2018

# Machine Learning HW #1

MALWARE ANALYSIS & DETECTION

AHMED HESHAM ELSHIEKH, 1873337

## Contents

# Introduction

## What is a malware?

“Malware” is short for “malicious software” - computer programs designed to infiltrate and damage computers without the users’ consent. “Malware” is the general term covering all the different types of threats to your computer safety such as viruses, spyware, worms, trojans, rootkits and so on.

## How does it affect computers?

The most known form of malwares is viruses, which infects your PC and delete important files, even whole sectors of your hard drive, and/or clogging up your PC’s memory. As well as, spywares which are very dangerous which might steal your personal data such as passwords, credit cards it keeps a record of your information such as your IP address, the websites visited and your overall computer information, causing you financial harm.

## How do you get infected by a malware?

They mostly come hidden or coupled with other drive by downloads on websites. Downloading apps from unknown sources or providers which might have a malware waiting to attack your handheld device/PC.

## What can be done to prevent those malwares?

There is a lot of machine learning algorithms which tries to learn which pieces of software are malwares and which are safe based on a database containing both files/apps. All it does in a nutshell is extracting a set of features and give them weights so the algorithm can differentiate between both.

## Problem and approach

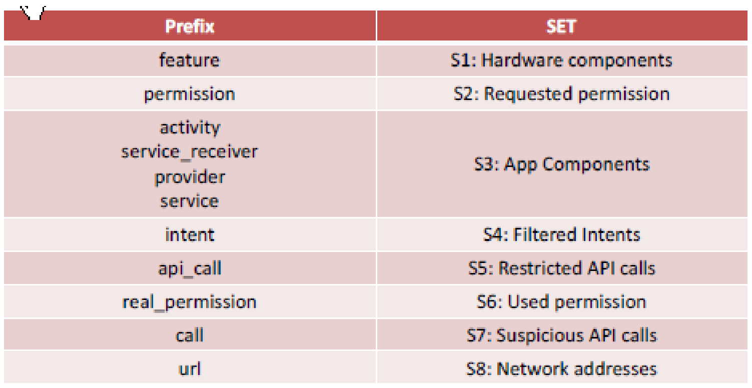
### Methodology

Problem chosen is detect malwares from non-malwares

1. After downloading the Drebin dataset, files were categorized to safe and malware files using the csv file.
2. Features were extracted from each file, files were labeled by 1 if malware, 0 otherwise. All of the files were used to get the number of features occurrences.
3. Depending on the features occurrences, some features will be chosen due to their impact on the results of the training of our model.
4. Calculating the accuracy, recall, f1_score Precision

### Features

We have a dataset downloaded from Drebin which contains a file, this file shows that an application with properties categorized among 8 properties/categories, which we use to extract a feature vector for the application which simply implies the occurrences of each category in the previously mentioned file [1, 5, 3, 4, 0, 10, 9, 0] this implies that feature S1 occurred only once, feature S6 occurred ten times and so on.



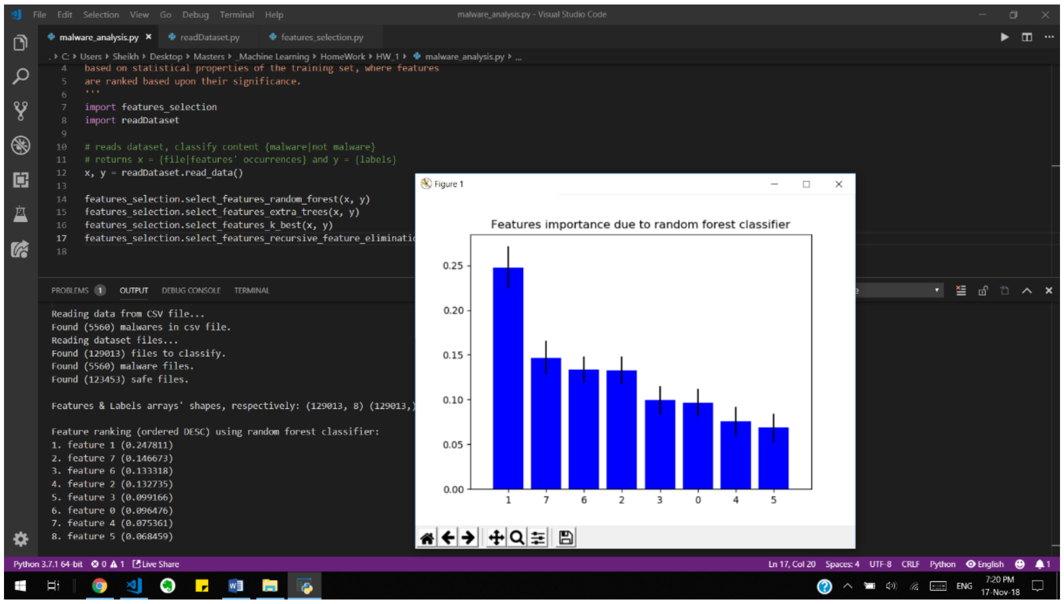
| Prefix | SET |
|---|---|
| feature | S1: Hardware components |
| permission | S2: Requested permission |
| activity<br>service_receiver<br>provider<br>service | S3: App Components |
| intent | S4: Filtered Intents |
| api_call | S5: Restricted API calls |
| real_permission | S6: Used permission |
| call | S7: Suspicious API calls |
| url | S8: Network addresses |

**FIGURE 1 SET OF FEATURES EACH FILE HAS**

## Features Selection

### using unbalanced dataset

Features can be selected using different approaches, either by statistical tests (Univariate selection) which outputs the features that have the strongest relationship with the output variable. Or by recursive feature elimination (RFE) which works by removing the attributes and build a model based on the non-eliminated features, and it uses model accuracy to identify which attributes and/or combination of attributes that contribute the most to predicting the target attribute. Or by using Feature Importance which is a bagged decision trees like random forest or extra tress can be used to determine the features' importance approximately. And other more feature selection techniques and algorithms


File Edit Selection View Go Debug Terminal Help
malware_analysis.py - Visual Studio Code
malware_analysis.py
readDataset.py
features_selection.py
C: › Users › Sheikh › Desktop › Masters › _Machine Learning › HomeWork › HW_1 › malware_analysis.py › ...
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
Figure 1
Features importance due to random forest classifier
Python 3.7.1 64-bit
Live Share
Ln 17, Col 20
Spaces: 4
UTF-8
CRLF
Python
English
7:20 PM
17-Nov-18

```
based on statistical properties of the training set, where features
are ranked based upon their significance.
'''
import features_selection
import readDataset

# reads dataset, classify content {malware|not malware}
# returns x = {file|features' occurrences} and y = {labels}
x, y = readDataset.read_data()

features_selection.select_features_random_forest(x, y)
features_selection.select_features_extra_trees(x, y)
features_selection.select_features_k_best(x, y)
features_selection.select_features_recursive_feature_eliminati
```

```
Reading data from CSV file...
Found (5560) malwares in csv file.
Reading dataset files...
Found (129013) files to classify.
Found (5560) malware files.
Found (123453) safe files.

Features & Labels arrays' shapes, respectively: (129013, 8) (129013,)

Feature ranking (ordered DESC) using random forest classifier:
1. feature 1 (0.247811)
2. feature 7 (0.146673)
3. feature 6 (0.133318)
4. feature 2 (0.132735)
5. feature 3 (0.099166)
6. feature 0 (0.096476)
7. feature 4 (0.075361)
8. feature 5 (0.068459)
```

