

# Classifying URLs

## A Machine Learning Approach Milestone Report II

Springboard Capstone 2 Project  
Author: Helga Wilde

This report contains information on decisions and steps taken to apply machine learning techniques for the URL Classification project.

### Milestone I Report Overview

**Project Objective:** A machine learning model with superior accuracy in classifying urls into one of three classes: benign, phishing or malicious.

**Project Approach:** The machine learning model is trained with features that rely mainly on url strings and limited third party data.

**Training & Testing Data:** Verified phishing and malicious urls were retrieved from PhishTank<sup>1</sup> and the URLHAUS project<sup>2</sup> from abuse.ch. Benign urls were collected by crawling Alexa's top 2500 websites<sup>3</sup>. Each 'benign' url's reputation was checked via VirusTotal's reputation service<sup>4</sup>. Virus scans were initiated when necessary.

**Feature Creation:** Two sets of features were created, trained and tested. The intent is to determine which feature set is the best for training a url classification model.

Feature Set 1 contains 96 predictor features created from lexical properties of the url and its segments. One predictor relies on third party data and signifies whether a url's domain is in the Alexa top 500 list.

Feature Set 2 contains records consisting of each url's tokens so that natural language processing (NLP) techniques can be applied during machine learning.

**Exploratory Data Analysis:** For feature set 1, exploratory data analysis and statistical techniques were used to investigate, analyze and summarize characteristics of the url strings and their components. Pairwise correlation and predictor versus target correlation analysis was completed.

---

<sup>1</sup> <http://phishtank.org/index.php>

<sup>2</sup> <https://urlhaus.abuse.ch/>

<sup>3</sup> <https://www.alexa.com/topsites>

<sup>4</sup> <https://developers.virustotal.com/v3.0/reference>

## New Project Updates

### Datasets

The machine learning models leverage an equal proportion of benign, malicious and phishing urls – 10,000 records of each class. To achieve this balance, it was necessary to reduce the number of benign urls. This was done by sampling benign urls from our initial dataset.

### Model Evaluation Metrics

Metrics were gathered for our multi-class classification models and some are documented in this report.

Here is a brief review of potential url classification outcomes:

Classification Outcome	Description
True Positive (TP)	Model correctly predicts the positive class. URLs are correctly classified. E.g. A truly benign url is classified as benign.
False Positive (FP)	The model incorrectly predicts the positive class. URLs are incorrectly classified. E.g. An actual malicious url is misclassified as benign.
True Negative (TN)	The model correctly predicts the negative class. URLs are correctly classified. E.g. An actual malicious url is not classified as benign.
False Negative (FN)	The model incorrectly predicts the negative class. URLs are incorrectly classified. E.g. An actual malicious url is not classified as malicious.

Scores in our reporting:

#### ***Log Loss***

An important classification metric based on probabilities. Log Loss quantifies the accuracy of a classifier by penalizing false classifications. A good metric for comparing models, with lower log loss values meaning better predictions.

#### ***Accuracy***

Fraction of the total samples that were correctly classified as benign, phishing or malicious.  
Overall accuracy of the model.

#### ***Classification Report***

Precision: The fraction of predictions as a positive class that were actually positive ( $TP/(TP + FP)$ ).

Recall: The True Positive Rate: the fraction of all positive samples that were correctly predicted as positive by the classifier ( $TP/(TP + FN)$ )

F1: A weighted averaged of the precision and recall scores, where an F1 score of 1 means it is 100% accurate. ( $2TP/(2TP + FP + FN)$ )

Note: Our primary goal is to accurately classify malicious or phishing url links, attaining high true positive rates and low false negative rates. Therefore, recall scores for phishing and malicious urls are highly valuable.

### Confusion Matrix

Confusion matrices allow us to visualize TP / TN / FP / FN scores for each class. Confusion matrices for multi-class problems are not straight-forward. For example, in the matrix below<sup>5</sup>, True positive classifications scores for malicious urls are obvious. Other measurements like TN, FP, FN can be derived with manual calculations. One can also rely on precision, recall and F1 scoring, which are based on these TP / TN / FP / FN prediction results.

