Spam or Ham?

109502503 楊沛蓉
109502007 張原鳴
109502518 陳洛鈞

Motivation

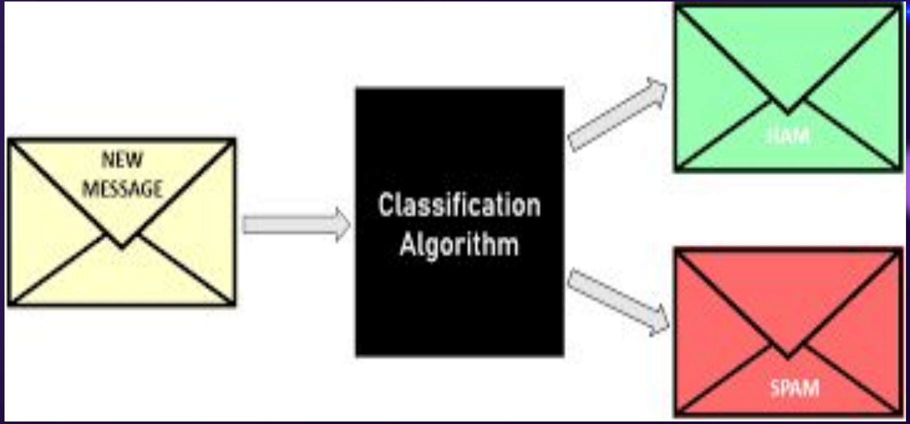
Out of curiosity, we want to understand how Gmail classifies various emails as spam.

Problem

- What type of mail is classified as spam?
- What words often appear in spam the most?

Goal

- Distinguish those contents of emails belonging to spam, and find the best type of classifier by the final accuracy rate.
- Moreover, find the top 50 English words that appear frequently from the results of classification by each classifier.



Research Outline

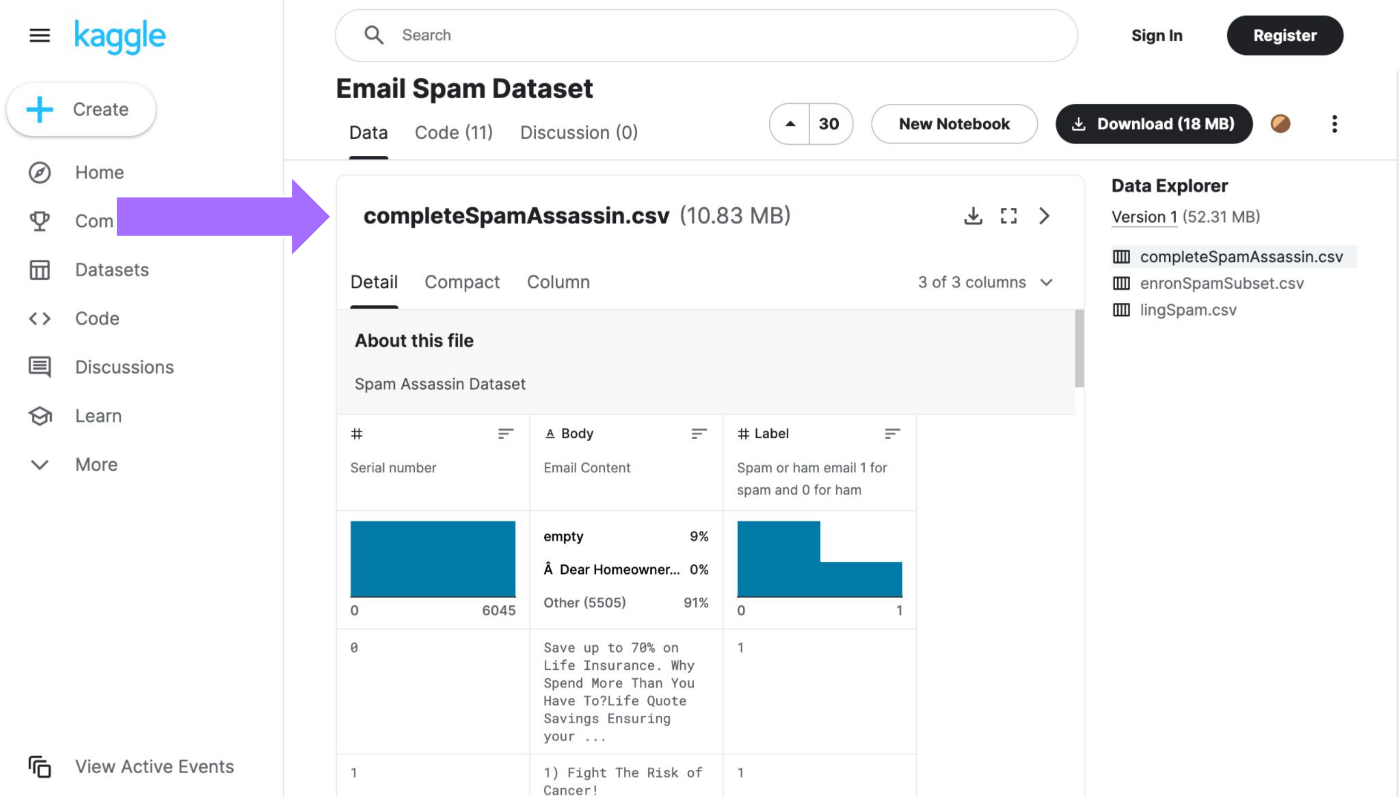
- Find a suitable dataset
- Preprocess the dataset
- Separate the dataset into training and testing data(7:3)
- Place those data into different models
- Execute text analysis on those predictions
- Show the results