**FIGURE 2: FIGURING OUT WHICH FEATURES ARE THE MOST AFFECTING THE CLASSIFICATION USING RANDOM FOREST CLASSIFIER USING UNBALANCED DATASET**

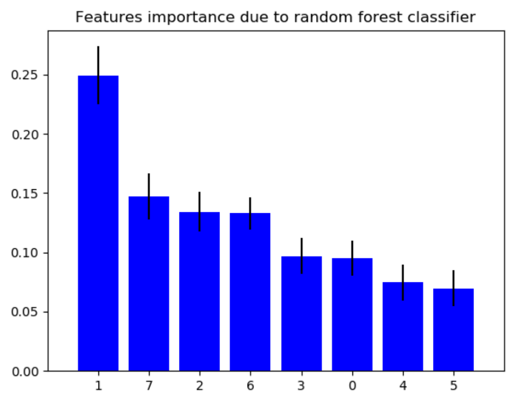
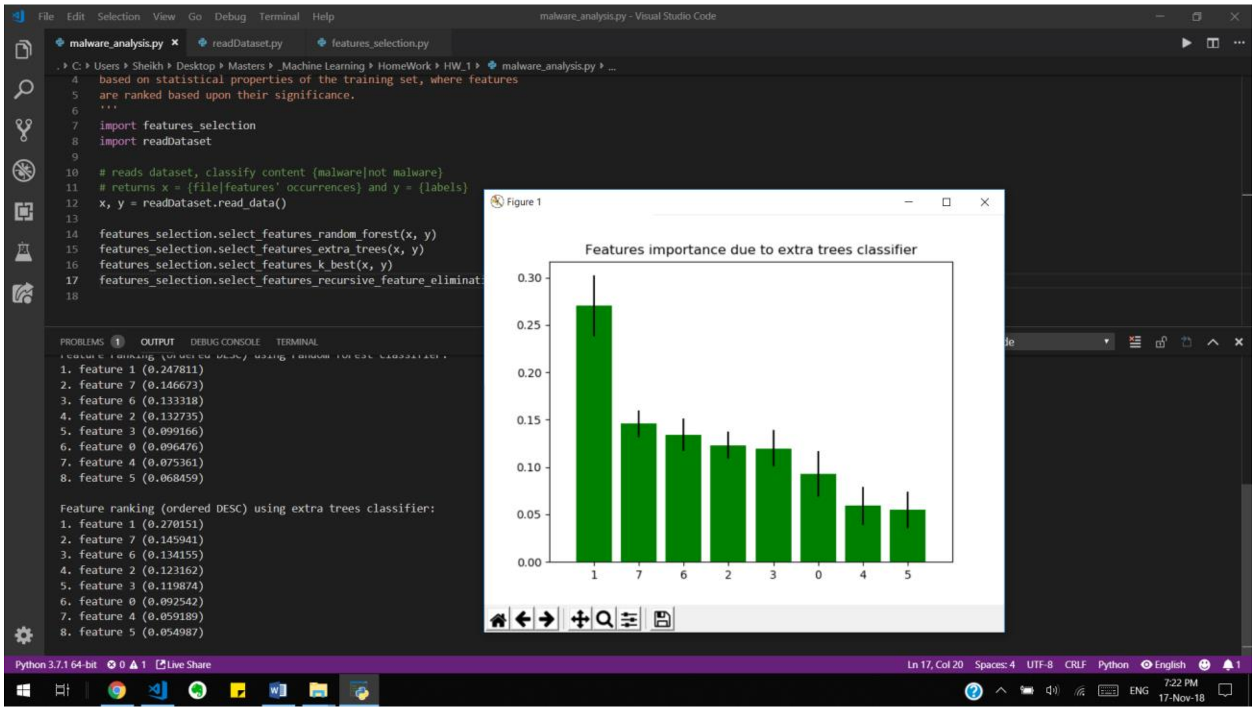


**FIGURE 3 FEATURES IMPORTANCE AS PER RANDOM FOREST CLASSIFIER USING UNBALANCED DATASET**



```
based on statistical properties of the training set, where features
are ranked based upon their significance.
'''
import features_selection
import readDataset

# reads dataset, classify content {malware|not malware}
# returns x = {file|features' occurrences} and y = {labels}
x, y = readDataset.read_data()

features_selection.select_features_random_forest(x, y)
features_selection.select_features_extra_trees(x, y)
features_selection.select_features_k_best(x, y)
features_selection.select_features_recursive_feature_eliminat
```

```
1. feature 1 (0.247811)
2. feature 7 (0.146673)
3. feature 6 (0.133318)
4. feature 2 (0.132735)
5. feature 3 (0.099166)
6. feature 0 (0.096476)
7. feature 4 (0.075361)
8. feature 5 (0.068459)

Feature ranking (ordered DESC) using extra trees classifier:
1. feature 1 (0.270151)
2. feature 7 (0.145941)
3. feature 6 (0.134155)
4. feature 2 (0.123162)
5. feature 3 (0.119874)
6. feature 0 (0.092542)
7. feature 4 (0.059189)
8. feature 5 (0.054987)
```

malware_analysis.py - Visual Studio Code
malware_analysis.py
readDataset.py
features_selection.py
C: › Users › Sheikh › Desktop › Masters › _Machine Learning › HomeWork › HW_1 › malware_analysis.py › ...
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
Figure 1
Features importance due to extra trees classifier
Python 3.7.1 64-bit
Live Share
Ln 17, Col 20
Spaces: 4
UTF-8
CRLF
Python
English
7:22 PM
17-Nov-18

**FIGURE 4 FIGURING OUT WHICH FEATURES ARE THE MOST AFFECTING THE CLASSIFICATION USING EXTRA TREES CLASSIFIER USING UNBALANCED DATASET**

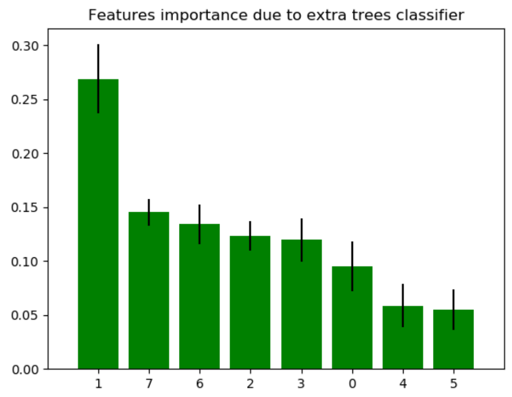
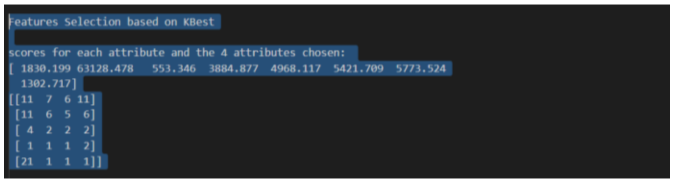
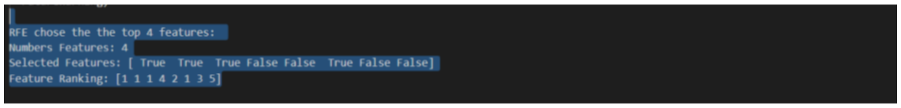