		PREDICTED		
		Benign	Phishing	Malicious
ACTUAL	Benign	TN	TN	FP
	Phishing	TN	TN	FP
	Malicious	FN	FN	TP

Chart 1: Confusion Matrix Guide for Malicious URL Classification

### Feature Set 1 Machine Learning Model

Feature set 1 contains 96 predictor features created from lexical properties of the entire url string or its components, such as scheme, netloc, domain, path, fragment, parameters and queries.

### Baseline Models

Supervised learning algorithms capable of multi-class classification were explored. Prior to any pre-processing or normalization, several classifiers were trained on the feature set. The RandomForestClassifier achieved the best overall accuracy score and TP scores<sup>6</sup>.

			Benign	Phishing	Malicious
	Accuracy	Log Loss	Confusion Matrix - TP Scores		
KneighborsClassifier	0.852	1.587	0.932	0.817	0.807
DecisionTreeClassifier	0.885	3.961	0.936	0.867	0.853
RandomForestClassifier	0.928	0.222	0.974	0.903	0.906
AdaBoostClassifier	0.843	1.016	0.907	0.822	0.799
GradientBoostingClassifier	0.897	0.297	0.950	0.885	0.856
XGBClassifier	0.883	0.330	0.951	0.860	0.839

Table 1: Overview of Baseline Scores

<sup>5</sup> Chart 1: Confusion Matrix Guide for Malicious URL Classification

<sup>6</sup> Table 1: Overview of Baseline Scores

## Random Forest Classifier

Based on the baseline scores, the Random Forest Classifier was selected as the algorithm to develop for this model.

### Step 1. Normalization of features

I adopted MinMaxScaler for normalization, constraining the range of all numeric values between 0 and 1. Since we bound the range from 0 to 1, we will suppress the effect of outliers.

### Step 2. Parameter Tuning

I utilized GridSearchCV, which completes a thorough check for the parameter set that returns the best accuracy score. It does this by using all possible parameter combinations. I settled on these parameter settings:

1. Max depth of the tree = 30
2. Max features to consider when looking for the best split = 10
3. Min samples required to be at a leaf node = 1
4. Min samples split required to split an internal node = 2
5. Number of estimators (number of trees in the forest) = 800

### Step 3. Validation of Scores

In the previous models, we split the dataset into 80/20 subsets, 80% used for training the model, 20% for testing. To ensure that our newly-tuned model will return similar results in real life applications (with new urls), a validation test is run.

StratifiedKFold was selected as the validation technique. It is a variation of KFold that preserves the percentage of samples for each class. Both the train and test partitions have equal portions of benign, phishing and malicious urls. Just like KFold, the validation test trains/tests on alternating sets of data. Accuracy scores are taken following each test.

The mean accuracy score following our StratifiedKFold cross-validation with 5 splits is 93.14, with a standard deviation of .0035. So, our accuracy scores are pretty solid.

### Step 4. Prediction and Final Scores

Our final step is to split our dataset 80/20, fit and predict on the model, evaluate our scores and feature importance. As indicated in Table 2, our final model scores are not higher across the board, but close to initial baseline and other models.<sup>7</sup>

---

<sup>7</sup> Table 2: Random Forest Classifier Score Review

RandomForestClassifier	Accuracy	Log Loss	Benign	Phishing	Malicious
			Confusion Matrix – TP scores		
Baseline Scores	0.928	0.222	0.974	0.903	0.906
Baseline w/ MinMaxScaler	0.932	0.205	0.971	0.907	0.919
Predict after Tuning	0.931		0.973	0.905	0.916

Table 2: Random Forest Classifier Score Review

Our final Random Forest Classification report shows some variability in our model's classification accuracy for benign versus phishing and malicious urls.

#### Classification Report

	precision	recall	f1-score	support
Benign	0.9358	0.9727	0.9539	1979
Phishing	0.9429	0.9055	0.9238	1989
Malicious	0.9154	0.9158	0.9156	2032
accuracy			0.9312	6000
macro avg	0.9314	0.9313	0.9311	6000
weighted avg	0.9313	0.9312	0.9310	6000

Domain and netloc features predominate in feature importance scores for our Random Forest Classifier<sup>8</sup>.

