University of Toronto
School of Continuing Studies
SCS 3251 Statistics for Data Science

Spam Email Classification

Student: Arjie Cristobal
Professor: Dr. Sergiy Nokhrin
August 2023

Table of Contents

Objective

Introduction

Spam email[1] is unsolicited and unwanted junk email sent out in bulk to an indiscriminate recipient list. Typically, spam is sent for commercial purposes. It can be sent in massive volume by botnets, networks of infected computers.

Spam email can be dangerous. It can include malicious links that can infect the computer with malware. Dangerous spam emails often sound urgent, so the recipient feels the need to act. The common types of spam are Commercial advertisements, Antivirus warnings, Email spoofing, Sweepstakes winners, and Money scams.

Purpose of Report

The primary purpose of this study is to classify if the email is spam or not using Binary Classification[2].

The study also aims to determine whether feature selection can help improve in classifying spam emails. The study will explore the use of Stepwise Backward Elimination Technique, Pearson Correlation and Chi-square in feature selection.

Data Preparation

The dataset used in the study is called Spambase[4] from University of California Irvine (UCI) - Machine Learning Repository. It is a standard machine learning dataset and open data.

The dataset is in good quality because it is already sanitised. There are no null values in the dataset and all continuous variables are in the correct type. There are also no categorical variables present in the dataset that requires OneHotEncoding[5].

The dataset consists of 4601 rows and 58 features. The 57 features are continuous variables and predictors while the remaining one feature is the Label which classifies whether the email is spam or not.

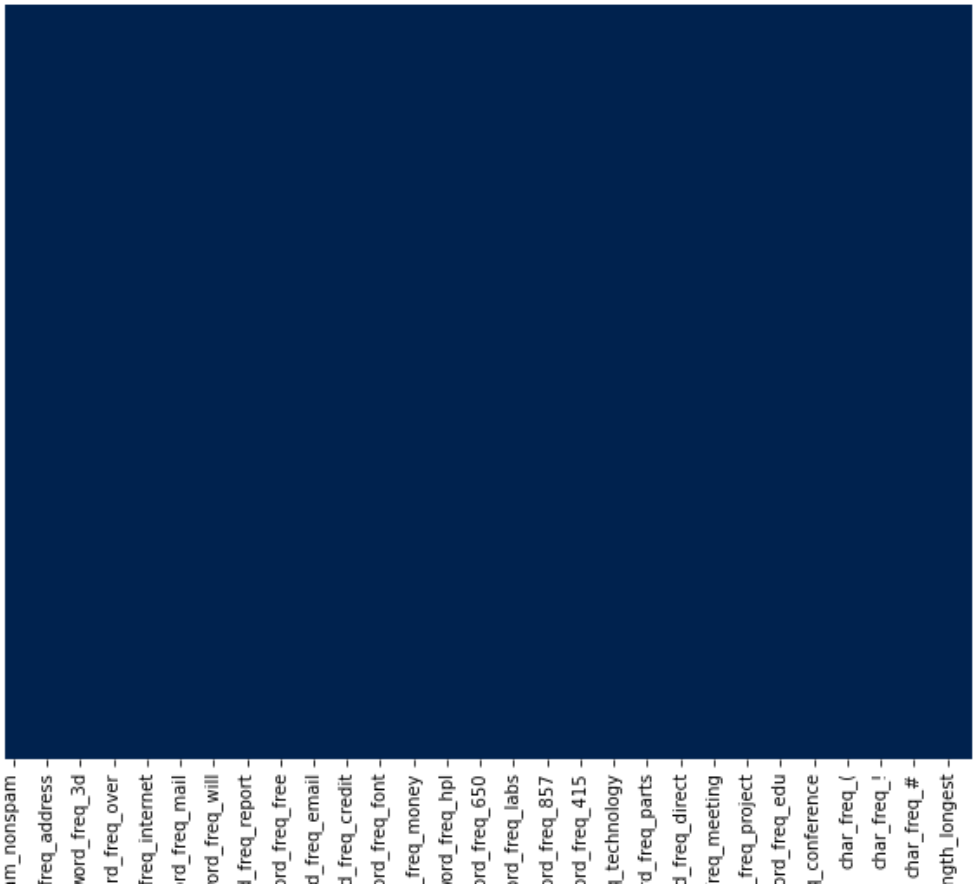

Fig. 1.0 There are no null values in the dataset.

The study procure the data by referring directly to the University of California Irvine (UCI) - Dataset Repository. Directly getting the dataset from Dataset Repository requires additional steps but offers more flexibility.

An alternative source of dataset is from Kaggle[6], an online community of data scientists and machine learning practitioners. However, the dataset in Kaggle was already prepared into two files, one for train data and the other is for test data.

The study introduced the SpambaseDatasetLoader[8] (see Appendix A Figure 1 and Figure 2), a reusable code responsible for retrieving the dataset from a repository.

The Python library Panda was used in data preparation and Matplotlib and Seaborn libraries for data visualisation. The Statsmodels library was used in feature selection and model analysis.

The SpambaseDatasetLoader[8] also offers functions for data preparations aside from loading the dataset (see Appendix A Figure 3).

Challenges:

The main challenges encountered in the study are imbalance data which can greatly affect a Binary Classification and a large number of predictive features.

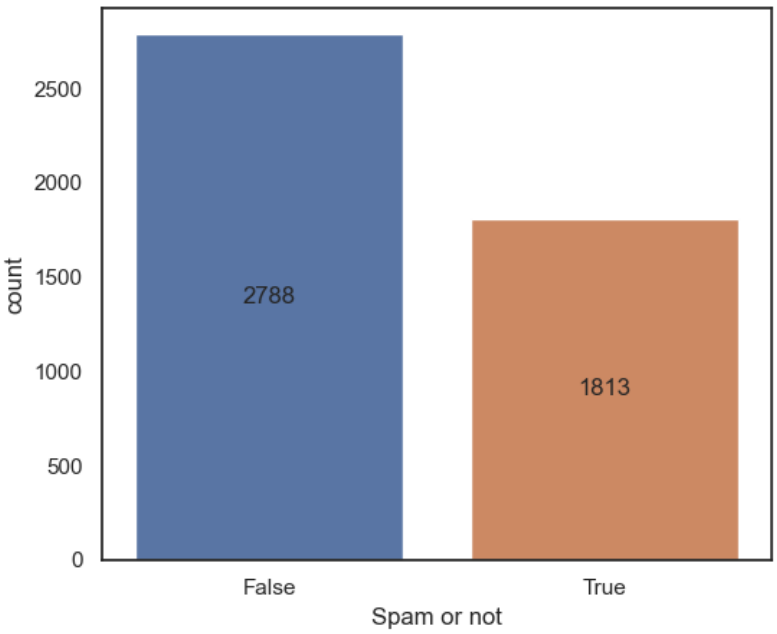

Fig.1.1 Imbalance data on dataset.
There are less spam records compared to non spam.

The large number of features is another challenge because it will add complexity to identify the correlations between the predictive features and the predictions. The lack of detailed description about each feature also added a challenge.

