Identification of Phishing Websites

CS-UY 4563: Machine Learning

Professor Linda N. Sellie

Final Project Report

April 27, 2023

Andy Chen, Rakeeb Hossain

Introduction

Phishing Websites are a form of fraud that involves an attacker disguising a webpage as a credible entity to trick a victim into supplying sensitive information. In the technological age, this attack has become increasingly common, and identifying phishing websites is important for our cybersecurity.

The “PhishingWebsites” dataset from OpenML is a reliable public training dataset that covers all the important features that have been proven to be effective in predicting phishing websites as well as some additional features. The dataset is a large one with 11055 different instances. The target of this dataset is 2 distinct binary values representing ‘true’ or ‘false’ in regards to whether the instance is a phishing website or not. There are 31 original features to this dataset, some of which include: URL length, port, age of the domain, has an IP address, and more.

The goal of identifying phishing websites is a binary classification problem. This report will cover three main machine learning models of Logistic Regression, Support Vector Machines (SVM), and Neural Networks to train models to classify phishing websites accurately. In addition, this report will also cover various feature transformations, regularization techniques, and modifications to the dataset in an attempt to improve the accuracy of each of the models.

Preprocessing

There was only a small amount of preprocessing required to prepare this dataset for modeling. One of the original worries we had when selecting this dataset was that the database would be unbalanced with only a small percentage of the dataset being classified as a phishing website. Fortunately, this proved to be untrue as the following distribution plot shows that the number of classified phishing websites (1) was greater than the number of websites classified as not phishing (-1).

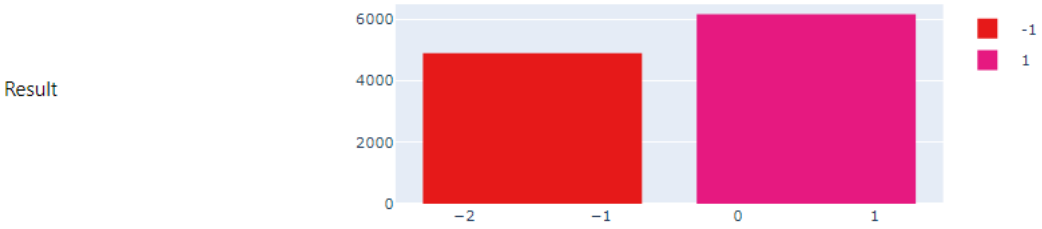

Figure 1: Phishing Dataset Results Plot (Source: OpenML)

The preprocessing we did do was removing unnecessary features, such as a feature named ‘id’, which acted as a label number for all the examples. Since this dataset had a large number of features originally (31 to be exact), we also felt that to help prevent overfitting we should remove some additional features. We removed four more features we deemed unnecessary, which we determined using the following OpenML RandomForest Feature importance graph of the dataset.

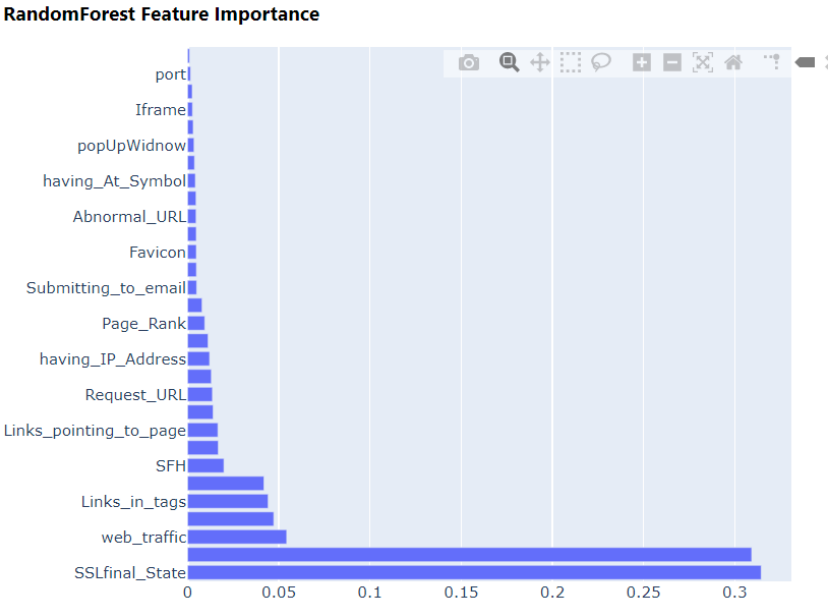

Figure 2: (Source: OpenML Phishing Dataset Analytics)

The data were then scaled using sklearn’s StandardScaler() function, which standardized the data by removing the mean and scaling to unit variance. This scaling was necessary to reduce variance and meet the requirements to correctly use Sklearn’s learning algorithms. Finally, sklearn’s train_test_split() function was used to split the dataset into a training set and a test set. The training set made up 80% of the data and the rest of the data was used in the validation set.