---

<sup>8</sup> Table 3: Feature Importance Scores; Chart 2: Feature Importance Scores

Feature	Score
n_netloc_let	0.0489
avg_netloc_tok_len	0.0409
pc_netloc_let	0.0338
n_domain_let	0.0336
len_netloc	0.0306
netloc_entropy	0.0290
pc_domain_let	0.0278
pc_domain_spec	0.0236
pc_path_spec	0.0217
longest_path_len	0.0217
pc_netloc_num	0.0202
avg_path_token_len	0.0192
avg_domain_tok_len	0.0189
n_domain_dots	0.0182
n_let	0.0178
entropy	0.0175
n_domain_tok	0.0174
pc_let	0.0173
pc_path_uppercase	0.0173
domain_entropy	0.0170

Table 3: Feature Importance Scores

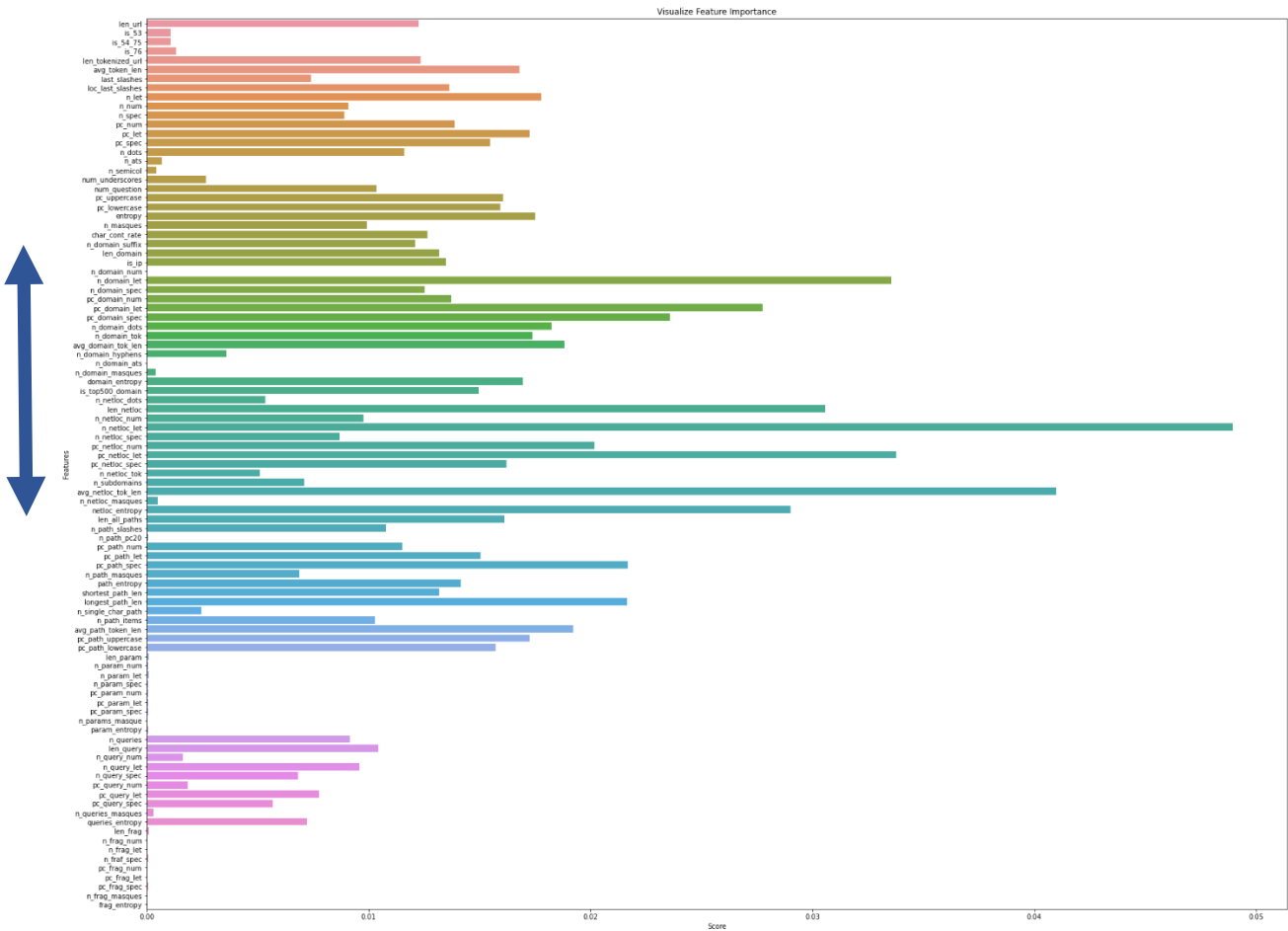


Chart 2: Feature Importance Scores

In order to improve the efficiency of our model and remove some redundancy in our dataset (as indicated with the feature importance scores), I decided to make alterations to the dataset. Details follow.

