CP8305 - Knowledge Discovery Final Project

Knowledge Discovery on Spambase Dataset

Advanced Data Analysis With Machine Learning and Deep Learning Models .

Ahmed Al-Daeni

12th NOV 2021

Table of Contents

Problem Statement

The "spam" concept is diverse, It can be from advertisements for products/websites, make money fast schemes, chain letters. In this project, we will be analyzing the spambase dataset[1] and implementing several machine learning models and deep learning models using python to be able to predict if an email is spam or not. This project will provide details on how to analyze the data and implement various machine learning algorithms , and implement deep learning neural networks to find the best model with the highest accuracy.

Data set: Spambase dataset

We used the Spambase dataset [1] for our assignment. The dataset contains 4601 instances and 57 attributes. Spambase dataset[1] contains a collection of spam emails that came from individuals who had filed spam, and a collection of non-spam emails came from filed work and personal emails. There were no missing values in each feature of the data set.

48 continuous real [0,100] attributes of type word_freq_WORD
= percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.

6 continuous real [0,100] attributes of type char_freq_CHAR]
= percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail

1 continuous real [1,...] attribute of type capital_run_length_average
= average length of uninterrupted sequences of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_longest
= length of longest uninterrupted sequence of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_total
= sum of length of uninterrupted sequences of capital letters
= total number of capital letters in the e-mail

1 nominal {0,1} class attribute of type spam
= denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

The list of the attribute and their data type

```
RangeIndex: 4601 entries, 0 to 4600
Data columns (total 58 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   word_freq_make              4601 non-null   float64
 1   word_freq_address           4601 non-null   float64
```

2	word_freq_all	4601 non-null	float64
3	word_freq_3d	4601 non-null	float64
4	word_freq_our	4601 non-null	float64
5	word_freq_over	4601 non-null	float64
6	word_freq_remove	4601 non-null	float64
7	word_freq_internet	4601 non-null	float64
8	word_freq_order	4601 non-null	float64
9	word_freq_mail	4601 non-null	float64
10	word_freq_receive	4601 non-null	float64
11	word_freq_will	4601 non-null	float64
12	word_freq_people	4601 non-null	float64
13	word_freq_report	4601 non-null	float64
14	word_freq_addresses	4601 non-null	float64
15	word_freq_free	4601 non-null	float64
16	word_freq_business	4601 non-null	float64
17	word_freq_email	4601 non-null	float64
18	word_freq_you	4601 non-null	float64
19	word_freq_credit	4601 non-null	float64
20	word_freq_your	4601 non-null	float64
21	word_freq_font	4601 non-null	float64
22	word_freq_000	4601 non-null	float64
23	word_freq_money	4601 non-null	float64
24	word_freq_hp	4601 non-null	float64
25	word_freq_hpl	4601 non-null	float64
26	word_freq_george	4601 non-null	float64
27	word_freq_650	4601 non-null	float64
28	word_freq_lab	4601 non-null	float64
29	word_freq_labs	4601 non-null	float64
30	word_freq_telnet	4601 non-null	float64
31	word_freq_857	4601 non-null	float64
32	word_freq_data	4601 non-null	float64
33	word_freq_415	4601 non-null	float64
34	word_freq_85	4601 non-null	float64
35	word_freq_technology	4601 non-null	float64
36	word_freq_1999	4601 non-null	float64
37	word_freq_parts	4601 non-null	float64
38	word_freq_pm	4601 non-null	float64
39	word_freq_direct	4601 non-null	float64
40	word_freq_cs	4601 non-null	float64
41	word_freq_meeting	4601 non-null	float64
42	word_freq_original	4601 non-null	float64
43	word_freq_project	4601 non-null	float64
44	word_freq_re	4601 non-null	float64
45	word_freq_edu	4601 non-null	float64
46	word_freq_table	4601 non-null	float64
47	word_freq_conference	4601 non-null	float64

48	char_freq_;	4601 non-null	float64
49	char_freq_(	4601 non-null	float64
50	char_freq_[	4601 non-null	float64
51	char_freq_!	4601 non-null	float64
52	char_freq_$	4601 non-null	float64
53	char_freq_#	4601 non-null	float64
54	capital_run_length_average	4601 non-null	float64
55	capital_run_length_longest	4601 non-null	int64
56	capital_run_length_total	4601 non-null	int64
57	spam	4601 non-null	int64

dtypes: float64(55), int64(3)

Data Pre-processing & Analysis of the dataset

The spambase dataset[1] contains two files spambase , which are data and spambase. names. The first step was to recuperate the column's names in spambase.names document, then convert the spambase.data to CSV format and add the names of the columns in spambase.names to the dataset.

Analysis of the dataset

1. Bar chart:

The bar chart shows the numbers of safe and spam emails. The safe emails were 2788 emails and 1813 spam emails. The bar chart indicates that safe emails are higher than spam as shown below.

