KEVIN JOHNSON

ASSIGNMENT 5

BUILDING A SPAM FILTER AS A PRODUCTION MODEL

I. Introduction

The Spambase dataset was originally developed by Hewlett Packard using a collection of spam and non-spam e-mails from filed work and personal e-mails. The dataset contains 4,601 observations on 58 variables. Of those observations, 1,813 are labelled as spam. Using this dataset, we intend to evaluate several different spam classification models and then select one to run in production. To run this analysis, the dataset was split 50-50 between the training and test sets. The dimensions are (2318, 58) for the training set and (2283, 58) for the test set. Below are the names of all variables. In this analysis, Spam will be the response variable.

"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_semicolon"	"char_freq_parenthesis"	"char_freq_bracket"	"char_freq_xpoint"
"char_freq_dollar"	"char_freq_hashtag"	"capital_run_length_average"	"capital_run_length_longest"
"capital_run_length_total"	"spam"		

II. Computational EDA

Computational exploratory data analysis was performed on the dataset using three different models: Logistic Regression with backward variable selection, a One Rule model, and XGBoost with 10 rounds.

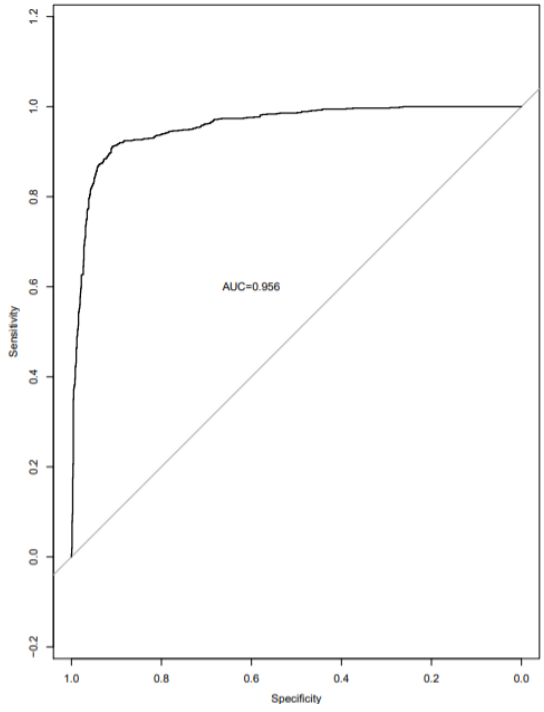
Logistic Regression with Backward Variable Selection

The first model run was a standard logistic regression model with backward variable selection. Backward variable selection is a form of stepwise regression that starts with every candidate variable and works its way down. At each step, the variable that is the least significant is removed. This process continues until there are no insignificant variables remaining. Below are the results of the model:

	0	1
0	0.910	0.090
1	0.089	0.911

Metric	Score
Accuracy	0.911
Precision	0.910
Recall	0.911
Specificity	0.910
F1	0.910

The logistic regression model had an accuracy score of .911. The true positive rate (recall) and true negative rate (specificity) are nearly identical, scoring 0.911 and 0.910, respectively. The AUC for this model is 0.956, as shown in the ROC curve below. A more detailed summary of the logistic regression model is included on the following page.



Logistic Regression with Backward Selection

	Dependent variable:
	spam
word_freq_make	-0.050**
	(0.024)
word_freq_address	-0.010*
	(0.006)
word_freq_all	0.051***
	(0.015)
word_freq_3d	0.016***
	(0.004)
word_freq_our	0.096***
	(0.010)
word_freq_over	0.082***
	(0.024)
word_freq_remove	0.208***
	(0.018)
word_freq_internet	0.092***
	(0.017)
word_freq_order	0.037
	(0.026)
word_freq_receive	0.089**
	(0.037)
word_freq_will	-0.026***
	(0.008)
word_freq_free	0.098***
	(0.010)
word_freq_business	0.061***
	(0.016)
word_freq_email	0.063***
	(0.014)

word_freq_you	0.009**
	(0.004)
word_freq_credit	0.053***
	(0.012)
word_freq_your	0.046***
	(0.007)
word_freq_font	0.050***
	(0.008)
word_freq_000	0.232***
	(0.023)
word_freq_money	0.094***
	(0.019)
word_freq_hp	-0.021***
	(0.005)
word_freq_hpl	-0.031***
	(0.010)
word_freq_george	-0.010***
	(0.002)
word_freq_labs	-0.071***
	(0.023)
word_freq_data	-0.039***
	(0.011)
word_freq_415	0.087***
	(0.029)
word_freq_parts	-0.065
	(0.041)
word_freq_meeting	-0.052***
	(0.010)
word_freq_original	-0.079**
	(0.031)
word_freq_project	-0.035***