#### Machine Model with Reduced Dataset

As stated previously, netloc and domain features dominated the top 20 features of importance in our Random Forest Classifier. I decided to train models with a reduced feature set. Using the feature importance list and the pairwise correlation data, I carefully selected a set of twenty-two predictor variables<sup>9</sup>, that covered the essential url characteristics. I then completed the same machine learning workflow steps, as with the full dataset.

Selected Predictor Features	
pc_num	len_netloc
pc_uppercase	n_netloc_tok
pc_spec	pc_netloc_spec
entropy	pc_netloc_num
len_domain	netloc_entropy
is_top500_domain	path_entropy
pc_domain_spec	avg_path_token_len
pc_domain_num	longest_path_len
n_domain_suffix	pc_path_uppercase
n_domain_tok	len_all_paths
domain_entropy	pc_path_num

Table 4: Reduced Feature Set

As the following results show<sup>10</sup>, I did not achieve higher scores for baseline models and the final Random Forest Classifier model<sup>11</sup>. Multicollinearity does not negatively affect prediction accuracy.

			Benign	Phishing	Malicious
	Accuracy	Log Loss	Confusion Matrix - TP Scores		
<b>KneighborsClassifier</b>	81.813	1.874	0.88	0.79	0.78
<b>DecisionTreeClassifier</b>	87.453	4.333	0.91	0.87	0.84
<b>RandomForestClassifier</b>	92.147	0.247	0.97	0.89	0.91
<b>AdaBoostClassifier</b>	82.080	1.016	0.89	0.78	0.79
<b>GradientBoostingClassifier</b>	87.586	0.342	0.94	0.84	0.85
<b>XGBClassifier</b>	86.760	0.367	0.94	0.83	0.84

Table 5: Baseline Classifier Scores with Reduced Feature Set

<sup>9</sup> Table 4: Reduced Feature Set

<sup>10</sup> Table 5: Baseline Classifier Scores with Reduced Feature Set

<sup>11</sup> Table 6: Random Forest Classifier scores

RandomForestClassifier	Accuracy	Log Loss	Benign	Phishing	Malicious
			Confusion Matrix – TP Scores		
Baseline	0.920	0.245	0.964	0.884	0.911
Baseline w/ MinMaxScaler	0.928	0.222	0.966	0.897	0.920
Predict	0.922		0.967	0.885	0.914

Table 6: Random Forest Classifier Scores, Reduced Feature Set

The tables below compare feature importance scores for both the full and reduced feature sets<sup>12</sup>. Multicollinearity may affect the ability to interpret the parameters learned by our models. We cannot say that the features with the largest weights are the most important when features are highly correlated with each other.

The 96 predictor feature set's top scorers stem from the domain/netloc section of the url. After reducing the feature set to 22 predictors, we still have a high proportion of domain/netloc features, but new features appear and features like entropy and the percent of special characters move up significantly in the importance list.

96 Predictors	Score	22 Predictors	Score
n_netloc_let	0.0489	len_netloc	0.0740
avg_netloc_tok_len	0.0409	pc_domain_spec	0.0677
pc_netloc_let	0.0338	entropy	0.0639
n_domain_let	0.0336	pc_spec	0.0605
len_netloc	0.0306	netloc_entropy	0.0602
netloc_entropy	0.0290	pc_netloc_spec	0.0522
pc_domain_let	0.0278	longest_path_tok	0.0495
pc_domain_spec	0.0236	n_domain_tok	0.0491
pc_path_spec	0.0217	pc_netloc_num	0.0490
longest_path_len	0.0217	avg_path_token_len	0.0485
pc_netloc_num	0.0202	len_all_paths	0.0459
avg_path_token_len	0.0192	pc_num	0.0458
avg_domain_tok_len	0.0189	pc_uppercase	0.0416
n_domain_dots	0.0182	domain_entropy	0.0401
n_let	0.0178	n_domain_suffix	0.0392
entropy	0.0175	pc_path_uppercase	0.0369
n_domain_tok	0.0174	path_entropy	0.0362
pc_let	0.0173	len_domain	0.0333
pc_path_uppercase	0.0173	pc_domain_num	0.0323
domain_entropy	0.0170	pc_path_num	0.0306


 Indicates feature not represented in reduced feature set

Table 7: Feature Importance Comparison

<sup>12</sup> Table 7: Feature Importance Comparison



## Feature Set 2 Machine Learning Model

For this approach, natural language processing (NLP) techniques were used to train and test our models.

