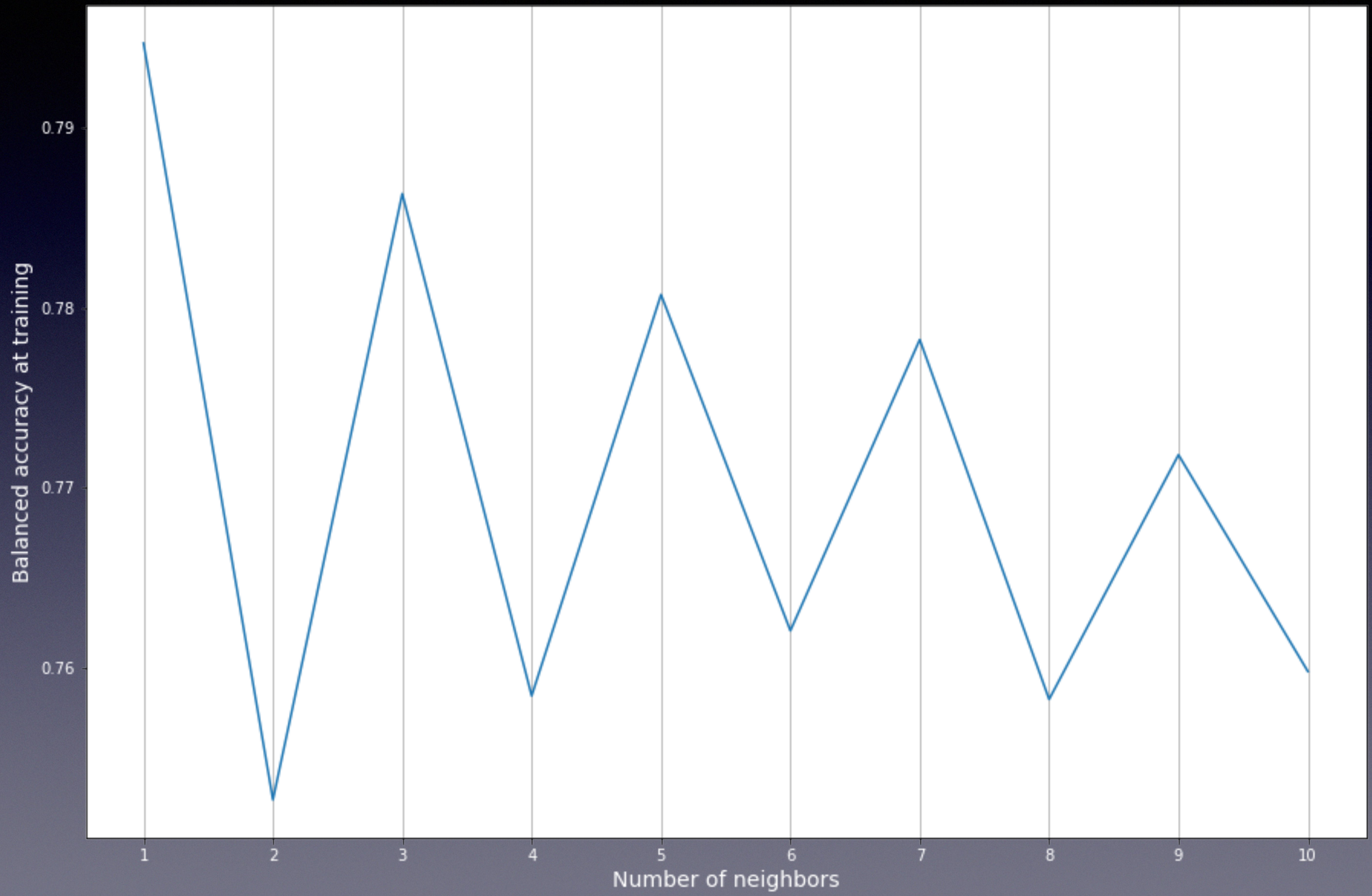
Here is the balanced accuracy plotted for several values of the number of trees. Notice that though it seems to vary a lot, a look at the y-axis shows it is not the case.
With a near 95% score both at training and final test, this model achieves a very good performance.



K-Nearest Neighbours

- For this model, we focused on the number of neighbour points used to classify a given point, the k in the model name.
- Here, too, we designed a function to try several values and find the the one giving the best performance at training.
- After making predictions on new data, we compared the results with the actual labels of that data to get a final performance.

Below is the balanced accuracy at training for different values of k. The actual variation of the score is more significant for this model.
This KNN model performed a 79% score at training and almost 81% on new data.