**FIGURE 5 FEATURES IMPORTANCE AS PER EXTRA TREES CLASSIFIER USING UNBALANCED DATASET**



```
Features Selection based on KBest

scores for each attribute and the 4 attributes chosen:
[ 1830.199 63128.478   553.346  3884.877  4968.117  5421.709  5773.524
  1302.717]
[[11  7  6 11]
 [11  6  5  6]
 [ 4  2  2  2]
 [ 1  1  1  2]
 [21  1  1  1]]
```

FIGURE 6 FIGURING OUT WHICH FEATURES ARE THE MOST AFFECTING THE CLASSIFICATION USING K BEST USING UNBALANCED DATASET



```
RFE chose the the top 4 features:
Numbers Features: 4
Selected Features: [ True  True  True False False  True False False]
Feature Ranking: [1 1 1 4 2 1 3 5]
```

FIGURE 7 FIGURING OUT WHICH FEATURES ARE THE MOST AFFECTING THE CLASSIFICATION USING RECURSIVE FEATURES ELIMINATION USING UNBALANCED DATASET

But why are we doing this?

3 benefits of performing feature selection before modeling your data are:

1. Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
2. Improves Accuracy: Less misleading data means modeling accuracy improves.
3. Reduces Training Time: Less data means that algorithms train faster.

What can we deduct from the above that

- Random Forest & Extra trees chose S2, S3, S7, S8
- KBest chose S2, S7, S6, S5
- Random features elimination chose S1, S2, S3, S6

So, these values have the most impact on the training models, thus it makes up the criteria for the evaluation process

## using balanced dataset

So, basically the very same thing was done except changing the dataset size instead of 5560 malware files and 120,000+ safe files, we are now using 5560 safe files as well as 5560 malware files, thus avoiding having biased models

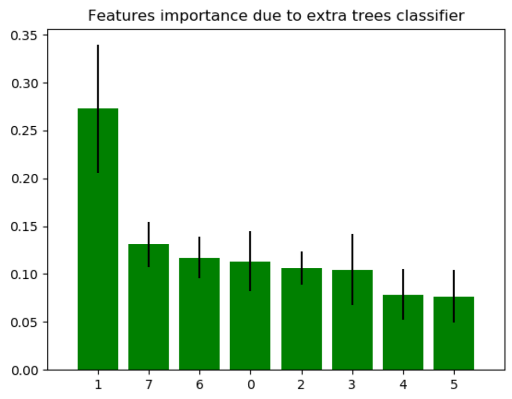


**FIGURE 8 FEATURES IMPORTANCE AS PER EXTRA TREES CLASSIFIER USING BALANCED DATASET**

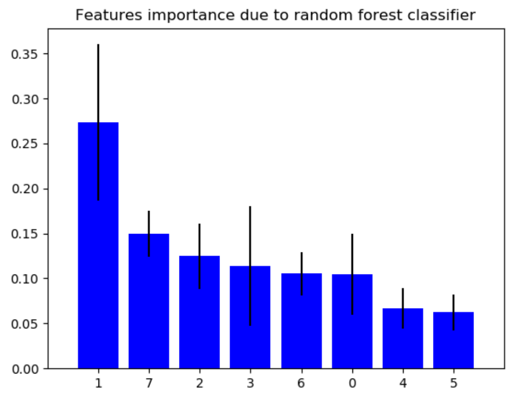
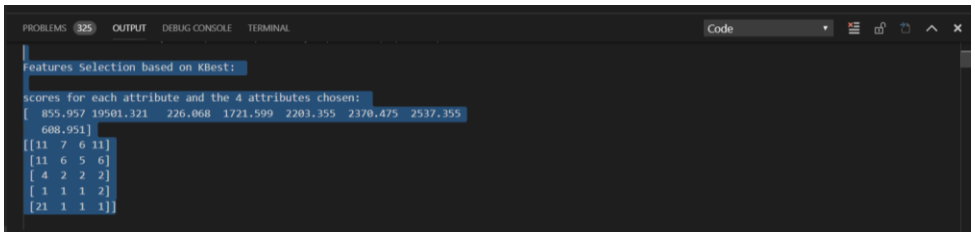
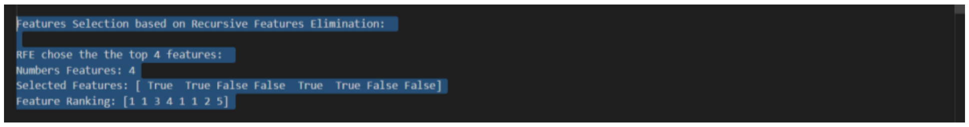


**FIGURE 9 FEATURES IMPORTANCE AS PER RANDOM FOREST CLASSIFIER USING BALANCED DATASET**



```
Features Selection based on KBest:

scores for each attribute and the 4 attributes chosen:
[  855.957 19501.321   226.068  1721.599  2203.355  2370.475  2537.355
   608.951]
[[11  7  6 11]
 [11  6  5  6]
 [ 4  2  2  2]
 [ 1  1  1  2]
 [21  1  1  1]]
```

**FIGURE 10 FIGURING OUT WHICH FEATURES ARE THE MOST AFFECTING THE CLASSIFICATION USING K BEST USING BALANCED DATASET**



```
Features Selection based on Recursive Features Elimination:

RFE chose the the top 4 features:
Numbers Features: 4
Selected Features: [ True  True False False  True  True False False]
Feature Ranking: [1 1 3 4 1 1 2 5]
```

**FIGURE 11 FIGURING OUT WHICH FEATURES ARE THE MOST AFFECTING THE CLASSIFICATION USING RECURSIVE FEATURES ELIMINATION USING BALANCED DATASET**

What can we deduct from the above that

- Extra trees chose S1, S2, S6, S8
- Random Forest chose S2, S8, S3, S4
- KBest chose S2, S7, S6, S5
- Random features elimination chose S1, S2, S5, S6

So, these values have the most impact on the training models, thus it makes up the criteria for the evaluation process

## Model Training

Algorithms used were

1. SVM

   SVM is a supervised machine learning algorithm which can be used for classification or regression problems. it is capable of doing both classification and regression.

   SVM for classification. In particular non-linear SVM. Non-linear SVM means that the boundary that the algorithm calculates doesn't have to be a straight line. The benefit is capturing much more complex relationships between datapoints without having to perform difficult transformations. The downside is that the training time is much longer as it's much more computationally intensive.