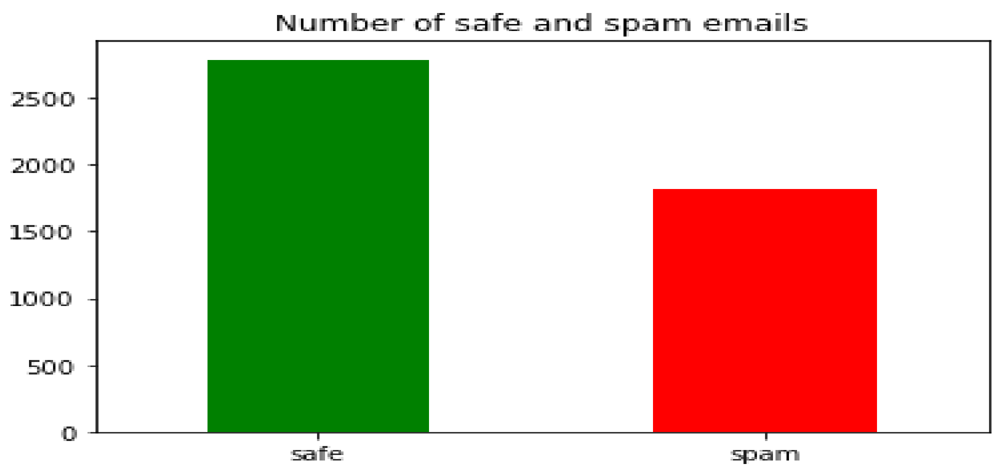


Figure 1

2. Scatter plot

The second visualization of the dataset is Scatterplot. The visualization of the individuals' distribution according to their type (non-spam or spam) and categorical target (0 or 1). To analyze this visualization, In figure 2 shows that the word "000" frequency is higher in spam email than in non-spam frequency. The spam is higher than 5%, according to our dataset a model which should predict the input as spam. The second example is the word "original", their frequency is higher in non-spam email than in spam frequency, and the non-spam is higher than 3.5%, which should predict the input as a non-spam.

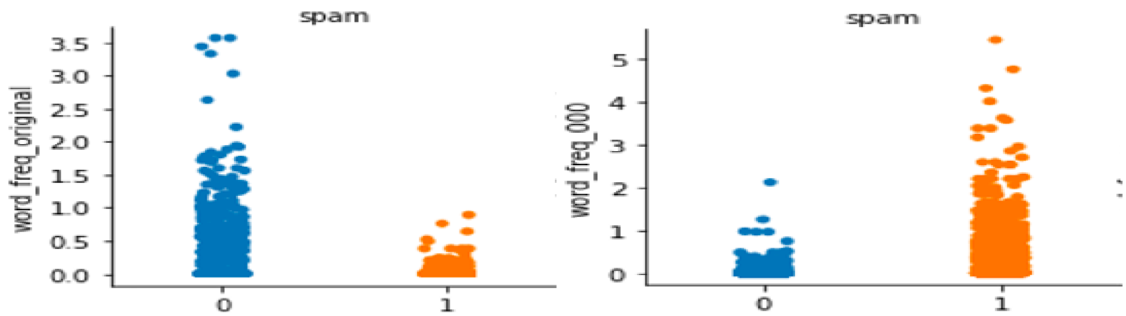


Figure 2

Below in figure 3 listed all the visualization of the attributes of the datasets to predict whether if the frequency is non-spam email or in spam.