### Feature Creation

For NLP we require a corpus, a collection of texts. For this url classification project, the corpus consists of url tokens. We start with just a dataframe of 30,000 url strings, each classified with a benign, phishing or malicious label. With the help of the `nlk.tokenize.WordPunctTokenizer()` method, tokens are extracted from each url string and stored in a new 'tokenized\_url' feature. For example, applying the `WordPunctTokenizer`<sup>13</sup> method on <https://www.google.com> returns the following tokens: `https`, `://`, `www`, `.`, `google`, `.`, `com`.

### Vectorization & Transformation

Our url tokens need to be transformed into vector representations since machine learning algorithms require a numeric feature set. Our url token lists are of varying length, but after the numeric transformation step (vectorization), our vector representations will be of uniform length.

There are several vectorization methods and tools to perform the transformation. For this project we leverage TF-IDF and utilize scikit-learn's `TfidfVectorizer` to do the conversion and return a matrix of TF-IDF features.

#### TF-IDF

TF-IDF stands for Term-Frequency-Inverse Document Frequency. With this approach each url token is assigned a number that is proportional to its frequency in the url string and inversely proportional to the number of url strings in which it occurs.

Other approaches exist, like bag-of-words representations. But they only take individual documents (urls, in our case) into consideration, and not the context of the entire corpus.

Formulas:

$N$  - number of urls we have in our dataset

$d$  - a given url from the dataset

$D$  - the collection of all urls

$W$  - a given token from a url

$f(w,d)$  - frequency of word  $w$  in document  $d$

---

<sup>13</sup> <https://www.nltk.org/api/nltk.tokenize.html#nltk.tokenize.regexp.WordPunctTokenizer>

### Term Frequency (TF) Formula

$$tf(w,d) = \log(1+f(w,d))$$

### Inverse Document Frequency (IDF) Formula

$$idf(w, D) = \log(N/f(w,D))$$

### TF-IDF formula

$$Tfidf(w,d,D) = tf(w,d)*idf(w,D)$$

Following TF-IDF scoring of our url records, we are left with a matrix of td-idf scores with one row per url, and as many columns as there are different tokens in the entire corpus.

#### [Scikit\\_learn TfidfVectorizer](#)

TF-IDF scores may be calculated manually, but our models utilize Scikit-Learn's transformer, TfidfVectorizer, which uses an estimator to count the occurrences of tokens (TF), followed by TfidfTransformer which normalizes these counts by the inverse document frequency(IDF). The vectorizer returns a sparse matrix representation in the form of ((doc, term), tfidf score) where each key is a document and term pair.

#### Baseline Models

Our matrix is finally split into train and test subsets and multiple algorithms are fitted to gather baseline accuracy scores and confusion matrices. The LinearSVC model achieved the best overall accuracy score and true-positive rates (with the exception of TP scores for malicious urls)<sup>14</sup>.

			Benign	Phishing	Malicious
	Accuracy	Log Loss	Confusion Matrix - TP Scores		
Logistic Regression	94.8830	0.2222	0.9735	0.9478	0.9250
LinearSVC	96.2330		0.9805	0.9684	0.9380
MultinomialNB	93.2000	0.2573	0.9825	0.9188	0.8949
Random Forest	95.1500	0.1754	0.9705	0.9363	0.9475

Table 8: Baseline Classifier Scores

<sup>14</sup> Table 8: Baseline Classifier Scores

Since these baseline scores are higher than our model using a 96 predictor feature set, we continue to develop the top 3 classifiers - LinearSVC, Random Forest Classifier and Logistic Regression.

#### Random Forest Classifier – LinearSVC – Logistic Regression

For each of these classifiers, the following steps were visited:

##### Step 1. Normalization of features

No further steps required.