Dataset

Kaggle — Email Spam Dataset

(https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset?select=completeSpamAssassin.csv)

include 6045 valid data



Preprocess

Delete the useless data in original dataset

```
##data preprocessing
data = pd.read_csv("completeSpamAssassin.csv")
data.drop("Unnamed: 0",inplace=True,axis=1)
data.dropna(inplace=True) #delete the data with NAN
#remove all link
no_link=[re.sub(r"http\S+",'',i) for i in data['Body']]
#remove character except for alphanumeric character
clean=[re.sub(r"[^a-z0-9A-Z]",' ',i) for i in no_link]
lower=[i.lower() for i in clean] #lower all texts
tokens=[nltk.word_tokenize(i) for i in lower] #texts to tokens
lemma=WordNetLemmatizer()
#make the words in the same form
lemmatized=[[lemma.lemmatize(w) for w in token] for token in tokens]
stopwords = nltk.corpus.stopwords.words("english")
#remove all the stopword
no_stopwords=[[w for w in text if w not in stopwords] for text in lemmatized]
```

Separate

The ratio of training and testing data
7 : 3

```
vectorizer=CountVectorizer(max_features=200)
X=vectorizer.fit_transform([' '.join(text) for text in no_stopwords]).toarray()
y=data['Label']
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1,test_size=0.3)
```

Model

1. Naïve Bayes
2. Logistic Regression
3. Decision Tree
4. Support Vector Machine(SVM)

Model – Naïve Bayes

```
nb_model=GaussianNB()
nb_model.fit(X_train,y_train)
nb_model.score(X_test,y_test)
y_pred=nb_model.predict(X_test)
```

Model – Logistic Regression

```
##linear regression
logr=linear_model.LogisticRegression()
logr.fit(X_train,y_train)
y_pred=logr.predict(X_test)
```

Model – Decision Tree

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Model – Support Vector Machine(SVM)

```
#svm
model = svm.SVC(probability=True)
model.fit(X_train,y_train)
#features_test = vectorizer.transform(X_test)
model.score(X_test,y_test)
y_pred=model.predict(X_test)
```

Analysis & Result

- Do the text analysis on the results of classifiers by using CountVectorizer, calculating the number of every word appearing in all the spam predicted
- We look over the final results by using classification report, ROC curve, confusion matrix in scikit-learn

Method – 1

- Delete all the link words and special symbols
- Transform all the English letters into lower-case
- Transform all the English words into their original words
  - ex: organizes, organizing, organized ⇒ organize
- Filter out all the stop words that have no real meaning
  - ex: a, the, on

Method – 2

- Separate the dataset into training and testing data
    - The ratio of training and testing data = **7 : 3**
- Put the training and testing data into different models
- Get the data index categorized as spam by the prediction of testing data

