Machine Learning Engineer Nanodegree

Capstone Project

Abdelrhman Magdy Mohamed

March 1, 2019

I. Definition

Project overview

- The project consisted to evaluate different classification models to predict malicious and benign websites, based on application layer and network characteristics.

Problem statement

- Confidentiality and Privacy is the biggest challenge faced. If a user accesses the malicious website through a search engine with no perception, malicious scripts usually launch attacks to install rogue program, steal personal identities and credentials [1], so our Problem is to develop an algorithm which accurately predicts the malicious and the benign URLs.
- This is a binary classification supervised problem that takes the URL as an input and returns whether it contains malicious content or not as an output.
- We've got our dataset from Kaggle platform [2]:
  - Data set characteristics: Multivariate, Time-series
  - Attribute characteristics: Integer, real, characters
  - Associated tasks: Classification
  - Number of instances: 1782
  - Number of attributes: 21
  - Data format: csv
  - The data is imbalanced and contains 1782 labeled URL:
  - 216 malicious URLs (12 %)

- in This project I'll evaluate different supervised algorithms to classify the URLs based on this data and the features related to each URL like URL length and the number of special characters in the URL.

- The project will be divided into two part:
    - Applying the supervised algorithms to the imbalanced data [3] and evaluating the model based on the precision metric.
    - Using over and under sampling to get quiet balanced data and apply the same supervised algorithms and evaluating the suitable metrics.

Metrics

- As malicious URLs wrongly classified as benign is more important than the benign that classified as malicious So, we interested to minimize the FN.
- For our imbalanced data we will consider recall as our metric besides the Confusion Matrix.

1- Recall

$$recall = \frac{TP}{TP + FN}$$

- TP = True positive examples (malicious URLs and correctly classified as malicious)
- FN = False negative examples (malicious URLs and wrongly classified as benign)
- So, we interested to see the accuracy of our model to classify the malicious URLs.

In the second part of the project we'll try to balance data using the over and under sampling techniques so we we'll use the accuracy metric besides the recall metric for this part.

2- Accuracy

$$accuracy = \frac{true\ positives + true\ negatives}{dataset\ size}$$

Accuracy metric is a common metric that considers both true positives and true negatives with equal weights.

II. Analysis

Data exploration

The data attributes:

• URL: it is the anonymous identification of the URL analyzed in the study

• URL_LENGTH: it is the number of characters in the URL

• NUMBER_SPECIAL_CHARACTERS: it is a number of special characters

identified in the URL, such as, “/”, “%”, “#”, “&”, “. “, “=”

• CHARSET: it is a categorical value and its meaning is the character encoding

standard (also called character set).

• SERVER: it is a categorical value and its meaning is the operative system of

the server got from the packet response.

• CONTENT_LENGTH: it represents the content size of the HTTP header.

• WHOIS_COUNTRY: it is a categorical variable; its values are the countries we

got from the server response (specifically, our script used the API of Whois).

• WHOIS_STATEPRO: it is a categorical variable; its values are the states we got

from the server response (specifically, our script used the API of Whois).

• WHOIS_REGDATE: Whois provides the server registration date, so, this

variable has date values with format DD/MM/YYY HH:MM

• WHOIS_UPDATED_DATE: Through the Whois we got the last update date

from the server analyzed

• TCP_CONVERSATION_EXCHANGE: This variable is the number of TCP packets

exchanged between the server and our honeypot client

• DIST_REMOTE_TCP_PORT: it is the number of the ports detected and

different to TCP

• REMOTE_IPS: this variable has the total number of IPs connected to the

honeypot

• APP_BYTES: this is the number of bytes transferred

• SOURCE_APP_PACKETS: packets sent from the honeypot to the server

• REMOTE_APP_PACKETS: packets received from the server

• APP_PACKETS: this is the total number of IP packets generated during the

communication between the honeypot and the server

• DNS_QUERY_TIMES: this is the number of DNS packets generated during the

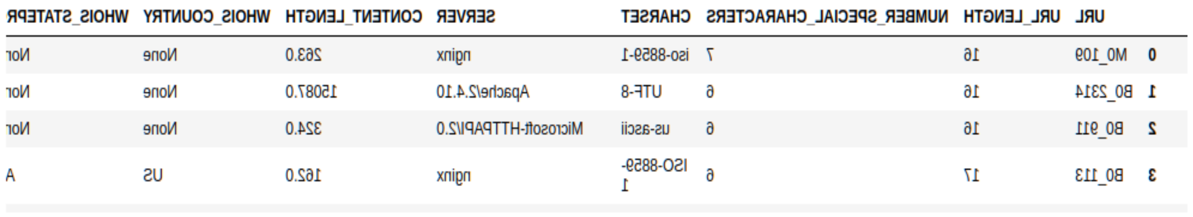
communication between the honeypot and the server