	(0.013)
word_freq_re	-0.035***
	(0.007)
word_freq_edu	-0.032***
	(0.007)
word_freq_table	-0.178**
	(0.078)
word_freq_conference	-0.051***
	(0.020)
char_freq_semicolon	-0.150***
	(0.030)
char_freq_parenthesis	-0.057*
	(0.034)
char_freq_xpoint	0.050***
	(0.007)
char_freq_dollar	0.222***
	(0.029)
char_freq_hashtag	0.031
	(0.020)
capital_run_length_longest	0.0002***
	(0.0001)
capital_run_length_total	0.0001***
	(0.00002)
Constant	0.190***
	(0.016)
Observations	2,318
Log Likelihood	-655.499
Akaike Inf. Crit.	1,394.998

One Rule Model

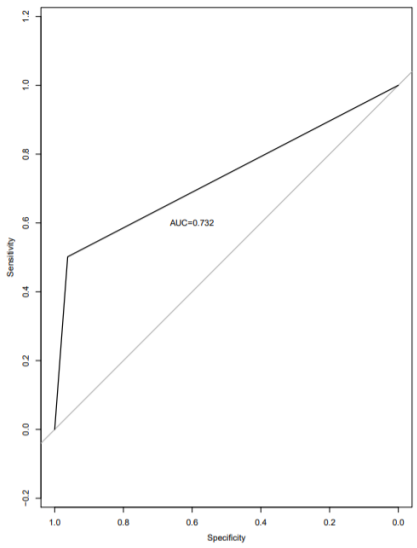
The second Computational EDA model to be run is the One Rule model. In this model, one rule is generated for each predictor in the data. The rule with the smallest total error is then selected as its "one rule." In this analysis, the rule that was selected used the Char_freq_dollar variable. Below are the results:

```
Rules:
If char_freq_dollar = (-0.0053,0.0746] then spam = 0
If char_freq_dollar = (0.0746,5.31]    then spam = 1
```

	0	1
0	0.748	0.252
1	0.103	0.896

Metric	Score
Accuracy	0.822
Precision	0.780
Recall	0.896
Specificity	0.748
F1	0.834

Although the true positive rate scored close to the logistic regression model above, the one rule model significantly underperformed the true negative rate. Specifically, there were a large amount of false positives. This would be problematic in a spam filtering model because legitimate e-mails would end up getting filtered. The AUC for the one rule model was 0.732, as shown in the below ROC curve:



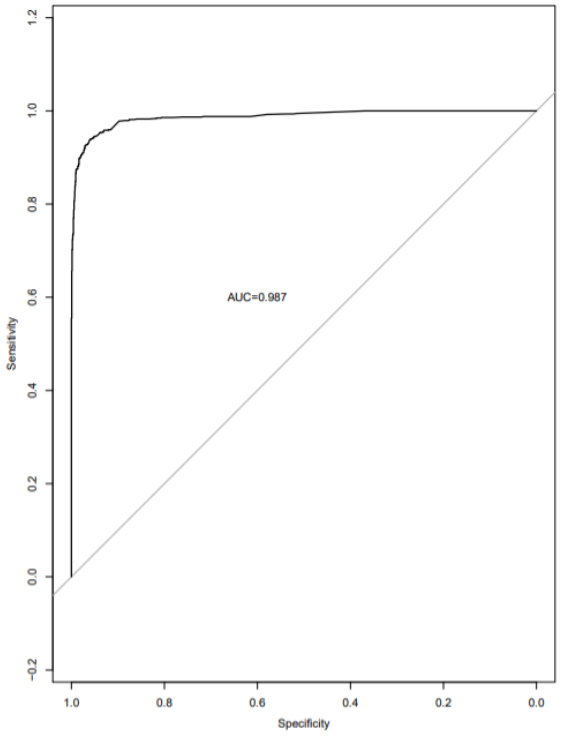
XGBoost

The final model performed for the Computational EDA was an XGBoost model with the following parameters: max_depth = 4, n_rounds = 10. Below are the results:

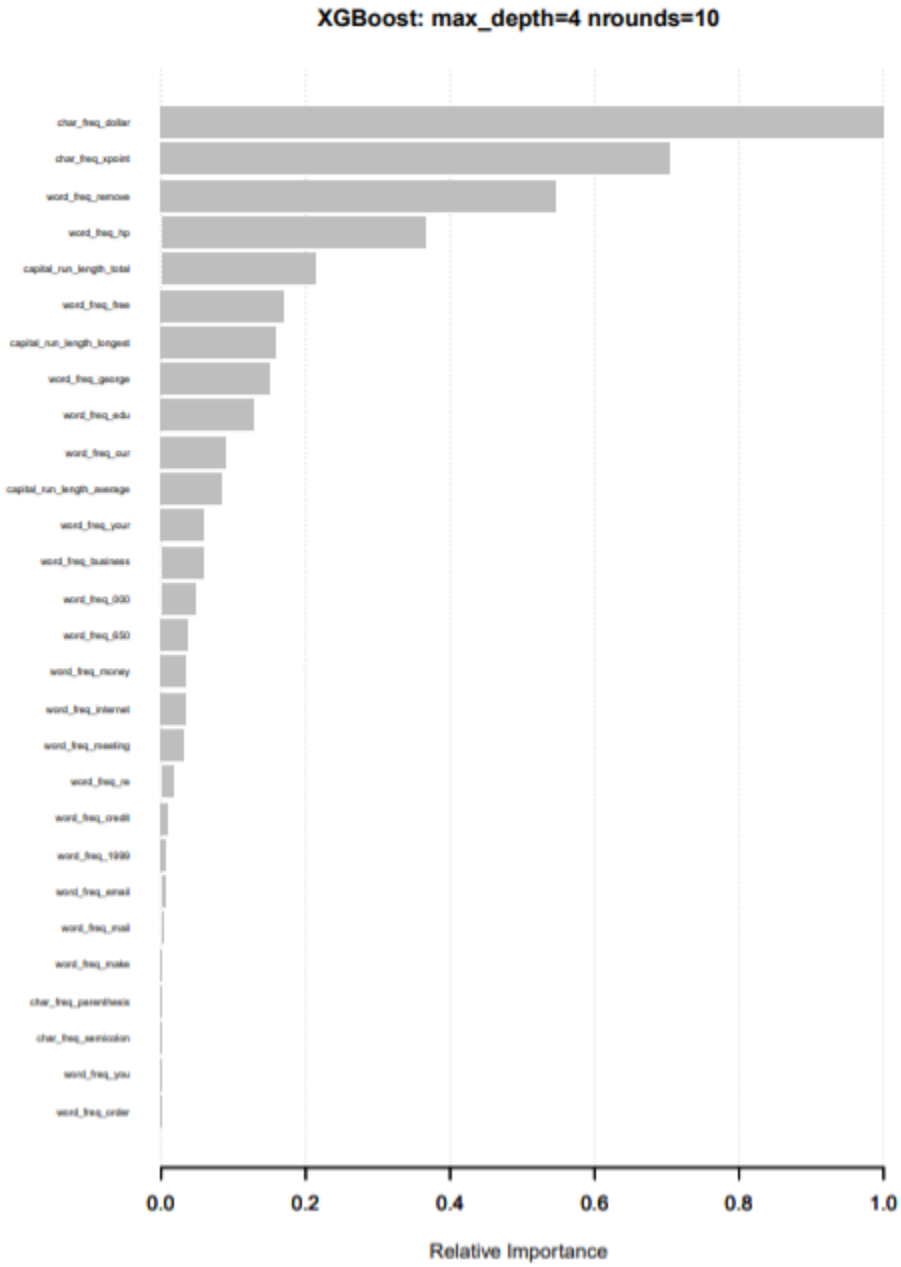
	0	1
0	0.960	0.040
1	0.061	0.939

Metric	Score
Accuracy	0.950
Precision	0.959
Recall	0.939
Specificity	0.960
F1	0.949

With an accuracy score of 0.950, the XGBoost model performed the best of the three models. The true negative rate performed slightly better than the true positive rate. In addition, the AUC for this model was 0.987.