Method – 3

- Set a counter that aims to calculate the top 2000 English words that appear frequently
- Calculate the number of every word appearing in all the predicted spam by using this counter and the indexes of spam
- Vectorize contents of spam and transfer them into the structure of Pandas in Python
- Delete those numbers in front of each line of data

Method – 4

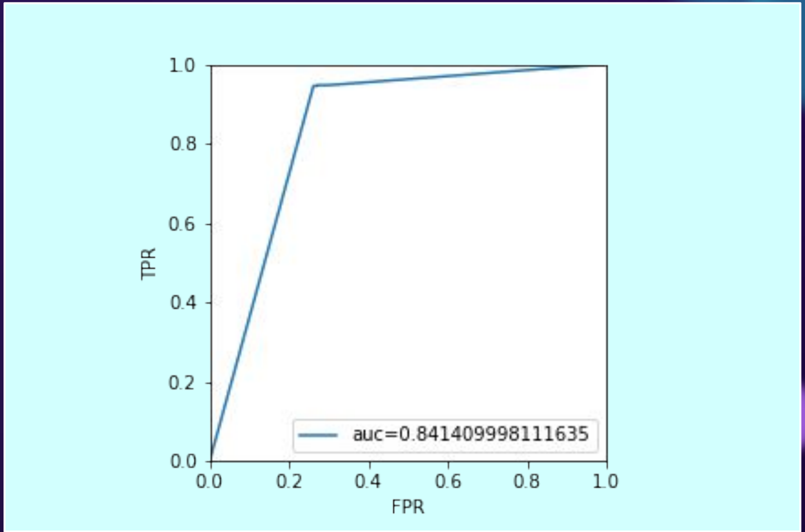
- Print the results based on testing data run in different models
    - classification results
    - ROC curve
    - confusion matrix
    - top 50 key words that appear the most in spam predicted

Naive Bayes

Classification report

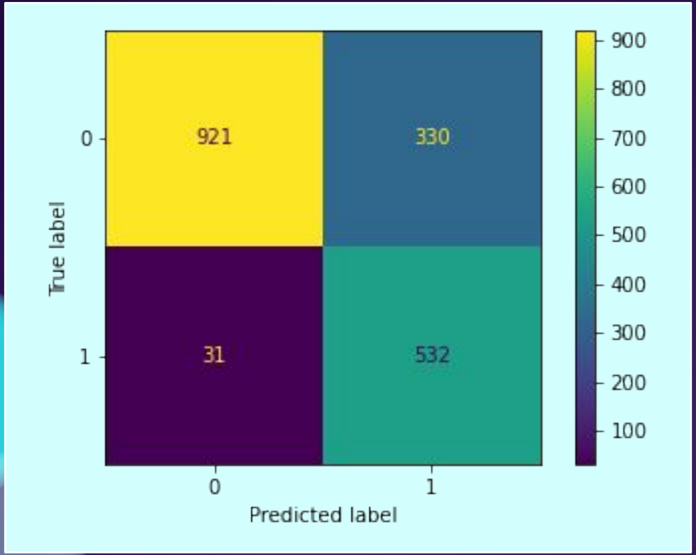
```
Naive Bayes's accuracy:  0.8009922822491731
              precision    recall  f1-score   support

         Ham       0.97      0.74      0.84      1251
        Spam       0.62      0.94      0.75       563

    accuracy                           0.80      1814
   macro avg       0.79      0.84      0.79      1814
weighted avg       0.86      0.80      0.81      1814
```