2. Recursive Features Elimination

3. Extra Trees

   An "extra trees" classifier, otherwise known as an "Extremely randomized trees" classifier, is a variant of a random forest. Unlike a random forest, at each step the entire sample is used and decision boundaries are picked at random, rather than the best one. In real world cases, performance is comparable to an ordinary random forest, sometimes a bit better.

4. Random Forest

   Is a very robust algorithm and usually perform very well in practice. However, it is complicated to tune.

5. Naïve Bayes

   is a collection of classification algorithms based on Bayes Theorem. It is not a single algorithm but a family of algorithms that all share a common principle, that every feature being classified is independent of the value of any other feature. The algorithm simply allows us to predict a class, given a set of features using probability.

Difference between both Random Forest & Extra Trees:

It seems these are the difference for ET:

1) When choosing variables at a split, samples are drawn from the entire training set instead of a bootstrap sample of the training set.

2) Splits are chosen completely at random from the range of values in the sample at each split.

The result from these two things are many more "leaves".

## Performance

A good metrics for this problem is F1 score since it combines precision and recall. And in this problem, both precision and recall are important. Because people want to find all the positive examples and also not to label a negative example as positive.

Data was split into 80% for training and 20% for testing, all data from training set was used.

### Confusion Matrices

| Algorithm | SVM | Extra Trees | RFE | Random Forest |
|---|---|---|---|---|
| **Confusion Matrix** | 24585 88 | 24597 76 | 24545 128 | 24599 74 |
| | 690 440 | 205 925 | 936 194 | 215 915 |

### Using unbalanced Dataset

| Algorithm | Precision | Accuracy | Recall | F1 Score |
|---|---|---|---|---|
| **SVM** | 0.833333 | 0.96984847 | 0.389381 | 0.530760 |
| **Extra Trees** | 0.924076 | 0.98910979 | 0.818584 | 0.868137 |
| **RFE** | 0.602484 | 0.95876448 | 0.171681 | 0.267218 |
| **Random Forrest** | 0.925177 | 0.98879975 | 0.809735 | 0.863615 |

**TABLE 1 DIFFERENT PERFORMANCE MEASURES FOR DIFFERENT ALGORITHMS USED. USING UNBALANCED DATASET**

From the above data we can deduce that Random forest classifier & Extra trees were the best in our problem -among all the algorithms used- due to the use of decision trees and taking the max vote in case of random forest, both random forest classifier and extra trees are meant to tackle variance.And for SVM -It works by using a hyperplane to classify classes into 2 sets (positive and negative for instance) has biggest possible margin in between classes. Plus, the Error allowance (C value) improves accuracy if best c is used, since Recursive features elimination (REF) prunes features to have a smaller feature vector it has less performance issues with big data however it misses on features to train the model efficiently thus we have the above results.

Note that Naïve Bayes was not included in the previous table because it does not work with an unbalanced dataset due to prior probability of the classes calculated will be very big thus the algorithm does not work, in brief.

## Confusion Matrices

| Algorithm | SVM | Extra Trees | RFE | Random Forest | Naïve Bayes |
|---|---|---|---|---|---|
| **Confusion** | 994 108 | 1041 61 | 929 173 | 1031 71 | 174 928 |
| **Matrix** | 160 962 | 63 1059 | 336 786 | 56 1066 | 74 1048 |

## Using balanced Dataset

| Algorithm | Precision | Accuracy | Recall | F1 Score |
|---|---|---|---|---|
| **SVM** | 0.899065 | 0.87949640 | 0.857398 | 0.877737 |
| **Extra Trees** | 0.945536 | 0. 94424460 | 0.943850 | 0.944692 |
| **RFE** | 0.819604 | 0.77113309 | 0.700535 | 0.755406 |
| **Random Forrest** | 0.937555 | 0.94289568 | 0.950089 | 0.943780 |
| **Naïve Bayes** | 0.530364 | 0.54946043 | 0.934046 | 0.676566 |

**TABLE 2 DIFFERENT PERFORMANCE MEASURES FOR DIFFERENT ALGORITHMS USED, USING BALANCED DATASET**

## What was used during implementation?

(1) I used python 3.7.1(64 bits) and pip 18.1 to install the following libraries
(2) Pandas to import and read csv file
(3) NumPy & SciPy for data reading as well as building of matrices
(4) Sklearn for machine learning algorithms and utilities

## Conclusion

Machine learning algorithms are very useful in this problem of malware detection/classification. Some algorithms perform very well due to selection of features, and how interesting insights may tell more about the app and how to examine it for malware.

## Appendix

(1) Drebin effective and explainable detection of android malware in your pocket.pdf
(2) Output of the code is shown in file output.txt, will be found in the zip file along with dataset and codes

# Malware-Analysis

using Drebin dataset to distinguish between malwares and not malwares

## Libraries

- python 3.7.1 (x64)
- pip 18.1
- NumPy 1.15.4
- SciPy 1.1.0
- Sklearn 0.20.0
- Pandas 0.23.4

## How it works

1. Open `malware_analysis.py` and run it

    1. we first use `x, y = read_dataset.read_data()`

        - which uses pandas to import the csv file of drebin dataset
        - prints out the size of the dataset and classify if the file is malware or not based on if the file name is found in csv file or not, and prints number of malwares found and number of safe files
        - Extract features found in each file and if malware labels it 1, 0 otherwise
            - `sample = features_extraction.extract_features(file_content)`
                - uses method implemented in `features_extraction.py`
                - define a features set dictionary including all the features that can be used to detect a malware
                - the method `extract_features(file_content)` creates an empty dictionary to have the number of features occurrences in the input -which is a text file-
                - Whenever a feature extracted from file content, and this feature is found in the features set dictionary it increments its corresponding in the occurrences dictionary we created
                - in the end we copy the values of the dictionary to an array and return it
        - convert the arrays of feature vectors and labels to numpy arrays and return them in variables x & y

    2. And then we select the features we want for demonstration

        - basically all do the same thing in terms of coding them -not in terms of what happens under the hood-, we fit all the data in to get to know which features have the most impact on the models and thus we use them
            - `features_selection.select_features_k_best(x, y)`
                - scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.

        - uses the chi squared (chi^2) statistical test for non-negative features to select 4 of the best features from the dataset
    - `features_selection.select_features_recursive_feature_elimination(x, y)`
        - It works by recursively removing attributes and building a model on those attributes that remain.
        - It works by recursively removing attributes and building a model on those attributes that remain.
    - `features_selection.select_features_extra_trees(x, y)`
    - `features_selection.select_features_random_forest(x, y)`
        - both of random forest and extra trees, we can say they are approximately the same in terms of how they work Bagged decision trees like Random Forest and Extra Trees can be used to estimate the importance of features.

