**Development of Spam Filter**

**Case Study**

This project goal is to build a machine learning model to filter and classify messages as spam or not spam, developed by using the CRISP-DM (business understanding, data understanding, data preparation, modelling, evaluation, and deployment) process, which includes stages of a proposal about how to build the project through the folder structure of a Git repository. The project objectives are to organize the project structure, analyze the provided data set, pre-process the data, vectorize using Tf-Idf, use L1 or L2 regularization, and cross validation to avoid overfitting, and then build the machine learning model. This study compares several algorithms, such as Decision Tree, Logistic Regression, Naive Bayes, Support Vector Machine, to determine which one is the best model for classifying and filtering messages as spam or not. Then the error analysis will be performed to understand the weaknesses and limitations of the model and raise confidence level from business partner about the model. The results of the analysis will be presented with suggestions for future steps in this area, and the graphical user interface will be proposed along with a diagram concept plan for the model's integration into daily work.

Cloud link:
https://hennypurwadi-spam-classifier-spam-classifier-ubc344.streamlit.app/

Github code:
https://github.com/hennypurwadi/spam_classifier

https://github.com/hennypurwadi/spam_classifier/blob/main/DLBDSME01_Spam_Classifier.ipynb

**Table of Contents**

**List of Figures**



**List of Tables**



**List of Abbreviations**

| | |
|---|---|
| CRISP-DM | Cross Industry Standard Process for Data Mining |
| GUI | Graphic User Interface |
| CV | Count Vectorizer |
| ML | Machine learning |
| NB | Naiive Bayes |
| Tf-Idf | Term Frequency-Inverse Document Frequency |
| SVM | Support Vector Machine |

# 1 Introduction

## 1.1 Business Understanding

Because the client needs a classification model to identify spam from messages, this project was created for data analysis to answer business problems by following CRISP-DM (Cross Industry Standard Process for Data Mining). The stages are Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment.

## 1.2 Project Aim and Objective

The purpose of this project is to build a Machine Learning model to classify messages as spam or not spam.

**Project Objective**

1 To create a proposal about how to build the project through folder structure of a Git repository.

2 To do pre-processing data, such as cleaning and filtering dataset, remove irrelevant data, and prepare the data to be ready for the next stage.

3 To vectorize datasets with Count Vectorizer and Tf-Idf, to turn texts into numbers, to make them understandable by the computer.

4 To use regularization to help to avoid overfitting, and to use Cross validation by folding the data into several smaller folds and selecting subsets as training and validation sets. The model can be evaluated multiple times on smaller portions of data, to give better performance on unseen data.

5 To compare performances of several machine learning algorithms and select the best performance model.

6 To classify a cleaned dataset as different label, either spam or ham (not spam), using the selected best performing model, to predict labels in new data.

7 To present and discuss the results of the analysis, to make business partners understand the weaknesses of the approach.