ROC curve

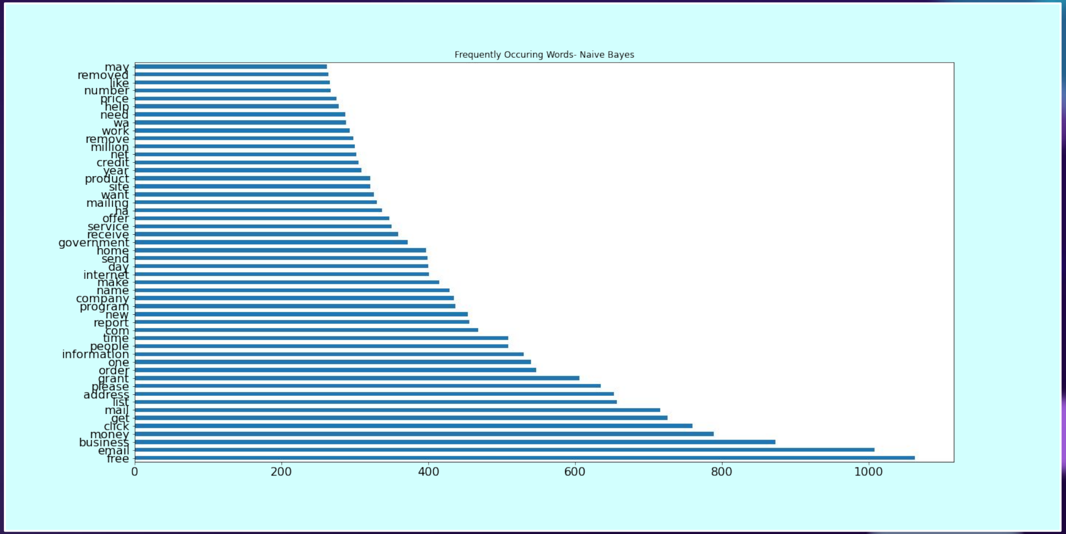


Naive Bayes

Confusion Matrix



Top 50 key words occur in spam

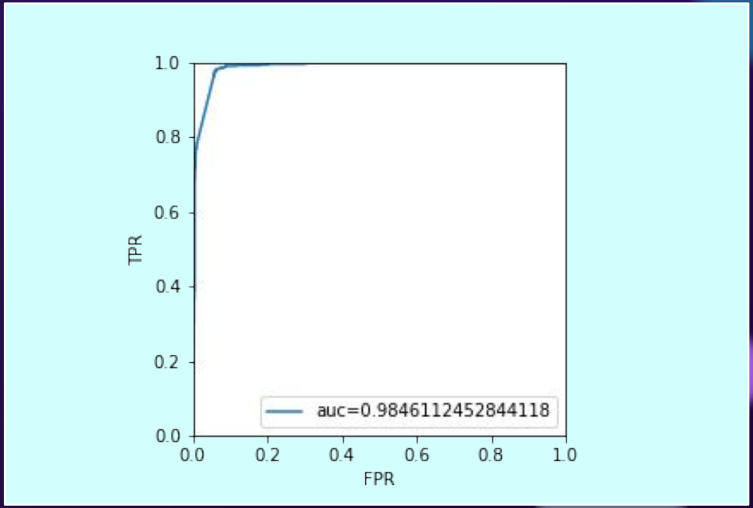


Logistic Regression

Classification report

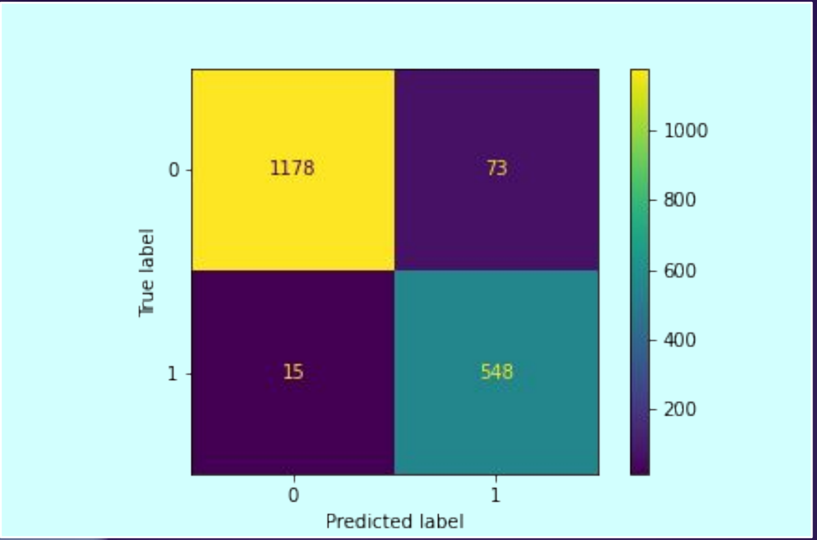
```
Logistic Regression's accuracy:  0.9514884233737596
              precision    recall  f1-score   support

         Ham       0.99      0.94      0.96      1251
        Spam       0.88      0.97      0.93       563

    accuracy                           0.95      1814
   macro avg       0.93      0.96      0.94      1814
weighted avg       0.95      0.95      0.95      1814
```