One of the benefits of XGBoost is it provides estimates of variable importance in a trained model. For the next section of this analysis, the Top 10 most important features will be used to build a predictive model. A plot of the relative importance for every variable is show below:

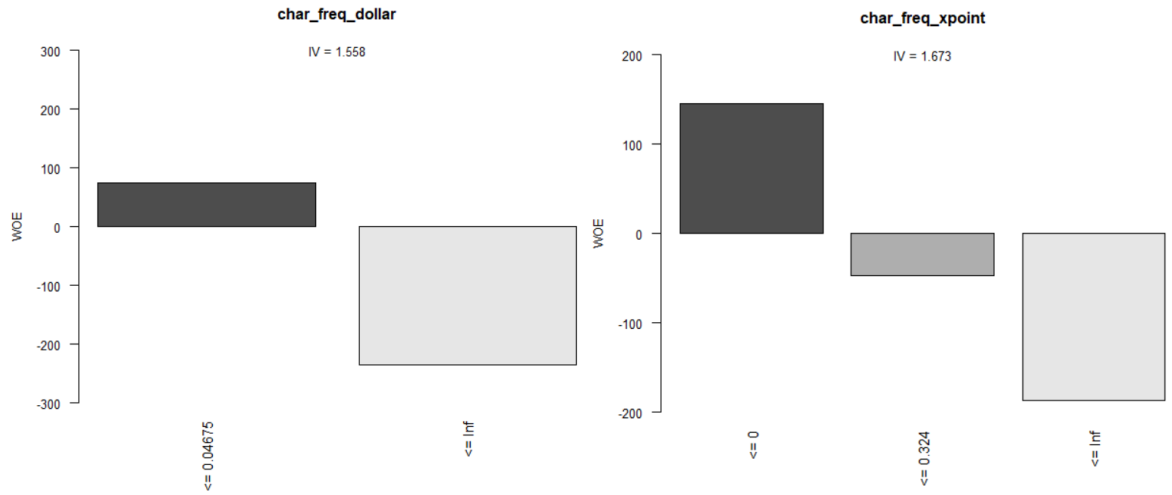


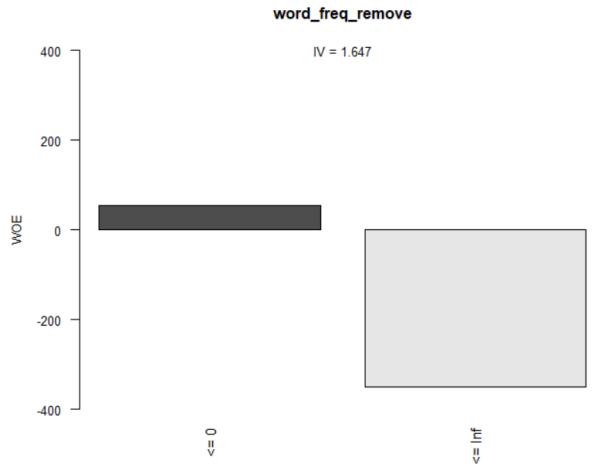
III. Feature Engineering

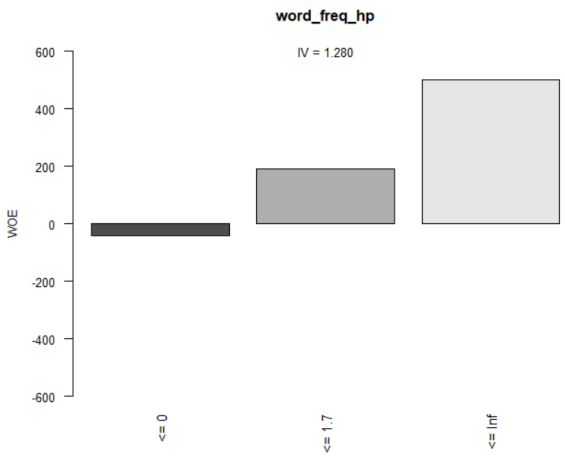
Raw predictor variables do not always have the best predictive value. To improve this, transformations can be performed on these values. WOE transformations work by transforming the predictor variable into a set of bins based on similarity to the target variable distribution. For this analysis, a Weight-of-Evidence transformation will be performed on the Top 10 variables from the XGBoost model shown in the previous section. The variables selected are shown below:

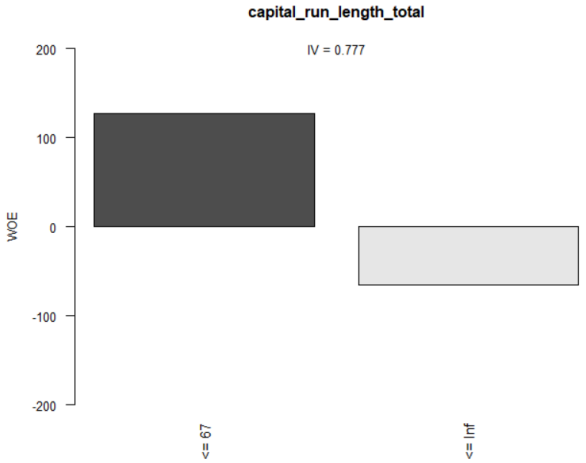
- Target variable: Spam
- Predictor variables, in order of importance:
    1. Char_freq_dollar
    2. Char_freq_xpoint
    3. Word_freq_remove
    4. Word_freq_hp
    5. Capital_run_length_total
    6. Word_freq_free
    7. Capital_run_length_longest
    8. Word_freq_george
    9. Word_freq_edu
    10. Word_freq_our