8 To propose daily work model integration through a diagram and GUI (Graphical User Interface) proposal.

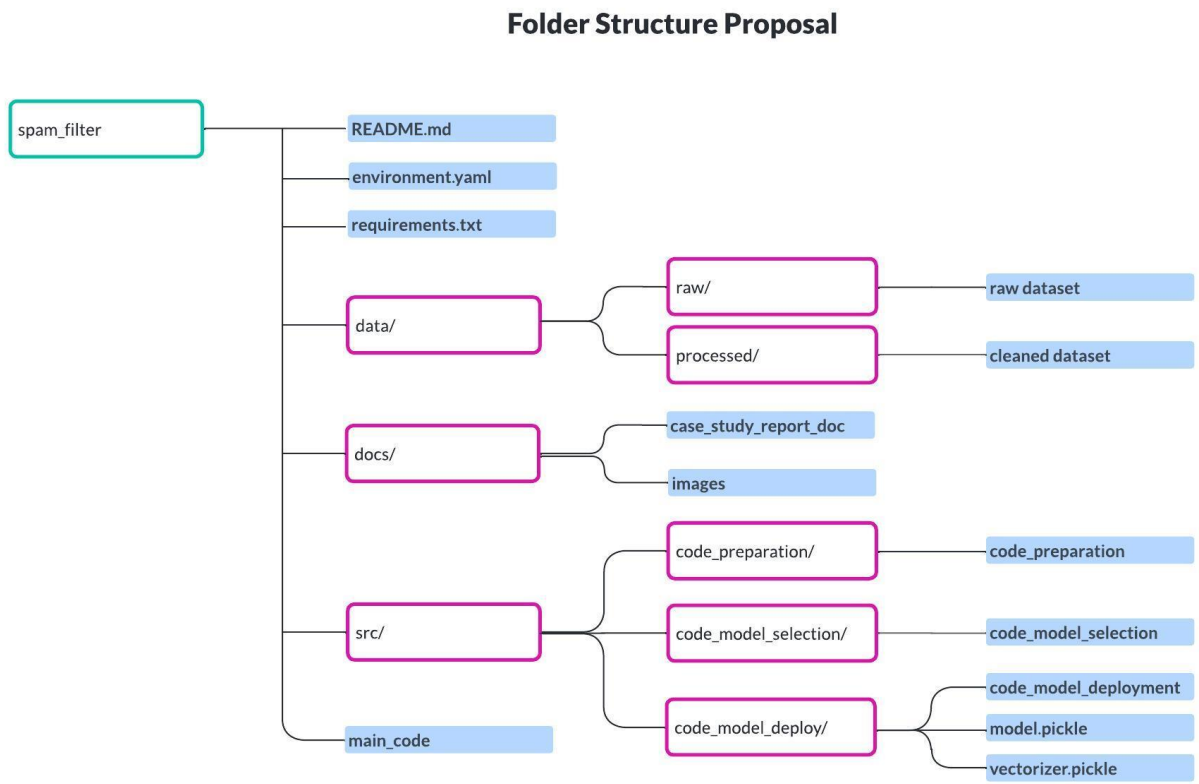
9 To provide a summary and conclusion of the study and suggestions for future steps in this area.

## 2 Preparation

### 2.1 Folder Structure

Below is the proposal of the folder structure of a Git repository for the project.

**Figure 1. Folder Structure Proposal**




**Source: Own Representation**

The folder structure is used to help business partners to understand the project and makes the project updatable and repeatable.

### 2.1 Terminology and Definitions

Terminology and definitions related to machine learning algorithms for classification, such as Logistic Regression, Decision Tree, and Multinomial Naive Bayes.

#### Logistic Regression

Logistic regression is an algorithm for binary classification tasks, which compares the two sets of features to one another, to find the ideal parameters the coefficients (weight) that provide the best fit for the data. (Raschka, 2015)

Formula: **P(y=1|x) = 1 / (1 + e^ (-w^T x - b))**

p(y = 1|x) is the conditional probability that a sample with x belongs to class 1. The inverse logit function predicts the chance that a sample belongs to a class.

w is the weight vector, y is the target class, x is the feature, b is the bias term.

**Decision Tree**

Decision Tree is algorithm for both binary and multi-class classification tasks, with acyclic graph uses a tree-like model of choices used to make decisions. It has branches with nodes of the graph. (Burkov, 2019).

**Multinomial Naïve Bayes**

Multinomial Naive Bayes is a probabilistic method based on Bayes' theorem. It's effective when features are independent to each others. (Pati, & Pradhan, 2020, p. 7).

Formula: **P(Q|R) = P(Q)*P(R|Q) / P(R),** where Q and R are events.

P(Q|R) is a conditional probability of Q event occurring given that A is true,

P(R|Q) is a conditional probability of R event occurring given that B is true,

P(Q) and P(R) are the probabilities observing events Q & R

**Tokenization and Stopwords.**

Text tokenization breaks sentences into smaller units called "tokens", such as words or phrases. Text tokenization can be done by dividing the space character, using regular expressions, or using natural language processing methods, or using Stopwords. (Raschka. (2015, p. 269).