ROC curve

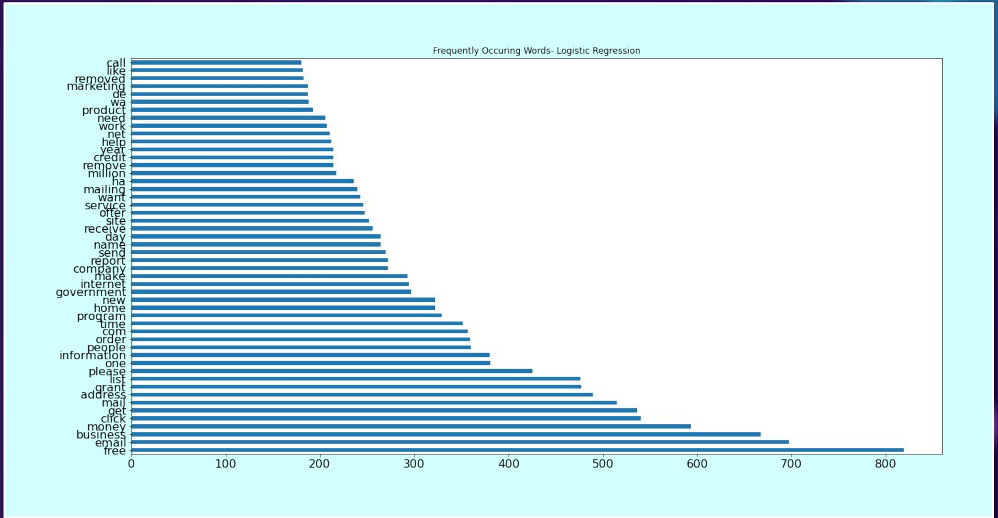


Logistic Regression

Confusion Matrix



Top 50 key words occur in spam



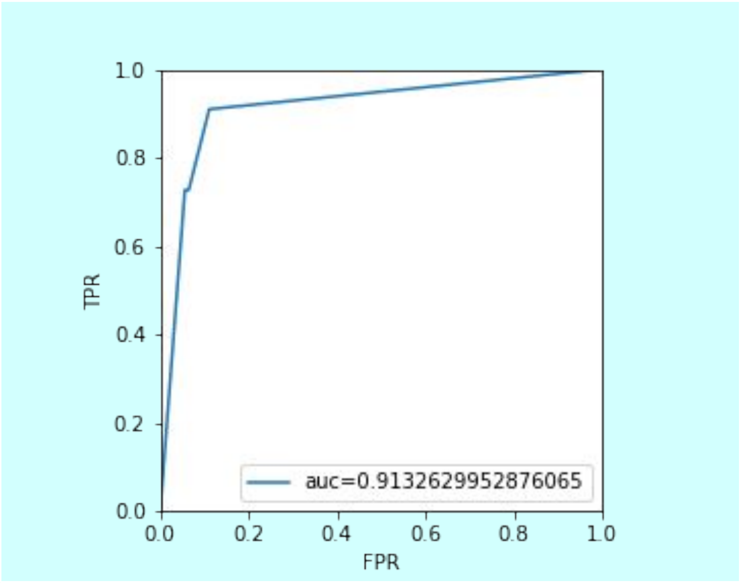
Decision Tree

Classification report

Decision Tree's accuracy: 0.8969128996692393

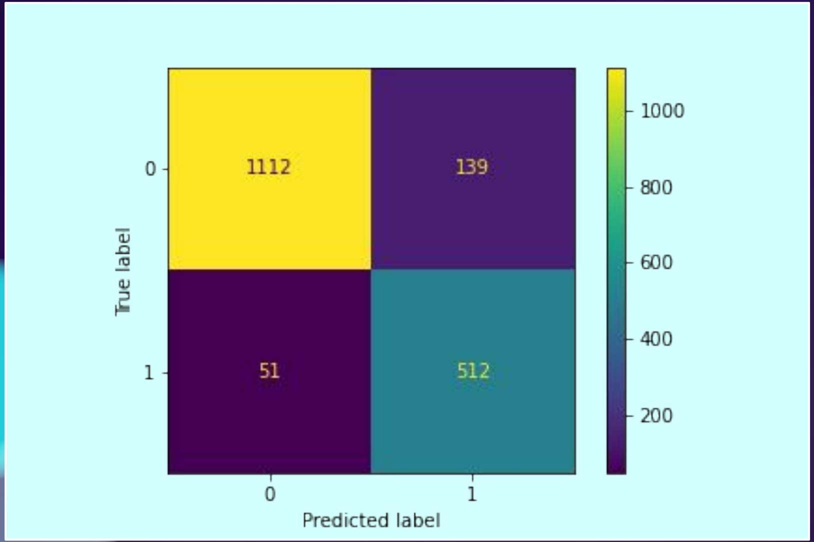
	precision	recall	f1-score	support