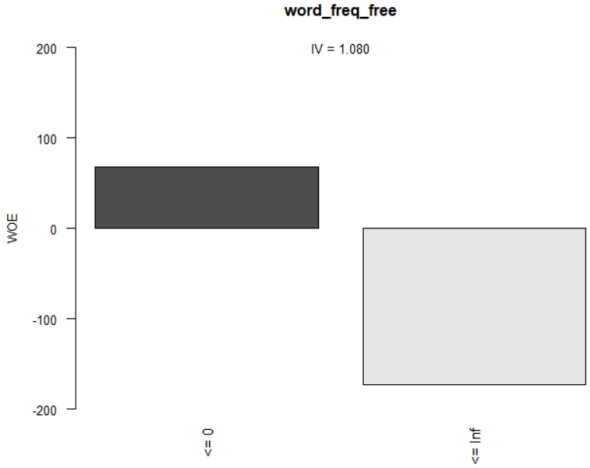
A strong predictor variable will ideally have greater separation between each bin. One way to observe this is by creating a separation plot for the transformed variable. Another way to evaluate the usefulness of a predictor value is through its information value. Variables with information values further away from 0 will be stronger predictors in a model. The separation plots and information value for each variable are shown below:

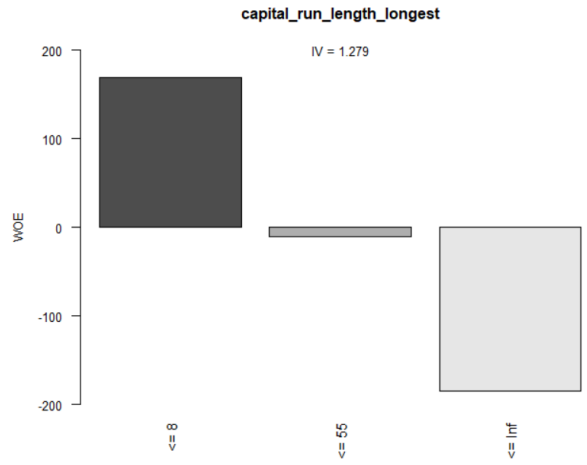


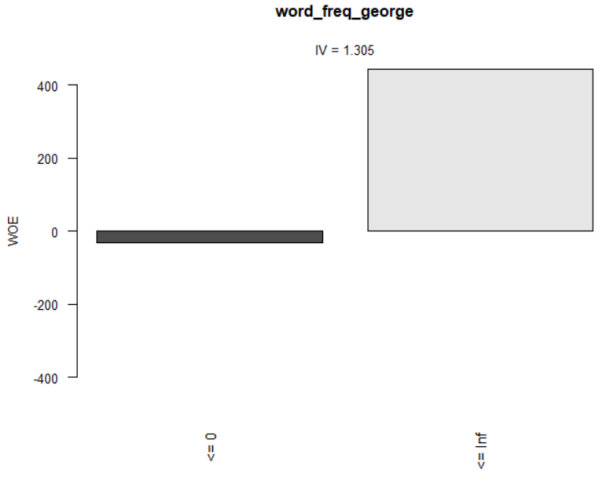

word_freq_remove
IV = 1.647
WOE
400
200
0
-200
-400
<= 0
<= Inf

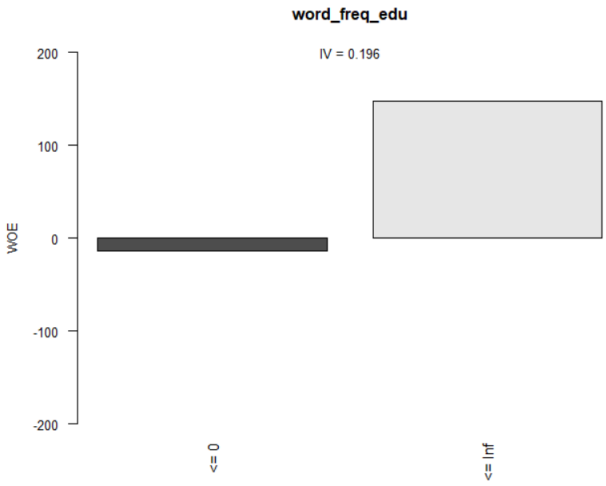

word_freq_hp
IV = 1.280
WOE
600
400
200
0
-200
-400
-600
<= 0
<= 1.7
<= Inf

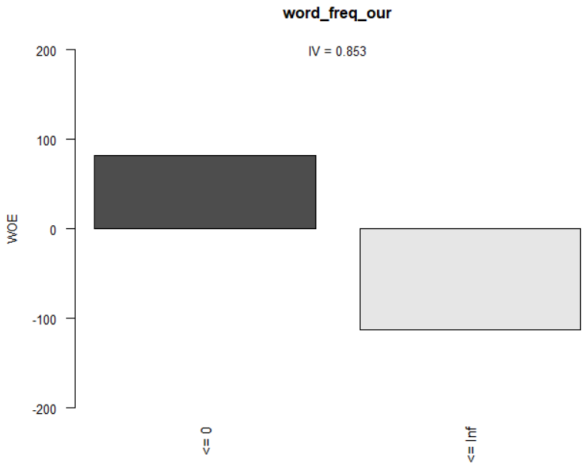

capital_run_length_total
IV = 0.777
WOE
200
100
0
-100
-200
<= 67
<= Inf


word_freq_free
IV = 1.080
WOE
200
100
0
-100
-200
<= 0
<= Inf


capital_run_length_longest
IV = 1.279
WOE
200
100
0
-100
-200
<= 8
<= 55
<= Inf


word_freq_george
IV = 1.305
WOE
400
200
0
-200
-400
<= 0
<= Inf





Variable	Information.Value
char_freq_xpoint	1.673
word_freq_remove	1.647
char_freq_dollar	1.558
word_freq_george	1.305
word_freq_hp	1.280
capital_run_length_longest	1.279
word_freq_free	1.080
word_freq_our	0.853
capital_run_length_total	0.777
word_freq_edu	0.196

One noticable takeaway from transforming the variables is though Char_freq_dollar was the most important variable in the Computational EDA, it is now the third most important. Some bins also showed to be more useful than others within the same variable. For example, in the Word_freq_george variable, e-mails containing the name "George" had a value of 1.217, while those that did not contain the name had a value of 0.088. One possible takeaway is that e-mails containing names could tend to be more personalized, which would suggest they are strong indicators of Not-Spam.