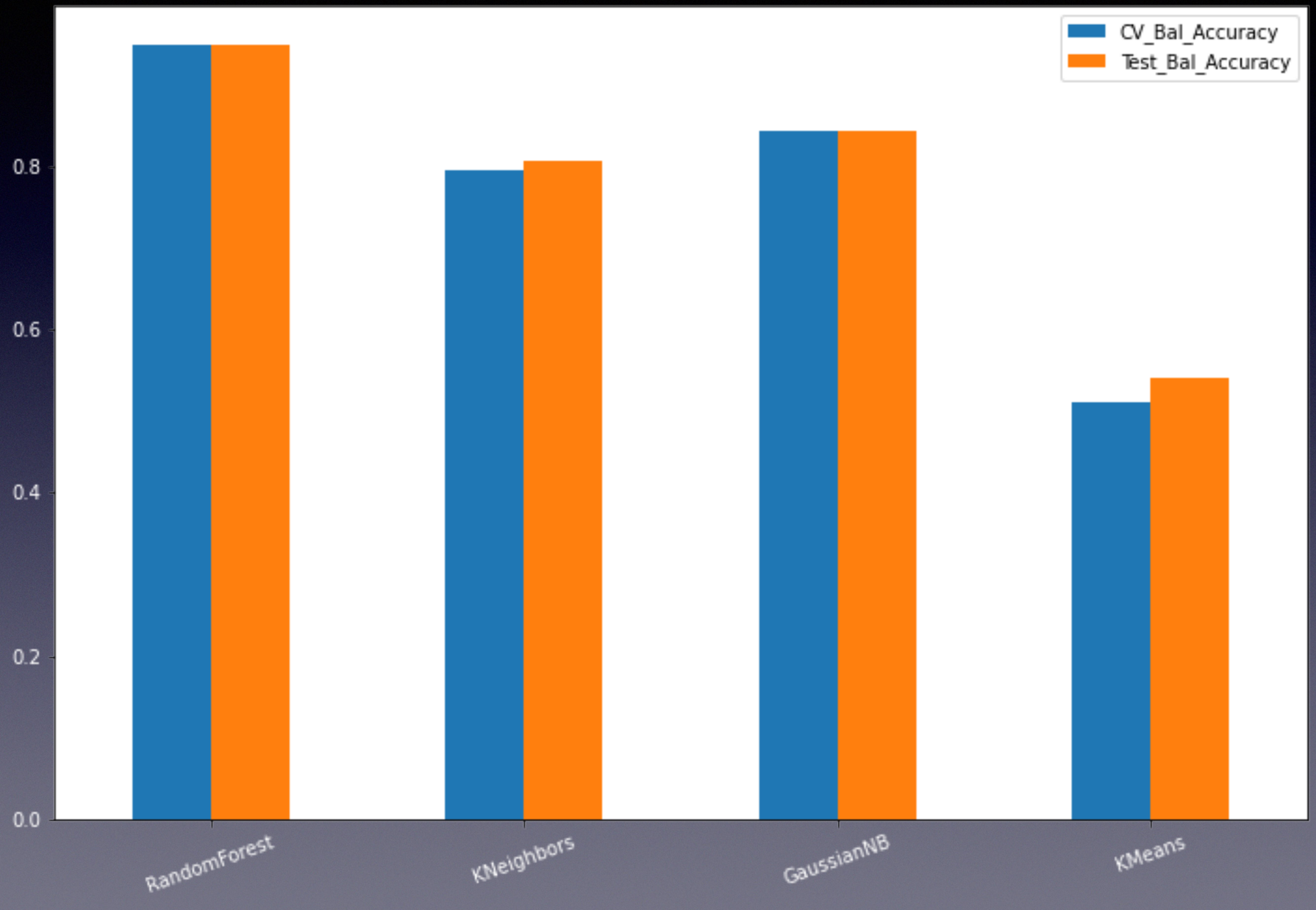
Gaussian Naive Bayes

- This Naives Bayes model has much fewer input parameters than the previous ones.
- We didn’t give any input and therefore didn’t have to try different input values.
- We trained the model, assessed its performance at training and a second time with new data.
- Performance at training: 84%. And the same again at testing.

K-means clustering

- This is an unsupervised learning model making predictions on input data without prior training, classifying based on resemblance of points.
- Final performance: 54%.