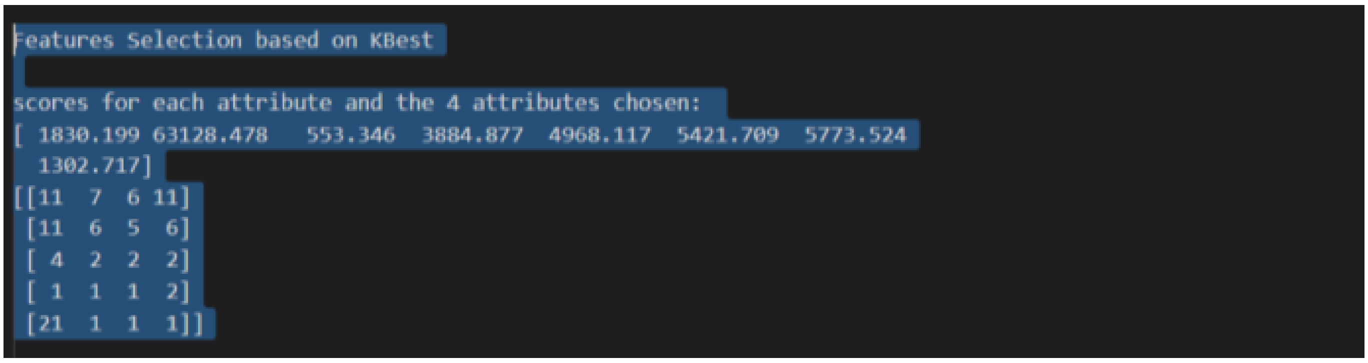
---

*Note* that, the code balances the dataset based on the malware files size, you can change that by removing the part the states 'remove this for using unbalanced dataset' in read_dataset.py file

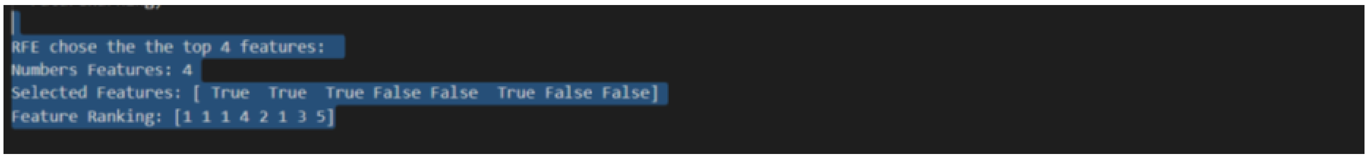
---

**Output:**

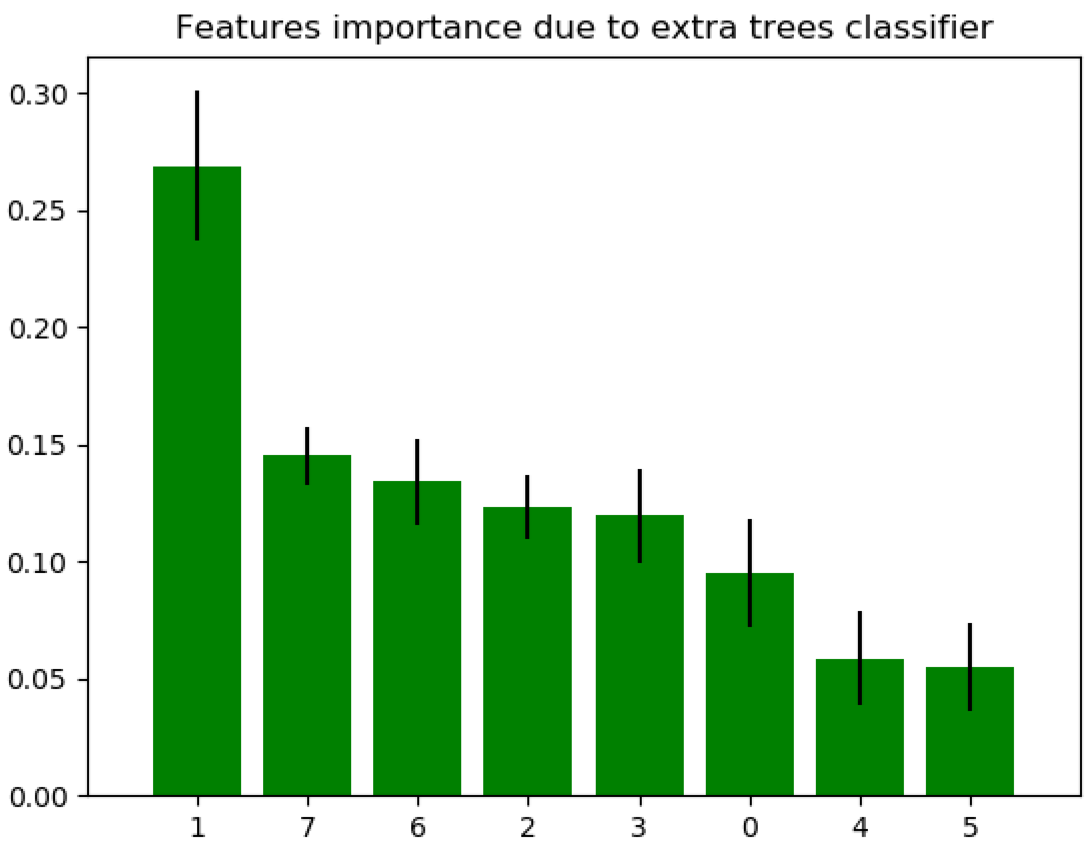
## Features Selection using **unbalanced data**

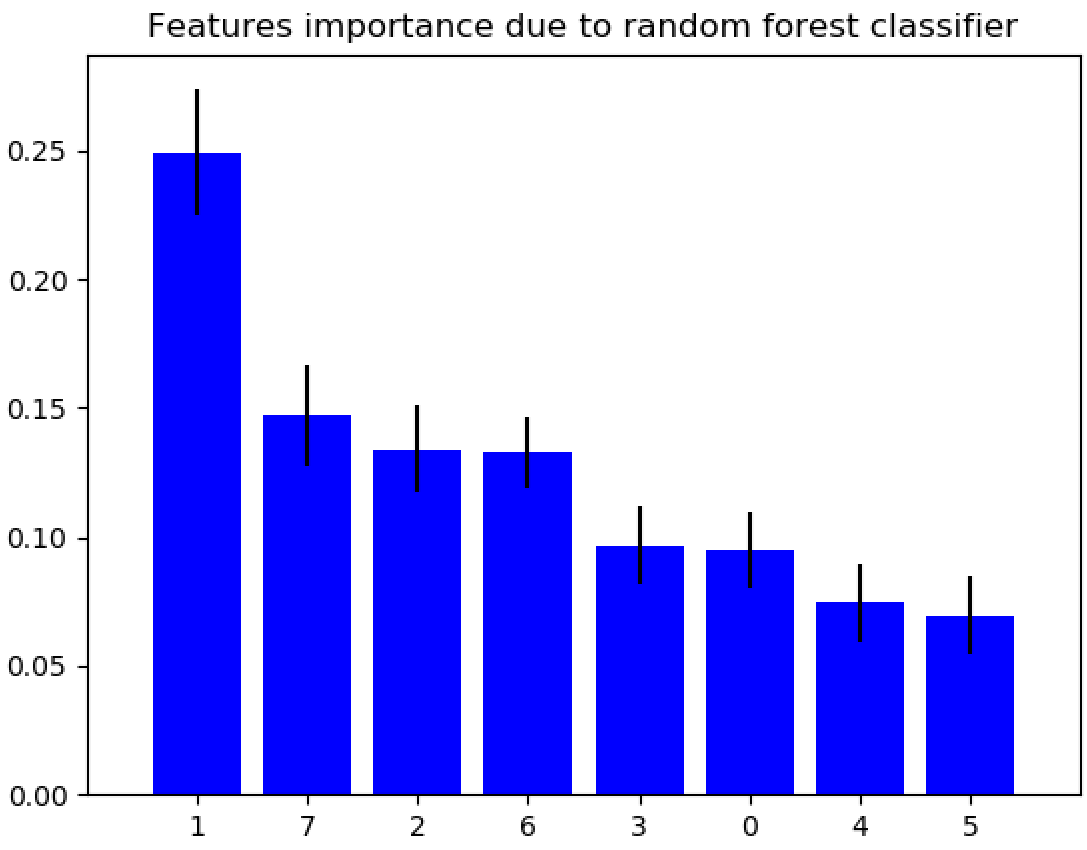


```
Features Selection based on KBest

scores for each attribute and the 4 attributes chosen:
[ 1830.199 63128.478   553.346  3884.877  4968.117  5421.709  5773.524
  1302.717]
[[11  7  6 11]
 [11  6  5  6]
 [ 4  2  2  2]
 [ 1  1  1  2]
 [21  1  1  1]]
```