Logistic Regression

Overview

The first model we used for the prediction of phishing attacks is Logistic Regression. The goal was to find the best hyperparameters of the model to achieve the highest accuracy score on the test data. Below are the steps that were taken to find the most favorable hyperparameters:

1. Create polynomial transformations on the training data without any regularization techniques.
2. Both L1 and L2 regularization techniques will be used with various lambda values.

Polynomial Transformation

To perform a polynomial transformation, we used Sklearn's PolynomialFeatures to generate a new features matrix. To find which transformation was best, we utilized K-Fold Cross Validation with K = 5. The degrees that were experimented with were 1, 2, and 3. Anything above 3 led to extremely long training times and training accuracy was starting to increase. This is expected, as having higher degrees of polynomial transformation may lead to the model fitting to the noise.

Below is a graph of the results:

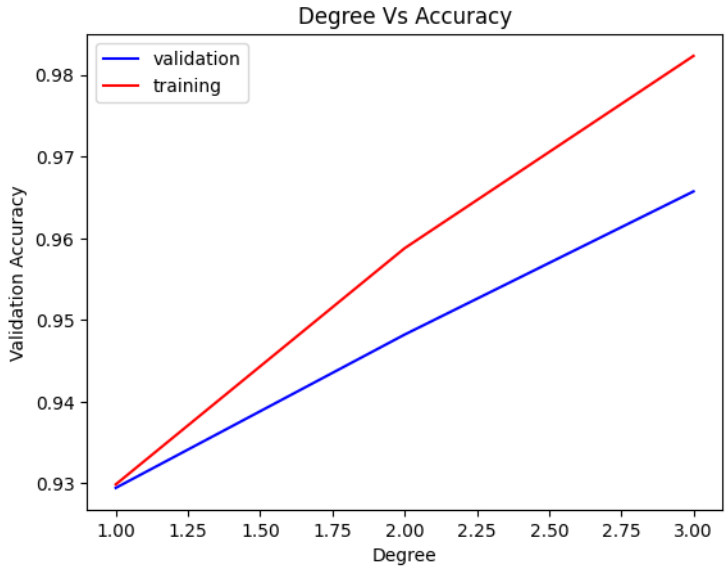

Figure 3: Logistic Regression Degree vs. Accuracy

Table 1: Logistic Regression Degree vs. Accuracy

Polynomial Degrees	1	2	3
Train	0.929867701901815	0.958785638808933	0.982332674825769
Validation	0.929443482397959	0.9482139614118464	0.965739968640411

Regularization

We continued with a polynomial transformation of degree 3 for regularization to try and see if we can prevent overfitting with any of the regularization techniques. K-Fold Cross Validation is still utilized here to find the best lambda values. We kept K = 5 as a fixed value and tested three lambda values. The accuracy for both training and validation did not change much. Below are the results for both L1 and L2 regularization.

L1 Regularization

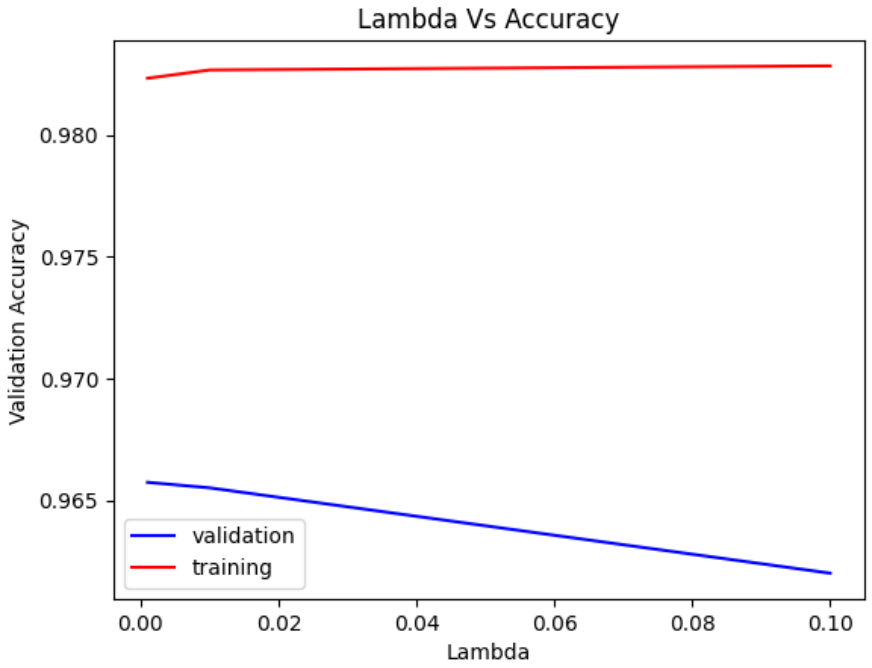

Figure 4: Logistic Regression L1 Lambda vs. Accuracy

Table 2: Logistic Regression L1 Lambda vs. Accuracy

Lambda (L1)	0.001	0.01	0.1
Validation	0.965739968640411	0.965513724296519	0.962008599587159
Training	0.982332674825769	0.982671877465658	0.982841488773079