Ham	0.96	0.89	0.92	1251
Spam	0.79	0.91	0.85	563
accuracy			0.90	1814
macro avg	0.87	0.90	0.88	1814
weighted avg	0.91	0.90	0.90	1814

ROC curve

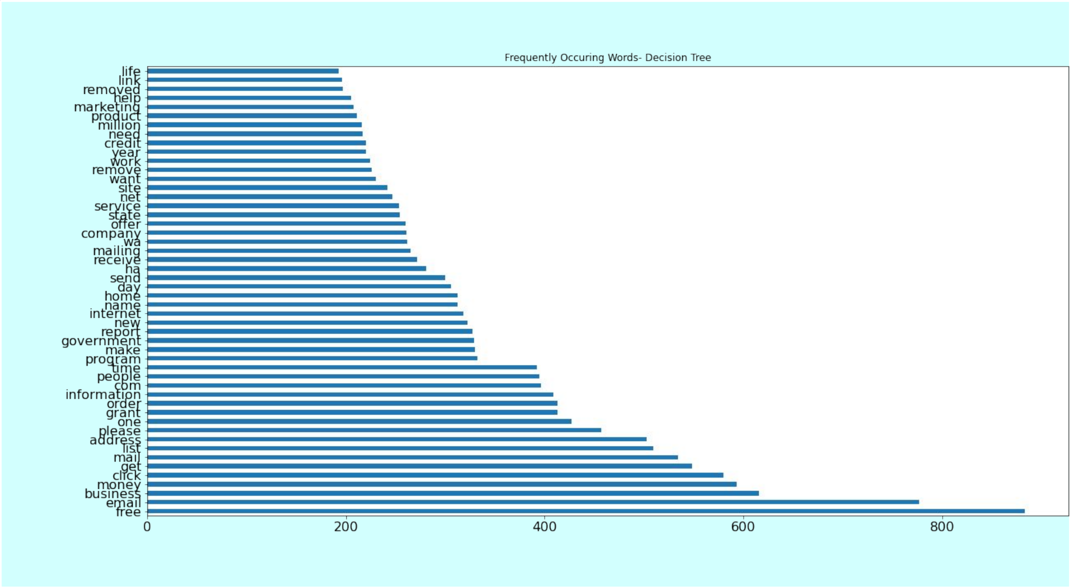


Decision Tree

Confusion Matrix



Top 50 key words occur in spam

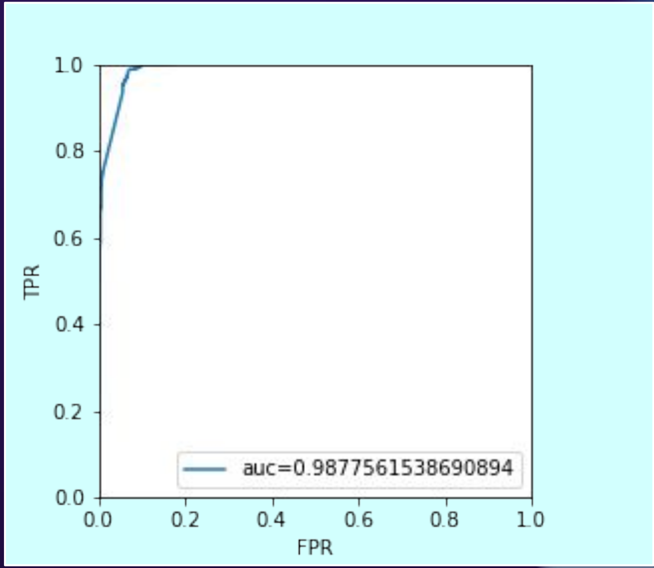


Support Vector Machine(SVM)