**Stemming** is reducing words to their stem/ root word is known as stemming. This equalizes related terms for the sake of comparison or sharing. When tokenizing sentences, the process of stemming aids in their analysis. (Mueller & Massaron, 2021, p. 355).

**Lemmatize** is to acquire grammatically accurate versions of individual words, or lemmas. Lemma is computationally more complex and costly than stemming and have minimal influence on text classification performance. (Raschka. (2015, p. 271).

**Tf-idf vectorizer**

TF-IDF stands for "Term Frequency-Inverse Document Frequency". It is used to find the importance of a word to a document inside a collection or corpus of documents. The "term frequency calculates the number of times a particular word appears in a document. The "inverse document frequency" measures how common or rare a word is in the entire corpus of documents. (Mueller & Massaron, 2021, p. 353)

**Confusion Matrix**

The confusion matrix is a table that summarizes the classification to predict different classes. One axis of the confusion matrix represents the label predicted by the model, while the other axis represents the actual label. (Burkov, A., 2019, p. 65)

Based on confusion matrix output, this research used four effective measures:

True Positive (TP) = Truly predicted as Positive. True Negative (TN) = Truly predicted as Negative. False Positive (FP) = Falsely predicted as Positive. False Negative (FN) = Falsely predicted as Negative.

**Precision** is the proportion of **correctly positive predictions** divided by the **total** number of **positive predictions**. Precision = TP/(TP+FP)

**Recall** is proportion of **correctly positive predictions** divided by the **total** number **of actual positive.** Recall(R) = TP/(TP+FN)

**Accuracy** is proportion of **correct predictions** divided by the **total examples** (Burkov, A., 2019, p. 67). Accuracy(A) = (TP+TN) / (TP + TN + FP + FN)

**F1-Score** is balancing precision and recall. The worst value is 0, and the best value is 1.

F1-score = 2 * (Precision * Recall) / (Precision + Recall)
To analyze performance of several machine learning models, will need to compare their accuracy, precision, recall, and f1-score.

**Normalized Confusion Matrix** is confusion matrix which normalized become numbers between 0 - 1 to simplify it and make it become easier to interpret.

## 3 Model Selection

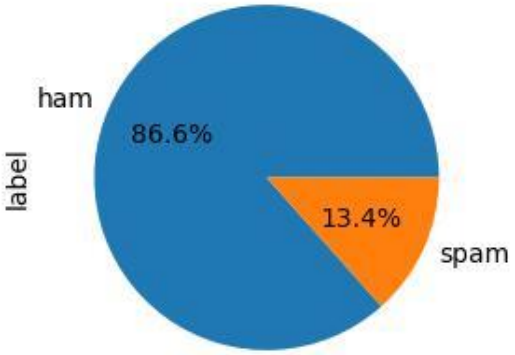
### 3.1 Data Understanding

Based on data visualization presented below, there is an imbalance distribution of data between spam and ham, with 86.6% of data labelled as ham, and only 13.4% of them labelled as spam. If the dataset is not balanced, it can have significant impact on the performance of machine learning algorithms.

**Figure 2. Data visualization**

```
df = df[['text','label']]
df.head()
```

| | text | label |
|---|---|---|
| 0 | Go until jurong point, crazy.. Available only ... | ham |
| 1 | Ok lar... Joking wif u oni... | ham |
| 2 | Free entry in 2 a wkly comp to win FA Cup fina... | spam |
| 3 | U dun say so early hor... U c already then say... | ham |
| 4 | Nah I don't think he goes to usf, he lives aro... | ham |




**Source: Own Representation**

**Figure 3. Words in spam Categories**


**Source: Own Representation**

Using the WordCloud Python library, we can investigate the most common words that appear in spam categories. Words such as "free", "won", "win", "awarded", "cash" "prize", "phone", "urgent", "offer", "now", and "claim" seem often appear.