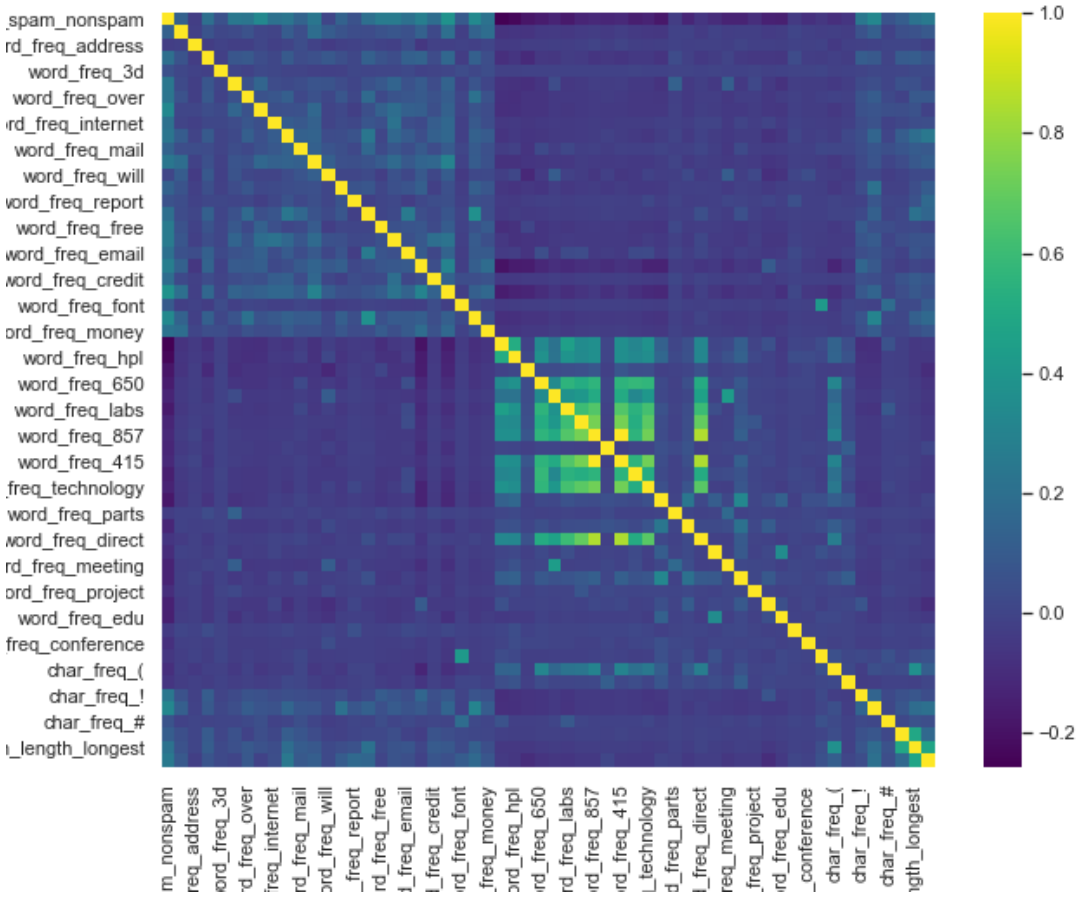

Fig.1.2 The correlation heatmap of Spambase dataset.

Solutions:

There are various ways of handling imbalance data such as collecting more data, using a different performance metric, resampling the dataset and generating synthetic samples.

For this study, the imbalance data was mainly handled by resampling the dataset[7] while the use of different performance metrics such as F1-score was also explored during model analysis.

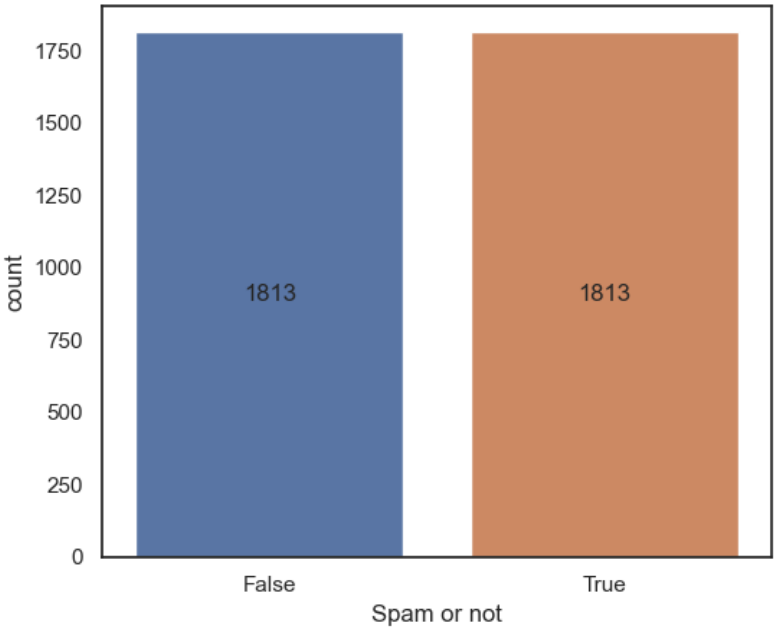

Fig.1.3 The distribution after resampling the data.

Meanwhile, the large number of features was addressed using the feature selection offered by SpambaseDatasetLoader[8] (see Appendix A Figure 3).

Model

The study introduced SpambaseModelHelper[9] (See Appendix B Figure 1) to produce a predictive model.

However, before fitting the dataset to the model, the study first performed feature engineering in the form of feature selection.

There are four options available offered by SpambaseDatasetLoader[8] for feature selection:

- Stepwise Backward Elimination - “This backward elimination technique uses a Logistic regression-based model which selects the features based on the p-value score of the feature. The features with p-value less than 0.05 are considered to be the more relevant feature.”[10] (see Appendix A Figure 4)

  There are 35 features left after performing this approach.

- Pearson Correlation - “This is used to construct a correlation matrix that measures the linear association between two features and gives a value between -1 and 1 indicating how related the two features are to one another. This measures the degree to which two features are interdependent by computing the association between each feature and the target variable, the one exerting high impact on the target can be picked out”[10]
  (see Appendix A Figure 5)

  “A value of 1 indicates a positive correlation, -1 indicates a negative correlation and 0 indicates no correlation between the features.”[10]

  There are only 19 features left after performing this feature selection.

- Chi-square - “A Chi-square test is used in statistical models to check the independence of attributes. The model measures the degree of deviation between the expected and actual response. The lower the value of Chi-square, the less dependent the variables are to one another, and the higher the value the more is their correlation.”[10]
  (see Appendix A Figure 6)

  For this study, the feature selection that uses Chi-square used k=19 similar to Pearson Correlation.

- None - No feature selection will be performed on the dataset. The number of features remains the same.

After the feature selection, the dataset will then be split to 80% training and 20% test dataset before fitting the training dataset to the model.

The study uses Logit from StatsModels to perform Logistic Regression to classify Spam emails.

After training the model, prediction was performed using the test dataset. The Logit Regression Summary was then printed (see Appendix C), a confusion matrix heatmap was displayed, and the score metrics such as Accuracy, Precision, Recall and F1-Score were all calculated, displayed and rendered in a graph.

Evaluation

Pseudo R-squared

The result of Pseudo R-squared shows positive results. The Pseudo R-squared is on a scale from 0 to 1, with higher values meaning a better fit.