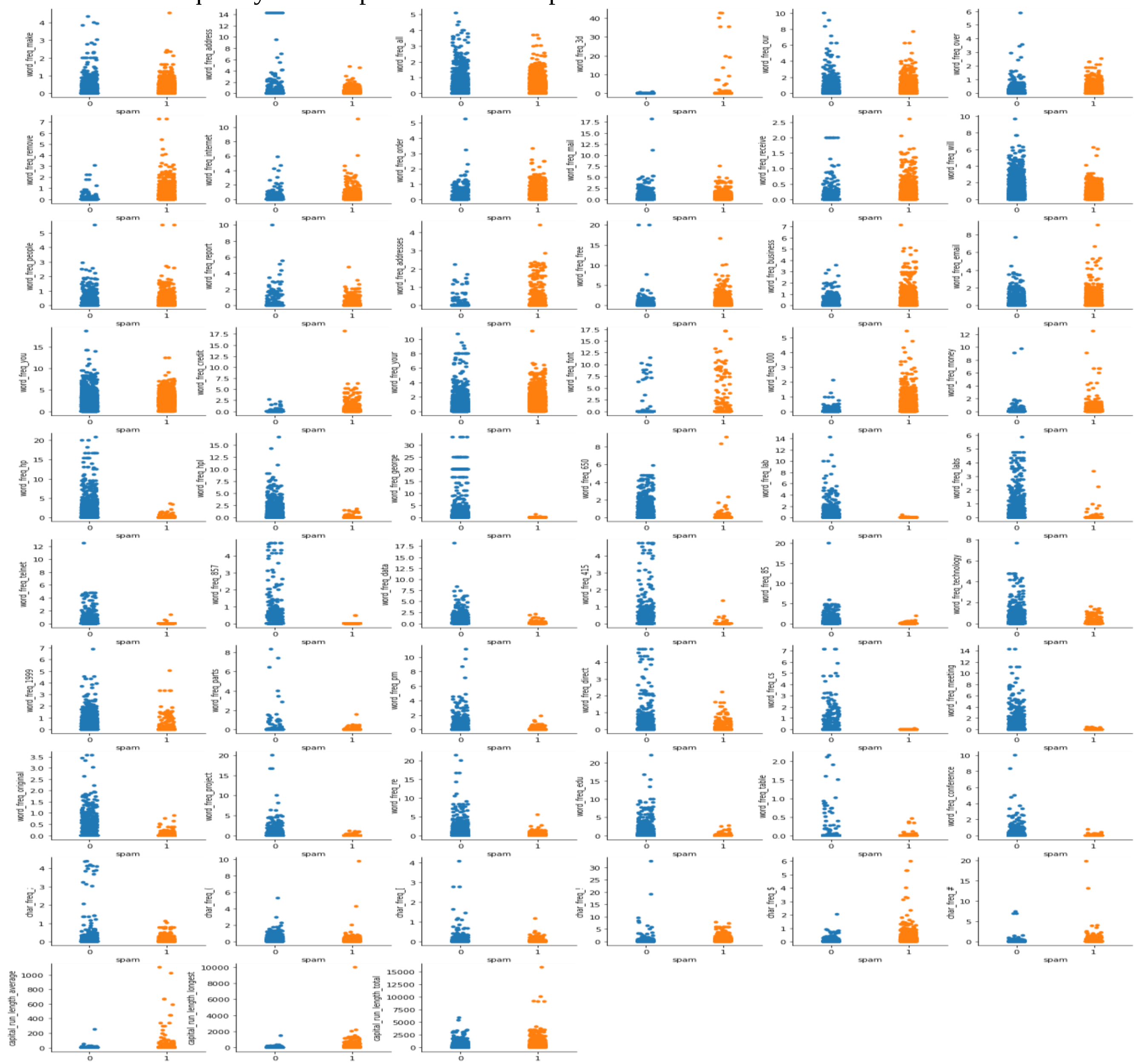


Figure 3

3. Boxplot

The third visualization of the dataset is Boxplot**.** For better visualization of our data, I used a logarithmic scale for our boxplot. In figure 3 some boxplots show attributes for which spam emails generally have higher frequencies.

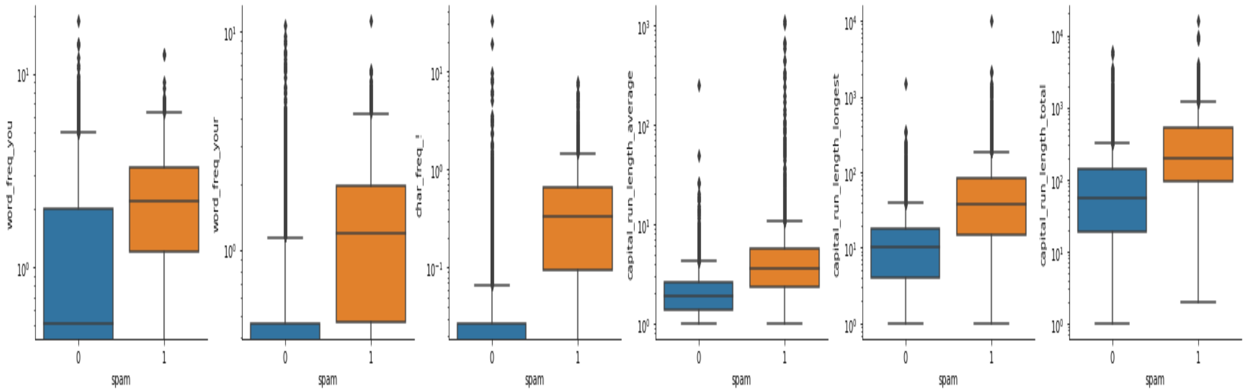


Figure 4

4. Barplot

Then, plot the sum of frequency for each attribute for a safe mail and spam and only plot attribute names for the attributes with a total frequency higher than 500. The word "you" is more present in both kinds of mail but more in spam even if they are fewer in our dataset than safe mail. While the word "George" and "hp" are also very present in safe mail.