• TYPE: this is a categorical variable, its values represent the type of web page

analyzed, specifically, 1 is for malicious websites and 0 is for benign websites

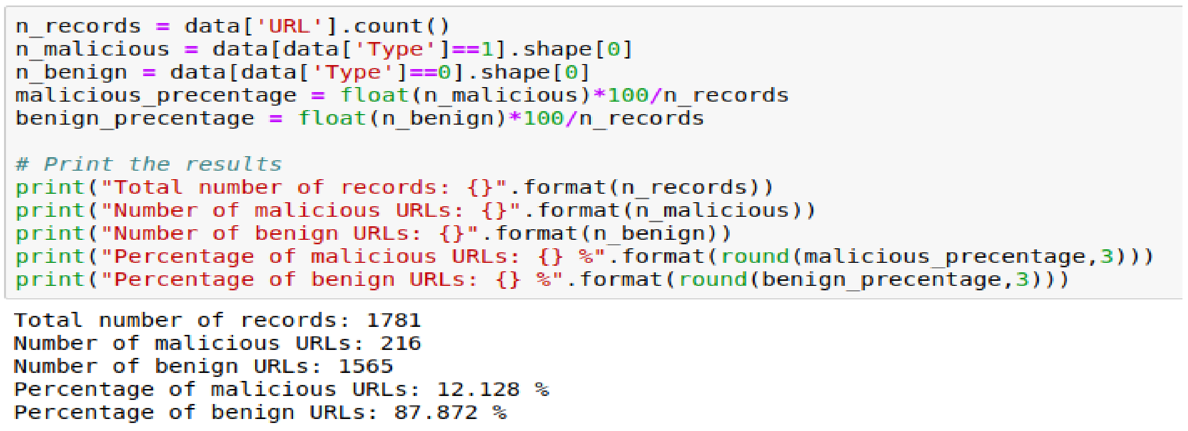
Sample from the data:

The shape of the data (1782, 21)



	URL	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CHARSET	SERVER	CONTENT_LENGTH	WHOIS_COUNTRY	WHOIS_STATEPR
0	M0_109	16	7	iso-8859-1	nginx	263.0	None	Nor
1	B0_2314	16	6	UTF-8	Apache/2.4.10	15087.0	None	Nor
2	B0_911	16	6	us-ascii	Microsoft-HTTPAPI/2.0	324.0	None	Nor
3	B0_113	17	6	ISO-8859-1	nginx	162.0	US	A

Figure 1 sample from the dataset



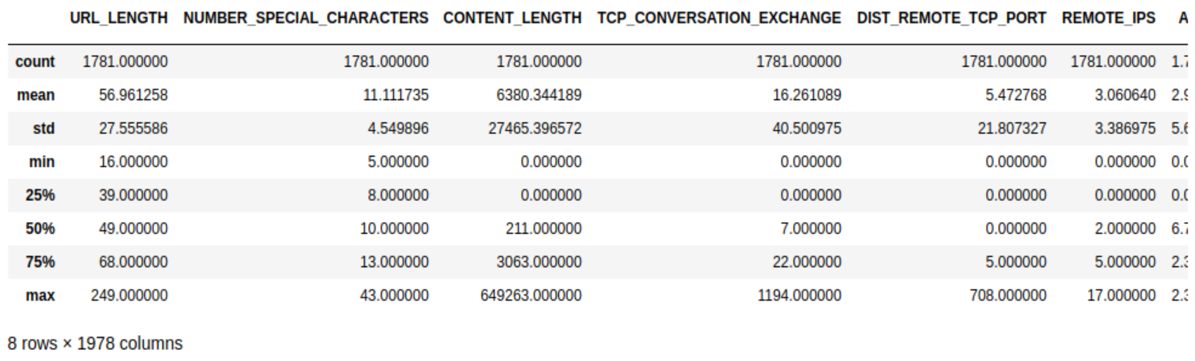
```
n_records = data['URL'].count()
n_malicious = data[data['Type']==1].shape[0]
n_benign = data[data['Type']==0].shape[0]
malicious_precentage = float(n_malicious)*100/n_records
benign_precentage = float(n_benign)*100/n_records

# Print the results
print("Total number of records: {}".format(n_records))
print("Number of malicious URLs: {}".format(n_malicious))
print("Number of benign URLs: {}".format(n_benign))
print("Percentage of malicious URLs: {} %".format(round(malicious_precentage,3)))
print("Percentage of benign URLs: {} %".format(round(benign_precentage,3)))
```

```
Total number of records: 1781
Number of malicious URLs: 216
Number of benign URLs: 1565
Percentage of malicious URLs: 12.128 %
Percentage of benign URLs: 87.872 %
```

