Machine Learning Engineer Nanodegree

Capstone project

Anand kumar

June 26,2018

Definition

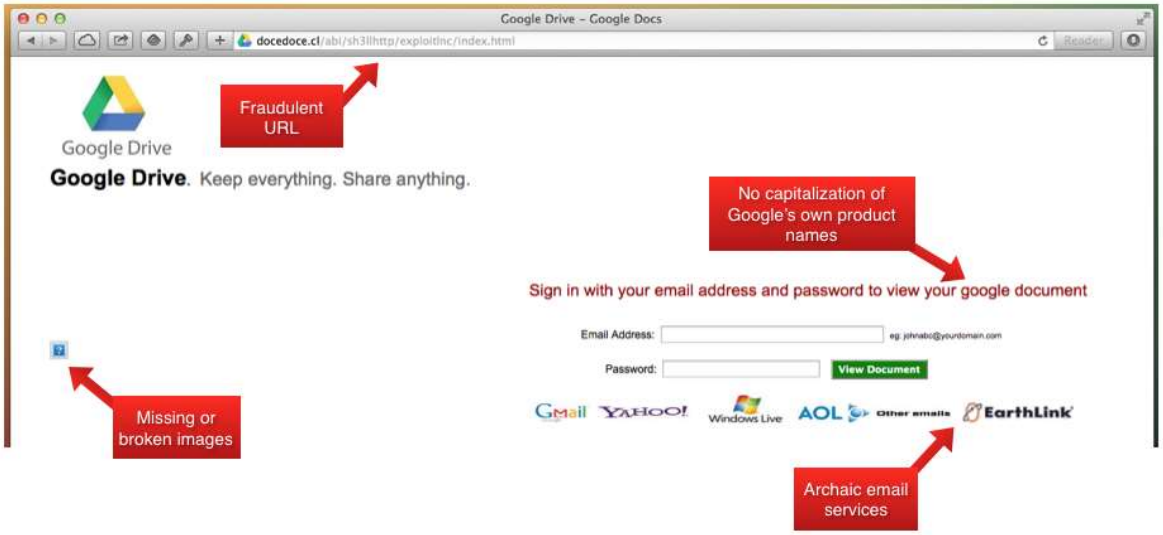
Project Overview

The project is about the detecting whether the website is phishy, suspicious or Legitimate. In this technological world Everything is getting online and most of the money transactions are happening through online every day. The usage of Internet has increased a lot and at the same time cyber crimes are also increasing due to presence of vulnerabilities and loop holes in the network.

It is a very serious issue, the users won't be able to find the difference between the websites which are good and which are bad. They mimic the original website and make the people use and collect the sensitive information. Normal people couldn't find whether the website they are using is good or bad because they don't know what is happening in the internet and what each thing in the website do . So, I want to apply machine learning algorithm to detect whether the website is bad or good using the individual features (like URL length, IP address) by knowing whether the features itself are good or bad to decide whether the website is malicious or legitimate.

https://www.ijcaonline.org/archives/volume147/number5/aldiabat-2016-ijca-911061.pdf

https://archive.ics.uci.edu/ml/datasets/Website+Phishing



Problem Statement

The problem is to detect whether the website is phishy or suspicious or Legitimate by making use of the dataset features(which represents the website information like URL length ,IP addresses etc). considering the individual features of the website which are responsible for fraudulent websites and which look suspicious can be used to identify whether the whole website is phishy or legitimate. Even though some of the websites look suspicious , but they might not be malicious, This dataset is designed taking into consideration of suspicious websites also. This dataset contains three classes, it is a multi-class classification, so we need classification algorithms to classify the data. we can use classification algorithms like svm and Gradient Boosting Algorithm to train and test the data.

Metrics

The Evaluation metrics are necessary for every machine learning Algorithm. We need to know how good our model performs. Based on those metrics , we need to evaluate the model. There are many metrics but we need to choose the one which is suitable for the data. The metric I want to use here is f1_score and mcc(Matthews correlation coefficient).Here Accuracy score doesn't workout because the classes are imbalanced, so accuracy score will take the highest class.so we need precision and recall inorder to know how good our model is.

$$f1\_score = 2*(precision*recall)/(precsion+recall)$$

f1 score and mcc score is a way of combing the precision and recall because both are essential to know how good our model is.