##### Step 2. Parameter Tuning

I utilized GridSearchCV to test all parameter combinations and provide the best parameters

The following parameters<sup>15</sup> were returned:

LinearSVC	Logistic Regression	Random Forest Classifier
C = 1	C = 100	max depth = None
penalty = l2		max features = 30
		min samples leaf = 1
		number of estimators = 1000

Table 9: Best Parameters per Classifier

##### Step 3. Validation of Scores

StratifiedKfold was selected as the validation technique, to preserve the percentage of samples for each class. For each algorithm, our accuracy scores remained consistent across all folds. The mean accuracy scores and standard deviation following StratifiedKfold cross-validation with 5 splits<sup>16</sup>:

	Mean Accuracy	Standard Deviation
LinearSVC	96.31	0.0015
Logistic Regression	96.21	0.0021
Random Forest Classifier	96.21	0.0021

Table 10: StratifiedKfold Results

##### Step 4. Prediction and Final Scores

Our final step was to split our dataset 80/20, fit and predict on each model and evaluate our scores.

Results<sup>17</sup>:

- LinearSVC and Random Forest Classifier tie on the overall accuracy score
- LinearSVC has the highest True Positive detection rate for benign and phishing urls
- Random Forest Classifier has the best True Positive scoring for malicious urls

---

<sup>15</sup> Table 9: Classifier Best Parameters

<sup>16</sup> Table 10: StratifiedKfold Results

<sup>17</sup> Table 11: Classifier Final Results

		Benign	Phishing	Malicious
	Accuracy	Confusion Matrix - TP Scores		
Logistic Regression	94.8800	0.9735	0.9479	0.9250
LinearSVC	96.2500	0.9835	0.9649	0.9390
Random Forest	96.2500	0.9765	0.9555	0.9555

Table 11: Classifier Final Results

Our models' Classification Reports<sup>18</sup> cover the following:

Precision	TP / (TP + FP)	The fraction of predictions as a positive class that were actually positive.
Recall	TP / (TP + FN)	The fraction of all positive samples that were correctly predicted as positive by the classifier.
F1	2TP / (2TP + FP + FN)	A combined measure of both precision and recall.

	LinearSVC		
	precision	recall	f1-score
benign	0.9825	0.9835	0.9830
phishing	0.9286	0.9649	0.9464
malicious	0.9781	0.9391	0.9582

	Random Forest		
	precision	recall	f1-score
benign	0.9839	0.9765	0.9802
phishing	0.9371	0.9554	0.9462
malicious	0.9671	0.9555	0.9613

	Logistic Regression		
	precision	recall	f1-score
benign	0.9659	0.9735	0.9697
phishing	0.9136	0.9479	0.9304
malicious	0.9691	0.9251	0.9466

Table 18: Classification Reports for All Models

#### Best Recall Scores:

The Random Forest Classifier model will accurately classify 95.55% of actual malicious urls as malicious.

The LinearSVC model will accurately classify 98.35% of actual benign urls as benign, and 96.49% of actual phishing urls as phishing.

<sup>18</sup> Table 18: Classification Reports for All Models

## Machine Learning Summary

While both features sets produced scores of 90% or higher, the feature set leveraging natural language processing achieved the best scores<sup>19</sup>. LinearSVC and RandomForest Classifiers trained on matrices with url token TF-IDF scores tied with an overall accuracy score of 96.25.

Focusing on recall scores, the Random Forest Classifier model returned the highest score for predicting positive malicious urls, while the LinearSVC model returned the highest recall scores for benign and phishing urls.

	Features	Accuracy	Benign	Phishing	Malicious
			Recall Scores		
<b>RandomForestClassifier</b>	Lexical	93.3310	0.9727	0.9055	0.9158
<b>LinearSVC</b>	TF-IDF	96.2500	0.9835	0.9649	0.9391
<b>RandomForestClassifier</b>	TF-IDF	96.2500	0.9765	0.9554	0.9555
<b>LogisticRegression</b>	TF-IDF	94.8800	0.9735	0.9479	0.9251

Table 19: Overview of Models: Accuracy & Recall Scores

It's important to note the lightweight nature of both models, which leverage url strings as the basis for feature development.

## Next Steps

Additional steps, such as stacking multiple classifiers, may be taken to improve model performance.

---

<sup>19</sup> Table 19: Overview of Models: Accuracy & Recall Scores