Figure 2 dataset statistics

Data statistics

Some statistics about the data that gives some intuition after a hot encoding step to convert the string fields of the data into int64.



	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CONTENT_LENGTH	TCP_CONVERSATION_EXCHANGE	DIST_REMOTE_TCP_PORT	REMOTE_IPS	A
count	1781.000000	1781.000000	1781.000000	1781.000000	1781.000000	1781.000000	1.7
mean	56.961258	11.111735	6380.344189	16.261089	5.472768	3.060640	2.9
std	27.555586	4.549896	27465.396572	40.500975	21.807327	3.386975	5.6
min	16.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	39.000000	8.000000	0.000000	0.000000	0.000000	0.000000	0.0
50%	49.000000	10.000000	211.000000	7.000000	0.000000	2.000000	6.7
75%	68.000000	13.000000	3063.000000	22.000000	5.000000	5.000000	2.3
max	249.000000	43.000000	649263.000000	1194.000000	708.000000	17.000000	2.3

8 rows × 1978 columns

Figure 3 dataset statistics after one hot encoding

Exploratory Visualization

Data Distribution

A count plot indicates how many times does a value in a column occurs. This is the most suitable plot to start with as it is simple and easy to understand.

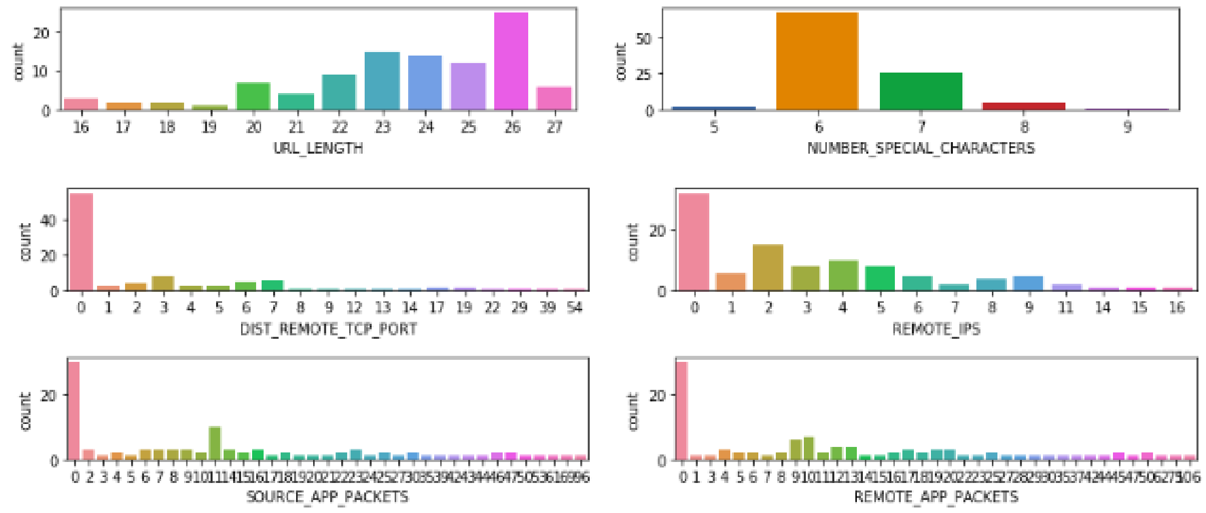


Figure dataset distribution plot

- Plot the features against the its label (malicious = 1, benign = 0)