IV. Model Development

Before deciding on a final model, we will build a few different models and compare the results. We will also compare the results to the models from the Computational EDA section. One item we want to observe is how the transformed variables perform compare to their raw values. The models to compare will be One Rule, Logistic Regression, and Naïve Bayes.

One Rule Model with Transformed Variables

The first model to evaluate will be a One Rule model using the transformed variables. Just like in the Computational EDA, the rule selected uses the Char_freq_dollar variable. Consequently, performs similarly to the raw value model.

```
Rules:
If char_freq_dollar.binned = (-Inf,0.04675] then spam = 0
If char_freq_dollar.binned = (0.04675, Inf] then spam = 1
```

	0	1
0	0.766	0.234
1	0.128	0.872

Metric	Score
Accuracy	0.819
Precision	0.788
Recall	0.872
Specificity	0.766
F1	0.828

It should be noted that the benefit of a One Rule model is its simplicity and interpretability. Due to the lack of factors, it is expected that this model will not perform as well as more complex models. Overall, the One Rule model works better as a baseline rather than a final model in production.

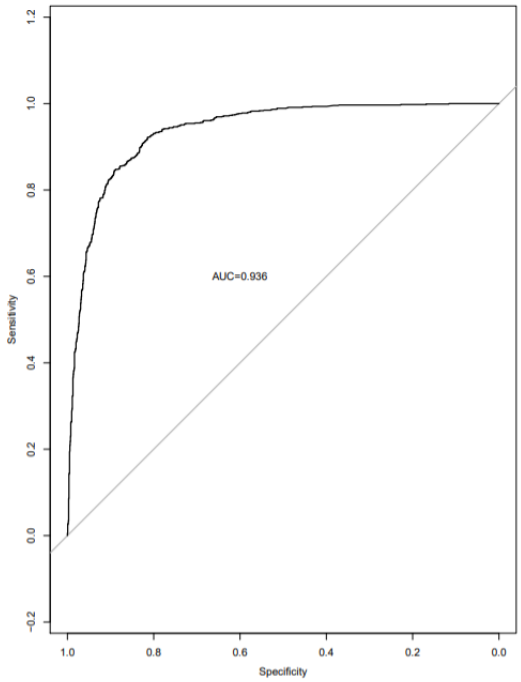
Logistic Regression with Raw Variables

The next model is a standard logistic regression using the top 10 variables from the XGBoost model. The variables in this model are the raw predictor variables.