L2 Regularization

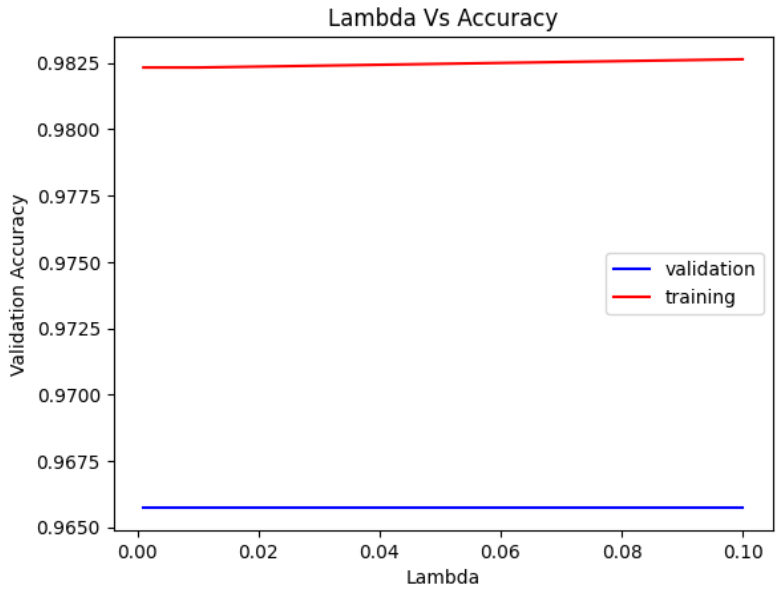

Figure 5: Logistic Regression L2 Lambda vs. Accuracy

Table 3: Logistic Regression L2 Lambda vs. Accuracy

Lambda (L2)	0.001	0.01	0.1
Validation	0.965739968640411	0.965739968640411	0.965739968640411
Training	0.982332674825769	0.982332674825769	0.982643608914421

Analysis

The model with no polynomial transformation and regularization started with an accuracy of ~92% on the test data. As we transformed the features with varying degrees, validation, and

training accuracy improved. However, the higher the degree, the training accuracy started to be significantly higher than the validation accuracy. This is expected, as higher degrees of transformation can lead to the model overfitting and adapting to the noise due to the complexity of the features. The variance was increasing as the degree increased, although bias seemed to remain low since the model had a high accuracy in predicting the validation set. We utilized regularization techniques afterward to see if we can reduce overfitting and converge the training and validation accuracy. However, using regularization did not alter the accuracy of both training and validation sets. This can mean the model did not overfit, but this may also not be the case. Using third-degree polynomial transformation on the features and L2 regularization to fit our model, we obtained the highest accuracy of 96.11% on the test data. Calculating the final model's F1 score, we obtained a high score of 0.9651, which suggests the model performed quite well in minimizing false predictions. The precision is 0.951239 and the recall is 0.979423. Below is the confusion matrix.

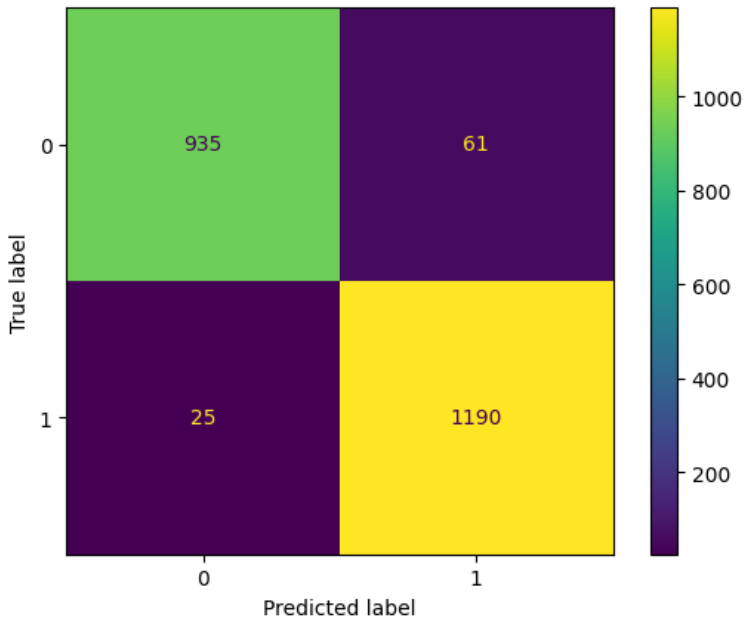

Figure 6 Logistic Regression Confusion Matrix

Support Vector Machine

Overview

The second model we experimented with on our dataset was Support Vector Machines (SVMs). The goal of Support Vector Machines is to find a hyperplane that distinctly separates the classes of data with the maximum margin. The best hyperplane that gets selected is based on a variety of hyperparameters that can be changed. We implemented SVMs using Sklearn's svm library, which allowed us to experiment with L2 regularization parameters and different kernel functions. To achieve the best SVM model, these are the parameters we experimented with

1. The Linear kernel function. We experimented with different values of the L2 regression ([.00001,.0001,.001,.01,.1,1,10,100]).
2. The Polynomial kernel function. In addition to experimenting with the L2 regression values the same way above, we also experimented with tuning the polynomial's degrees of freedom.
3. The Radial Based kernel function. Similarly to the other models, we experimented with the L2 regression values using the same values of [.00001,.0001,.001,.01,.1,1,10,100].