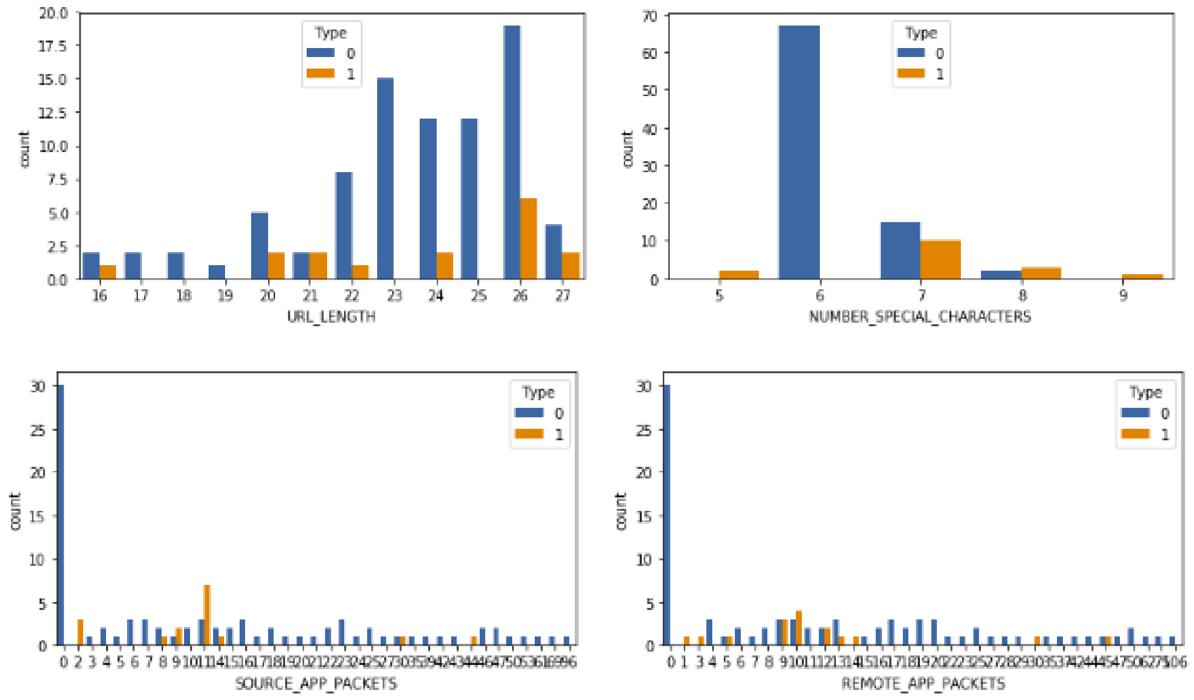


Figure features against its label

observations

The output classes are randomly distributed across the data distribution and there is not enough information to observe the correlation between the features or to indicate the most important features for our model.

<u>Algorithms and techniques:</u>

As we know our dataset is imbalanced and for that reason, I'll use the SMOTE algorithm to balance the data.

1- SMOTE [4]

SMOTE algorithm (Synthetic Minority Oversampling Technique) is most common algorithm that solves the problem of imbalanced data. At a high level, SMOTE creates synthetic observations of the minority class (malicious URLs) by:

1. Finding the k-nearest-neighbors for minority class observations (finding similar observations)

2. Randomly choosing one of the k-nearest-neighbors and using it to create a similar, but randomly tweaked, new observation.

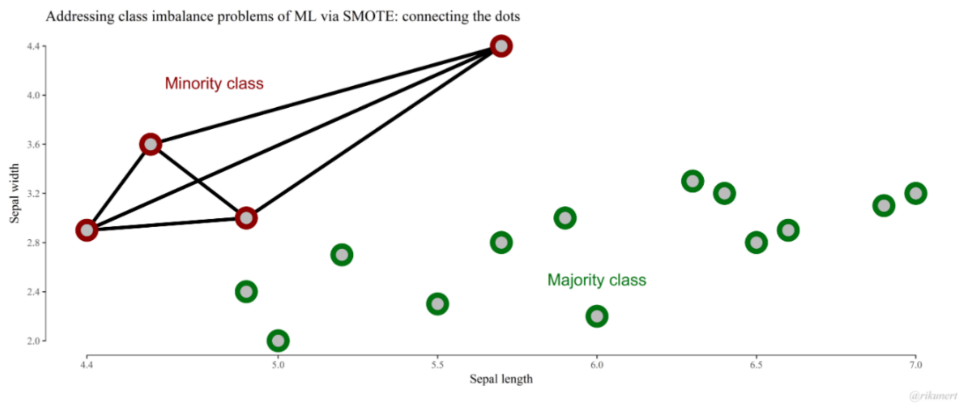


Figure 4 SMOTE algorithm

After applying the algorithm, the dataset should have equal output classes.

