Machine Learning Approach for Detection of Website Phishing

José Castanheira, 76545
Pedro Santos, 76532

Website Phishing

Website phishing is a common scam, where fraudulent websites try to pass as legitimate ones in order to steal sensitive information from the users. This kind of scam is particularly present in the e-banking and e-commerce industry.

Data Description - Labels

- Data is labelled in 3 different classes: Phishing (-1), Suspicious (0), and Legitimate (1)
- Dataset comprises 1353 examples
- 702 Phishing URLS, 103 Suspicious, and 548 Legitimate

Data Description - Features

- Each dataset example is composed by 9 features:
  - URL Anchor (element defined by <a> tag)
  - Request URL
  - SFH (**S**erver **F**orm **H**andler)
  - URL Length
  - Pop-Up Windows
  - SSL Final State
  - Having IP Address
  - Web Traffic
  - Domain Age
- All features assume **integer** values
- There are **no** missing values

Data Description - Features

- URL anchor, request URL, SFH, URL length, pop-up windows, SSL final state, web traffic = {-1, 0, 1}
- Having IP address = {0, 1}
- Age of domain = {-1, 1}

Data separation

- 65% training set
- 20% cross-validation set
- 15% test set

Algorithms Applied

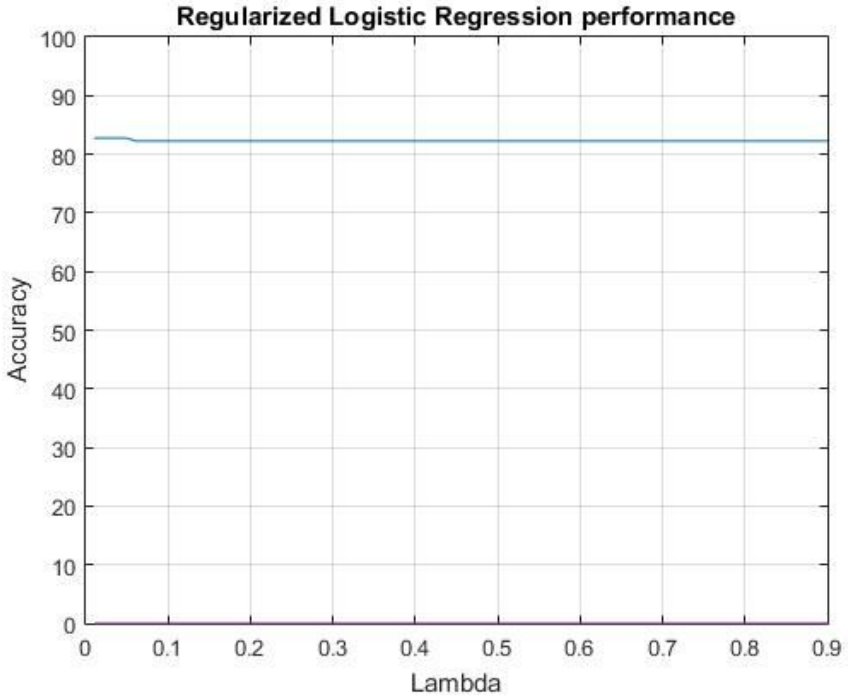
- Logistic Regression
- Artificial Neural Network
- Gradient Boosting Trees
- State Vector Machine

Logistic Regression

- Starting, logistic regression was applied with no data preprocessing and no regularization.
- Accuracy (training set): 82.821%
- Accuracy (cross validation set): 86.345%

	Phishing	**Suspicious**	**Legitimate**
Predicted Phishing	124	11	10
Predicted Suspicious	1	0	0
Predicted Legitimate	8	7	110