Linear Kernel

Using Sklearn's SVC() function to create the SVM, we set the kernel parameter to "linear" and the L2 regression parameter to its default value of 1. The L2 regression parameter is strictly increasing and as a result, couldn't be set to 0. With these parameters, the accuracy of the model was ~92.28%. Using Sklearn's built-in K-Fold Cross Validation function with the number of folds set to 5, the accuracy was ~92.36%. Below are the results for L2 Regularization parameter tuning on the training and validation set using K-Fold Cross Validation.

L2 Regularization

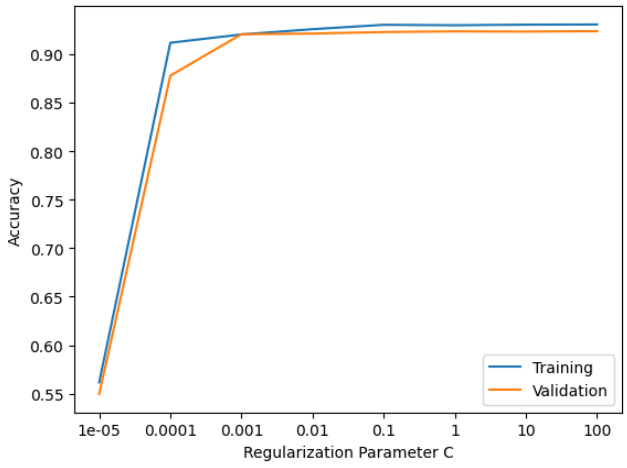

Figure 7: SVM Linear Kernel C vs. Accuracy

Table 4: SVM Linear Kernel C vs. Accuracy

C Values	Training Accuracy	Validation Accuracy
0.00001	0.5617368018449627	0.5497512590433826
0.0001	0.9118026690126518	0.877885829690416
0.001	0.920396089570255	0.920629393869673
0.01	0.925823907906447	0.9213060817547358
0.1	0.9303466352050066	0.9228890252319963
1	0.9298940328551571	0.9235675026203441
10	0.9304973511055342	0.9233415139197791
100	0.9306481806681285	0.923794002607562

Polynomial Kernel

The default accuracy using the polynomial kernel function set to its default degree of three and C=1 gave us an accuracy of ~94.66% and a K-Fold Cross Validation score of ~93.35%. Below are the results for L2 Regularization parameter tuning on the training and validation set using K-Fold Cross Validation with K=5.

L2 Regularization

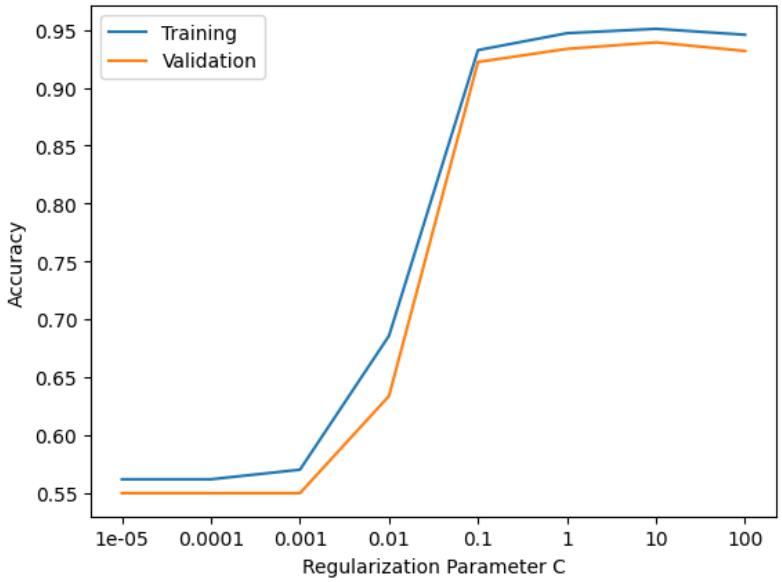

Figure 8: SVM Polynomial Kernal C vs. Accuracy

Table 5: SVM Polynomial Kernal C vs. Accuracy

C Values	Training Accuracy	Validation Accuracy
0.00001	0.5617368018449627	0.5497512590433826
0.0001	0.5617368018449627	0.5497512590433826
0.001	0.5700286769394441	0.5497512590433826
0.01	0.6856609619675358	0.6336495641281285
0.1	0.9324569987985921	0.9222108034869748
1	0.9470815559427643	0.9335184191016694
10	0.9508501354283526	0.9391719712656903
100	0.9457248855138832	0.9317066748472532

Polynomial Degree Tuning

Using the best C value of 10, the results of the polynomial degree tuning are below.