Analysis

Data Exploration

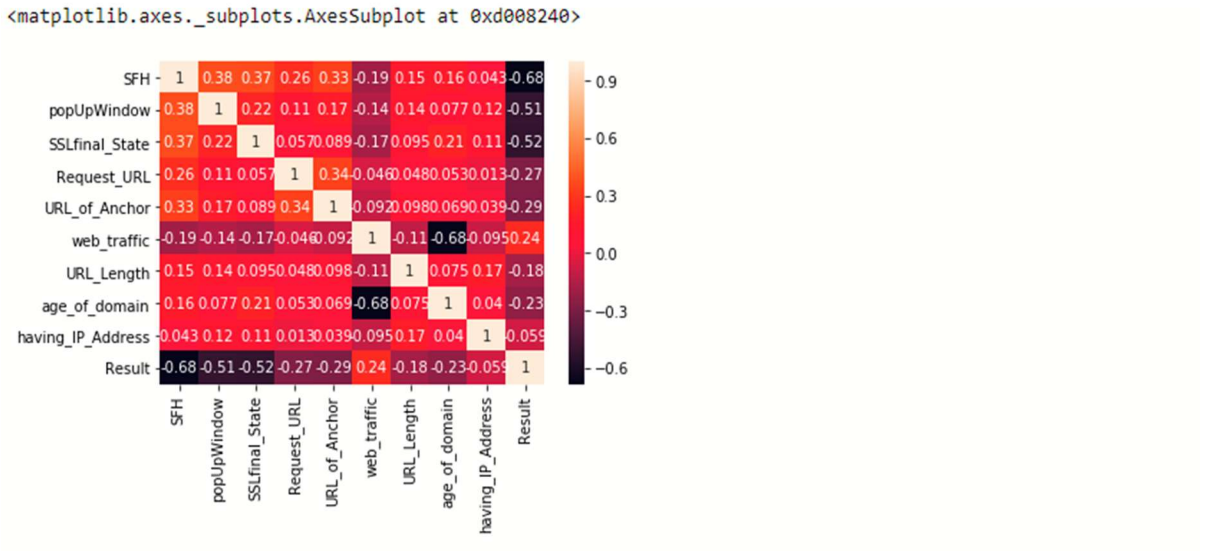
The dataset is from the UCI Machine Learning Repository. The dataset contains 1353 instances with 10 features each representing the information about the websites. Each feature and output label represents three categories (Phishy , Suspicious , Legitimate) and they are represented as (-1,0,1) as numerical features. In those 1353 instances there are total of 702 phishy websites(represented as -1) and 548 Legitimate websites (represented as 1) and 103 suspicious websites (represented as 0).The classes are imbalanced. Take the result column as output label and the input is remaining 9 features. There are no abnormalities in the data that we can see.

```
# Load the PhishingData dataset and store it in data
data = pd.read_csv("PhishingData.csv")

# display
display(data.head(n=5))
```

	SFH	popUpWindow	SSLfinal_State	Request_URL	URL_of_Anchor	web_traffic	URL_Length	age_of_domain	having_IP_Address	Result
0	1	-1	1	-1	-1	1	1	1	0	0
1	-1	-1	-1	-1	-1	0	1	1	1	1
2	1	-1	0	0	-1	0	-1	1	0	1
3	1	0	1	-1	-1	0	1	1	0	0
4	-1	-1	1	-1	0	0	-1	1	0	1

Exploratory Visualization



By looking at the above visualisation we can see that there is no good correlation between any of the features, so every feature is necessary for the model to detect the phishy websites.

We can also see that classes are imbalanced , there are very less suspicious websites, so we need to select a good metric to evaluate the model or else we may end up having a model chooses the highest class most of the time.It is regarded as Accuracy paradox.

Algorithms and Techniques

The problem deals with multi-class classification. So, we need classification algorithms to classify the data. There are many algorithms available. The algorithms that I choose to test are

Logistic Regression: It works well when the data is linearly separable. It also works well when there are less features. This is prone to overfitting. It underperforms when there is a nonlinear decision boundary.