Logistic Regression with Regularization

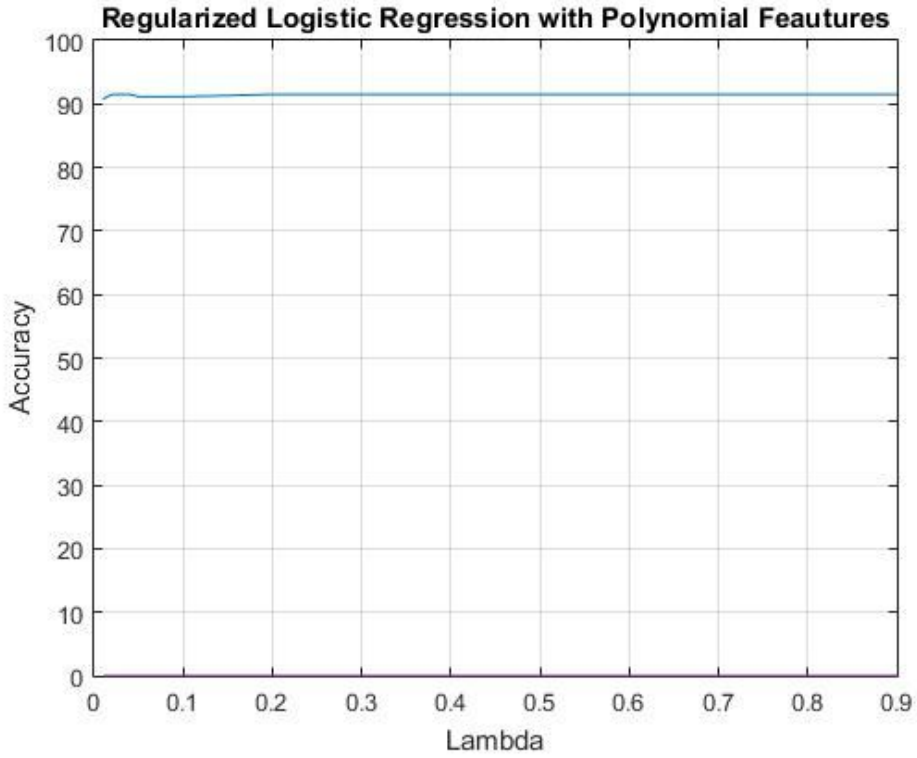


Logistic Regression with Polynomial Features

- With polynomial features of degree 2 accuracy increased to 88.939%
- With degree 4, we get the following confusion matrix (accuracy of 91.593%):

	Phishing	**Suspicious**	**Legitimate**
Predicted Phishing	124	1	9
Predicted Suspicious	4	16	3
Predicted Legitimate	5	1	108

Again no effect with regularization



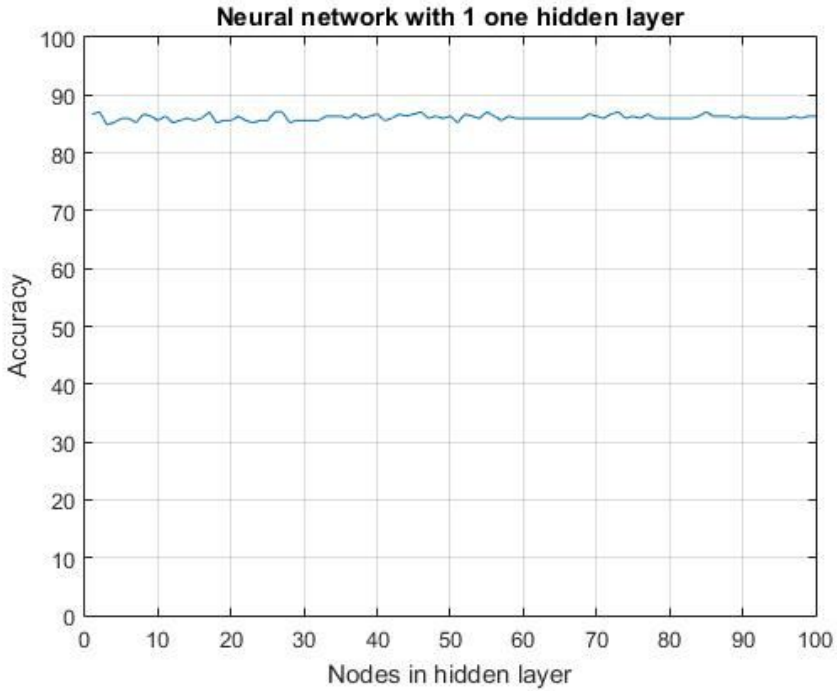
Artificial Neural Network

Several parameters to adjust:

- Number of hidden layers
- Number of hidden nodes
- Regularization parameter
- Epochs
- Activation functions