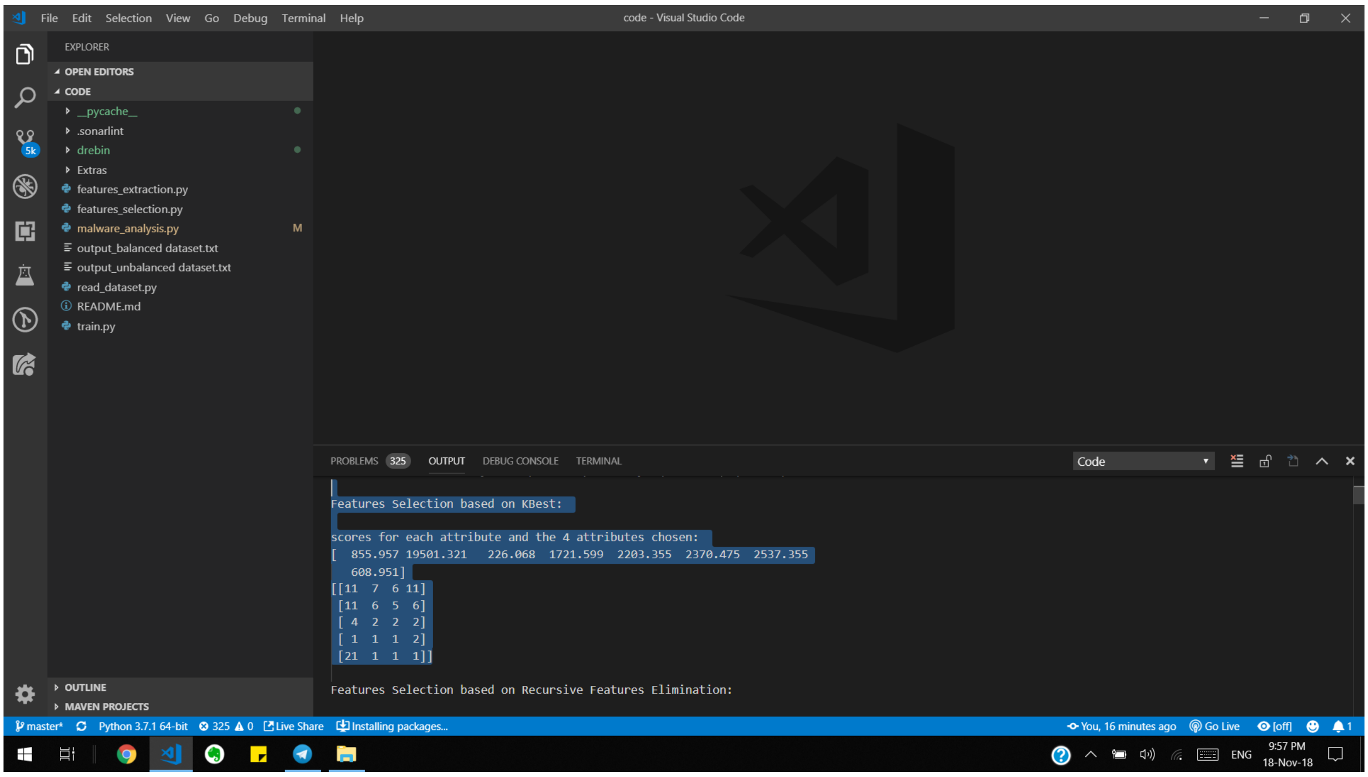


```
RFE chose the the top 4 features:
Numbers Features: 4
Selected Features: [ True  True  True False False  True False False]
Feature Ranking: [1 1 1 4 2 1 3 5]
```