In contrast, these words do not appear in ham categories.

**Figure 4. Words in ham Categories**


**Source: Own Representation**

### 3.2 Data Preparation

Data preparation process is including cleaning, lemmatize, stemming, tokenize, imputation, vectorize, hyperparameter tuning, and feature selection.

**Cleaning**

```
def clean_text(text):
    import re
    from string import punctuation
    text=re.sub(r'(http|ftp|https):\/\/([\w\-_]+(?:(?:\.[\w\-_]+)+))([\w\-\.,@?^=%&:/~\+#]*[\w\-\@?^=%&/~\+#])?',
            ' ', text)
    text=re.sub(r'['+punctuation+']',' ',text)
    text=re.sub(r'#(\w+)',' ',text)
    text=re.sub(r'@(\w+)',' ',text)
    text = text.lower() # Convert  to lowercase

    token=RegexpTokenizer(r'\w+')
    tokens = token.tokenize(text)

    lemmatizer = WordNetLemmatizer()
    stems = [lemmatizer.lemmatize(t) for t in tokens]
    stemmer = PorterStemmer()
    stems = [stemmer.stem(t) for t in stems]

    return ' '.join(stems)

def tokenize(text):
    token=RegexpTokenizer(r'\w+')
    tokens = token.tokenize(text)

    return tokens
```

For this dataset, imputation is not necessary, as there are no null rows.

```
df.shape
```

```
(5574, 2)
```

```
df['text'].isna().sum()
```

```
0
```

```
df['label'].isna().sum()
```

```
0
```

### 3.3 Vectorization and Regularization

Vectorization is a way to turn words into numbers to make computers understand them. One approach to do this is by using TF-IDF which assigns weight to each word in a text.

**Vectorizer tf-idf**

```
cv=TfidfVectorizer(lowercase=True,preprocessor=clean_text,stop_words='english',
                   ngram_range=(1,3),tokenizer=tokenize)

text_counts=cv.fit_transform(df['text'].values.astype('U'))
```

Adding L1/ Lasso regularization is good for feature selection by adding penalty to cost function. Adding L2/ Ridge regularization is also useful to prevent overfitting.Besides, using Cross validation is useful to improve model performance by divided datasets into multiple subset.

## 4 Results and Findings

### 4.1 Comparison of Model Performance and Evaluation

Example of model with high accuracy but low precision and recall can be shown in this performance of Logistic Regression.

**Table 1. Machine Learning Model Performance**

| ML Algorithms | accuracy | precision | recall | F1-score |
|---|---|---|---|---|
| Logistic Regression | 0.88 | 0.00 | 0.00 | 0.00 |
| Bernoulli NB | 0.93 | 0.98 | 0.41 | 0.58 |
| Gaussian NB | 0.93 | 0.67 | 0.89 | 0.76 |
| Linear SVC | 0.96 | 0.99 | 0.73 | 0.84 |
| SVM | 0.98 | 0.98 | 0.82 | 0.89 |
| Decision Tree | 0.96 | 0.83 | 0.85 | 0.84 |
| Multinomial NB | 0.99 | 0.98 | 0.93 | 0.96 |

**Source: own representation**

Logistic Regression's performance result show how struggle it is to classify spam. It predicts all messages as ham.

```
Enter your message:You won lucky prize $20,000
['ham']
```

```
1 performance_evaluation()
```

