Comparing OVR Logistics Regression and OVR Support Vector Classification in Phishing detection

1. Introduction

Phishing attacks is a form of social engineering attacks. Often posing as trustworthy systems and hitting directly on users' needs, phishing attacks use malicious website to trick users into giving private information. For instance, a phishing attack may pose as a banking service problem report and ask users to login into the fake website to resolve the issue. The moment users submit their information to this fake website, hackers will be able to access their accounts using the given information and exploit the data for bad intents.

If there is a way to detect phishing websites, many costs will be saved and many data will be kept secured, which can be greatly applicable to the Cyber Security field. This report offers a solution to this problem and discuss it in 6 main sections: **Section 1** is a brief introduction of the Machine Learning problem and the project. **Section 2** contains the grounding definitions for this machine learning problem. **Section 3** discusses the ML methods being used. **Section 4** describes the results, compares the used methods. **Section 5** concludes the method and seeks improvement for the chosen method. **Section 6** contains reference list of the materials used in this project and the appendix to the code file.

2. Problem definition

The label is the prediction (an Integer) whether a website contain phishing attacks: -1: phishing, 0: suspicious, 1: no phishing. The datapoints represent websites, which can be either phishing, suspicious or legitimate. The datapoints are characterized by 8 selected features. The features in the dataset have already been normalized (according to the author's rules) so that they are Integers labelled with -1, 1 or 0 (phishing, legitimate or suspicious, respectively):

URL based features:

- IP Address: Whether the website uses IP address as an alternative of domain name. If the URL has IP Address → -1, otherwise → 1.
- URL Length: Phishing websites can use long URL to hide suspicious part
  If the URL Length < 54 → 1, 54 ≤ URL Length ≤ 75 → 0, URL Length > 75 → -1
- SSL state: A website without SSL certificate or a trusted issuer is often unsecured websites. This can be seen in the URL as "http" (unsecured) or "https" (secured).
  Using https and issuer trusted → 1, https but issuer not trusted → 0, otherwise → -1

Domain based features:

- Age of domain: Phishing websites usually last shortly (less than a year).
  Age of domain ≥ 1 → 1, age of domain < 1 → -1
- SFH (Server Form Handler): A phishing website's SFH is usually "about: blank" or empty, serving no purpose but to save the entered data for malicious intent.
  SFH is "about: blank" or empty → -1, SFH redirects to a different domain → 0, otherwise → 1

HTML based features:

- Pop-up windows: Phishing attack is hidden using pop-up windows leading to different websites. Pop-up disabling right click → -1, pop-up with alert → 0, otherwise → 1
- Request URL: IFrame requests URL from other websites
  % Of request URL > 61% → -1, 22% ≤ % of request URL ≤ 61% → 0, otherwise → 1
- URL of Anchor: A phishing website's HTML anchor tags can contain a different website domain.
  % Of request URL > 67% → -1, 31% ≤ % of request URL ≤ 67% → 0, otherwise → 1

3. Methods

3.1. Dataset description:

The dataset is retrieved from Kaggle: https://www.kaggle.com/ahmednour/website-phishing-data-set

The dataset has a total of 1353 data points, consisting in 548 legitimate websites, 702 phishing websites and 103 suspicious websites, which are labelled as 1, -1 and 0, respectively. None of the datapoint is missing. The phishing websites were from Phishtank archive http://www.phishtank.com/. The legitimate and suspicious websites were collected from Starting Point Directory using PHP script by the author.

The dataset has 10 columns; 8 of which are selected as features (having_IP_Address, age_of_domain, URL_Length, SSLfinal_State, popUpWindow, SFH, Request_URL, URL_of_Anchor) and the "Result" column is chosen as the label. By visualizing the data of each column with a heatmap and column charts (find all diagrams in **6.1.Appendices**), 8 features showing stronger correlation to one of the three labels -1, 0, 1 are selected. The selection process ensures that the chosen features present strong correlation to the labels, therefore giving more accurate results.

3.2. One-vs-rest (OVR) Logistic Regression model:

Although this machine learning problem is formulated as a multiclass classification problem, the binary classification Logistic Regression model will be used. The logistic model will be combined with the "one-vs-rest" classification strategy, one of the most common approaches to multilabel classification problems.

To be more detailed, Logistics Regression simply aims to models the probability of an event taking place. It finds the relationship between features and the given input labels then gives probability of the labels (between 0 and 1). In the meanwhile, the main idea behind the "one-vs-rest" method is to treat the problem as a binary classification problem - applying logistic regression N times to find the probability of each of N label from the rest N-1 labels. New predictions are then made using the trained hypothesis that returns the highest value of our prediction, meaning it's most likely to be that label. That is, to train a logistic regression classifier $h_\theta(x)$ for each class to predict the probability that y = i (label i ∈ { -1, 1, 0 }); then make prediction on a new x' by using the class that maximizes $h_\theta(x')$.

The reason I use this approach is because Logistic Regression and OVR model fit well when working with a small number of labels (3 labels), and they are also very simple to implement with the sklearn library: The class "sklearn.linear_model.LogisticRegression" will be used to perform Logistic Regression. The parameter "multi_class" is set to "ovr" to get the model into working as a One-vs-rest model.

The hypothesis space of the Logistics Regression model is all function of the form:

$$h_w(x) = \sigma(w^T x) = \frac{1}{1 + exp(-w^T x)} \quad [1]$$

where $h_w(x)$ is the predicted label $h_w(x) \in [0,1]$, x is the vector containing the value of the features, w is the vector containing coefficients for each feature. That is, a linear function, composed with a sigmoid function σ (the logistic function).