2- SVM [5]

Support vector machine is most common algorithm used in binary classification for supervised learning Because the basic principle of SVM is to maximize the distances of samples to a boundary that separates the classes. This means we are necessarily talking about a 2-class problem. But SVM's can be used for multi-classification as well, by combining multiple 2-class SVM's.

SVM used in paper and makes a great result.

There for I'll use this algorithm in this project and try to find the best parameters for better accuracy using the search grid.

3- Decision tree algorithms [6]

Random Forest, CART is the most common decision tree algorithms that works well with classification problems, we will use it and compare the results with the SVM algorithm.

Random Forest is an “ensemble learning” technique consisting of the aggregation of a large number of decision trees, resulting in a reduction of variance compared to the single decision trees

The CART algorithm is based on Classification and Regression Trees. A CART tree is a binary decision tree that is constructed by splitting a node into two child nodes repeatedly, beginning with the root node that contains the whole learning sample.

Bench marking model

For this project, I will be using simple guessing as a benchmark to prove that the model has at least learned something from the data. Sklearn has a Dummy classifier [7] for this that will help implement the simple guessing. If my model does not beat simple guessing, then it really did not learn much from the data.

III. Methodology

Data Processing

Steps used to preprocess the data

1. Separating the target variable 'target' column into another variable to use it for splitting the dataset.

2. Replace the nan values with a zero

3.Get dummy objects to Convert categorical variable into dummy/indicator variables

4. Splitting the dataset into training and testing sets with testing ratio of 30%.

5. Applying SMOTE algorithm to get balanced data and the ration of the output classes are equal (1565 for each class)

Implementation

- As mentioned, the dataset was loaded and analyzed to extract some useful information. Then, the dataset was cleaned and encoded. Then the dataset splatted into two parts: 70% training and 30% testing. It is time to implement our machine learning algorithms:
- Fitting the models:

  The 'Support Vector Machine' was the first model to be trained on the dataset. After that, all the models were trained one by one ('Decision Tree', 'Random Forest').

- Evaluating the models:

  Using training and testing sets and evaluate the metrics for both training and testing sets for each model.

- Comparing the results:

  Using our evaluation metrics as mentioned in the metrics section.

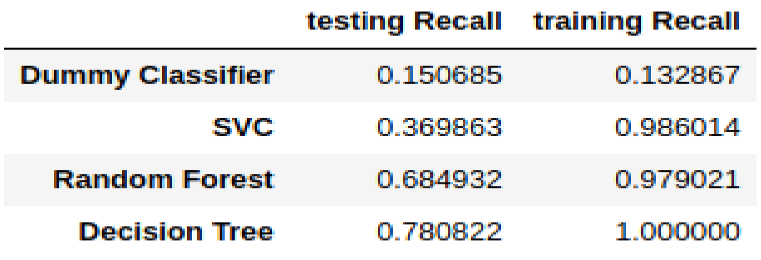
  Those steps will be followed for both the imbalanced and balanced data.

- Improvements:

  Enhance the best model Using hyper-parameter-tuning search grid technique.

  Results:

  For imbalanced data:



	testing Recall	training Recall
Dummy Classifier	0.150685	0.132867
SVC	0.369863	0.986014
Random Forest	0.684932	0.979021
Decision Tree	0.780822	1.000000

Figure 5 results of different algorithms for imbalanced data

As we see the best algorithm based on the recall metric is the Decision Tree algorithm with 1.0 for the training and 0.78 for the testing.

There is a high variance between testing and training results, and it’s may be because of the overfitting. This is a problem in this project and to overcome the overfitting we need to more training data.