	0	1
0	0.891	0.109
1	0.154	0.846

Metric	Score
Accuracy	0.869
Precision	0.886
Recall	0.846
Specificity	0.891
F1	0.865

The standard logistic regression model has an accuracy of 0.869. The true negative rate performs better than the true positive rate. The AUC is 0.936 and is illustrated on the below ROC curve:



Comparing to the logistic regression model with backward selection, the top 10 raw variables has a slight decrease in accuracy. The backward selection model had an accuracy of .911 and a F1 score of .910, while the top 10 raw variable model had an accuracy of .869 and F1 score of .865. A more detailed summary for the standard logistic regression model can be found on the next page:

Standard Logistic Regresion

	Dependent variable:
	spam
char_freq_dollar	0.377***
	(0.032)
char_freq_xpoint	0.058***
	(0.008)
word_freq_remove	0.268***
	(0.020)
word_freq_hp	-0.044***
	(0.005)
capital_run_length_total	0.0001***
	(0.00002)
word_freq_free	0.139***
	(0.011)
capital_run_length_longest	0.0004***
	(0.0001)
word_freq_george	-0.014***
	(0.002)
word_freq_edu	-0.042***
	(0.008)
word_freq_our	0.118***
	(0.012)
Constant	0.225***
	(0.011)
Observations	2,318
Log Likelihood	-1,007.316
Akaike Inf. Crit.	2,036.631

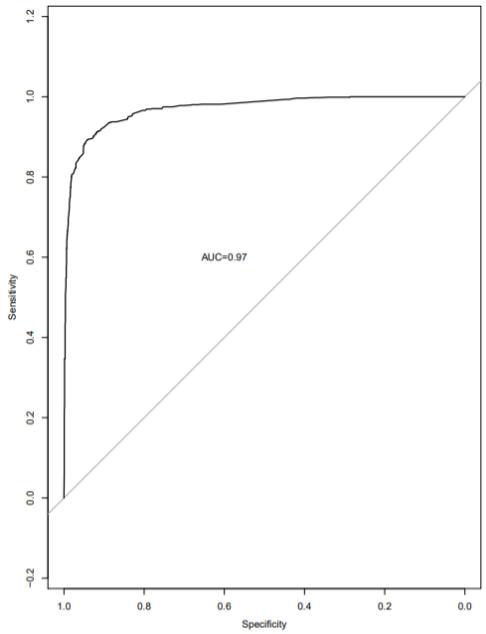
Logistic Regression with Transformed Variables

This next model will test whether the raw values or engineered values work as better predictor variables. This is another standard logistic regression model. However, the predictor variables have been engineered using WOE transformation, as shown in the Feature Engineering section.

	0	1
0	0.941	0.059
1	0.106	0.894

Metric	Score
Accuracy	0.918
Precision	0.938
Recall	0.894
Specificity	0.941
F1	0.916

With an accuracy of 0.918 and F1 score of 0.916, the logistic regression with transformed variables is the best performing logistic regression model. This model also has a strong true negative rate of 0.941. The AUC of the model is 0.970, as illustrated in the ROC curve below:



A more detailed summary for the engineered logistic regression model can be found on the next page:

Logistic Regression with Transformed Variables

	Dependent variable:
	spam
char_freq_xpoint.binned(0,0.324]	0.079***
	(0.016)
char_freq_xpoint.binned(0.324, Inf]	0.223***
	(0.017)
word_freq_remove.binned(0, Inf]	0.214***
	(0.018)
char_freq_dollar.binned(0.04675, Inf]	0.200***
	(0.016)
word_freq_george.binned(0, Inf]	-0.163***
	(0.017)
word_freq_hp.binned(0,1.7]	-0.300***
	(0.019)
word_freq_hp.binned(1.7, Inf]	-0.316***
	(0.022)
capital_run_length_longest.binned(8,55]	0.189***
	(0.018)
capital_run_length_longest.binned(55, Inf]	0.197***
	(0.024)
word_freq_free.binned(0, Inf]	0.115***
	(0.015)
word_freq_our.binned(0, Inf]	0.097***
	(0.014)
capital_run_length_total.binned(67, Inf]	0.074***
	(0.018)
word_freq_edu.binned(0, Inf]	-0.284***
	(0.019)
Constant	0.119***