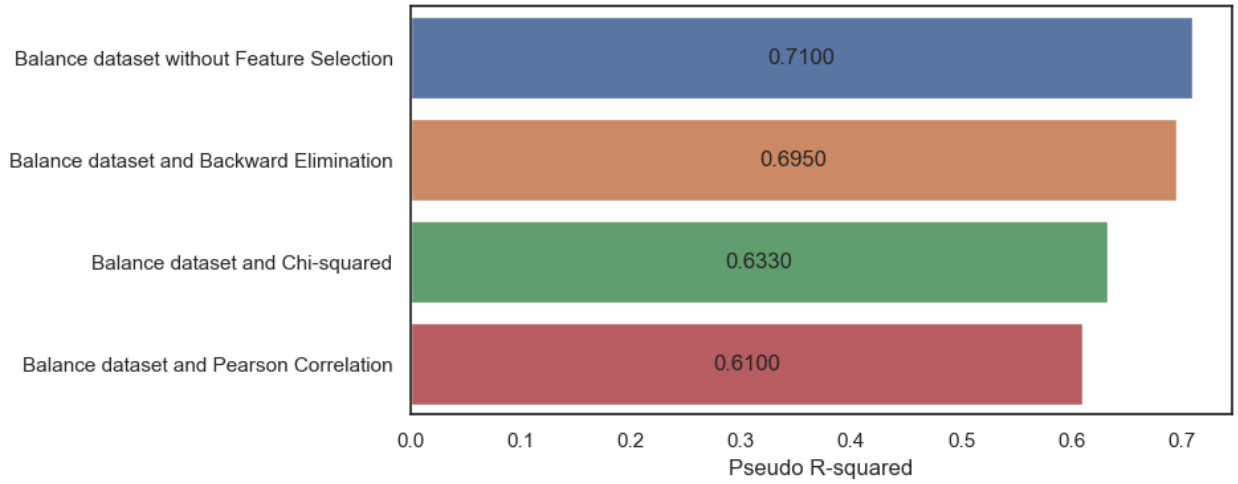

Fig. 1.4 The Pseudo R-squared result.

Confusion Matrix

The confusion matrix shows greater numbers of True Positive and True Negative compared to False Positives and False Negatives which is a positive sign for a good model.

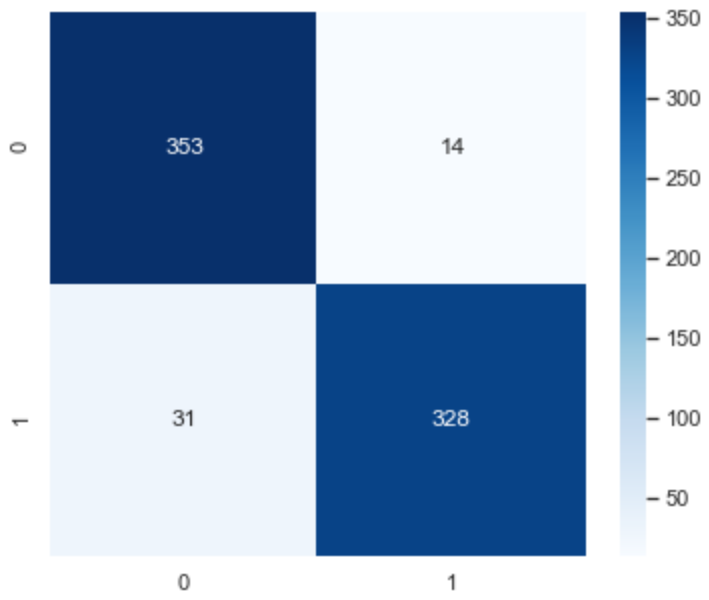

Fig. 1.5 Confusion Matrix of Balance Dataset and Backward Elimination Technique

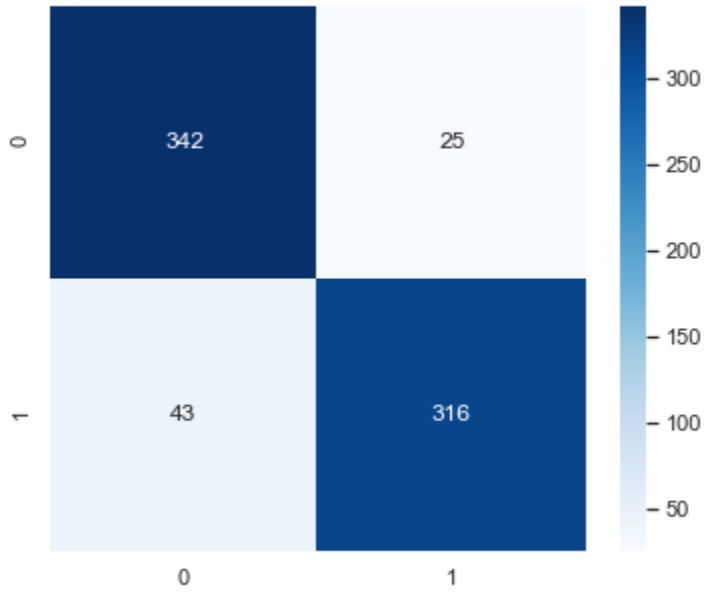

Fig. 1.6 Confusion Matrix of Balance Dataset and Pearson Correlation

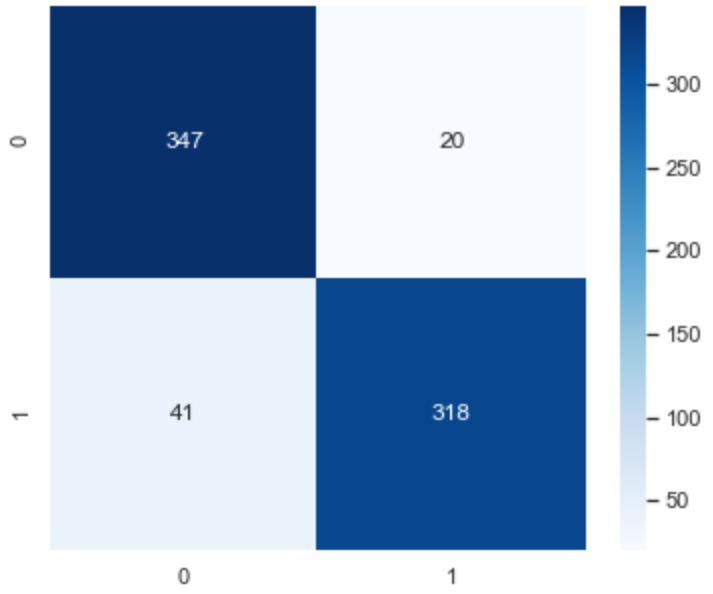

Fig. 1.7 Confusion Matrix of Balance Dataset and Chi-squared

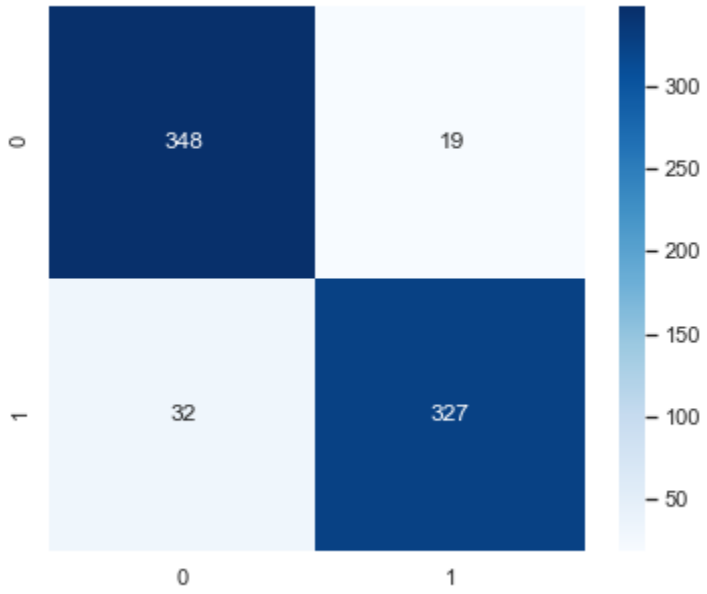

Fig. 1.8 Confusion Matrix of Balance Dataset without Feature Selection

Statistical Accuracy

For a balanced dataset we can use Accuracy as metric. The result shows that the model that undergoes Backward Elimination technique produces the best result.