SVM: Support vector machine work well even if the data is not linearly separable. It makes use of the higher dimension to find a hyper plane that separates the data. It tries to find a maximum margin hyper plane

Gradient Boosting Classifier: It is a boosting algorithm. It takes simple rules (weak learners to make a complex learner). It is an ensemble of weak models. It is used mostly with decision tress.

The technique is to take any algorithm, train the model using training data and testing the model using testing data. Based on the metric, evaluate the performance of the model. We can also use grid search method to choose the best parameters which yields the best results.

Benchmark

The Benchmark algorithm means naive algorithm which decently generalizes the data but it is not intelligent. We can use this algorithm to compare it with the best model we choose. The benchmark algorithm that I choose here is Logistic Regression, it is a simple classification algorithm. It acts as the good benchmark.

The f1 score for the benchmark here is 0.79, which seems good.

Methodology

Data Pre-processing

There is no need of Data pre-processing and scaling for this dataset. As Every feature of the dataset only contains 3 categories (-1,0,1). There are no missing and abnormal values in the dataset. Data processing is only done when there are missing values and if the non-numerical data is present because most of the machine learning algorithms want the numerical input. so, they must be encoded using some encoding technique.

Implementation

First split the data into training and testing sets by taking 20% data for testing.80% of the data for training the data.

Shuffle and Split Data

```
#
from sklearn.cross_validation import train_test_split

# Split the 'features' and 'result' data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    result,
                                                    test_size = 0.2,
                                                    random_state = 5)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

```
Training set has 1082 samples.
Testing set has 271 samples.
```

After the Evaluation of benchmark model (here Logistic Regression) then deploy some supervised algorithms to know which one yields the best results.

Then Create a training and predicting pipeline for these algorithms.

This function will take learner and training and testing data as parameters and fit the data using training data and predict the data using testing data. Then the metrics like f1 score ,mcc and accuracy score are returned as a dictionary.

```
def predict(learner, X_train, y_train, X_test, y_test):

    results = {}

    learner = learner.fit(X_train,y_train)
    predictions_test = learner.predict(X_test)

    results['f1_score'] = f1_score(y_test,predictions_test,average='weighted')
    results['mcc_score'] = matthews_corrcoef(y_test,predictions_test)
    results['acc_score'] = accuracy_score(y_test,predictions_test)

    return results
```

Test these three supervised algorithms by choosing some random state.

```
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

# TODO: Initialize the three models
clf_A = AdaBoostClassifier(random_state=0)
clf_B = SVC(random_state=0)
clf_C = GradientBoostingClassifier(n_estimators=250,random_state=0)

print "AdaBoostClassifier : ",predict(clf_A,X_train, y_train, X_test, y_test)
print "svm : ",predict(clf_B,X_train, y_train, X_test, y_test)
print "GradientBoostingClassifier : ",predict(clf_D,X_train, y_train, X_test, y_test)
```

```
AdaBoostClassifier :  {'f1_score': 0.821322434179438, 'acc_score': 0.8339483394833949, 'mcc_score': 0.6927992587916685}
svm :  {'f1_score': 0.8125582239428951, 'acc_score': 0.8450184501845018, 'mcc_score': 0.7113921424929533}
GradientBoostingClassifier :  {'f1_score': 0.8963936028186082, 'acc_score': 0.8966789667896679, 'mcc_score': 0.814150273944441
4}
```

Test these three models by training and testing to get the score so that we can choose the best model among these. As you can see on the above data we can say that Gradient boosting classifier has got the good f1 score and mcc score. So, the best model to choose is gradient boosting algorithm.

Refinement

We have chosen the best model and we need to refine the model by choosing the appropriate parameters which are responsible for high score. We need to tune the model. In order to tune the model, we need to check every possible parameter and it is tedious task for us.so we need to use the grid search technique so that it will choose the best possible parameters. Hence the score of the model will be increased a little bit.