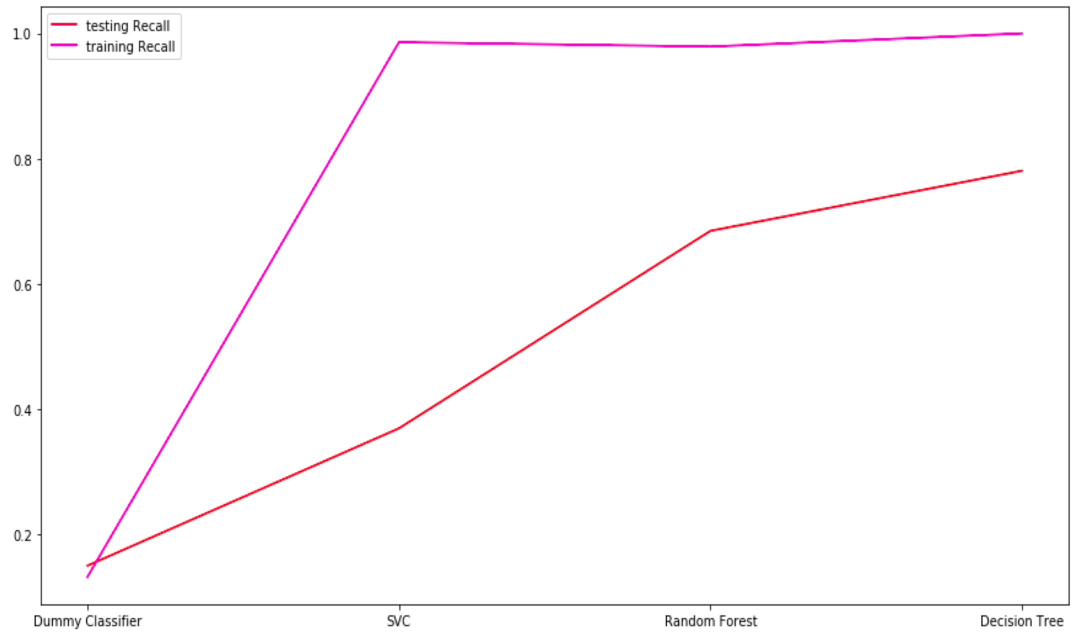
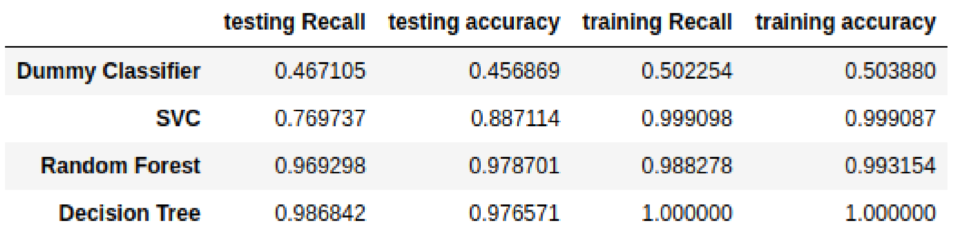


Figure 6 Model evaluation metric plot

For Balanced data:

After applying SMOTE technique to balance the dataset:



	testing Recall	testing accuracy	training Recall	training accuracy
Dummy Classifier	0.467105	0.456869	0.502254	0.503880
SVC	0.769737	0.887114	0.999098	0.999087
Random Forest	0.969298	0.978701	0.988278	0.993154
Decision Tree	0.986842	0.976571	1.000000	1.000000

Figure 7 results of different algorithms for balanced dataset

Again, the Decision Tree algorithms beat the others with 0.986 for the recall metric and 0.976 for the accuracy metric which secure a good save model to predict the malicious URLs that classify 98% of URLs accurately.

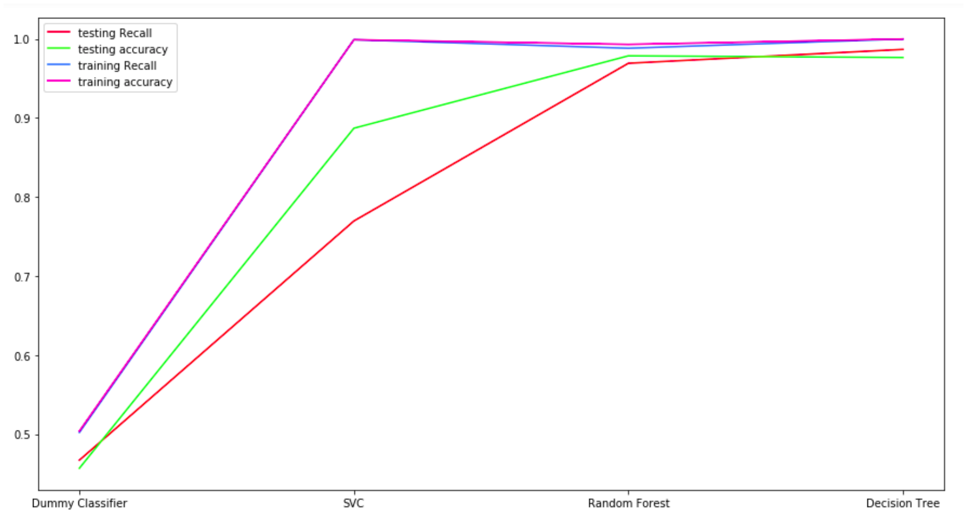


Figure 8 Model evaluation plot

Refinement

- Using search grid technique to find out the best collection of parameters that enhance our model using brute force.