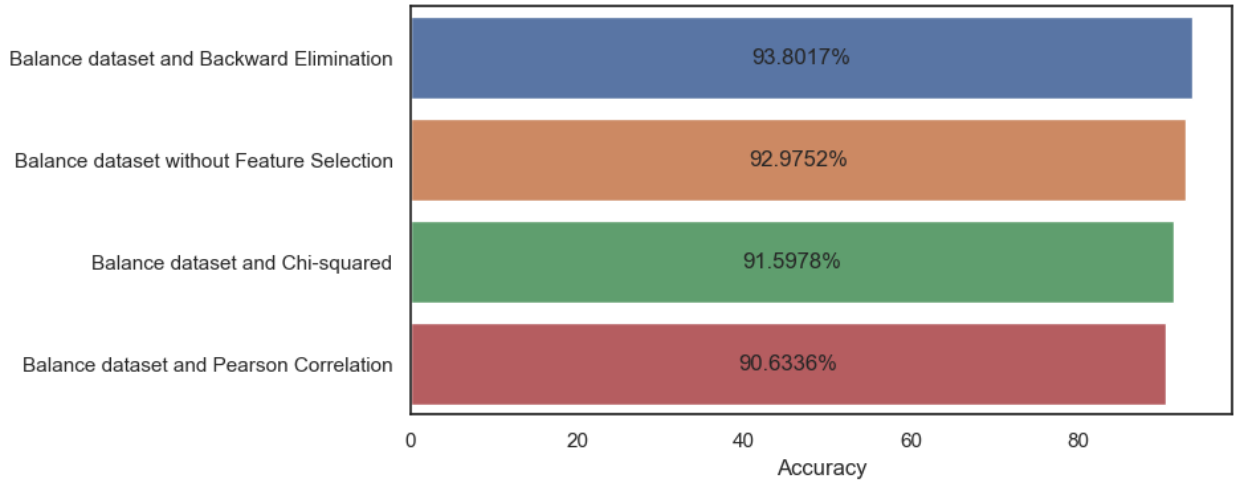

Fig.1.9

Full / Imbalance Dataset

This study also performs a similar analysis using a full dataset that has imbalanced data. The metric that was used is F1-Score because of the presence of imbalance data. The result is similar to a balanced dataset with Backward Elimination technique having the best result.

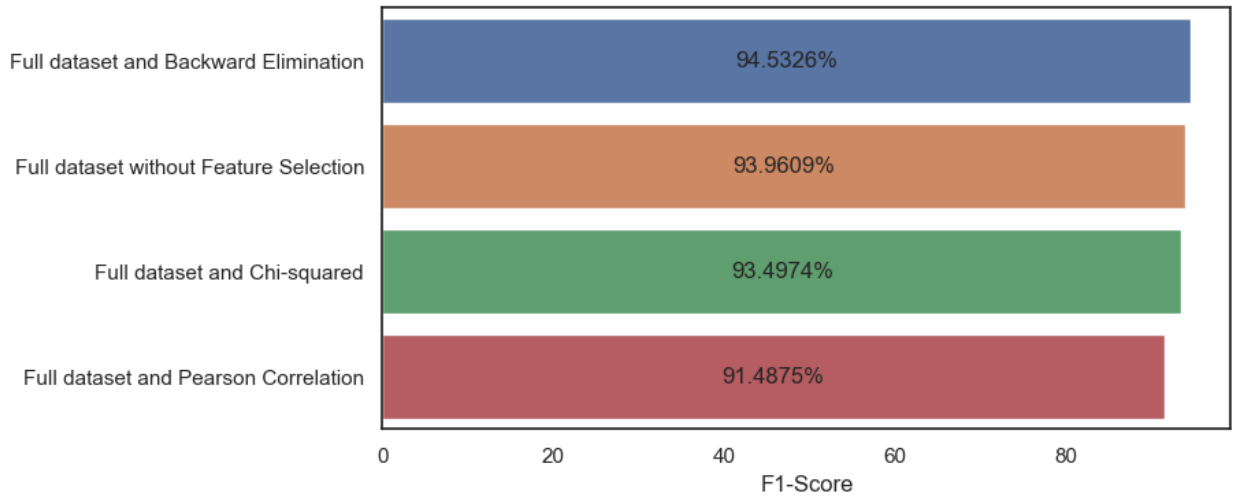

Fig. 1.10

Features with p-value greater than 0.05

The evaluation noted in the Logit Summary that there are features from Pearson Correlation and Chi-squared that have p-value greater than 0.05. (see Appendix C Figure 2 and Appendix C and Figure 3). Those features can be dropped as part of improvement. However, for this current study those features will not be deleted but instead will be kept as is but will be recommended to be deleted for future study.

Conclusion

The study was able to create a useful model for the classification of Spam email. It proves that Binary Classification is helpful in classifying if an email is a Spam or not.

Even though the difference in scores is not that significant between balance and imbalance dataset, the results show that imbalance data can affect the result because of unequal distribution. Hence, resampling of data is a best practice when doing Binary Classification.

The study recognized that the dataset was already old and new sources of data are needed. The use of other Classifications Models and hyperparameters are also highly recommended for future study.

Finally, the study shows the importance of the use of statistics in solving modern problems. This study can help serve as an introduction and reference to other researchers interested in topics related to both data science and internet security.

Appendix A. The Spambase Dataset Loader

Figure 1 - Initialize the Spambase Dataset Loader

```
import nbimporter
import spambase_dataset_loader_nb

loader = spambase_dataset_loader_nb.SpambaseDatasetLoader()
```

```
init Loader notebook
```

Figure 2 - Load Dataset

```
help(loader.load_dataset)
```

```
Help on method load_dataset in module spambase_dataset_loader_nb:

load_dataset(url='https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data') method of
    (string) --> None

    This function retrieves the spambase.data from University of California Irvine (UCI) - Dataset Repository
    and save into two files:

    1. spambase_all.csv - The CSV file that contain all the rows from spambase.data.
    2. spambase_balance.csv - The CSV file that rows balance between spam and not spam records.
```

Figure 3 - Perform Feature Selection

```
help(loader.perform_feature_selection)
```

```
Help on method perform_feature_selection in module spambase_dataset_loader_nb:

perform_feature_selection(data, feature_selection_type) method of spambase_dataset_loader_nb.SpambaseDatasetLoader instance
    (DataFrame, float) --> Dataframe


    This function performs feature selection based on provided feature selection type.

    Parameters
    ----------
    data: Dataframe that will be use in feature selection.

    feature_selection_type: The type of feature selection to be performed.
    1. stepwise_back - use stepwise backward elimation technique in feature selection..
    2. pearson - perform the feature selection using pearson correlation.
    3. chi2 - perform the feature selection using chi-squared.

    Returns
    ----------
    DataFrame : The DataFrame after performing feature selection.
```

Figure 4 - Backward Elimination Technique

```
help(loader.backward_elimitation)
```

```
Help on method backward_elimitation in module spambase_dataset_loader_nb:

backward_elimitation(data) method of spambase_dataset_loader_nb.SpambaseDatasetLoader instance
    (DataFrame) --> Array

    This backward elimitation technique used Logistic regression-based model which
    selects the features based on the p-value score of the feature.
    The features with p-value less than 0.05 are considered to be the more relevant feature
    Source: https://www.analyticsvidhya.com/blog/2021/04/discovering-the-shades-of-feature-selection-methods/

    Parameters
    ----------
    data: Dataframe that will be use in feature selection.

    Returns
    ----------
    Array of features
```