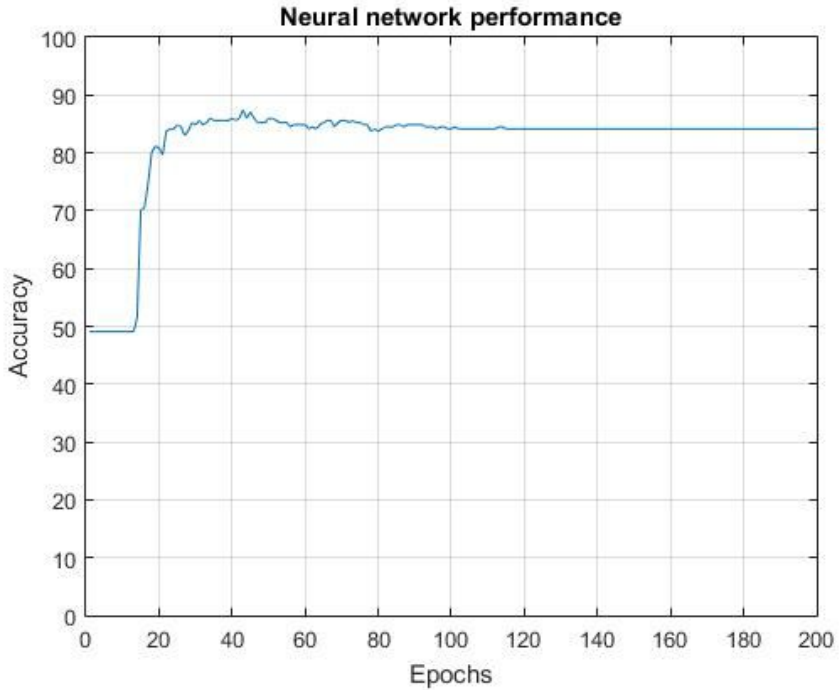
Artificial Neural Network - 1 Hidden Layer

- Activation function - Sigmoid Function
- Input Layer - 9 nodes
- Output Layer - 3 nodes
- Hidden Layer - from 1 to 100 nodes
- Best performance at 2 nodes with 87.084% accuracy (several other configurations also attained this)



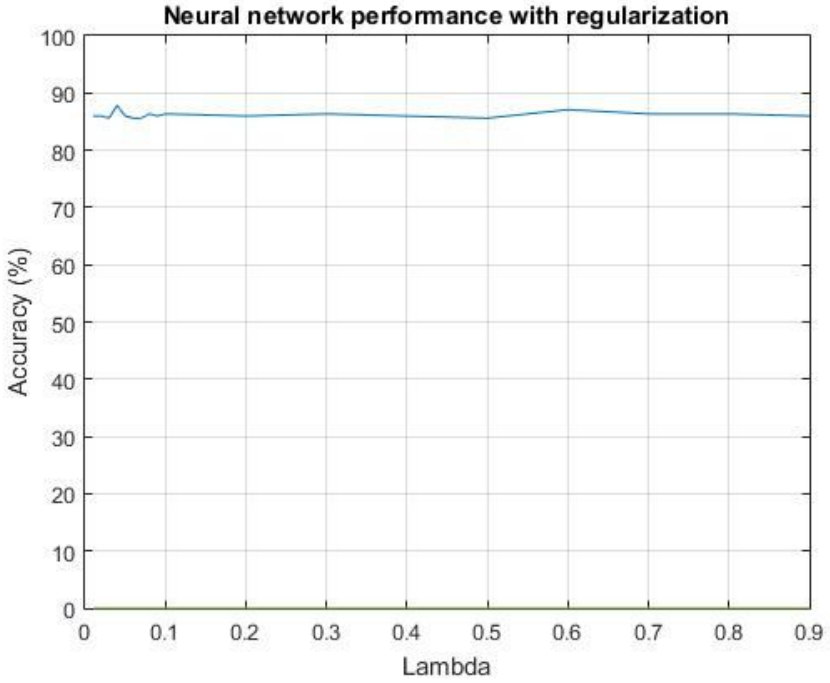
Artificial Neural Network - 1 Hidden Layer

- Neural Network with 2 hidden nodes
- Epochs from 0 to 200
- Best performance at 43 epochs with 87.454%