Classification report

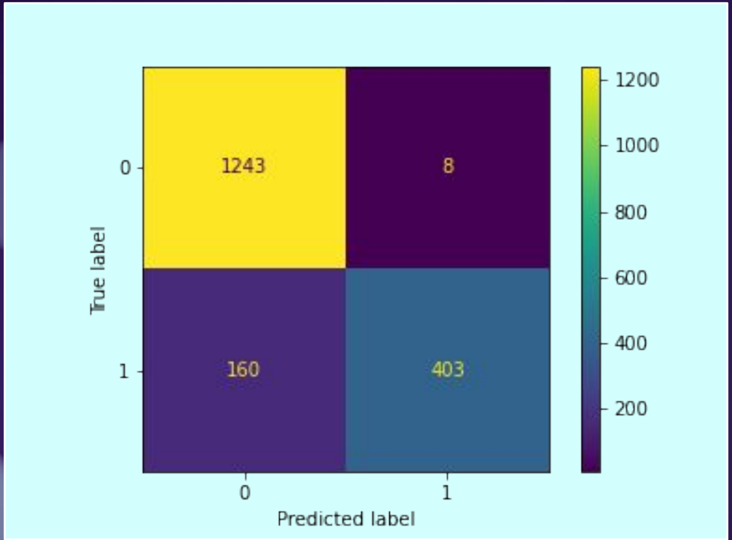
```
Support Vector Machine's accuracy:  0.9073869900771775
              precision    recall  f1-score   support

         Ham       0.89      0.99      0.94      1251
        Spam       0.98      0.72      0.83       563

    accuracy                           0.91      1814
   macro avg       0.93      0.85      0.88      1814
weighted avg       0.92      0.91      0.90      1814
```

ROC curve

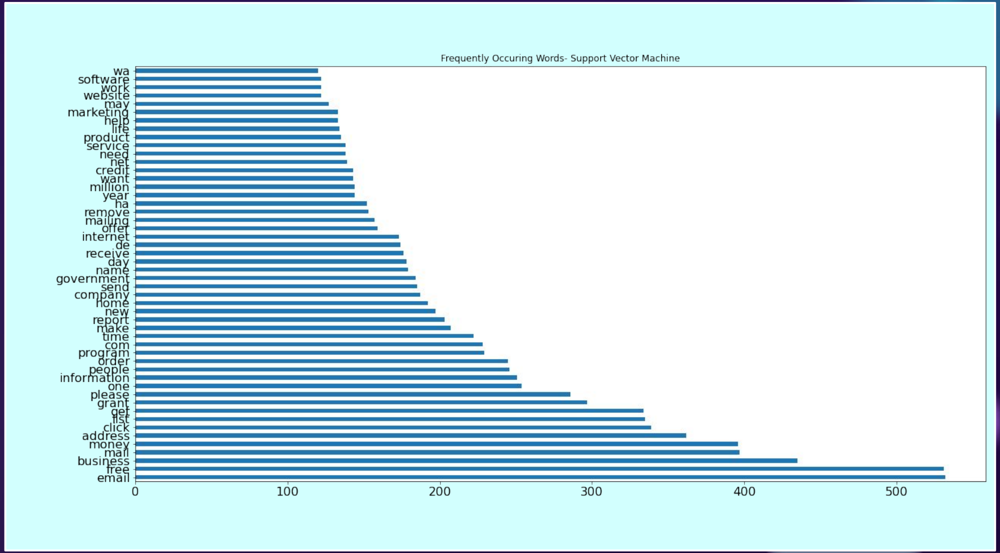


Support Vector Machine(SVM)

Confusion Matrix



Top 50 key words occur in spam



Result

Acuracy rate: Logistic Regression > SVM > Decision Tree > Naïve Bayes

Area under the ROC curve: SVM > Logistic Regression > Decision Tree > Naïve Bayes