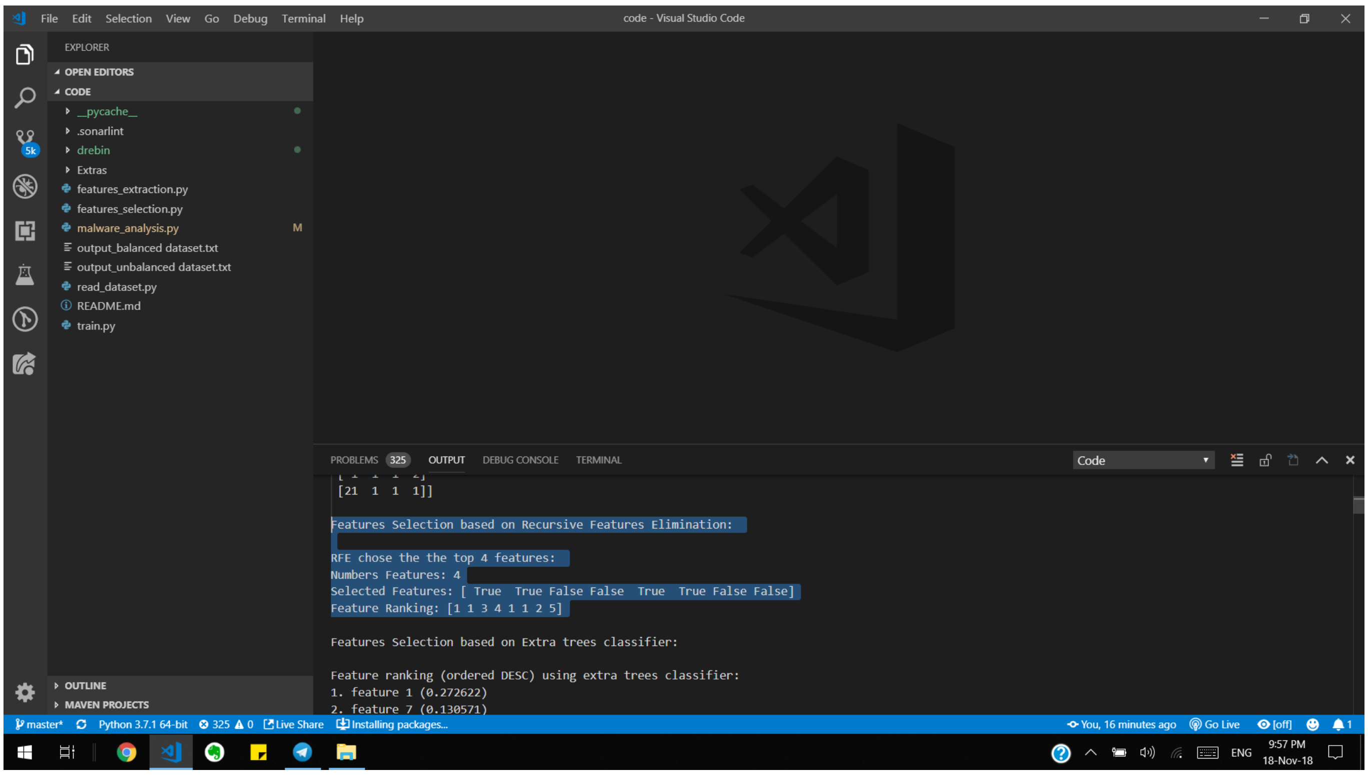




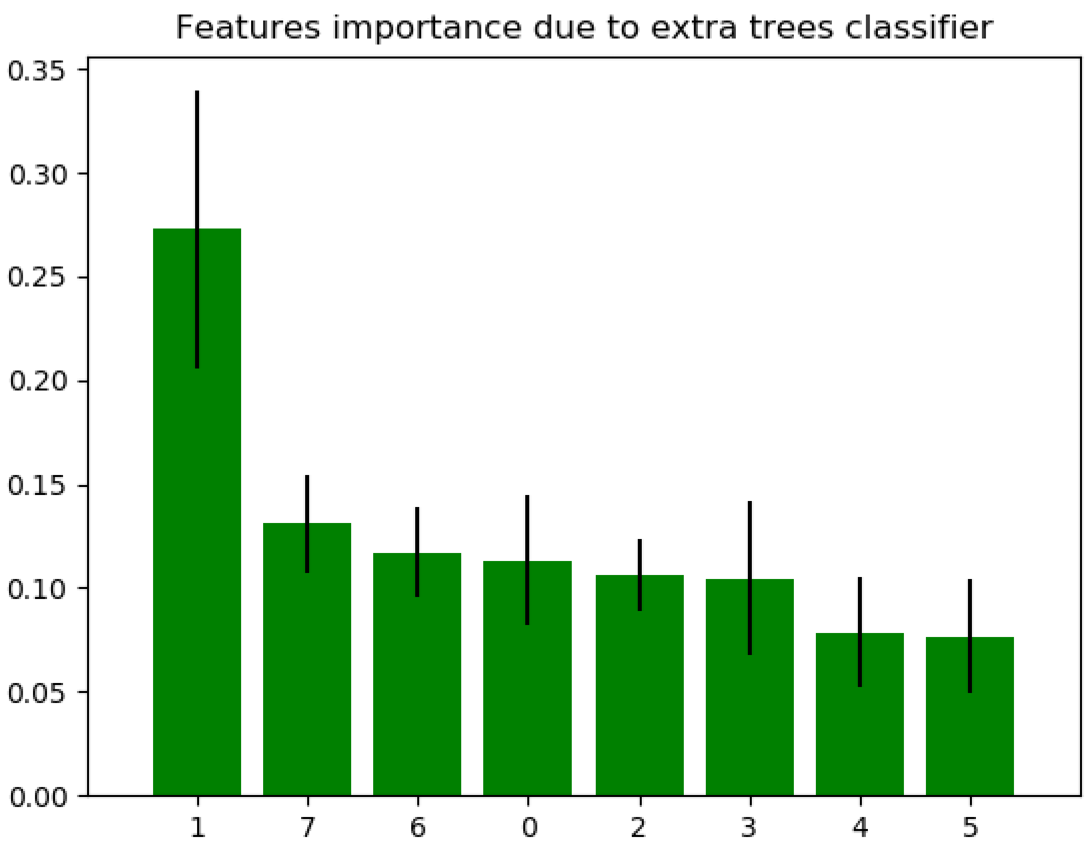
## Features Selection using **balanced data**

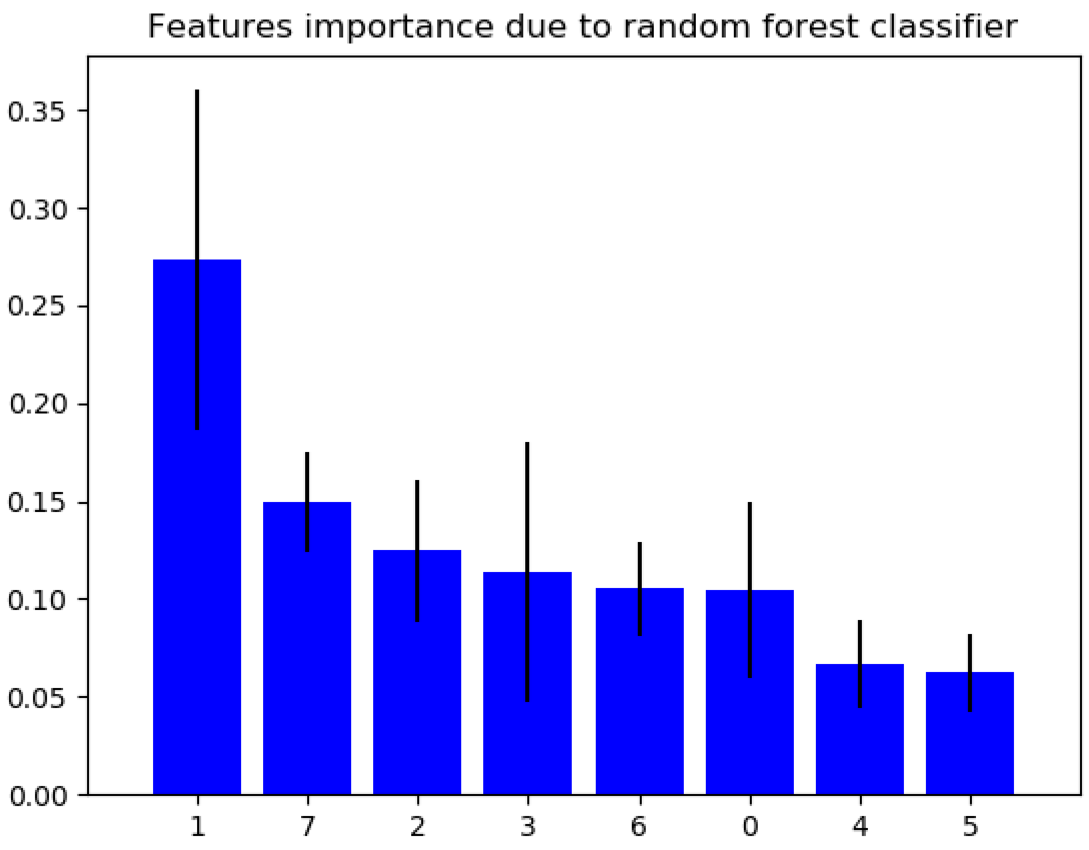


```
Features Selection based on KBest:

scores for each attribute and the 4 attributes chosen:
[  855.957 19501.321   226.068  1721.599  2203.355  2370.475  2537.355
   608.951]
[[11  7  6 11]
 [11  6  5  6]
 [ 4  2  2  2]
 [ 1  1  1  2]
 [21  1  1  1]]

Features Selection based on Recursive Features Elimination:
```



```
 [21  1  1  1]]

Features Selection based on Recursive Features Elimination:

RFE chose the the top 4 features:
Numbers Features: 4
Selected Features: [ True  True False False  True  True False False]
Feature Ranking: [1 1 3 4 1 1 2 5]

Features Selection based on Extra trees classifier:

Feature ranking (ordered DESC) using extra trees classifier:
1. feature 1 (0.272622)
2. feature 7 (0.130571)
```