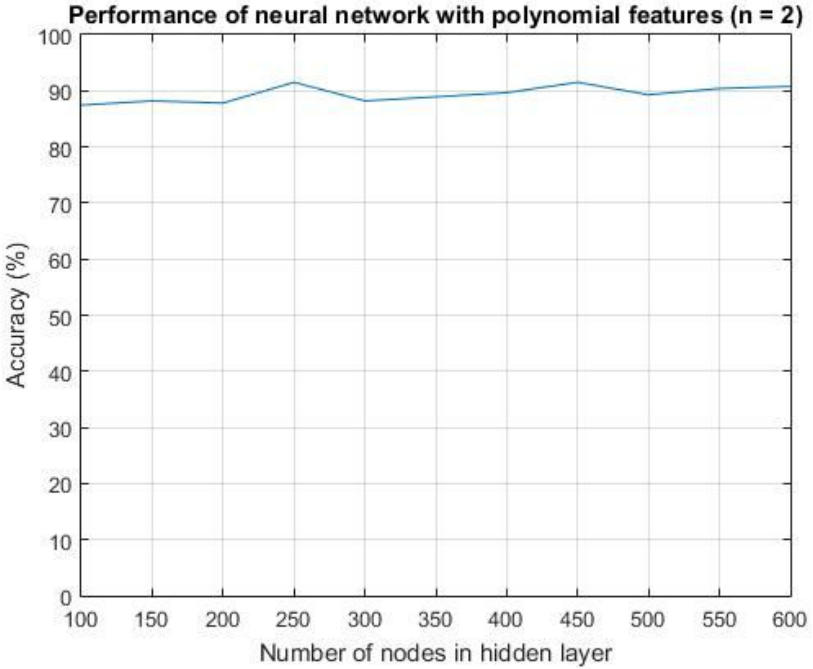
Artificial Neural Network - 1 Hidden Layer

- Neural Network with 2 hidden nodes, 43 epochs
- Test regularization
- Best performance at λ = 0.04, with 87.823%



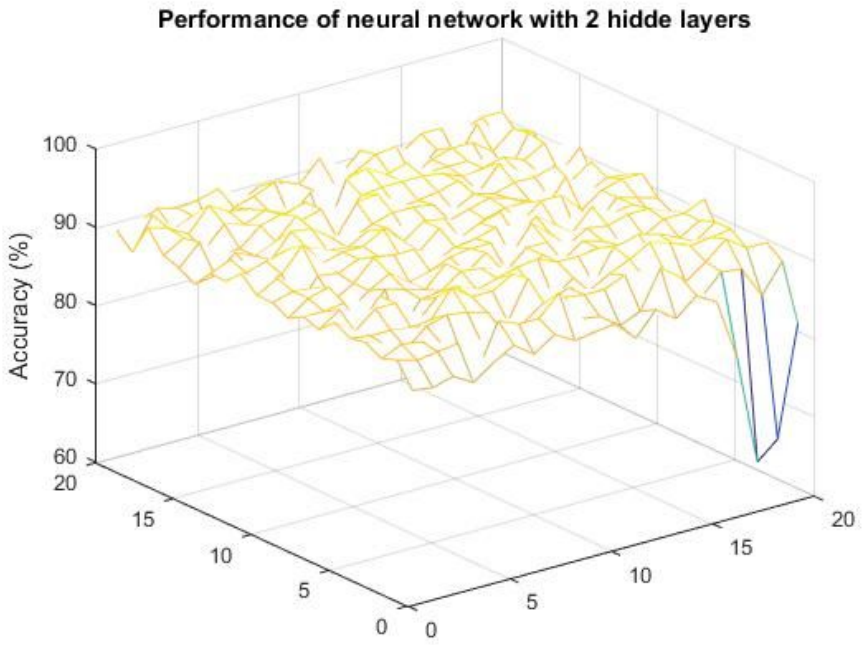
Artificial Neural Network - 1 Hidden Layer

- Neural Network with 2 hidden nodes, 43 epochs, $\lambda = 0.04$
- Test performance using polynomial features (degree 2) and different number of hidden nodes
- Best performance at 250 nodes with 91.513% (also the best for logistic regression)



Artificial Neural Network - 2 Hidden Layers

- Trained a 2 hidden layers neural network for a number of nodes varying between 1 and 20 in each layer
- Best performance with 17 hidden nodes on first layer and 11 on the second, yielding an accuracy of 92.593%



Artificial Neural Network - 2 Hidden Layers

- Confusion matrix

	Phishing	**Suspicious**	**Legitimate**
Predicted Phishing	129	1	9
Predicted Suspicious	3	13	3
Predicted Legitimate	1	3	108