Models comparison



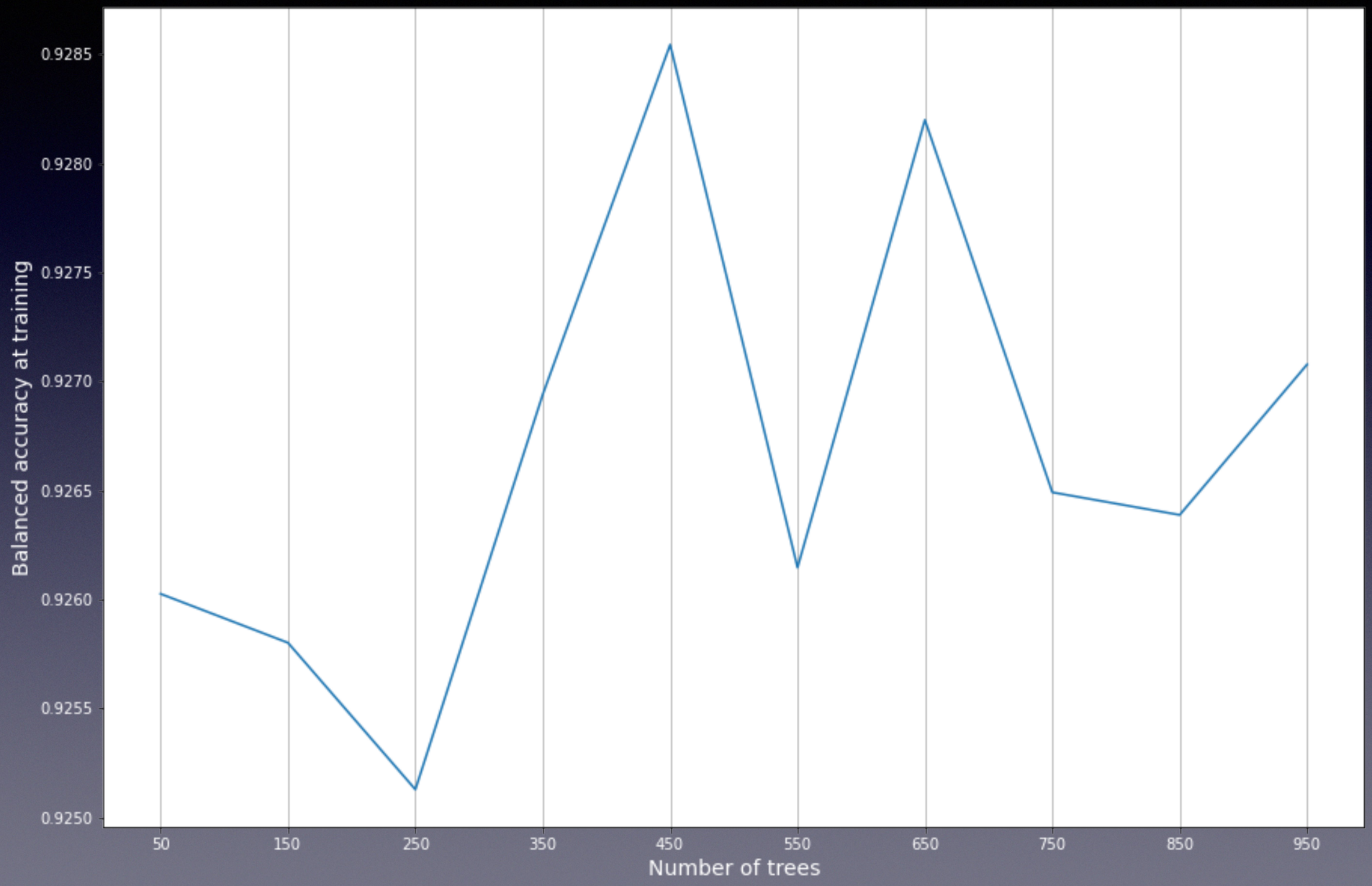
- The best performing model: Random Forest with an excellent score (95%).
- The K-Means model underperformed in this context but doesn't always in others.
- The other two still give good performances (between 80 and 85%).
- The Random Forest model is very good at predictions but the decision process behind is rather complex and obscure. In a situation where explainability matters, it might not be the best model.

A simpler Random Forest model

- Remember earlier we showed on graphs the most frequent words and characters in each target class. We will now this information to build a Random Forest model with fewer features.
- The most frequent words and characters in spam that we choose: ‘3d’, ‘000’, ‘remove’, ‘credit’, ‘$’, ‘addresses’, ‘money’.
- In non-spam: ‘cs’, ‘george’, ‘lab’, ‘857’, ‘meeting’, ‘telnet’, ‘hp’, ‘hpl’, ‘415’.
- We also add the average, longest and total lengths of sequences of uppercase letters.