Figure 5 - Pearson Correlation

```
help(loader.person_correlation)
```

```
Help on method person_correlation in module spambase_dataset_loader_nb:

person_correlation(data) method of spambase_dataset_loader_nb.SpambaseDatasetLoader instance
    (DataFrame) --> Array

    Pearson Correlation is used to construct a correlation matrix that measures the linear association
    between two features and gives a value between -1 and 1 indicating how related the two features are to one another.
    This measures the degree to which two features are interdependent by computing the association
    between each feature and the target variable, the one exerting high impact on the target can be picked out

    A value of 1 indicates a positive correlation, -1 indicates a negative correlation and 0 indicates no correlation between the features.
    Source: https://www.analyticsvidhya.com/blog/2021/04/discovering-the-shades-of-feature-selection-methods/

    Parameters
    ----------
    data: Dataframe that will be use in feature selection.

    Returns
    ----------
    Array of features
```

Figure 6 - Chi-Squared

```
help(loader.chi2)
```

```
Help on method chi2 in module spambase_dataset_loader_nb:

chi2(data) method of spambase_dataset_loader_nb.SpambaseDatasetLoader instance
    (DataFrame) --> Array

    A chi-square test is used in statistical models to check the independence of attributes.
    The model measures the degree of deviation between the expected and actual response.
    The lower the value of Chi-square, the less dependent the variables are to one another,
    and the higher the value more is their correlation.
    Source: https://www.analyticsvidhya.com/blog/2021/04/discovering-the-shades-of-feature-selection-methods/

    Parameters
    ----------
    data: Dataframe that will be use in feature selection.

    Returns
    ----------
    Array of features
```

Appendix B. The Spambase Model Helper.

Figure 1 - Initialize the Spambase Model Helper

```
import nbimporter
import spambase_model_helper_nb

model_helper = spambase_model_helper_nb.SpambaseModelHelper()
```

```
init Model Helper notebook
```

Figure 2 - Function to run StatsModels Logit

```
help(model_helper.run_statsmodels_logit)
```

```
Help on method run_statsmodels_logit in module spambase_model_helper_nb:

run_statsmodels_logit(data, feature_selection_type=None, verbose=False) method of spambase_model_helper_nb.SpambaseModelHelper instance
    This function run logistic regression based from stats models.

    (DataFrame, boolean, boolean) --> DataFrame

    Parameters
    ----------
    data: Dataframe that will be use in running the model.

    feature_selection_type: Type of Feature Selection to use. Available options are stepwise_back, pearson and chi2.
    verbose: True to display statsmodels summary, confusion matrix and its heatmap otherwise false.

    Returns
    ----------
    DataFrame : The DataFrame of statsmodels scores.
```

Appendix C. Logistic Regression Results

Figure 1 - Balance Dataset and Backward Elimination Technique

Results: Logit

Model:	Logit	Pseudo R-squared:	0.695
Dependent Variable:	target_spam_nonspam	AIC:	1297.5680
Date:	2023-07-30 14:01	BIC:	1506.6043
No. Observations:	2900	Log-Likelihood:	-613.78
Df Model:	34	LL-Null:	-2010.1
Df Residuals:	2865	LLR p-value:	0.0000
Converged:	1.0000	Scale:	1.0000
No. Iterations:	14.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-1.1334	0.1382	-8.2002	0.0000	-1.4043	-0.8625
word_freq_make	-0.6895	0.2807	-2.4562	0.0140	-1.2397	-0.1393
word_freq_address	-0.1362	0.0711	-1.9162	0.0553	-0.2755	0.0031
word_freq_our	0.7089	0.1440	4.9236	0.0000	0.4267	0.9910
word_freq_over	1.1924	0.3294	3.6199	0.0003	0.5468	1.8380
word_freq_remove	2.1662	0.3974	5.4512	0.0000	1.3874	2.9451
word_freq_internet	0.3860	0.1591	2.4264	0.0152	0.0742	0.6979
word_freq_order	1.1309	0.4254	2.6585	0.0078	0.2972	1.9646
word_freq_will	-0.1952	0.0899	-2.1702	0.0300	-0.3714	-0.0189
word_freq_addresses	1.7607	1.0987	1.6026	0.1090	-0.3926	3.9141
word_freq_free	1.0159	0.1817	5.5924	0.0000	0.6599	1.3720
word_freq_business	0.9559	0.2544	3.7567	0.0002	0.4572	1.4545
word_freq_credit	1.8183	0.7640	2.3798	0.0173	0.3208	3.3157
word_freq_your	0.3062	0.0617	4.9629	0.0000	0.1852	0.4271
word_freq_000	2.0768	0.5421	3.8307	0.0001	1.0142	3.1393
word_freq_money	0.5631	0.2708	2.0795	0.0376	0.0324	1.0938
word_freq_hp	-1.6858	0.3045	-5.5367	0.0000	-2.2825	-1.0890
word_freq_hpl	-1.2124	0.4897	-2.4761	0.0133	-2.1722	-0.2527
word_freq_george	-16.0712	2.3997	-6.6972	0.0000	-20.7745	-11.3679
word_freq_650	0.6702	0.3002	2.2329	0.0256	0.0819	1.2586
word_freq_data	-0.8431	0.3263	-2.5842	0.0098	-1.4826	-0.2037
word_freq_85	-2.4049	0.8758	-2.7458	0.0060	-4.1215	-0.6883
word_freq_technology	0.7379	0.3737	1.9744	0.0483	0.0054	1.4704
word_freq_pm	-0.8429	0.4824	-1.7473	0.0806	-1.7883	0.1026
word_freq_meeting	-2.9390	0.9369	-3.1369	0.0017	-4.7753	-1.1027
word_freq_project	-2.4142	0.7127	-3.3875	0.0007	-3.8110	-1.0174
word_freq_re	-0.7563	0.1769	-4.2756	0.0000	-1.1030	-0.4096
word_freq_edu	-1.4416	0.3019	-4.7742	0.0000	-2.0334	-0.8498
word_freq_conference	-4.3789	1.8943	-2.3116	0.0208	-8.0918	-0.6661
char_freq_;	-0.6352	0.3304	-1.9228	0.0545	-1.2827	0.0123
char_freq_!	0.2405	0.0650	3.7016	0.0002	0.1132	0.3679
char_freq_$	5.7986	0.9568	6.0603	0.0000	3.9233	7.6739
char_freq_#	2.8186	0.7155	3.9392	0.0001	1.4162	4.2210
capital_run_length_longest	0.0140	0.0024	5.8951	0.0000	0.0094	0.0187
capital_run_length_total	0.0005	0.0002	2.2815	0.0225	0.0001	0.0010