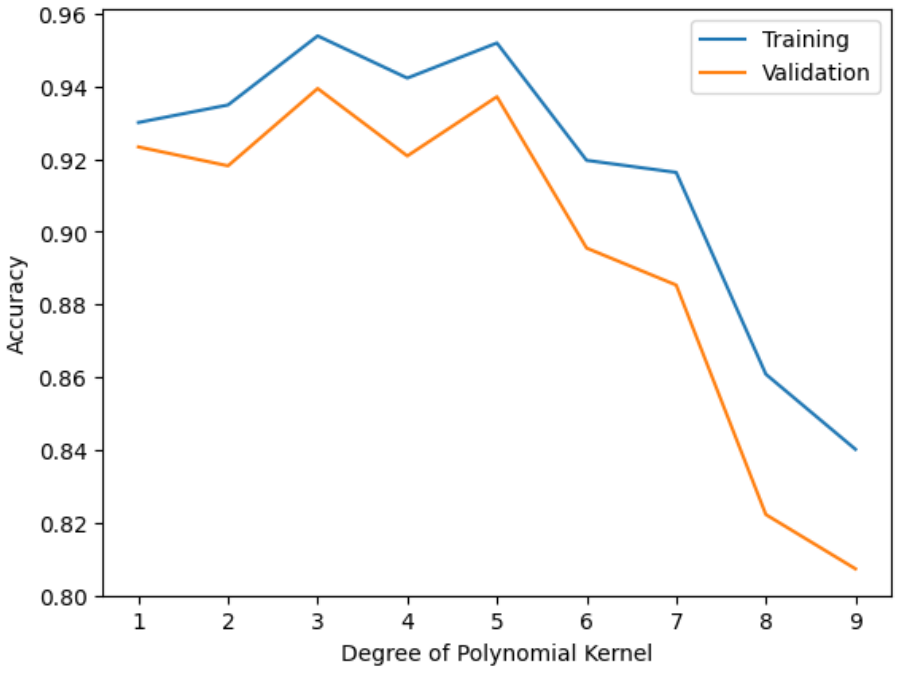

Figure 9: SVM Polynomial Kernel Degree vs. Accuracy

Table 6: SVM Polynomial Kernel Degree vs. Accuracy

Degree	Training Accuracy	Validation Accuracy
1	0.9300450897418848	0.9233415139197791
2	0.9348700444759668	0.9181345706470333
3	0.95386524907337	0.9393971930362758
4	0.9422585334638175	0.9208518035636681
5	0.9519052603941119	0.9407546590996242
6	0.9196431920400183	0.8955244522841731
7	0.9163250553250111	0.8853490809622414
8	0.8608461458898091	0.8222509394892246
9	0.8401901111728675	0.8073259708055321

Radial Based Kernel

The accuracy of the RBF kernel with the default C=1 gives an accuracy of ~94.52% and a K-Fold Cross Validation score of ~93.89% using K=5. Below are the results for L2 Regularization parameter tuning on the training and validation set using K-Fold Cross Validation with K=5.

L2 Regularization

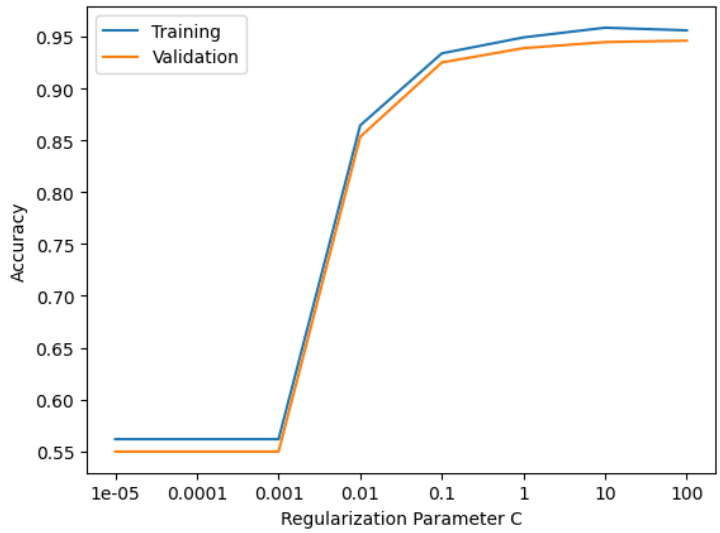

Figure 10: SVM RBF kernel C vs. Accuracy

Table 7: SVM RBF kernel C vs. Accuracy

C Values	Training Accuracy	Validation Accuracy
0.00001	0.5617368018449627	0.5497512590433826
0.0001	0.5617368018449627	0.5497512590433826
0.001	0.5617368018449627	0.5497512590433826
0.01	0.8644663963782719	0.8534660122197509
0.1	0.9339654080866016	0.9251517243142369
1	0.9493430900851443	0.9389439374185138
10	0.9586904311315854	0.9448224557097937
100	0.9561274651881504	0.946177365339878

Analysis

For the linear kernel, the highest validation accuracy came from when the value of C was 100 with a validation accuracy of ~92.37%. If I were to experiment with larger C values, based on Figure 7, I would expect the accuracy to increase but only by a small amount to the point that it is almost negligible. The model is overall solid as it avoids both overfitting and underfitting.