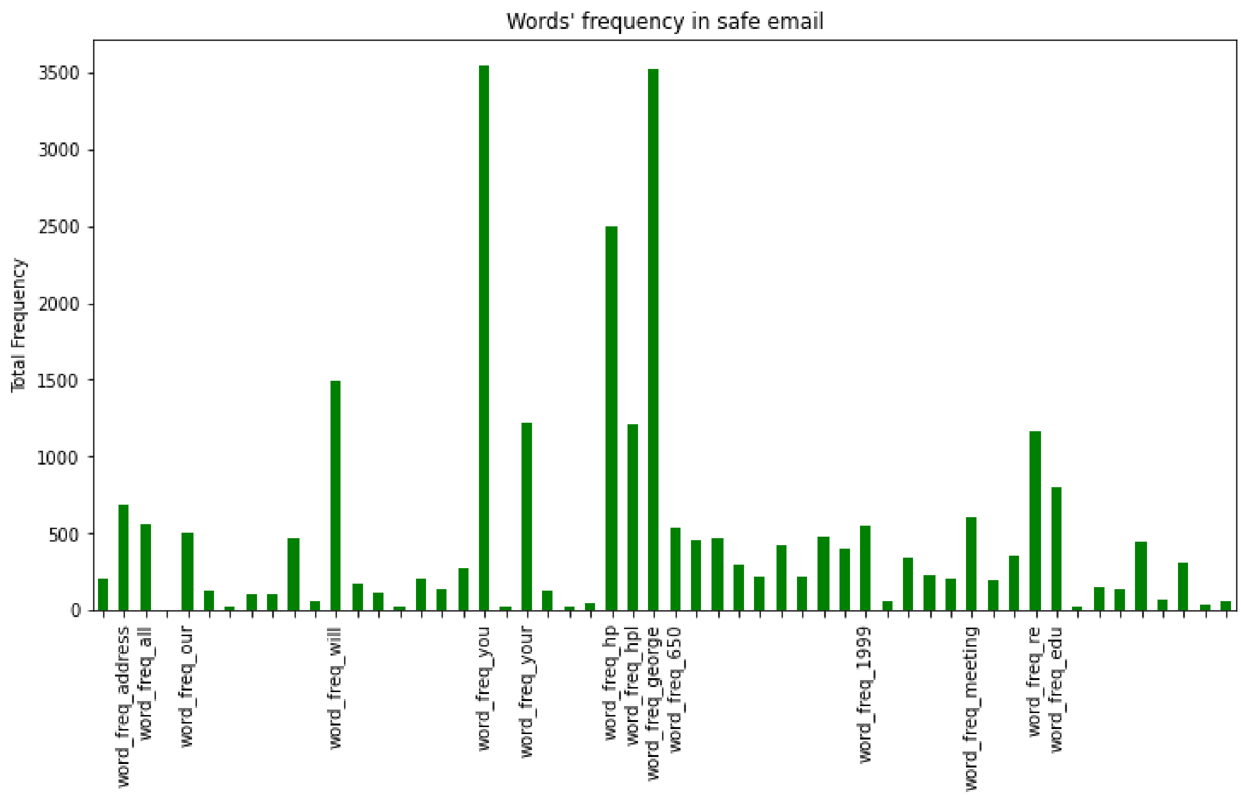


Figure 5

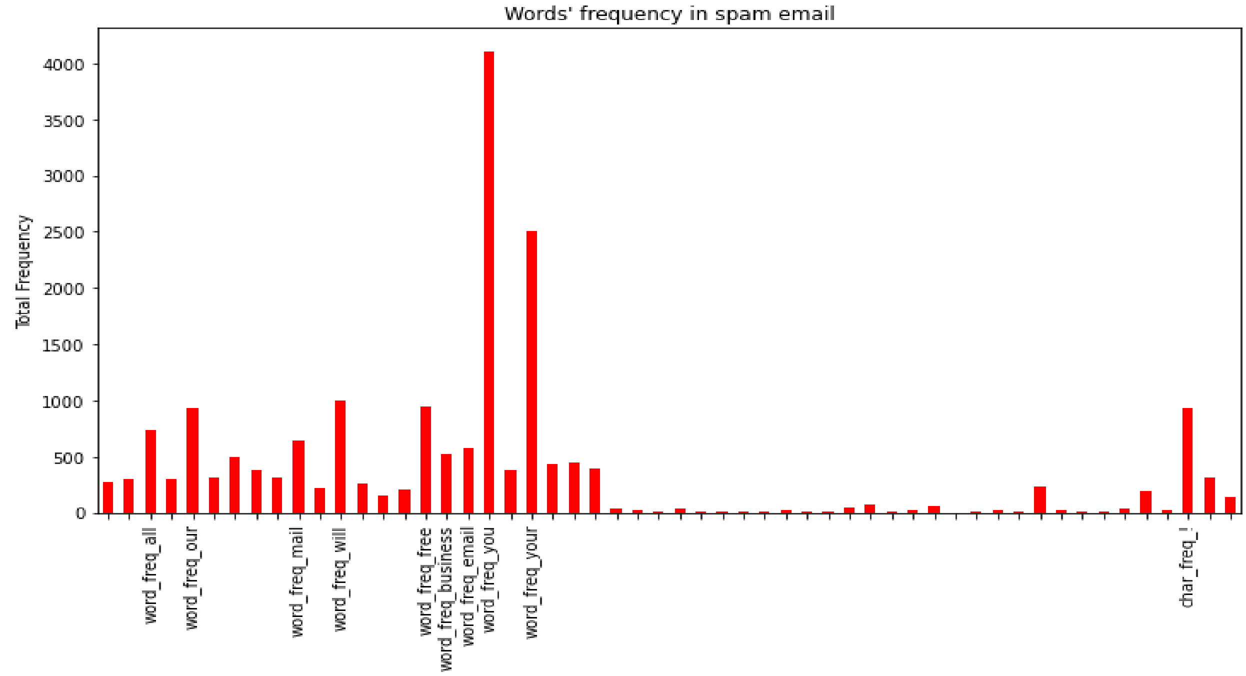


Figure 6

5. Correlation

Finally, in figure 7 I implemented a correlation matrix plot but it seems that result is not clear and interpretable result. so in figure 8 I applied only plot the correlation of attributes to the target. A positive correlation means a high frequency of the attribute implies the mail is spam and vice versa for a negative correlation. We can see that according to our previous plots, hp, hpl, and George presence tends to predict a non-spam while your or remove are more present in spam.