Figure 2 - Balance Dataset and Pearson Correlation

Results: Logit

Model:	Logit	Pseudo R-squared:	0.610
Dependent Variable:	target_spam_nonspam	AIC:	1609.2204
Date:	2023-07-30 14:01	BIC:	1728.6697
No. Observations:	2900	Log-Likelihood:	-784.61
Df Model:	19	LL-Null:	-2010.1
Df Residuals:	2880	LLR p-value:	0.0000
Converged:	1.0000	Scale:	1.0000
No. Iterations:	14.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-1.8312	0.1209	-15.1454	0.0000	-2.0682	-1.5942
word_freq_all	0.1417	0.1256	1.1283	0.2592	-0.1044	0.3877
word_freq_our	0.5874	0.1010	5.8140	0.0000	0.3894	0.7854
word_freq_over	1.3171	0.2837	4.6431	0.0000	0.7611	1.8731
word_freq_remove	3.0495	0.4520	6.7461	0.0000	2.1635	3.9355
word_freq_order	1.8773	0.3999	4.6940	0.0000	1.0934	2.6612
word_freq_receive	0.1043	0.3502	0.2979	0.7658	-0.5820	0.7907
word_freq_free	1.2005	0.1687	7.1153	0.0000	0.8698	1.5312
word_freq_business	1.1723	0.2480	4.7261	0.0000	0.6861	1.6584
word_freq_email	0.3444	0.1296	2.6566	0.0079	0.0903	0.5985
word_freq_you	0.0098	0.0368	0.2676	0.7890	-0.0622	0.0819
word_freq_your	0.3478	0.0579	6.0046	0.0000	0.2343	0.4614
word_freq_000	2.9013	0.5702	5.0887	0.0000	1.7838	4.0188
word_freq_money	0.7718	0.2713	2.8453	0.0044	0.2402	1.3035
word_freq_hp	-1.4906	0.2988	-4.9891	0.0000	-2.0762	-0.9050
word_freq_hpl	-1.5563	0.4946	-3.1464	0.0017	-2.5258	-0.5869
word_freq_george	-14.6271	3.2712	-4.4715	0.0000	-21.0386	-8.2157
char_freq_!	0.3693	0.0929	3.9752	0.0001	0.1872	0.5513
char_freq_$	5.3150	0.8136	6.5330	0.0000	3.7205	6.9096
capital_run_length_total	0.0012	0.0002	6.2442	0.0000	0.0008	0.0016

Figure 3 - Balance Dataset and Chi-squared

Results: Logit

Model:	Logit	Pseudo R-squared:	0.633
Dependent Variable:	target_spam_nonspam	AIC:	1516.5313
Date:	2023-07-30 14:01	BIC:	1635.9806
No. Observations:	2900	Log-Likelihood:	-738.27
Df Model:	19	LL-Null:	-2010.1
Df Residuals:	2880	LLR p-value:	0.0000
Converged:	1.0000	Scale:	1.0000
No. Iterations:	14.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-1.3129	0.1230	-10.6749	0.0000	-1.5540	-1.0719
word_freq_our	0.6396	0.1297	4.9312	0.0000	0.3854	0.8937
word_freq_remove	2.6880	0.4336	6.1994	0.0000	1.8382	3.5378
word_freq_free	1.0940	0.1728	6.3326	0.0000	0.7554	1.4326
word_freq_business	1.1303	0.2526	4.4752	0.0000	0.6353	1.6253
word_freq_you	-0.0031	0.0361	-0.0860	0.9315	-0.0739	0.0677
word_freq_credit	1.1573	0.6101	1.8969	0.0578	-0.0385	2.3531
word_freq_your	0.3264	0.0598	5.4606	0.0000	0.2092	0.4435
word_freq_000	3.2666	0.5681	5.7495	0.0000	2.1530	4.3801
word_freq_money	1.0977	0.3079	3.5651	0.0004	0.4942	1.7012
word_freq_hp	-1.5544	0.2755	-5.6425	0.0000	-2.0944	-1.0145
word_freq_hpl	-1.2441	0.4491	-2.7703	0.0056	-2.1243	-0.3639
word_freq_george	-18.3510	2.7401	-6.6973	0.0000	-23.7214	-12.9806
word_freq_lab	-3.2725	1.6146	-2.0268	0.0427	-6.4370	-0.1080
word_freq_meeting	-3.0557	0.9612	-3.1791	0.0015	-4.9396	-1.1718
word_freq_edu	-1.6748	0.3228	-5.1878	0.0000	-2.3075	-1.0420
char_freq_!	0.3087	0.0871	3.5446	0.0004	0.1380	0.4795
capital_run_length_average	0.0070	0.0202	0.3459	0.7294	-0.0326	0.0466
capital_run_length_longest	0.0167	0.0029	5.7848	0.0000	0.0111	0.0224
capital_run_length_total	0.0006	0.0002	2.6890	0.0072	0.0002	0.0010

Figure 4 - Balance Dataset without Feature Selection

Results: Logit

Model:	Logit	Pseudo R-squared:	0.710
Dependent Variable:	y	AIC:	1281.4857
Date:	2023-07-30 14:01	BIC:	1627.8887
No. Observations:	2900	Log-Likelihood:	-582.74
Df Model:	57	LL-Null:	-2010.1
Df Residuals:	2842	LLR p-value:	0.0000
Converged:	1.0000	Scale:	1.0000
No. Iterations:	15.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-1.2212	0.1767	-6.9123	0.0000	-1.5675	-0.8749
x1	-0.6867	0.3051	-2.2507	0.0244	-1.2847	-0.0887
x2	-0.1454	0.0774	-1.8798	0.0601	-0.2970	0.0062
x3	0.0306	0.1440	0.2126	0.8316	-0.2516	0.3129
x4	1.6498	1.3623	1.2111	0.2259	-1.0202	4.3199
x5	0.6578	0.1446	4.5498	0.0000	0.3744	0.9411
x6	1.2704	0.3491	3.6386	0.0003	0.5861	1.9546
x7	2.0182	0.3902	5.1723	0.0000	1.2534	2.7830
x8	0.4023	0.1657	2.4285	0.0152	0.0776	0.7271
x9	1.3133	0.4665	2.8152	0.0049	0.3990	2.2277
x10	0.1112	0.0969	1.1474	0.2512	-0.0788	0.3012
x11	-0.0780	0.3828	-0.2037	0.8386	-0.8282	0.6723
x12	-0.2336	0.0941	-2.4839	0.0130	-0.4180	-0.0493
x13	-0.1332	0.2973	-0.4479	0.6542	-0.7159	0.4496
x14	0.0572	0.1439	0.3977	0.6908	-0.2248	0.3393
x15	2.2060	1.4018	1.5737	0.1155	-0.5414	4.9534
x16	0.9560	0.1846	5.1779	0.0000	0.5941	1.3178
x17	1.0764	0.2835	3.7974	0.0001	0.5209	1.6320
x18	0.0750	0.1381	0.5432	0.5870	-0.1956	0.3456
x19	0.0612	0.0442	1.3845	0.1662	-0.0255	0.1479
x20	1.6784	0.8010	2.0955	0.0361	0.1085	3.2483
x21	0.3265	0.0691	4.7218	0.0000	0.1909	0.4620
x22	0.1395	0.1858	0.7507	0.4528	-0.2247	0.5038
x23	2.1051	0.5702	3.6919	0.0002	0.9875	3.2227
x24	0.3488	0.2335	1.4934	0.1353	-0.1090	0.8065
x25	-1.5649	0.3462	-4.5199	0.0000	-2.2434	-0.8863
x26	-1.1597	0.5169	-2.2436	0.0249	-2.1729	-0.1466
x27	-17.2049	2.7380	-6.2837	0.0000	-22.5714	-11.8385
x28	0.5492	0.2849	1.9277	0.0539	-0.0092	1.1076
x29	-2.1617	1.4833	-1.4573	0.1450	-5.0690	0.7457
x30	-0.9228	0.4703	-1.9621	0.0497	-1.8446	-0.0010
x31	0.6337	1.2400	0.5110	0.6093	-1.7968	3.0641
x32	1.3374	3.7459	0.3570	0.7211	-6.0045	8.6793
x33	-0.7775	0.3389	-2.2938	0.0218	-1.4418	-0.1132
x34	-18.7836	5.7273	-3.2797	0.0010	-30.0089	-7.5583
x35	-2.4404	0.9378	-2.6024	0.0093	-4.2784	-0.6024
x36	0.7722	0.3678	2.0997	0.0358	0.0514	1.4930
x37	0.0366	0.2210	0.1657	0.8684	-0.3965	0.4697
x38	0.5716	1.2825	0.4457	0.6558	-1.9420	3.0852
x39	-0.6771	0.4841	-1.3987	0.1619	-1.6259	0.2717
x40	0.3589	1.1178	0.3211	0.7481	-1.8319	2.5498
x41	-42.8453	40.1180	-1.0680	0.2855	-121.4751	35.7846
x42	-2.7829	0.9420	-2.9542	0.0031	-4.6292	-0.9366
x43	-1.7003	1.0850	-1.5671	0.1171	-3.8268	0.4262
x44	-2.1665	0.7323	-2.9587	0.0031	-3.6017	-0.7313
x45	-0.8614	0.1823	-4.7257	0.0000	-1.2186	-0.5041
x46	-1.1842	0.2882	-4.1091	0.0000	-1.7491	-0.6194
x47	-3.7920	2.5850	-1.4669	0.1424	-8.8585	1.2745
x48	-3.9483	1.9609	-2.0136	0.0441	-7.7916	-0.1051
x49	-1.0017	0.5313	-1.8854	0.0594	-2.0431	0.0396
x50	0.1509	0.3486	0.4328	0.6651	-0.5323	0.8340
x51	-1.8605	1.4796	-1.2574	0.2086	-4.7603	1.0394
x52	0.2281	0.0601	3.7972	0.0001	0.1104	0.3458
x53	5.9557	0.9628	6.1859	0.0000	4.0686	7.8427
x54	2.3343	1.6080	1.4516	0.1466	-0.8174	5.4859
x55	0.0204	0.0252	0.8115	0.4171	-0.0289	0.0698
x56	0.0119	0.0035	3.3758	0.0007	0.0050	0.0188
x57	0.0007	0.0003	2.5250	0.0116	0.0002	0.0012