  We search over the following parameters:

- Criterion:

  The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

  The values we used for this parameter: gini, entropy.

- Max depth:

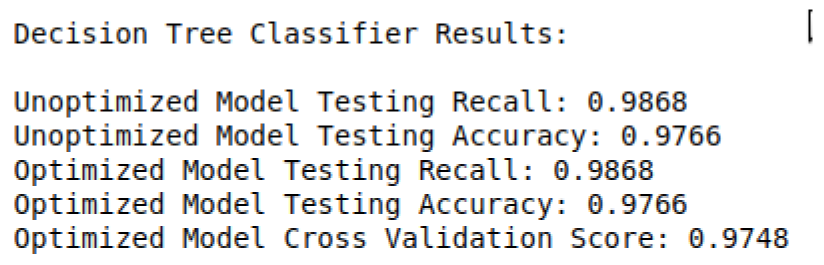
  The maximum depth of the tree. If none, then nodes are expanded until all leaves are pure or until all leaves contain less than min samples split samples.

  The values we used for this parameter: 2,5,10,15,None.

- We will choose those values to search over them.

  By applying this technique on decision tree model for balanced data set

The result is:


```
Decision Tree Classifier Results:

Unoptimized Model Testing Recall: 0.9868
Unoptimized Model Testing Accuracy: 0.9766
Optimized Model Testing Recall: 0.9868
Optimized Model Testing Accuracy: 0.9766
Optimized Model Cross Validation Score: 0.9748
```

The model enhanced by 0.0044 for the recall metric and 0.0053 for the accuracy metric, and the best collection of parameters that makes this this result is:

Entropy as a criterion and max tree depth 15.

IV. Results

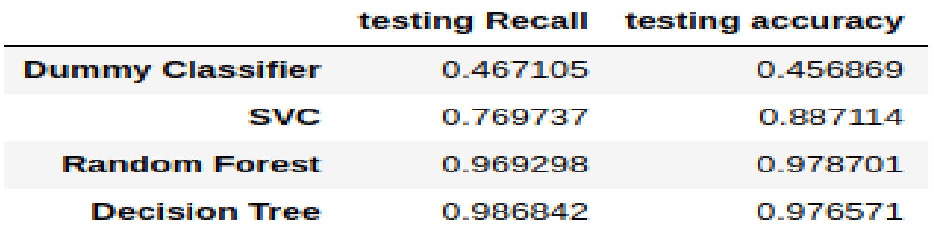
<u>Model evaluation and validation</u>

- The model tested using 30 % of the dataset and evaluated using the recall and accuracy metric
- The final chosen model is the model based on decision tree classifier with Entropy as a criterion and max tree depth 15 which give us 99.12% recall score and 98.19% accuracy score which is great. The model chosen over three different algorithms and the chosen one tuned using grid search technique.
- The final model evaluated using two different data set the original one that have imbalanced output classes and the second one balanced data set after applying the SMOTE algorithm on the original dataset. The imbalanced dataset has an overfitting problem and that problem solved using the balanced dataset.
- I used the cross-validation technique to validate the chosen model and the score was 0.9748 and that’s ensure our model efficiency to handle the different variation in the data.

Justification

The bench mark model was the dummy classifier in sklearn library, and it's recall result is 0.46 for the balanced data and 0.15 for the imbalanced data.