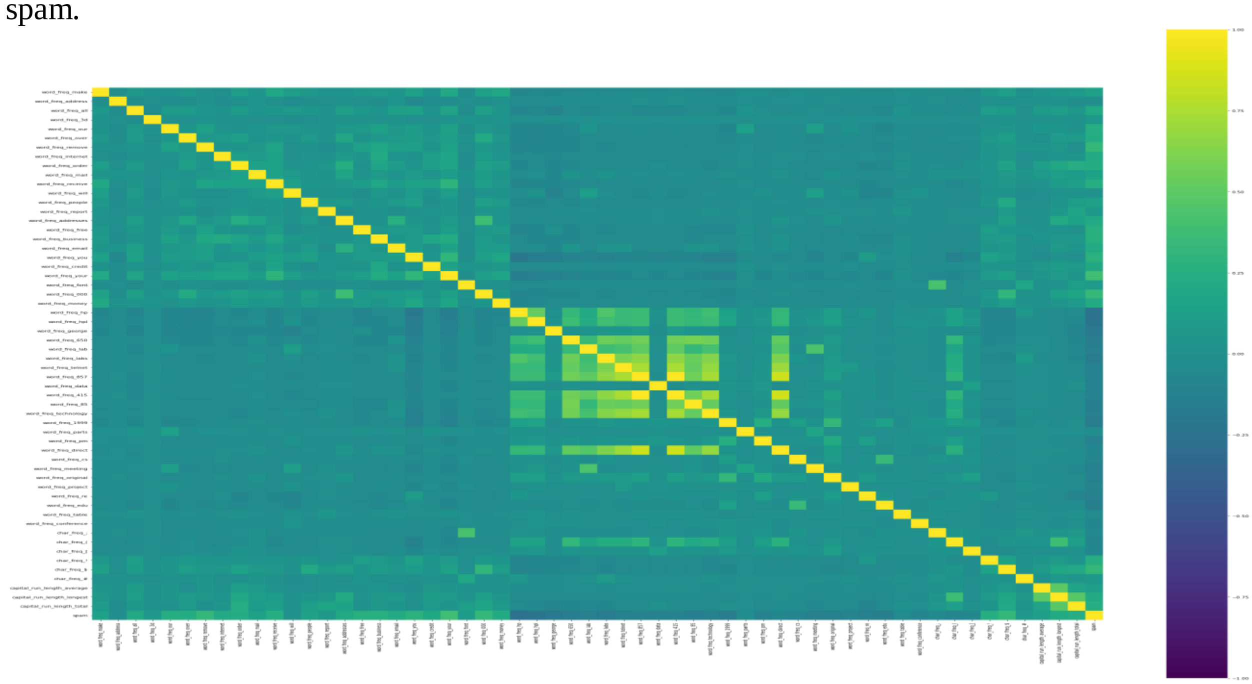


Figure 7

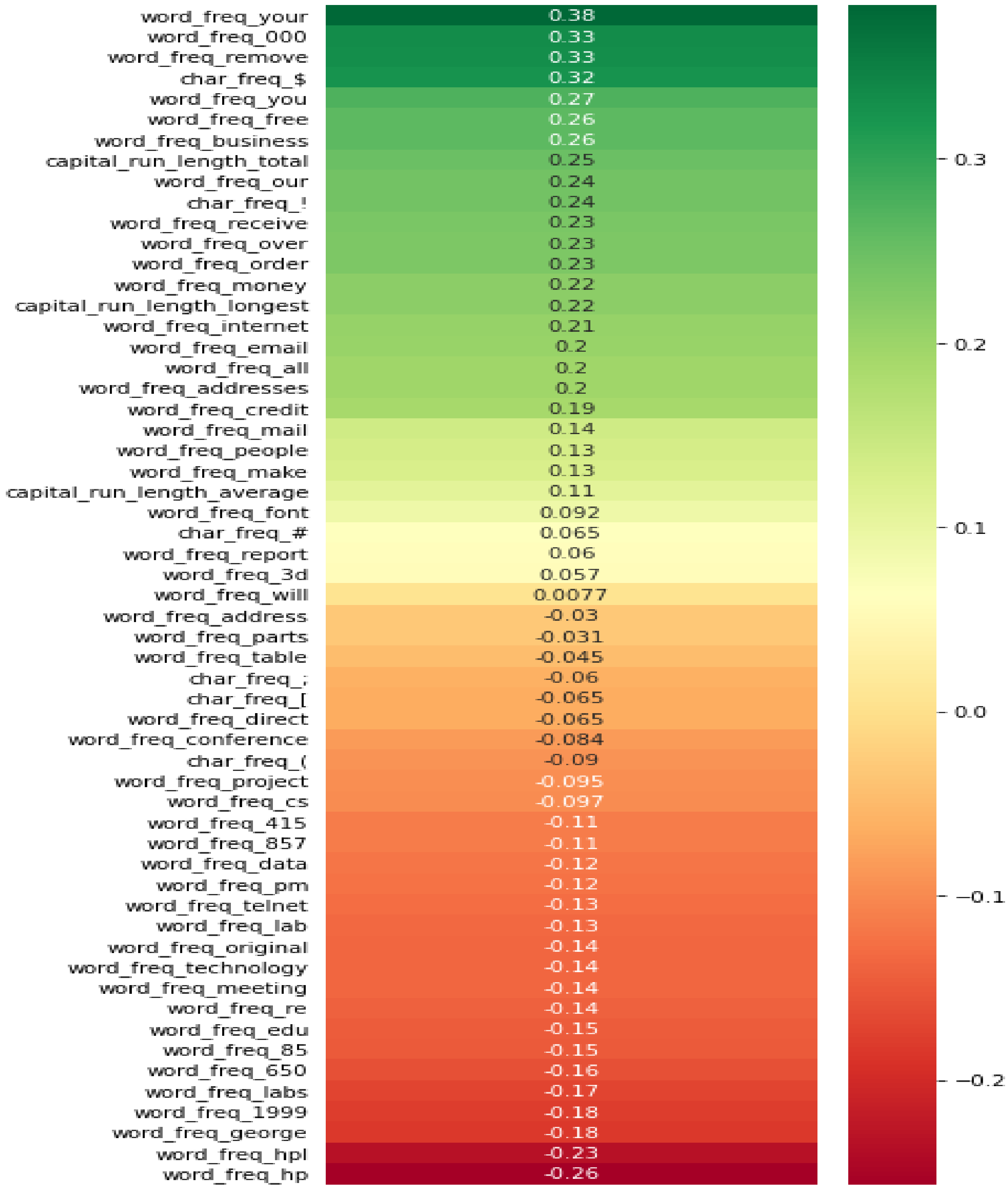


Figure 8

According to the correlation between some attributes and the target, I defined a threshold and selected attributes that have a higher correlation with the target than the positive threshold or a lower correlation than the negative threshold. The result of the selected attributes was 41.