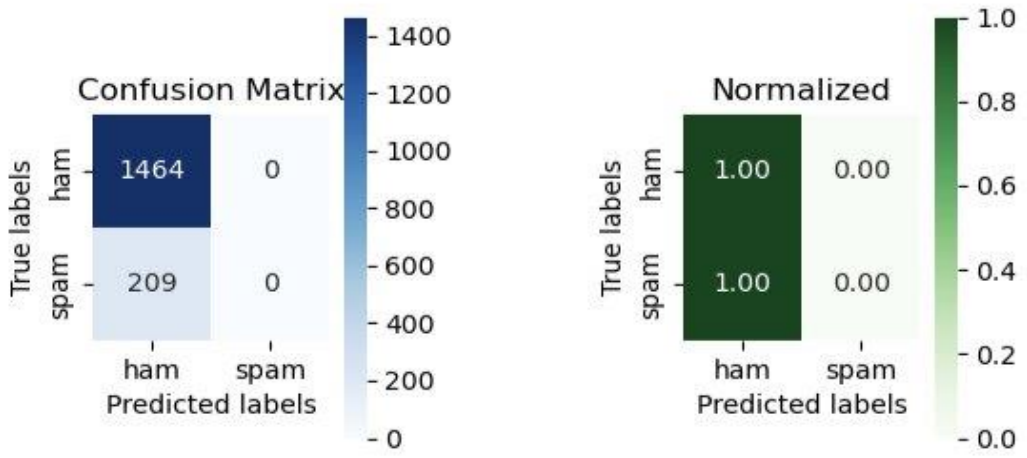
```
              precision    recall  f1-score   support

         ham       0.88      1.00      0.93      1464
        spam       0.00      0.00      0.00       209

    accuracy                           0.88      1673
   macro avg       0.44      0.50      0.47      1673
weighted avg       0.77      0.88      0.82      1673

[[1464    0]
 [ 209    0]]

[[1. 0.]
 [1. 0.]]
```




**The model is very strong in identifying ham, but very weak in identifying spam.**

**Recall = 0.00 for spam means: only can identify 0% of actual spam.**

**f1-score = 0.00 for spam means: this model's ability to detect spam is only 0%.**

**Source: own representation**

Among various classifiers in this project, Multinomial Naïve bayes has displayed very good performance, with accuracy of 99%, precision of 98% , 93% recall, and 96% f1-score. Therefore, this top performer model will be used to classify new data.

Multinomial NB performance result:

```
Enter your message:You won lucky prize $900000
['spam']
```

```
1 performance_evaluation()
```

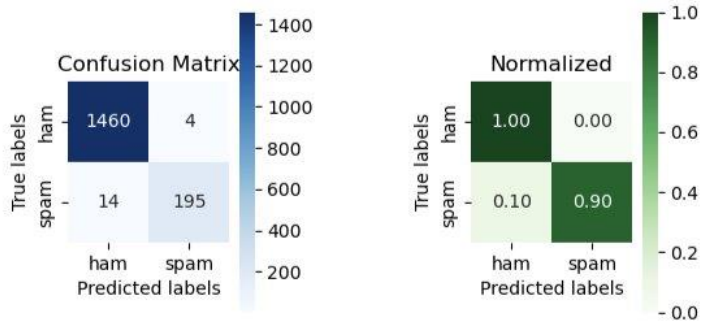
```
              precision    recall  f1-score   support

         ham       0.99      1.00      0.99      1464
        spam       0.98      0.93      0.96       209

    accuracy                           0.99      1673
   macro avg       0.99      0.97      0.97      1673
weighted avg       0.99      0.99      0.99      1673

[[1460    4]
 [  14  195]]

[[1.  0. ]
 [0.1 0.9]]
```







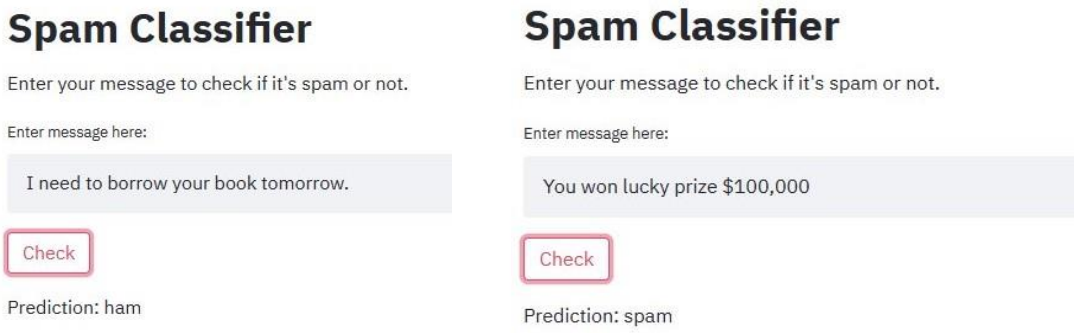
**This is the best model so far with highest precision, recall, f1-score, accuracy for both ham and spam.**