```
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import f1_score, make_scorer
from sklearn.metrics import matthews_corrcoef

clf = GradientBoostingClassifier(n_estimators=250,random_state=0)

parameters = {'n_estimators':[100,50,250],'learning_rate':[0.1,0.5]}

scorer = make_scorer(f1_score,average='weighted')
grid_obj = GridSearchCV(clf,parameters,scoring=scorer)
grid_fit = grid_obj.fit(X_train,y_train)
best_clf = grid_fit.best_estimator_

# Make predictions using the unoptimized and model
predictions = (clf_C.fit(X_train, y_train)).predict(X_test)
#best_predictions = best_clf.predict(X_test)
best_predictions = best_clf.predict(X_test)

print("Unoptimized model\n------")
print("f1_score core on testing data: {:.4f}".format(f1_score(y_test, predictions,average='weighted')))
print("\nOptimized Model\n------")
print("Final f1_score on the testing data: {:.4f}".format(f1_score(y_test, best_predictions,average='weighted')))
```

Results

Model Evaluation and Validation

The parameters which are used for final model based on the grid search result

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=0, subsample=1.0, verbose=0,
              warm_start=False)
```

Learning rate = 0.1 which is a reasonable learning rate which is not too slow as well. Number of estimators are chosen as 100 as we know the increase in the number of estimators increase the performance and a maximum depth of 3 gives the best result.

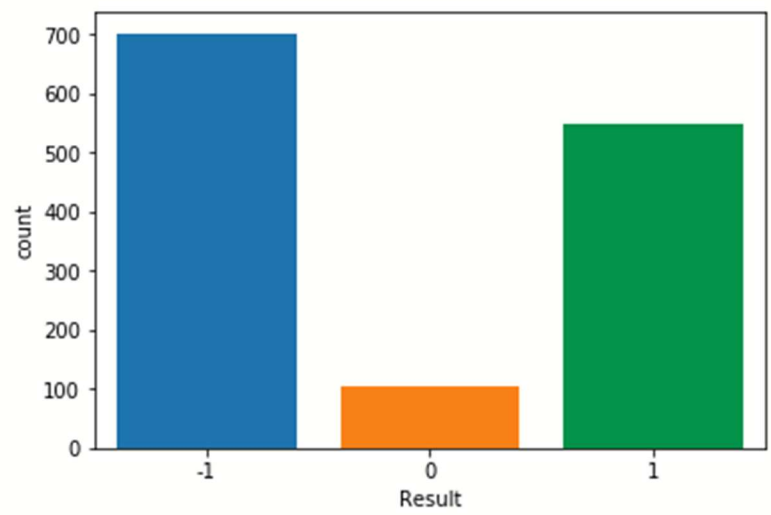
Justification

The benchmark model has got the f1 score of 0.79 and the final model which is chosen has got 0.89 after choosing necessary parameters. Obviously the final model has got the best score and is able to generalize the data well. Logistic regression has done a good job but not good comparatively as it is a simple classification algorithm. Which is also prone to overfitting .Also the number of features is 10, so it may not perform well. Gradient Boosting is a boosting algorithm, so it got high score. The final model and solution has got 0.89 f1 score, we can say it can significantly solve the problem.

Conclusion

Free-Form Visualization

```
<matplotlib.axes._subplots.AxesSubplot at 0x123c4c88>
```



From the above visualisation we can see that the result is plotted. we can observe here the category phishy (-1) is more and Legitimate (1) is little less but we can see that the suspicious (0) is very less. It leads to imbalances in the data. The models tend to be more towards the class which is having more instances (phishy). If we take balanced number of instances then there will be no ambiguity. Then our model will also perform well.

Reflection

1.Initially I choose the problem domain as fraud website detection.

2.Took necessary data set from the machine learning repository.

3.Downloaded necessary python modules and files required for the project.

4.Then the data is divided into input and output labels.

5.As the data is numerical with -1 or 0 or 1 which represents only three categories, there is no need of the data pre-processing and feature scaling.

6.Split the data into two parts as training data and testing data.20% of the data is used as testing.

7.Make some analysis using some benchmark model. Here logistic regression is used as benchmark model and the metric is noted.

8.use some supervised algorithms to test which model yields the best results.

9.After training and testing each and every model, choose the best model which gives the best results.

10.Model refinement should be done in order to improve the performance of the model by choosing the parameters which yields the better results by using grid search method.

Improving the performance seems challenging to me as it involves tuning of parameters that best fits the model and the dataset.

Improvement

By considering the score of the final model, I think it is a very good score and model can generalize well. May be we can use more powerful algorithms like xgboost,Light GBM that may improve the performance. If the instances are not like 702 phishy and 508 legitimate and very less suspicious, instead if they are little balanced so that the model can perform vey well.