	(0.012)
Observations	2,318
Log Likelihood	-324.771
Akaike Inf. Crit.	677.542

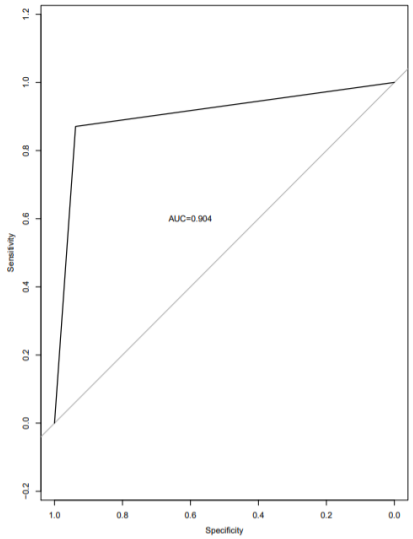
Naïve Bayes

The final model to evaluate will be a Naïve Bayes model. In a naïve bayes classifier, the probability of a sample of a certain is calculated based on prior knowledge. An assumption of the model is that the effect of a feature in independent of the other features. The model created contains the 10 variables that have been engineered using the Weight-of-Evidence transformation.

	0	1
0	0.938	0.062
1	0.129	0.871

Metric	Score
Accuracy	0.905
Precision	0.871
Recall	0.934
Specificity	0.879
F1	0.901

The naïve bayes model had an accuracy of 0.905 and F1 score of 0.901. The model had an AUC of 0.904, as shown in the below ROC curve:



V. Model Comparison

The models we ran in Section 4 have been compared with those in the Computational EDA. The results are below:

In-Sample Model Comparison

Metric	One.Rule.Raw	One.Rule.Eng.	XG.Boost	LR.Backward	LR.Standard	LR.Engineered	Naive.Bayes
Accuracy	0.822	0.819	0.950	0.911	0.869	0.918	0.905
Precision	0.780	0.788	0.959	0.910	0.886	0.938	0.871
Recall	0.896	0.872	0.939	0.911	0.846	0.894	0.934
Specificity	0.748	0.766	0.960	0.910	0.891	0.941	0.879
F1	0.834	0.828	0.949	0.910	0.865	0.916	0.901

Using the in-sample data, the XGBoost model from the Computational EDA section performed the best of all the models. Of the models built in the previous section, the Logistic Regression with Transformed variables was the best performing. One takeaway when comparing each model is that the transformed variables appear to be better predictors than the raw values.

Out-of-Sample Model Comparison

Metric	One.Rule.Raw	One.Rule.Eng.	XG.Boost	LR.Backward	LR.Standard	LR.Engineered	Naive.Bayes
Accuracy	0.824	0.817	0.934	0.899	0.858	0.925	0.911
Precision	0.784	0.790	0.938	0.899	0.884	0.921	0.910
Recall	0.894	0.862	0.928	0.899	0.824	0.930	0.911
Specificity	0.754	0.771	0.939	0.899	0.892	0.920	0.910
F1	0.836	0.824	0.933	0.899	0.853	0.925	0.911

The XGBoost model performs best on the out-of-sample data with the Engineered Logistic Regression as the next best. The Backward Logistic Regression decreased in accuracy out-of-sample. Considering the performance between the Engineered and Backward Logistic Regressions was so close with the in-sample data, this suggests that the transformed variables should generalize better in a production model.

VI. Model Recommendation

XGBoost vs. Engineered Logistic Regression

Metric	XG.Train	XG.Test	Logistic.Train	Logistic.Test
Accuracy	0.950	0.934	0.918	0.925
Precision	0.959	0.938	0.938	0.921
Recall	0.939	0.928	0.894	0.930
Specificity	0.960	0.939	0.941	0.920
F1	0.949	0.933	0.916	0.925

One drawback of XGBoost is that since it is an ensemble method, the model cannot handle categorical data. However, the XGBoost model performed at the beginning of the analysis still managed to outperform the other models. The decrease in accuracy of the XGBoost model on the out-of-sample data suggests that it may be overfit. Of the models we created, the Logistic Regression with Transformed Variables was the best performing. The better performance on the out-of-sample data suggests the model may be underfit. As a result, this model could be improved with more feature analysis.

In choosing to deploy a model to production, I would recommend either of these models. However, my preference would be for the Logistic Regression model due to it generalizing better between both samples of data.