The polynomial kernel with its default degree of 3, performs similarly to the linear kernel. The highest validation accuracy was ~93.92% with C=10. Compared to the linear kernel model, this model has issues with underfitting when the C values are below 0.1 as Figure 8 shows. When the polynomial degree is changed it best performs when the degree is 5 with a validation accuracy of ~94.07%. Increasing the degree over 5 decreases the accuracy for both the training set and validation set as the model begins to overfit the training data and its associated noise.

The radial-based function kernel performed slightly better than the other kernels but not significantly. The RBF kernel was very similar to the polynomial kernel with degree 3 as they both were underfitted for C values below 0.1 before reaching 90% accuracy with the C values being greater than 0.1. The RBF kernel best performed when C=100 with a validation score of ~94.61%, which is the highest of all the SVM models.

Using the best-performing SVM model (RBF kernel when C=100), we get the following confusion matrix with an F1 score of 0.9707. The precision is 0.959036 and the recall is 0.9827.

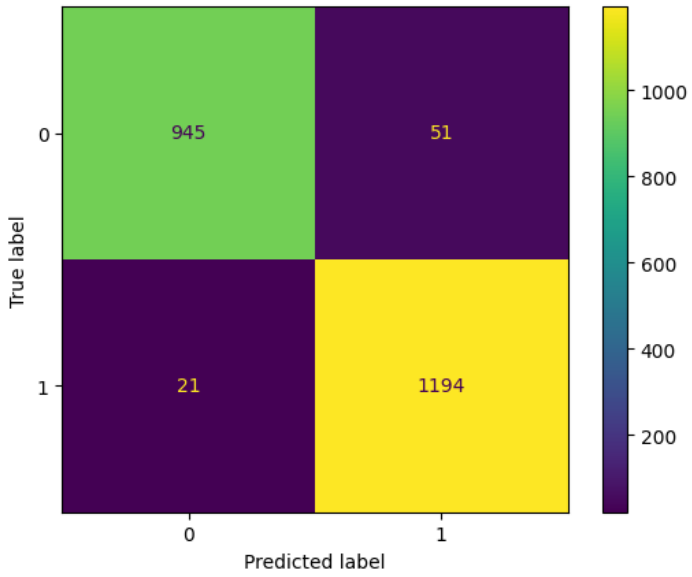

Figure 11: SVM Confusion Matrix

Neural Networks

Overview

The third model tested is neural networks. The goal was to find the best architecture for the neural network. For a classification problem like ours that did not have an extremely large pool of complex features, we opted to use one hidden layer and find the best number of neurons. We also wanted to see the impact of the activation function on our model, so logistic and ReLU were tested. To build our neural network, we utilized Sklearn's MLPClasslifier. The library allowed us to define the hidden layers and the number of iterations, which was set to 1000. Since the library only allowed L2 regularization implementations, we were only able to test various lambda values using L2 regularization.

1. Test validation scores on a varying number of neurons in the hidden layer.
2. Test varying degrees of polynomial transformation to see if the architecture we have from the previous two experiments can improve.
3. Test L2 regularization
4. Test logistic activation function on the built model

Number of Neurons

To find the best number of neurons, K-Fold Cross Validation was utilized. K was set to 5 as a constant to keep it in line with the previous models. The neurons tested were from the range of 1 to 10, inclusive. Increasing the number of neurons improved the validation score significantly. Below are the results:

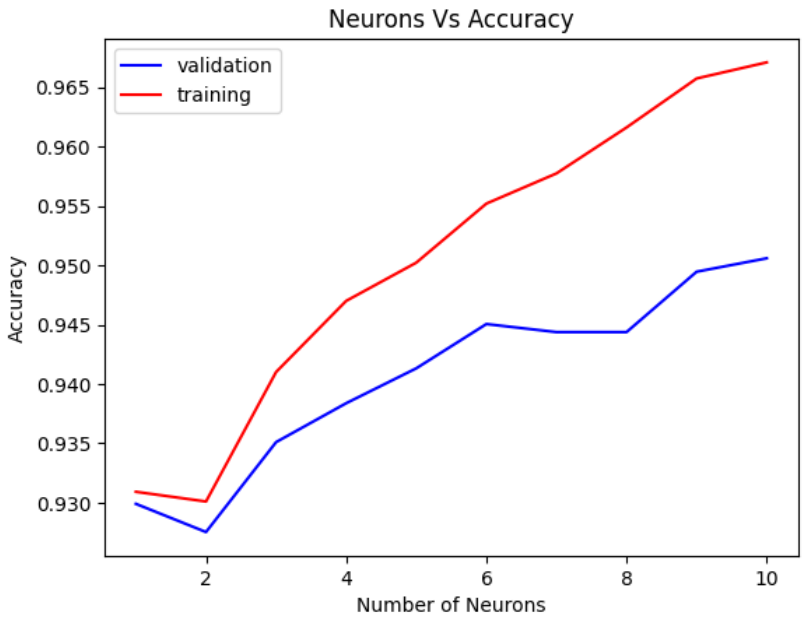

Figure 12: Neurons vs. Accuracy

Table 8: Neurons vs. Accuracy

Neurons	Training	Validation
1	0.9309135943526817	0.9298957152979034
2	0.9300938982515925	0.9275213007323206
3	0.9410050996050952	0.9350974167986106
4	0.9469979485724901	0.938376041376241
5	0.950220443563771	0.9413161307485118
6	0.9551956286816331	0.945047244013925
7	0.957739802287931	0.9443690865048893
8	0.9616124060428224	0.9443697259744877
9	0.9657395665835043	0.9494576658336509
10	0.9670963851330432	0.9505883759774292

The number of neurons that provided the highest validation accuracy is 10.

Polynomial Transformation

We utilized KFold Cross Validation to find the best polynomial transformation degree. The degrees tested are from 1 to 3 inclusive. Any degree above 3 led to extremely long training times so it was not feasible to continue, considering accuracy dropped after the second degree.

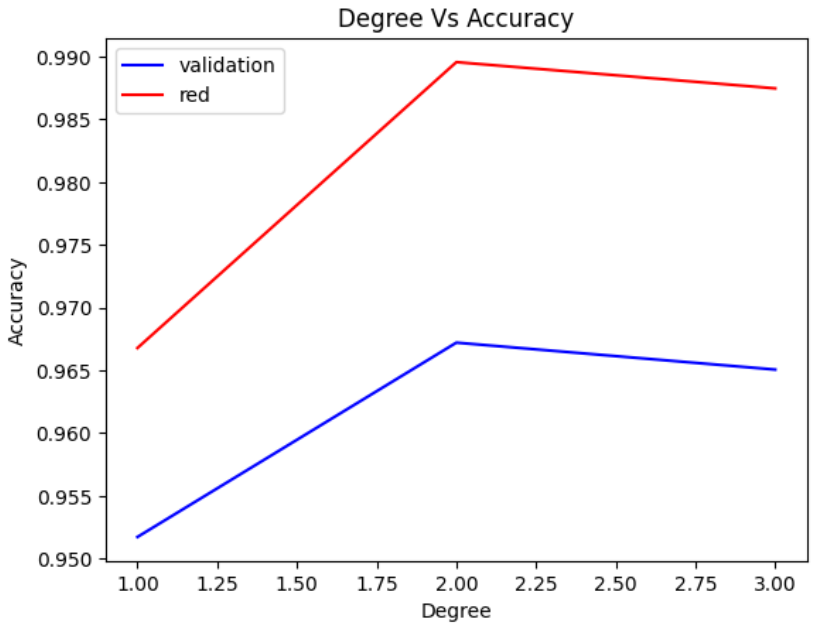

Figure 13: Neural Networks Degree vs. Accuracy

Table 9: Neural Networks Degree vs. Accuracy

Degrees	Training	Validation
1	0.9667853711445847	0.9517190221742478
2	0.9895692042179107	0.9672091500425888
3	0.9874773514013426	0.9650611716617767

L2 Regularization

The lambda values we used are 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10. We are using L2 regularization on a 2nd-degree polynomial transformation on the features. Below are the results:

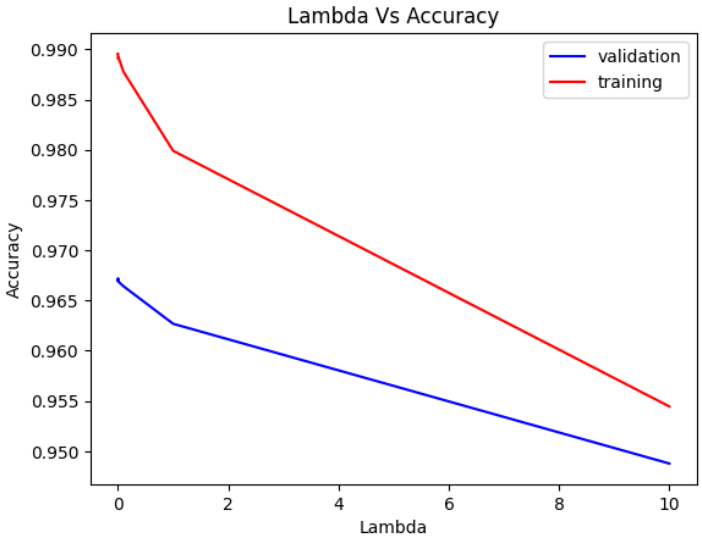

Figure 14: Neural Networks L2 vs Accuracy

Table 10: Neural Networks L2 vs Accuracy

Lambda	Training	Validation
0.00001	0.9895692042179107	0.9669829696456571
0.0001	0.9895692042179107	0.9672091500425888
0.001	0.9890603822806201	0.9669829056986972
0.01	0.9891734564855671	0.9668699753676311
0.1	0.9877600528936714	0.966417998255527
1	0.9799015993943595	0.9626866292022745
10	0.9544606703194194	0.9487794443776554