Preprocessing Data

Then split the dataset in half: a training set with 75% of the dataset and a test set with the last 25%. And we make the same operation with the selected attributes.

Training set with all attributes shape: (3450, 57)
Testing set with all attributes shape: (1151, 57)

Training set with selected attributes shape: (3450, 41)
Testing set with selected attributes shape: (1151, 41)

Training set with target value shape: (3450,)
Testing set with target value shape: (1151,)

Machine Learning Models

The purpose of this project is to make a model to predict spam emails are safe emails or spam. Accuracy is the most important of our model. However, I focused on the precision which is explained by the number of true spam among all instances predicted as spam. The machine learning models I used are Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and SVC. I have applied all the models to all attributes and selected attributes.

1- **Logistic Regression**
   I have used the sklearn library, specifically sklearn.linear_model to import the LogisticRegression function to implement Logistic Regression Model, the maximum iteration was 2000, and the random state was 10.
2- **Decision Tree**
   I have used the sklearn library, specifically sklearn.tree to import the DecisionTreeClassifier function to implement the Decision Tree Model, and the random state was 10.
3- **Random Forest**
   I have used the sklearn library, specifically sklearn.ensemble to import the RandomForestClassifier function to implement Random Forest Model, and the random state was 10.
4- **Gradient Boosting**
   I have used the sklearn library, specifically sklearn.ensemble to import the GradientBoostingClassifier function to implement Gradient Boosting Model, and the random state was 10.
5- **SVC**
   I have used the sklearn library, specifically sklearn.SVM to import SVC function to implement Gradient Boosting Model and the random state was 10.

I have applied each model to all and selected attributes. As it shown below in figure 9.

```
Logistic Regression
Accuracy -> all : 0.9348392701998263 vs selected : 0.9261511728931364
Precision -> all : 0.9282511210762332 vs selected : 0.9285714285714286

Desision Tree
Accuracy -> all : 0.9226759339704604 vs selected : 0.9157254561251086
Precision -> all : 0.9088888888888889 vs selected : 0.8964757709251101

Random Forest
Accuracy -> all : 0.9591659426585578 vs selected : 0.9609035621198957
Precision -> all : 0.9575892857142857 vs selected : 0.963963963963964

Gradient Boosting
Accuracy -> all : 0.9574283231972198 vs selected : 0.9530842745438749
Precision -> all : 0.9615384615384616 vs selected : 0.9548532731376975

SVC
Accuracy -> all : 0.7193744569939183 vs selected : 0.7185056472632494
Precision -> all : 0.7445255474452555 vs selected : 0.7435897435897436
```

Figure 9

The Graph of the models with all the attributes and the models with the selected attributes.