**The lowest number is 93% recall for spam**

### 4.2 Deployment

Multinomial naiive bayes as the best model, has been saved as both model.pkl and vectorizer.pkl. It has been loaded and deployed to cloud.

**Figure 6. Deployment into the cloud**



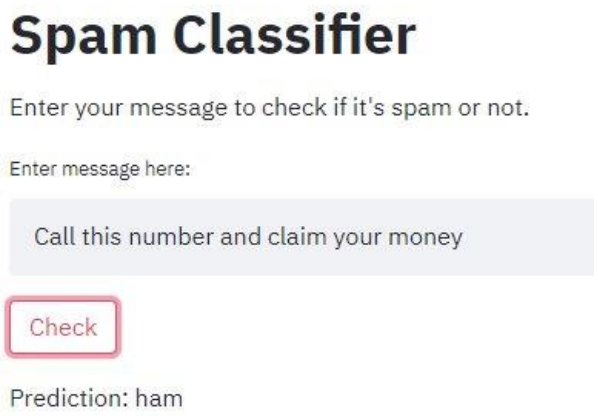

**Source: Own Representation**

## 5 Discussion

### 5.1 Error Analysis

Although Multinomial Naïve Bayes performed the best among other algorithms in training and testing data, it still shows several errors in classifying new data.

**Figure 7. Wrong prediction on new data**




**Source: Own Representation**

**This study has several limitations, such as:**

1 Too small dataset size, which has led to inadequate training.

2 Imbalance amount two categories, lack of training for the minor category, compared to the majority.

In datasets with severely class-imbalanced classifiers, the classifier will always “predict” the most common class without performing any feature analysis and will have a high degree of accuracy, but not the correct one.

Using simpler metrics like accuracy score only without comparing to other metrics can be misleading.

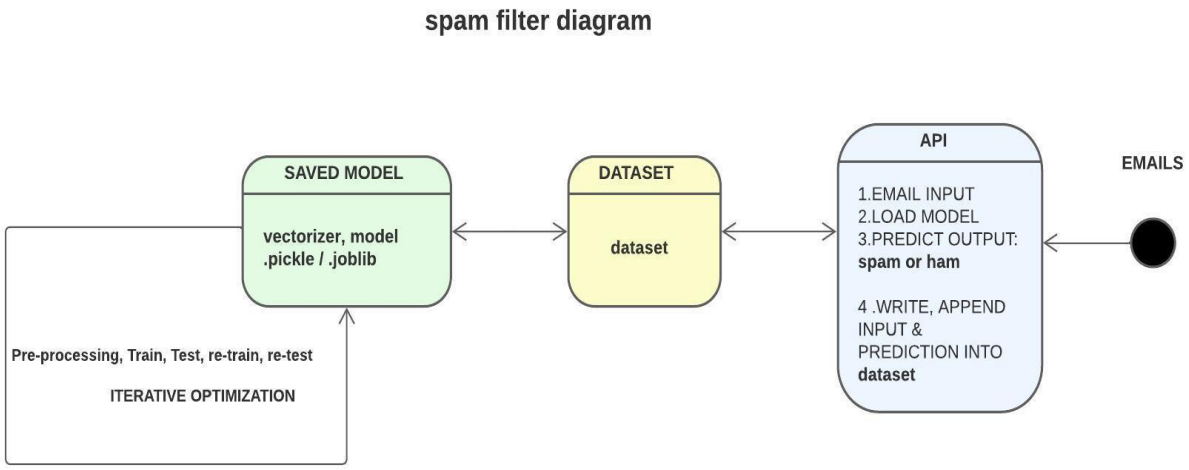
For example, in Logistic Regression, the model may show high accuracy just by predicting the majority class (ham), although it failed to identify minority class (ham). This can result in a high rate of false negative, where spam emails incorrectly classified as ham.