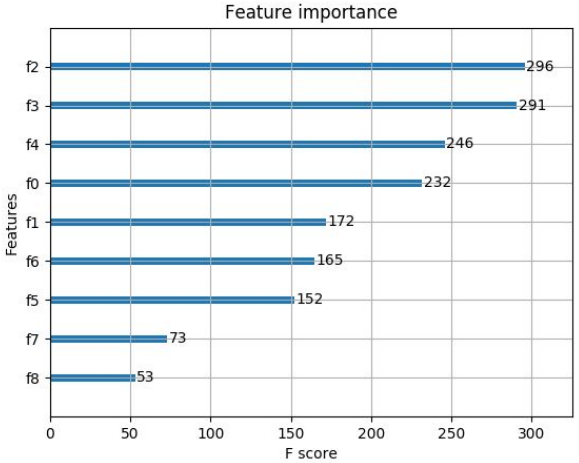
Gradient Boosted Trees

1. One of the most powerful techniques for building predictive models

2. It is possible to retrieve importance scores for each attribute and then it is possible use this score in other algorithms.

Gradient Boosted Trees - First Approach

1. First test was using the default values of the library
2. After training the model it is possible to see and plot the importance of each feature.



f0=SFH
f1=popUpWindow
f2=SSLfinal_State
f3=Request_URL
f4=URL_of_Anchor
f5=web_traffic
f6=URL_Length
f7=age_of_domain
f8=having_IP_Address

Gradient Boosted Trees - Training the model

3. The accuracy that it was obtained by training the model was 89.91%

4. Using the information of the most importants features the model was trained by using multiple thresholds being possible to test each subset of features by importance. First it was trained using all features and ending with a subset with the most important feature.

Gradient Boosted Trees - Results

Number of Features	Accuracy
9	89.91 %
8	89.91 %
7	90.20 %
6	89.34 %
5	86.46 %
4	84.44 %
3	76.66 %
2	73.78 %
1	73.49 %

Gradient Boosted Trees - Polynomial Features

	Degree 2	**Degree 3**
Best Accuracy	91.64	91.93
Second Best Accuracy	91.35	91.35
Third Best Accuracy	91.07	90.49

Gradient Boosted Trees - Best Arguments

How is it possible to improve the accuracy by changing the arguments?

max_dephs=range(3,10)

learning_rates=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8]

n_estimators=range(50,200,10)

min_child_weights=range(1,4)

max_depths=5

learning_rates=0.2

n_estimators=80

min_child_weights=1

Accuracy = 91.06 %

State Vector Machine

- SVM doesn’t support multiclass classification
- It was created two versions using SVM
  - Matlab
  - Python

State Vector Machine - Matlab

A. It was used the Svm version of libsvm
B. It was necessary create an algorithm to give support to multiclass classification
C. One vs One approach that consists in training a separate classifier for each different pair of labels.

State Vector Machine - Matlab Algorithm

- See the different classes that the dataset has.
- Calculate the combination of all the classes that the dataset has.
- The number of combinations is the number of models that it will be created.
- Iterate by the models and select from the dataset the data that is from the classes that will be tested by that model.
- Train the model and predict.
- The class that has more votes is the class that belongs to the given data.

State Vector Machine - Python Algorithm

- It was used the version of library sklearn
- This version has support for multi classification
- One vs Rest approach
- Gradient Boosted Tree to calculated the most important features to training the model with the set of feature that are more important.

State Vector Machine - Results

Python

Number of Features	Accuracy
9	83.86 %
8	83.57 %
7	84.44 %
6	83.00 %
5	83.57 %
4	81.27 %
3	74.64 %
2	73.78 %
1	73.49 %

Matlab
Accuracy: 84.29 %

Model Selection - Rank

1. Neural Network with 2 hidden layers (17 and 11 hidden nodes) -> 92.593%
2. Gradient Boosting trees -> 91.93 % using polynomial degree=3, features=16
3. Logistic Regression with polynomial features (degree 4), λ =0.04 -> 91.513%
4. SVM Python -> 84.44 % using 7 features
5. SVM Matlab -> 84.29 %

Best model performance - Gradient Boosting Trees

- Best model was gradient boosting trees
- After training with training set and cross validation set, accuracy on the test set: 91.742%

End