Features importance due to extra trees classifier
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1
7
6
0
2
3
4
5


Features importance due to random forest classifier
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1
7
2
3
6
0
4
5

---

3. Using `train_test_split()` function from `sklearn.model_selection` to split our data into 2 categories testing (20%) and training data (80%), and print out their shapes for representation purposes
4. Use `train.py` which includes all the training models, as well as their performance metrices
    1. fit the model using the training data
        - SVM
        - Extra Trees
        - Random forest
        - Recursive features elimination
        - Naive Bayes
    2. Predict the x_test
    3. Build confusion matrix, and use the evaluation metrices
        - Accuracy
        - Precision
        - Recall
        - F1 score

---

- But why we do features selection?

    - Three benefits of performing feature selection before modeling your data are:

    1. Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
    2. Improves Accuracy: Less misleading data means modeling accuracy improves.
    3. Reduces Training Time: Less data means that algorithms train faster.

---

- Difference between Extra trees classifier and Random forest classifier?
    - First thing first, extra trees is named Extremely Randomized Tress
    - Secondly, Random Forest and Extremely Randomized Trees differ in the sense that the splits of the trees in the Random Forest are deterministic whereas they are random in the case of an Extremely Randomized Trees (to be more accurate, the next split is the best split among random uniform splits in the selected variables for the current tree).
    - Extra trees seem to keep a higher performance in presence of noisy features.
    - When all the variables are relevant, both methods seem to achieve the same performance, Extra trees seem three times faster than the random forest (at least, in scikit learn implementation)

---

## Performance Metrices

- Accuracy

    - how many instances were classified correctly
    - (TP + TN) / (TP + FN + TN + FP)

- Precision:

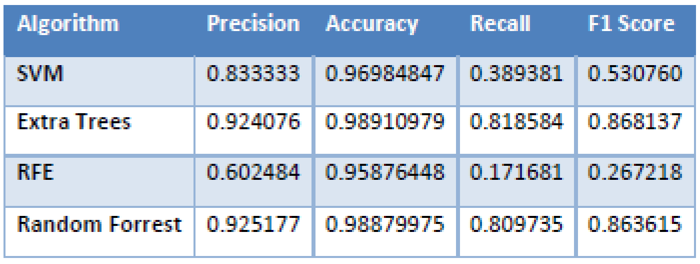
    - It talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.
    - It is a good measure to determine, when the costs of False Positive is high.
    - True positives per Total predicted positives (True Positive/ True Positive + False Positive)

- Recall
  - It actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive).
  - It shall be the model metric we use to select our best model when there is a high cost associated with False Negative.
  - True positives / True Positive + False Negative
- F1 Score
  - It might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives)
  - 2 x ((Precision x Recall) / (Precision + Recall))

---

**Performance metrices**

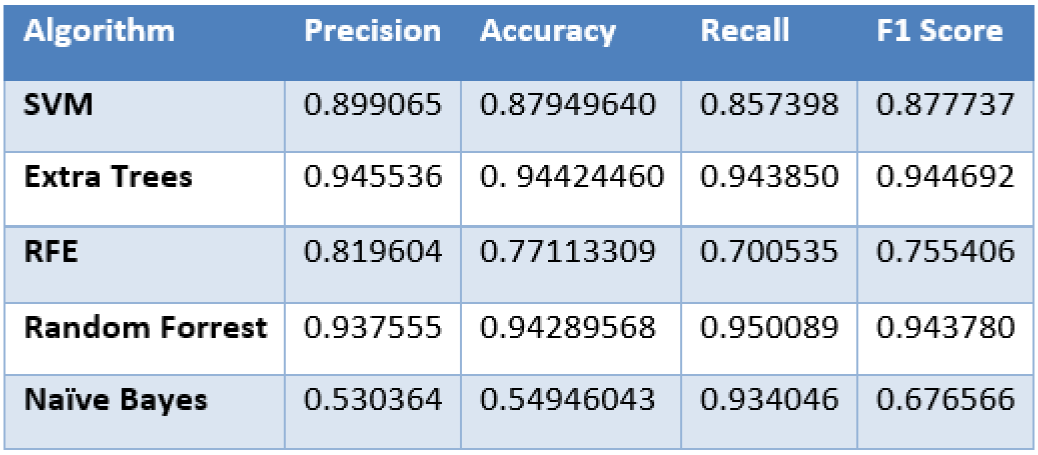
# Performance metrices of algorithms used to train model using **UNBALANCED dataset**



| Algorithm | Precision | Accuracy | Recall | F1 Score |
|---|---|---|---|---|
| SVM | 0.833333 | 0.96984847 | 0.389381 | 0.530760 |
| Extra Trees | 0.924076 | 0.98910979 | 0.818584 | 0.868137 |
| RFE | 0.602484 | 0.95876448 | 0.171681 | 0.267218 |
| Random Forrest | 0.925177 | 0.98879975 | 0.809735 | 0.863615 |

TABLE 1 DIFFERENT PERFORMANCE MEASURES FOR DIFFERENT ALGORITHMS USED

# Performance metrices of algorithms used to train model using **BALANCED dataset**



| Algorithm | Precision | Accuracy | Recall | F1 Score |
|---|---|---|---|---|
| SVM | 0.899065 | 0.87949640 | 0.857398 | 0.877737 |
| Extra Trees | 0.945536 | 0. 94424460 | 0.943850 | 0.944692 |
| RFE | 0.819604 | 0.77113309 | 0.700535 | 0.755406 |
| Random Forrest | 0.937555 | 0.94289568 | 0.950089 | 0.943780 |
| Naïve Bayes | 0.530364 | 0.54946043 | 0.934046 | 0.676566 |

---