Our model has developed using different machine learning algorithm and all these algorithms achieved higher score than the bench mark model



	testing Recall	testing accuracy
Dummy Classifier	0.467105	0.456869
SVC	0.769737	0.887114
Random Forest	0.969298	0.978701
Decision Tree	0.986842	0.976571

Figure 9 Bench mark model and different algorithms results

V. Conclusion

Free-Form Visualization

Plotting Decision Tree feature importance

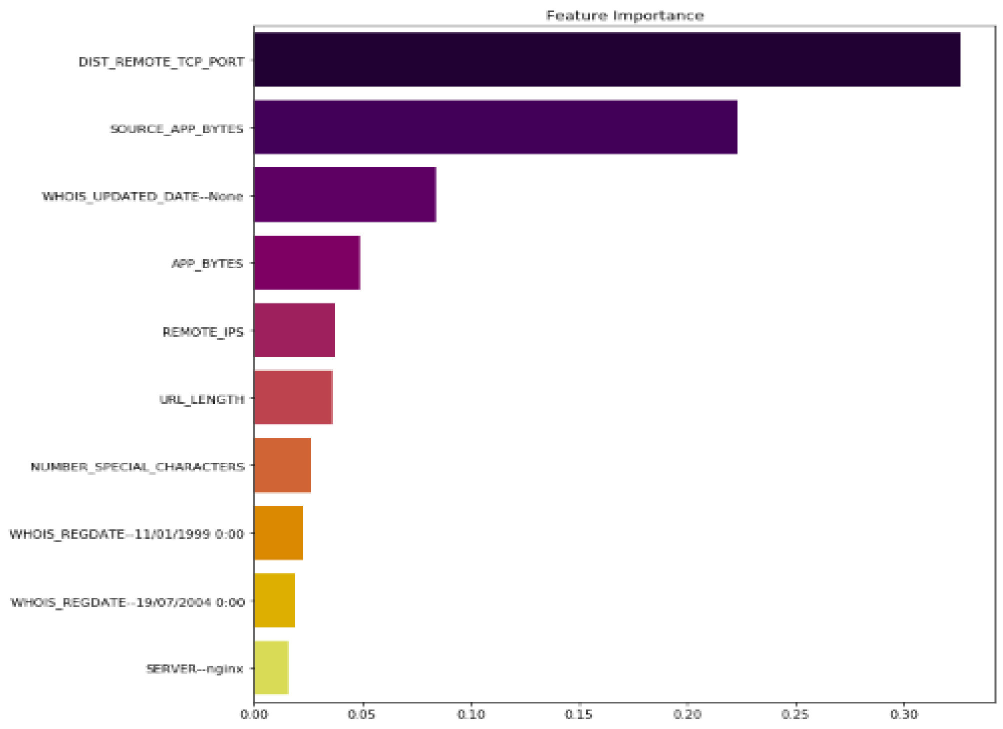


Figure 10 feature importance

This is the top 10 important features in our model that affect the output, as we see the DIST_REMOTE_TCP_PORT and SOURCE_APP_BYTES are the most useful features for our model.

Reflection

This is a binary classification project with a data set from Kaggle platform most related to the information security field.

The target is to classify the URLs whether it's a malicious or benign one.

Project steps:

1- data exploration and data analysis
- Importing the necessary libraries.
- Loading the data set into data frame object.
- Displaying the data and visualize its features.
- Printing some useful statistic about the data.

2- data processing
- Drop the unique features from our training data.
- Handling the nan values with zeros.

- Encode the dataset with one hot encoding using pandas dummies function.
- Using SMOTE algorithm to get a balanced dataset.
- split the dataset into train / test sets

3- **bench mark model**
- Building a simple bench mark model using sklearn dummy classifier.
- Fit the model with the dataset and get the model score using the recall metric.

4- **Supervised algorithm**
- Applying different machine learning algorithm (SVM, Random Forest, Decision Tree).

5- **Model Evaluation**
- Evaluate the algorithms using the recall and the accuracy metric.
- Compare the results and record the best result.
- Visualize the results for more intuitive meaning.

6- **improvement**
- Enhancing the chosen algorithm to tune its hyper parameters.

Our final model gets an 98.6% score for the recall metric with the Decision tree classifier.

Improvements

Improvements for future work

- Get more feature and using more complex neural network model.
- Get richer dataset with higher number of malicious URLs.

References:

[1] What is the malicious URLs [https://www.vircom.com/blog/what-is-a-malicious-url/]
[2] Kaggle dataset [https://www.kaggle.com/xwolf12/malicious-and-benign-websites]
[3] Imbalanced dataset [https://www.datascience.com/blog/imbalanced-data]
[4] SMOTE algorithm [http://rikunert.com/SMOTE_explained]
[5] Support vector machine [https://medium.com/@LSchultebraucks/introduction-to-support-vector-machines-9f8161ae2fcb]
[6] Decision Trees algorithms [https://medium.com/datadriveninvestor/decision-tree-and-random-forest-e174686dd9eb]
[7]DummyClassifier[https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html]