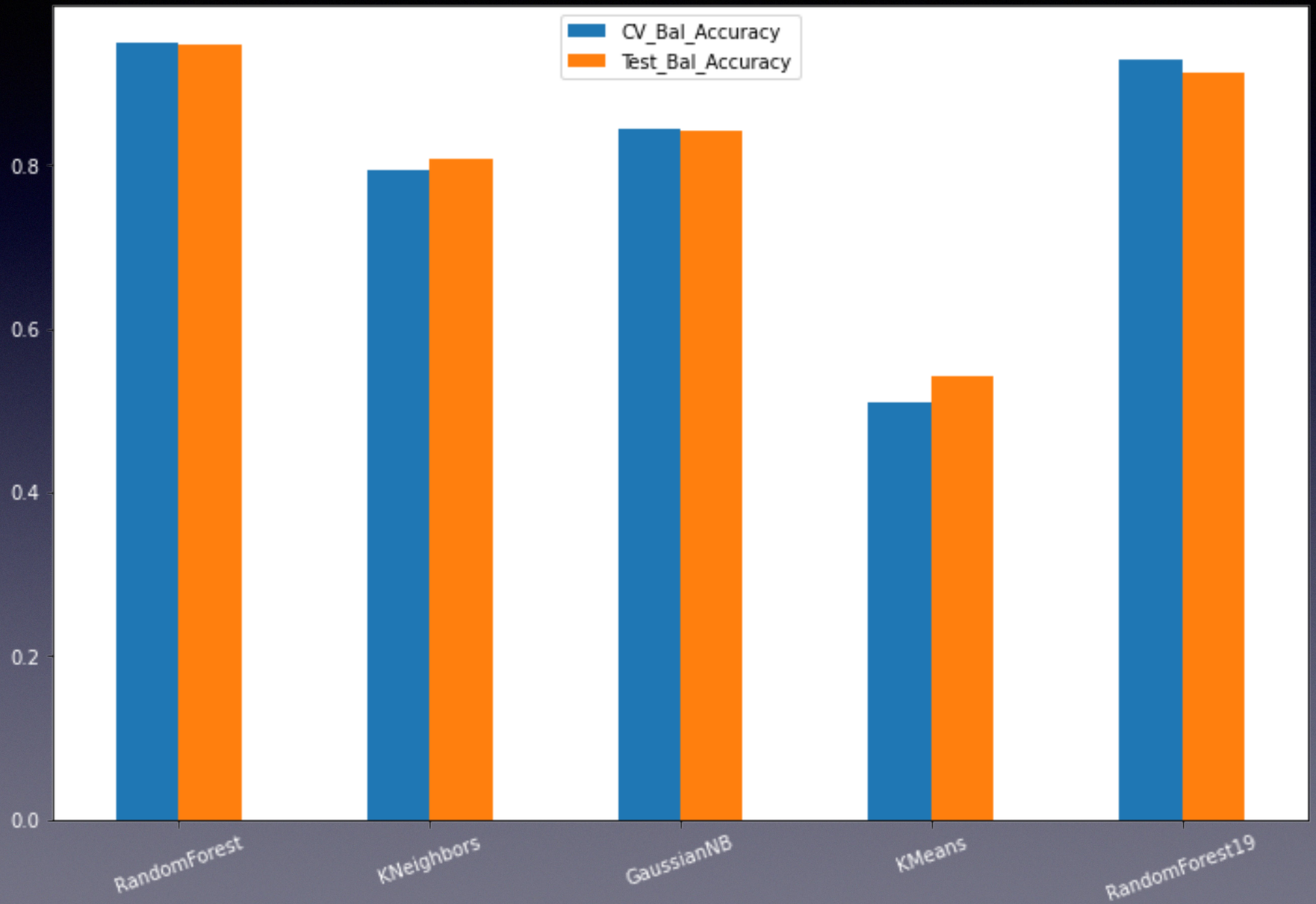
- We have a total of 19 features. That’s one third of our initial number (57).
- We follow the same process as with the first Random Forest model, adjusting the number of trees to get the best training performance and evaluating again with new data.

Below we have the different values of balanced accuracy depending on the number of trees.
Performance at training: around 92%. And similar again on new data.



Models comparison

- The new Random Forest model achieves a performance nearing that of the full model (92% vs 94-95%).
- And it outperforms all the others as well.
- This shows the importance of choosing the right features. Such a model is lighter and needs less resources. In real-world applications with much more data, this could make a major difference.



Thank you for watching