Precision rate: SVM > Logistic Regression > Decision Tree > Naïve Bayes

Recall rate: Logistic Regression > Naïve Byes > Decision Tree > SVM

True positives(TP): Logistic Regression > Naïve Bayes > Decision Tree > SVM

True negatives(TN): SVM > Logistic Regression > Decision Tree > Naïve Bayes

Ranking considering the above comparison criterion:
Logistic Regression (21) > SVM (14) > Decision Tree (12)>Naïve Bayes (10)

The top 5 word in spam of each model

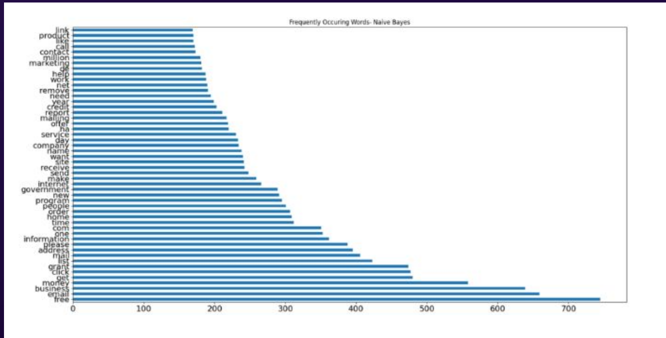
Naïve Bayes: free、email、business、money、get

Logistic Regression: free、email、business、money、click

Decision Tree: free、email、business、money、click

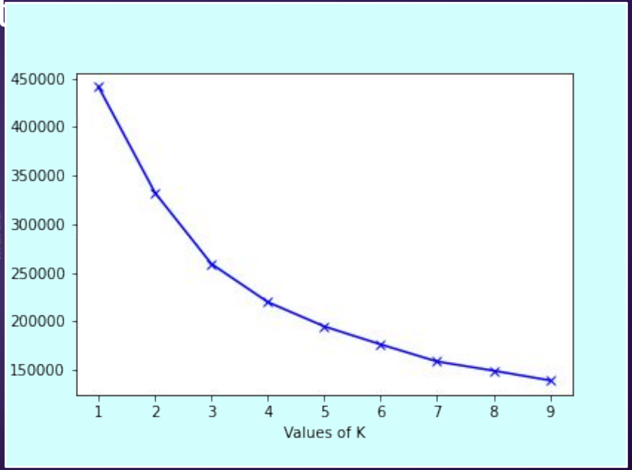
SVM: email、free、business、money、mail

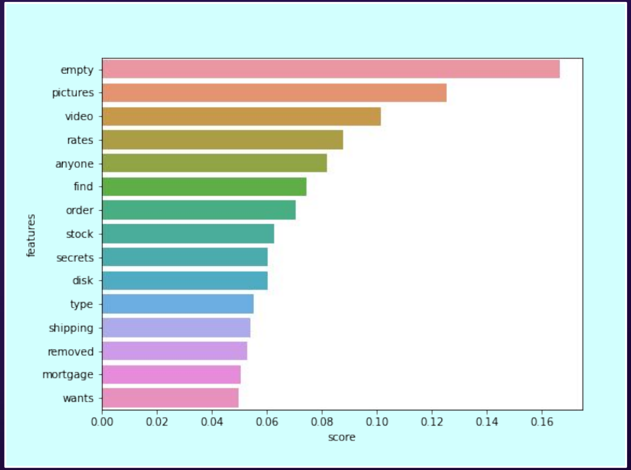
The top common words:
free, money, email, business, click



Extension

We want to find what kind of vocabularies in spam emails will cluster together. Thus, we applied the elbow method to find the optimal k of the k-means algorithm. Then fit the data to k-means algorithm to get the clustering center. Through the clustering center, we are able to find the top 20 words that have the highest score. Then print the bar chart about k clusters, showing the most 15 representative words in each clust





Conclusion

In terms of comprehensive considerations, choosing Logistic Regression will have the best results. But because Logistic Regression doesn’t perform best in all evaluation standards, so if you only want to focus on a certain measurement standard, other classifiers may have better performance results.

Through this final project, we can learn what words are usually contained in the content of spam. In future, if we receive a letter that is not classified as spam, we can also judge by ourselves to avoid being scammed.