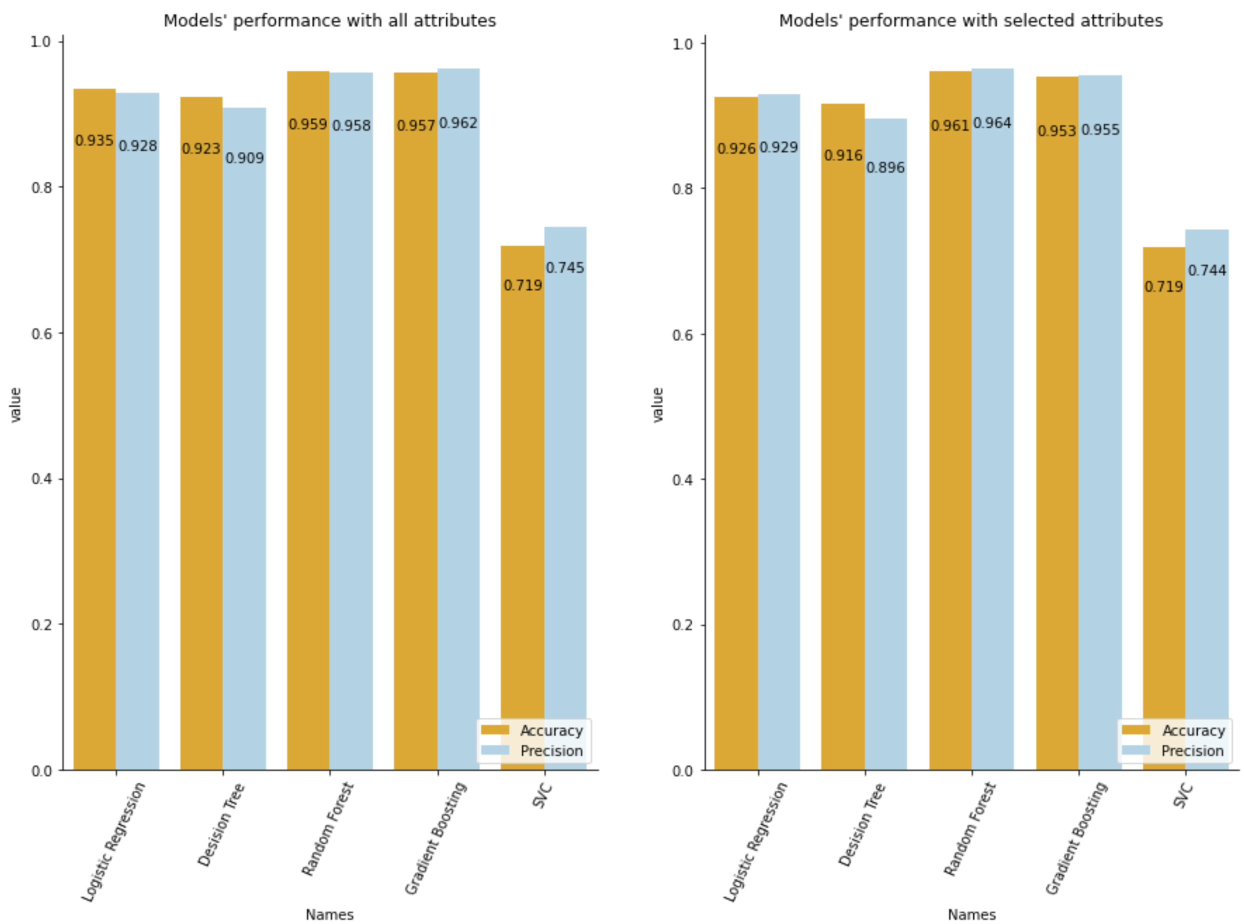


Figure 10

I have fitted 5 models on the training set with all attributes and the training set with the 41 selected attributes: Logistic Regression, Random Forest, Gradient Boosting and Support Vector Classifiers. The two best models are ensemble classifiers Random Forest and Gradient. Using the selected attributes also seems to be an interesting idea to use a lighter model with good performances. The result didn’t change that much compare to the model with all the attributes. Since the random forest model performed very well. I have implemented a confusion matrix of random forest and the result of the confusion matrix as shown below.

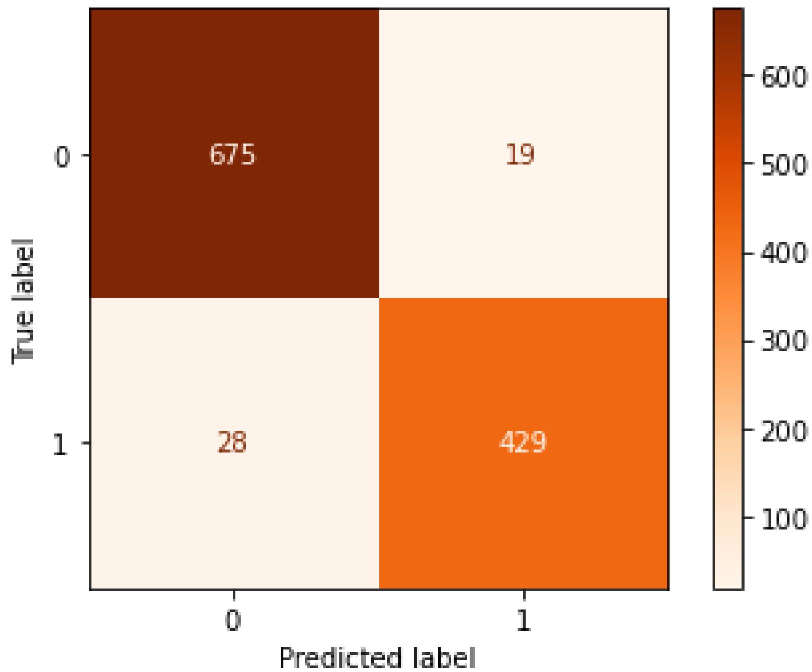


Figure 11

Deep Learning Model

I have used a Deep Learning model to predict safe emails or spam. I have used python liberals such as Keras, pandas, matplotib, and NumPy.

Load data & data Preprocessing

First, I converted the data to CSV and since the data is unbalanced, I balanced it in order to work with it to build our model. The total of the data is 4601 instances and 57 attributes. I have created a function to create two classes to train test split, 1500 indexes of class 0 and 1500 indexes class 1, for the training set so as to have a 70/30 training/test split.

```
0     1500
1     1500
Name: 57, dtype: int64
```

Figure 12

The rest of the observations will be used for testing Preprocessing the data which is 1601. The last step is to vectorize and normalize the data. I Created a validation set so randomly selected 600 data for validation, so as to have an 80/20 training/validation split, the total of the Validation set is 2400. Now we have a balanced dataset

Deep learning model and evaluation

I have built the model by using models.Sequential() function, I have added dense layers, relu, and sigmoid activation. As shown in figure 13.

```
model = models.Sequential()
model.add(layers.Dense(26, activation='relu', input_shape=(57,)))
model.add(layers.Dense(1, activation='sigmoid'))
```

Figure 13

Then complied the model. As shown in figure 14.

```
model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
```

Figure 14

I have implemented the model with 200 epochs and 200 batch size. As shown in figure 15.

```
Epoch 200/200
2400/2400 [==============================] - 0s 14us/step - loss: 0.0854 - accuracy: 0.9746 - val_loss: 0.1504 - val_
accuracy: 0.9417
```

Figure 15

The graph indicates that the training loss was 0.0854, and the validation loss was 0.1504. As shown in figure 16.

Figure 16

The graph indicates that the training accuracy was 0.9746, which was a really good result for the model, and validation accuracy was 0.94. As shown in figure 17.