### 5.2 Recommendations for Future

Resampling technique can be used to highly imbalanced datasets. Under-sampling will remove samples from the majority class, while over-sampling will add more examples for the minority class. (10 Techniques to deal with Imbalanced Classes in Machine Learning. 2020).

As final stage of the project, a way to incorporate the model into the service team's daily operations suggested, through spam filter diagram below.

**Figure 7. Spam Filter Diagram**




**Own Representation**

**Spam email classification machine learning model requires several steps:**

1 First, the model is trained with labelled dataset. Then, save the trained model.

2 Then the stored model predicts spam or not spam when user enters new email.

3 The input email and prediction are added to dataset, to increase model accuracy over time.

4 The model keeps re-trained and re-tested with new data. (Iterative optimization).

5 The machine learning lifecycle on dataset iteration keep looping to maintain accuracy.

6 The model will be incorporated into the service team's workflow to simplify email classification.

## 6 Conclusion

A good spam classifier created with machine Learning algorithm can increase service team's productivity and effectiveness, because it can simplify and make filtering easier.

Machine learning algorithms perform optimally when the number of samples in each class is about the same. When the data set is imbalanced, a high accuracy rate can be achieved by predicting the majority class, but this will lead to a failure to recognize the minority class, which is often the main objective of creating the model in the first place.
(10 Techniques to deal with Imbalanced Classes in Machine Learning. 2020).

**References**


Techniques to deal with Imbalanced Classes in Machine Learning. (2020, July 23). https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/

Burkov, A. (2019, January 1). The Hundred-Page Machine Learning Book.

Burkov, A. (2020, September 8). Machine Learning Engineering.

Chauhan, V. K., Dahiya, K., & Sharma, A. (2018, January 16). Problem formulations and solvers in linear SVM: a review - Artificial Intelligence Review. SpringerLink. Retrieved from https://link.springer.com/article/10.1007/s10462-018-9614-6

Dangeti, P. (2017, July 21). Statistics for Machine Learning.

Frost, J. (2022, January 26). Chi-Square Table. Statistics by Jim. Retrieved from https://statisticsbyjim.com/hypothesis-testing/chi-square-table/

MUKHERJEE. (2022). The Maths behind Linear SVC Classifier. Retrieved from https://www.kaggle.com/code/soham1024/the-maths-behind-linear-svc-classifier

Mohri, M., Rostamizadeh, A., Talwalkar, A., & Bach, F. (2012, September 7). Foundations of Machine Learning.

Nandi, S. (2021, July 1). Twitter Sentiment Analysis Using Machine Learning Approaches. Medium. Retrieved from https://nandisoham2017.medium.com/twitter-sentiment-analysis-using-machine-learning-approaches-14fba1b8e357

Mueller, J. P., & Massaron, L. (2021, February 9). Machine Learning for Dummies. For Dummies.

Pati, & Pradhan. (2020, December 12). Comparison Between Machine Learning Algorithms Used for Sentiment Analysis. *IAEME Publication.* Retrieved from https://iaeme.com/Home/article_id/IJARET_11_12_026

Raschka. (2015). Python Machine Learning Equation Reference. Retrieved from

https://github.com/rasbt/python-machine-learning-book

Redjeki, & Widyarto. (2022). View of Comparison of Seven Machine Learning Algorithms in the Classification of Public Opinion. View of Comparison of Seven Machine Learning Algorithms in the Classification of Public Opinion. Retrieved from https://jurnal.ubd.ac.id/index.php/te/article/view/1046/526

Streamlit.(2020). Retrieve from https://streamlit.io/