The loss function being used is Logistic Loss (true label y and probability estimate p):

$$L_{\log}(y,p) = -\big(y\ log(p) + (1-y) log(1-p)\big) \quad [2]$$

The reason that this loss function is being used is because it is default to logistic regression.

3.3. One-vs-rest (OVR) Support Vector Classification (SVC) model:

The Support Vector Classification model, although is a binary classification, will be utilized for this Machine Learning problem using the same One-vs-rest strategy explained **above**.

Support Vector Machine (SVC) is a widely used model in linear classification objectives. The idea behind this model is to find a line that separates two input labels apart. To explain more clearly, the SVC model will study the given input to give a decision boundary that maximizes the margin from both labels. On the other hand, OVR strategy aims to run SVC N times to separate each label with the rest N-1 labels. Predictions will then be made using the trained hypothesis that returns the highest value of our prediction, meaning it's most likely to be that label. That is, to train a Support Vector decision function $h_\theta(x)$ for each class to output a confidence score that y = i (label i ∈ { -1, 1, 0 }); then make prediction on a new x' by using the class that maximizes the score $h_\theta(x')$.

The reason that SVC and OVR strategy is being used is because they are memory-efficient, fit very well working with a small number of labels (3 labels), and they are also very simple to implement with the sklearn library: The class "sklearn.svm.SVC" will be used to perform SVC. The parameter "decision_function_shape" is set to "ovr" to get the model into working as a One-vs-rest model.

The hypothesis space of the Support Vector Classification model the classifier maximizing distance from the two labels, which is given by:

$$f(x) = \text{sign}(w^T x - b) \quad [4]$$

Where "sign" is the sign function. The w and b is solved from the function maximizing the distance:

$$\min \frac{1}{2} w^T w\,, \text{subject to } y_i(w^T x_i + b) \geq 1\ (1 \leq \text{i} \leq \text{n}) \quad [5]$$

Maximizing the distance between two hyperplanes, geometrically, $\frac{2}{||w||}$, is equivalent to minimizing $\frac{||w||}{2}$, or $\frac{w^T w}{2}$. Therefore, the notation is a "min" function but not a "max". A constraint function $y_i(w^T x_i + b) \geq 1$ is given so that datapoints will not fall into the margin.

Hinge Loss is being used as it is the default loss function to SVC model:

$$L_{\text{Hinge}(y_w, y_t)} = \max\{1 + y_t - y_w, 0\} \quad [6]$$

y_w is the predicted decision for true label, y_t is the maximum of the predicted decisions for other labels.

3.4. Additional model evaluation metrics and training-validation dataset splitting:

To evaluate the model's performance, besides the loss functions, the F1-score is also being used, instead of the accuracy score. This is because F1-score minimizes False Negative and False Positives predictions.

The wrong prediction that a website is a phishing website while it is not, or a website is legitimate while it is in fact a phishing can be highly costly to our model, so it is reasonable to minimize those. Moreover, the data shows high imbalances between the "0" class and the others ("-1", "1"), therefore the F1-score is taken into use as the F1-score deals better with imbalanced data.

The dataset will be split into training set and validation test with the ratio 80:20 using the "train_test_split" class available in sklearn. The reason for this choice is because it's a very commonly used ratio, allowing the training set to be reasonably larger than the test set: 1353 datapoints will be separated into 2 datasets: 1082 datapoints (training set) and 271 datapoints (validation set).

4. Results

	Training error	Validation error	F1-score
OVR Support Vector Classification	0.396	0.337	0.843
OVR Logistic Regression	0.447	0.393	0.838

It can be seen from the table that the training error and validation error for OVR Support Vector Classification is less than OVR Logistic Regression. Moreover, the F1-score of OVR SVC is greater than that of OVR Logistic Regression. This is enough to conclude that OVR Support Vector Classification is a better-performed model than OVR Logistic Regression for this Machine Learning problem.

The test set is then formed using the ratio 80:10:10 using "train_test_split" class available in sklearn. The reason for this choice is because it is again, a very commonly used ratio, allowing the training dataset to be large enough to train and leaving enough data remaining for validation and testing processes. 1353 datapoints will be separated into 3 datasets: 1082 datapoints (training set), 135 datapoints (validation set) and 136 datapoints (test set).

The test error (Hinge loss) of the chosen method, OVR Support Vector Classification, is 0.349 and the F1-score for the test set is 0.839.

5. Conclusion

For both methods, the training error is about 6% larger than the validation error. This implies that both models are slightly overfitting. This problem might be due to the small size of the dataset (only 1353 data points). There is also an imbalance between the class "0" and the classes "1", "-1", which may lead to class imbalances in training and validation datasets; therefore, cause some difference between the errors. To further improve the model, more datapoints can be collected so that the number of datapoints of each class are large enough and data selection and features selection can be done more carefully so that the class are balanced.

For the chosen method OVR Support Vector Classification, the training, validation, and test error is not low enough to say that it's an excellent model. That being said, the F1-score of the model is relatively high (≈ 0.84), and it is usually a better and more interpretable value to the model performance than the Hinge loss function. Although there could be room for more improvement, it is reasonably to assert that the model performs well enough for the detection of phishing websites as a beginner ML project.

6. References and Appendices

6.1. Code appendix:

The code file can be found here: https://github.com/Hungreeee/Phishing-Detection-Notebook

6.2. References:

[1] Formula is taken from https://svivek.com/teaching/lectures/slides/prob-learning/logistic-regression.pdf

[2] Formula is taken from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html

[4] [5] Formula is taken from https://en.wikipedia.org/wiki/Support-vector_machine

[6] Formula is taken from https://scikit-learn.org/stable/modules/model_evaluation.html#hinge-loss