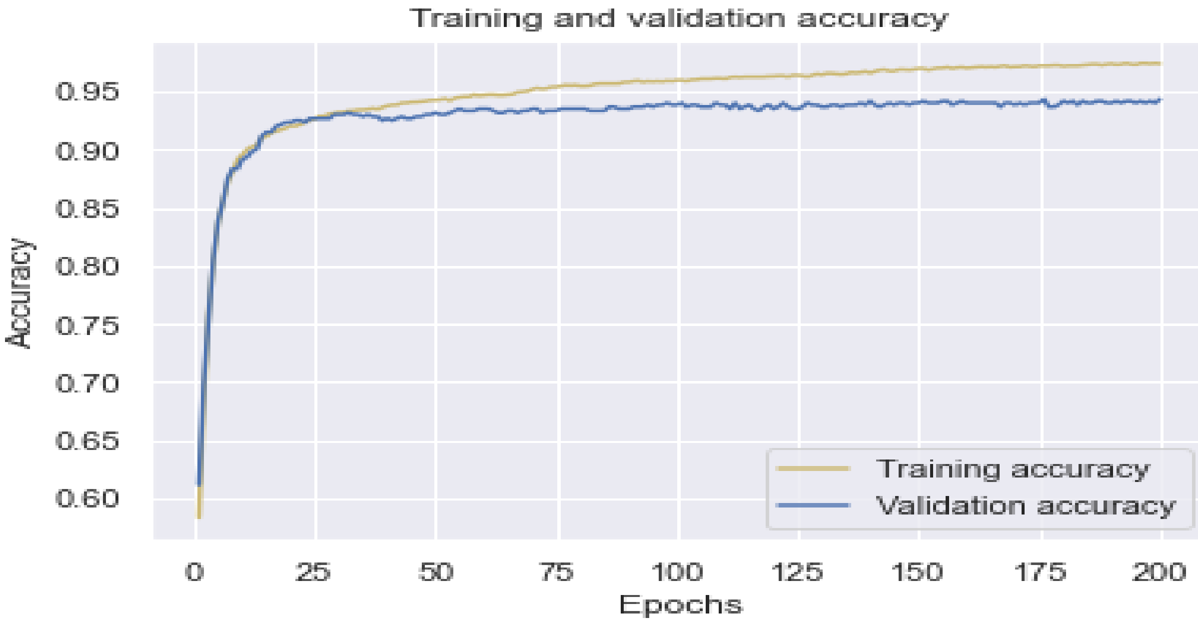


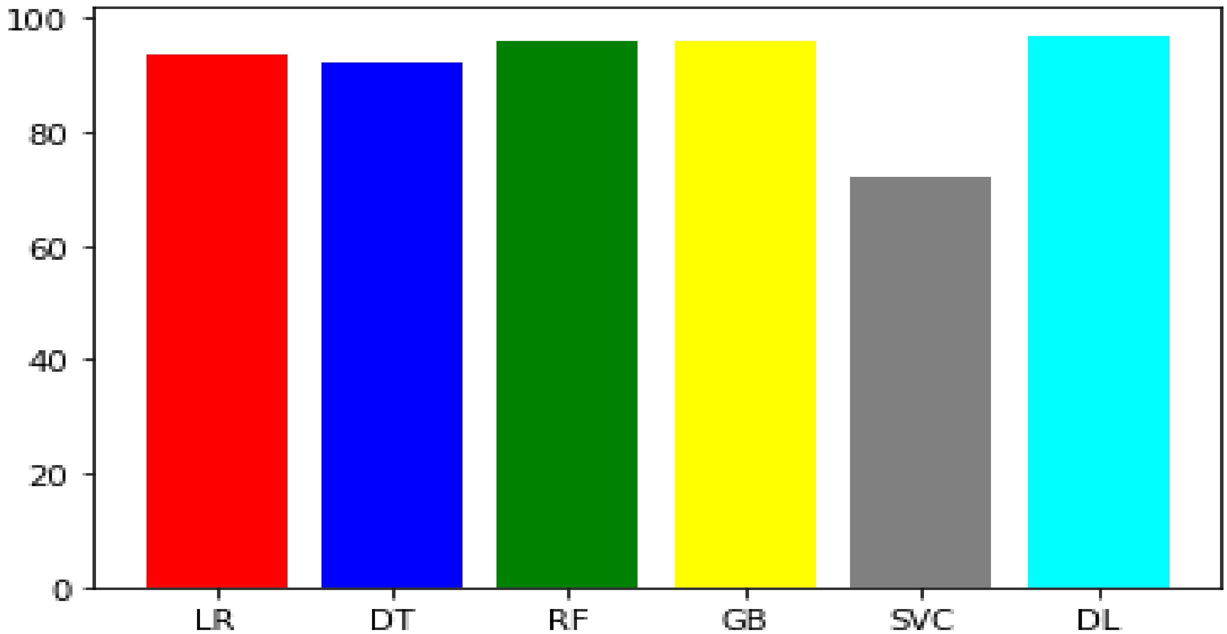
Figure 17

Evaluation Results

After implementing five machine learning algorithms, which are Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and SVC. The highest results based on the results for the machine learning algorithms were Random Forest with 96% and Gradient Boosting with 96%. The deep learning model achieved the highest accuracy with 97% which is higher than all the machine learning models. This indicates that the deep learning model is more efficient than the machine learning models. Despite that most of the models performed well except the svc model but the deep learning achieved the highest accuracy.

Machine learning algorithm	Accuracy
Logistic Regression	**0.934= 93%**
Decision Tree	**0.922=92%**
Random Forest	**0.959=96%**
Gradient Boosting	**0.957=96%**
SVC	**0.719=72%**

Deep learning Model	Accuracy
DL Neural Network	**0.97= 97%**



I have posted the coding part in GitHub[2] and the link in the reference section.

References

[1] Hopkins, M., Reeber, E., Forman, G., & Suermondt, J. (1999, July 1). UCI Machine Learning Repository: Spambase Data Set. Retrieved December 1, 2021, from https://archive.ics.uci.edu/ml/datasets/spambase.

[2] https://github.com/ahmedHamzah1997/Final-Project