Appendix D. Code - SpambasedDatasetLoader

```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import statsmodels.api as sm
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

class SpambaseDatasetLoader():

    def __init__(self):
        print('init Loader notebook')

    def load_dataset(self,
url='https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.dat
a'):
        '''

        (string) --> None

        This function retrieves the spambase.data from University of California
Irvine (UCI) - Dataset Repository
        and save into two files:

        1. spambase_all.csv - The CSV file that contains all the rows from
spambase.data.
        2. spambase_balance.csv - The CSV file that rows balance between spam and
not spam records.

        '''
        print('Loading dataset.')

        columns = [
            'word_freq_make',
            'word_freq_address',
            'word_freq_all',
            'word_freq_3d',
            'word_freq_our',
            'word_freq_over',
            'word_freq_remove',
            'word_freq_internet',
            'word_freq_order',
            'word_freq_mail',
            'word_freq_receive',
            'word_freq_will',
            'word_freq_people',
            'word_freq_report',
            'word_freq_addresses',
            'word_freq_free',
            'word_freq_business',
            'word_freq_email',
            'word_freq_you',
            'word_freq_credit',
            'word_freq_your',
```

```
            'word_freq_font',
            'word_freq_000',
            'word_freq_money',
            'word_freq_hp',
            'word_freq_hpl',
            'word_freq_george',
            'word_freq_650',
            'word_freq_lab',
            'word_freq_labs',
            'word_freq_telnet',
            'word_freq_857',
            'word_freq_data',
            'word_freq_415',
            'word_freq_85',
            'word_freq_technology',
            'word_freq_1999',
            'word_freq_parts',
            'word_freq_pm',
            'word_freq_direct',
            'word_freq_cs',
            'word_freq_meeting',
            'word_freq_original',
            'word_freq_project',
            'word_freq_re',
            'word_freq_edu',
            'word_freq_table',
            'word_freq_conference',
            'char_freq_;',
            'char_freq_(',
            'char_freq_[',
            'char_freq_!',
            'char_freq_$',
            'char_freq_#',
            'capital_run_length_average',
            'capital_run_length_longest',
            'capital_run_length_total',
            'spam_nonspam']

        # retrieve the spam data from icu.
        spam_data = pd.read_csv(url, header=None, names=columns, index_col=False)

        # make the label the first feature.
        spam_data.insert(0, 'target_spam_nonspam', spam_data['spam_nonspam'])
        spam_data = spam_data.drop('spam_nonspam', axis = 1)

        # save the data to the new csv.
        spam_data.to_csv('./datasets/spambase_all.csv', index = False)
        print('Loading spambase_all.csv completed.')

        spam = spam_data[spam_data['target_spam_nonspam']==1]
        non_spam = spam_data[spam_data['target_spam_nonspam']==0]
        non_spam = non_spam.sample(n=len(spam), random_state=101)
        spam_data_balance = pd.concat([spam,non_spam],axis=0)
        print('Loading spambase_balance.csv completed.')

    def get_full_dataset(self):
        '''

        (None) --> Dataframe

        This function returns the spambase imbalance dataset.
```

```
        '''
        print('get_full_dataset')
        return pd.read_csv('./datasets/spambase_all.csv')

    def get_balance_dataset(self):
        '''

        (None) --> Dataframe

        This function returns the spambase dataset based on the balance dataset.

        '''
        print('get_balance_data')
        return pd.read_csv('./datasets/spambase_balance.csv')

    def backward_elimitation(self, data):
        '''

        (DataFrame) --> Array

        This backward elimination technique used Logistic regression-based model
which
        selects the features based on the p-value score of the feature.
        The features with p-value less than 0.05 are considered to be the more
relevant feature
        Source:
https://www.analyticsvidhya.com/blog/2021/04/discovering-the-shades-of-feature-selec
tion-methods/

        Parameters
        ----------
        data: Dataframe that will be used in feature selection.

        Returns
        ----------
        Array of features

        '''
        print('Feature Selection using Backward Elimination')
        threshold = 0.05
        selected_columns = data.columns
        selected_columns = selected_columns[1:].values # remove the label

        X = data.iloc[:,1:]
        y = data.iloc[:,0]

        # Add constant to predictors for statsmodels
        X = sm.add_constant(X)

        logit_model = sm.Logit(y, X).fit()

        # Perform backward elimination
        while len(X.columns) > 1:
            # Fit Logit model with all predictors except one
            logit_model = sm.Logit(y, X.iloc[:, :-1]).fit()

            # Get p-values for each predictor
            p_values = logit_model.pvalues

            # Remove predictor with highest p-value
            max_p_value = p_values.idxmax()
            if p_values[max_p_value] > threshold:
```

```
                X = X.drop(max_p_value, axis=1)
            else:
                break

        X = X.drop('const', axis=1) # cleanup remove the const column before
returning
        return X.columns

    def person_correlation(self, data):
        '''

        (DataFrame) --> Array

        Pearson Correlation is used to construct a correlation matrix that measures
the linear association
        between two features and gives a value between -1 and 1 indicating how
related the two features are to one another.
        This measures the degree to which two features are interdependent by
computing the association
        between each feature and the target variable, the one exerting high impact
on the target can be picked out

        A value of 1 indicates a positive correlation, -1 indicates a negative
correlation and 0 indicates no correlation between the features.
        Source:
https://www.analyticsvidhya.com/blog/2021/04/discovering-the-shades-of-feature-selec
tion-methods/

        Parameters
        ----------
        data: Dataframe that will be used in feature selection.

        Returns
        ----------
        Array of features

        '''
        print('Feature Selection using Pearson Correlation')
        corr = data.corr()
        cor_target = abs(corr['target_spam_nonspam'])

        #Selecting highly correlated features
        relevant_features = cor_target[cor_target>0.2]
        selected_columns = relevant_features.keys().to_list()

        selected_columns.remove('target_spam_nonspam')
        return selected_columns

    def chi2(self, data):
        '''

        (DataFrame) --> Array

        A chi-square test is used in statistical models to check the independence of
attributes.
        The model measures the degree of deviation between the expected and actual
response.
        The lower the value of Chi-square, the less dependent the variables are to
one another,
        and the higher the value the more is their correlation.
        Source:
https://www.analyticsvidhya.com/blog/2021/04/discovering-the-shades-of-feature-selec
```

```
tion-methods/

        Parameters
        ----------
        data: Dataframe that will be used in feature selection.

        Returns
        ----------
        Array of features

        '''
        print('Feature Selection using Chi-squared')
        X = data.iloc[:,1:]
        y = data.iloc[:,0]

        chi2_features = SelectKBest(chi2, k=19)
        X_kbest_features = chi2_features.fit_transform(X, y)
        mask=chi2_features.get_support()
        selected_columns =[]
        for bool, feature in zip(mask, X.columns):
            if (bool):
                selected_columns.append(feature)

        return selected_columns


    def perform_feature_selection(self, data, feature_selection_type):
        '''

        (DataFrame, float) --> Dataframe


        This function performs feature selection based on the provided feature
selection type.

        Parameters
        ----------
        data: Dataframe that will be used in feature selection.

        feature_selection_type: The type of feature selection to be performed.
        1. stepwise_back - use stepwise backward elimination technique in feature
selection..
        2. pearson - perform the feature selection using pearson correlation.
        3. chi2 - perform the feature selection using chi-squared.

        Returns
        ----------
        DataFrame : The DataFrame after performing feature selection.

        '''
        print('The shape before feature selection: {}'.format(data.shape))

        if feature_selection_type == 'stepwise_back':
            selected_columns = self.backward_elimitation(data)
        elif feature_selection_type == 'pearson':
            selected_columns = self.person_correlation(data)
        elif feature_selection_type == 'chi2':
            selected_columns = self.chi2(data)
        else:
            raise ValueError('Unknown type of feature selection.')

        y = pd.DataFrame()
```

```
        y['target_spam_nonspam'] = data.iloc[:,0]

        X = pd.DataFrame(data = data.iloc[:,1:], columns = selected_columns)
        print('The shape after feature selection: {}'.format(X.shape))

        return X, y
```