Activations

Utilizing the architecture we found from previous experiments, we decided to train and test two models, one with logistics and the other with ReLU activation. The results: ReLU has an accuracy of 95.5676% on the test data and logistic has an accuracy of 94.1203%. ReLU provides higher accuracy, so we will use ReLU in our final model.

Analysis

The model we ended up using was a neural network with one hidden layer and 10 neurons in the hidden layer. The model also had L2 regularization with a lambda value of 0.0001 and a ReLU activation function. While determining the number of neurons, the training and validation accuracy did not vary significantly from each other. There was only an average of ~1% difference between the two scores. This meant that the neural network was able to capture the complexity of the model while achieving a good balance between bias and variance. Utilizing the current architecture, we wanted to see if polynomial transformations on the data prior to training can impact accuracy. We tested 3 degrees and noticed that at the 2nd degree, the validation score was highest, but the training score was higher than the validation by ~2%. The 3rd degree was tested with a lower validation score, and we decided that using a 2nd-degree transformation on the data could be useful while going further will lead to overfitting. To help combat overfitting, we used L2 regularization on the model and found that a lambda value of 0.0001 provided the highest validation score. However, validation scores did not vary significantly, and this may mean that L2 regularization is helping reduce overfitting or there may not be any overfitting.

The features also underwent a 2nd-degree polynomial transformation. The final accuracy of our model is 95.5676% on the test data, with an F1 score of 0.960579243765084. The precision is 0.9394 and the recall is 0.9827 This suggests that our model was able to predict correctly while minimizing its false predictions.

Below is the confusion matrix.

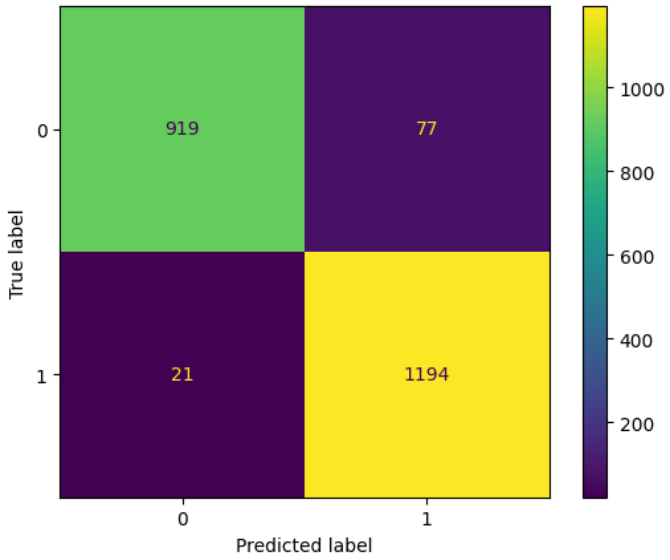

Figure 15: Neural Networks Confusion Matrix

Conclusion

The OpenML “PhishingWebsites” dataset with a minimal amount of preprocessing was used to create models to accurately predict and classify web pages as Phishing sites. There were 3 types of Machine Learning models created in order to do so: Logistic Regression, Support Vector Machines, and Neural Network models. K-fold Cross Validation was used as the method to gauge the accuracies of the models. The highest accuracies for each of the types of models are listed below. The best of these models was the Logistic Regression model with a Validation accuracy of 96.11%.

Table 11: Best Performing Model Accuracies

Logistic Regression	SVM	Neural Network
96.11%	94.61%	95.57%

Overall, the three different models had relatively high and similar accuracies. This leads us to believe that the task of classifying a website as a Phishing webpage isn’t a difficult one and the dataset used was fairly linearly separable.

Acknowledgments

Our Notebook: https://github.com/ac8736/phishing-prediction

Dataset from OpenML:
https://www.openml.org/search?type=data&sort=runs&id=4534&status=active

Description
Author: Rami Mustafa A Mohammad (University of Huddersfield", "rami.mohammad '@' hud.ac.uk", "rami.mustafa.a '@' gmail.com) Lee McCluskey (University of Huddersfield", "t.l.mccluskey '@' hud.ac.uk) Fadi Thabtah (Canadian University of Dubai", "fadi '@' cud.ac.ae)
Source: UCI

SVM Documentation:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

https://www.isanasystems.com/machine-learning-handling-dataset-having-multiple-features/

Neural Network Documentation:
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html