Appendix E. Code - SpambasedDatasetHelper

```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

import statsmodels.api as sm

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

import nbimporter
import spambase_dataset_loader_nb

class SpambaseModelHelper():

    def __init__(self):
        print('init Model Helper notebook')

    def run_statsmodels_logit(self, data, feature_selection_type=None,
verbose=False):
        '''

        This function runs logistic regression based on stats models.

        (DataFrame, boolean, boolean) --> DataFrame

        Parameters
        ----------
        data: Dataframe that will be used in running the model.

        feature_selection_type: Type of Feature Selection to use. Available options
are stepwise_back, pearson and chi2.
        verbose: True to display statsmodels summary, confusion matrix and its
heatmap otherwise false.

        Returns
        ----------
        DataFrame : The DataFrame of statsmodels scores.
        '''
        X = data.iloc[:,1:].values
        y = data.iloc[:,0].values

        if feature_selection_type != None:

            loader = spambase_dataset_loader_nb.SpambaseDatasetLoader()
```

```
            result = loader.perform_feature_selection(data, feature_selection_type)

            X = result[0]
            y = result[1]

        X = sm.add_constant(X)

        # split the dataset
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state=101)

        # building the model and fit the model using the training data
        statsmodels_lr = sm.Logit(y_train, X_train).fit()

        if verbose == True:
            print(statsmodels_lr.summary2())

        # performing predictions on the test dataset
        yhat = statsmodels_lr.predict(X_test)

        y_pred_statsmodels = list(map(round, yhat))

        cf_matrix = confusion_matrix(y_test, y_pred_statsmodels)

        TP = cf_matrix[0][0]
        TN = cf_matrix[1][1]
        FP = cf_matrix[0][1]
        FN = cf_matrix[1][0]
        if verbose == True:
            print('Confusion Matrix')
            print(cf_matrix)
            path = ('./images/cf_matrix_TP{}_TN{}_FP{}_FN{}.png'.format(TP, TN, FP,
FN))

            sns.set(rc={"figure.figsize":(6, 5)})
            sns_plot = sns.heatmap(cf_matrix, annot=True,  fmt='d', cmap='Blues')
            plt.savefig(path)

        acc = (TP + TN) / np.sum(cf_matrix)
        precision = TP / (TP + FP)
        recall = TP / (TP + FN)
        f1_score = (2 * (precision * recall)) / (precision + recall)

        statsmodels_scores_df = pd.DataFrame([[acc, precision, recall, f1_score]],
['Score'], ['Accuracy', 'Precision', 'Recall', 'F1-Score'])

        return statsmodels_scores_df


    def run_sklearn_log_reg(self, data, feature_selection_type=None):
        '''

        This function runs logistic regression based on sklearn models.

        (DataFrame, boolean, boolean) --> DataFrame

        Parameters
        ----------
        data: Dataframe that will be used in running the model.

        feature_selection_type: Type of Feature Selection to use. Available options
are stepwise_back, pearson and chi2.
```

```
        Returns
        ----------
        DataFrame : The DataFrame of statsmodels scores.
        '''
        X = data.iloc[:,1:].values
        y = data.iloc[:,0].values

        if feature_selection_type!=None:

            loader = spambase_dataset_loader_nb.SpambaseDatasetLoader()

            result = loader.perform_feature_selection(data, feature_selection_type)

            X = result[0]
            y = result[1]

        # split the dataset
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state=101)

        # instantiate the model
        sklearn_lr = LogisticRegression(max_iter=10000)

        # fit the model using the training data
        sklearn_lr.fit(X_train, y_train)

        # use model to make predictions on test data
        y_pred_sklearn = sklearn_lr.predict(X_test)

        # calculate the accuracy, precision, recall and f1-score
        acc = accuracy_score(y_test, y_pred_sklearn)
        precision = precision_score(y_test, y_pred_sklearn)
        recall = recall_score(y_test, y_pred_sklearn)
        f1 = f1_score(y_test, y_pred_sklearn)

        sklearn_scores_df = pd.DataFrame([[acc, precision, recall, f1]], ['Score'],
['Accuracy', 'Precision', 'Recall', 'F1-Score'])

        return sklearn_scores_df
```

References

1. Spam Email:
   https://www.cisco.com/c/en/us/products/security/email-security/what-is-spam.html

2. Binary Classification:
   https://deepchecks.com/glossary/binary-classification/

3. Backward Elimination Technique:
   https://www.kaggle.com/code/bbloggsbott/feature-selection-correlation-and-p-value/notebook

4. Spambase:
   https://archive.ics.uci.edu/dataset/94/spambase

5. OneHotEncoding:
   https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/

6. Kaggle:
   https://www.kaggle.com/datasets/monizearabadgi/spambase

7. Tactics to combat imbalance classes:
   https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

8. SpambaseDatasetLoader:
   https://github.com/quickheaven/scs-3251-statistics-for-data-science/blob/main/spambase_dataset_loader_nb.ipynb

9. SpambaseModelHelper:
   https://github.com/quickheaven/scs-3251-statistics-for-data-science/blob/main/spambase_model_helper_nb.ipynb

10. Feature Selection
    https://www.analyticsvidhya.com/blog/2021/04/discovering-the-shades-of-feature-selection-methods/