Malware Diagnosis in Computer Systems

Avadhesh Gaur (2021AIM1004)

Ishan Tripathi (2021AIM1009)

Contents

1) Motivation

- To implement the different Machine Learning techniques on a real-world dataset
- Making inferences from the dataset & to compare the results and runtime of the different methods used, and also make a note of the observations made along the way

2) Introduction

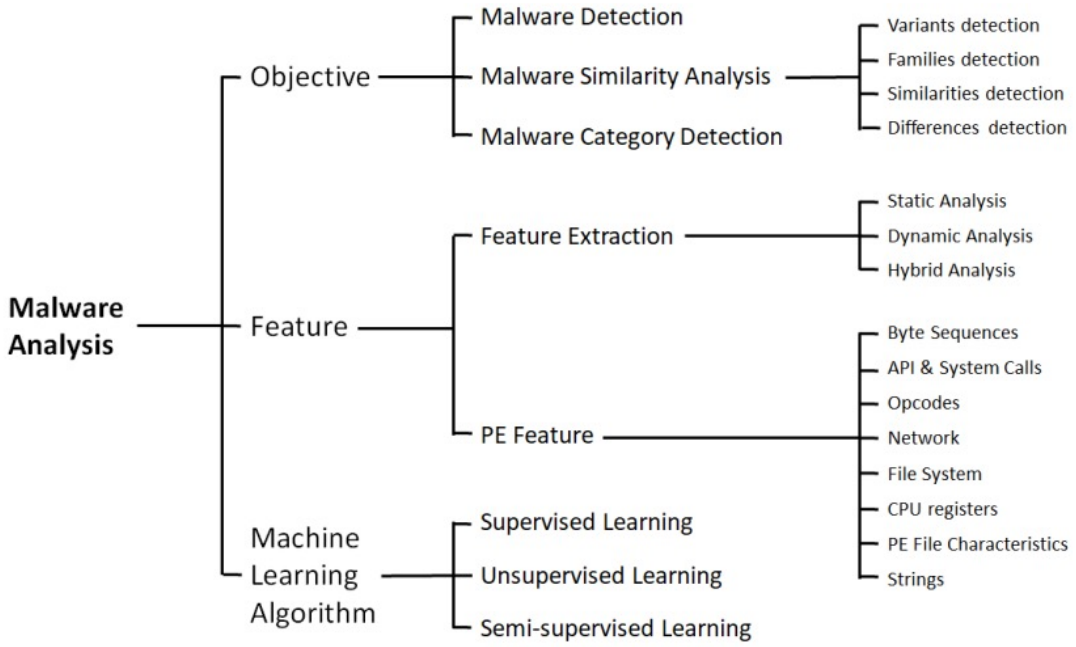
- The malware industry continues to be a well-organized, well-funded market dedicated to evading traditional security measures.
- Once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways.
- With more than *one billion* enterprise and consumer customers, tech companies, such as Microsoft, are taking this problem very seriously and are deeply invested in improving security.

3) Problem Statement

- To develop Machine Learning techniques to predict if a computer system will soon be hit with malware.
- The dataset used to achieve this goal is an unprecedented malware dataset provided by Microsoft to encourage open-source progress on effective techniques for predicting malware occurrences.
- The objective is to help protect more than one billion machines from damage BEFORE it happens.

4) Literature Review

- The paper '*Survey of Machine Learning Techniques for Malware Analysis*', published in 2019, presents a survey covering the wide range of topics given in the figure:
- The paper '*A Machine Learning Approach to Android Malware Detection*', published in 2012, presents a machine learning based system for the detection of malware on Android devices.
    - It extracts several features and trains a One Class Support Vector Machine in an offline (off-device) manner



5) Preliminaries

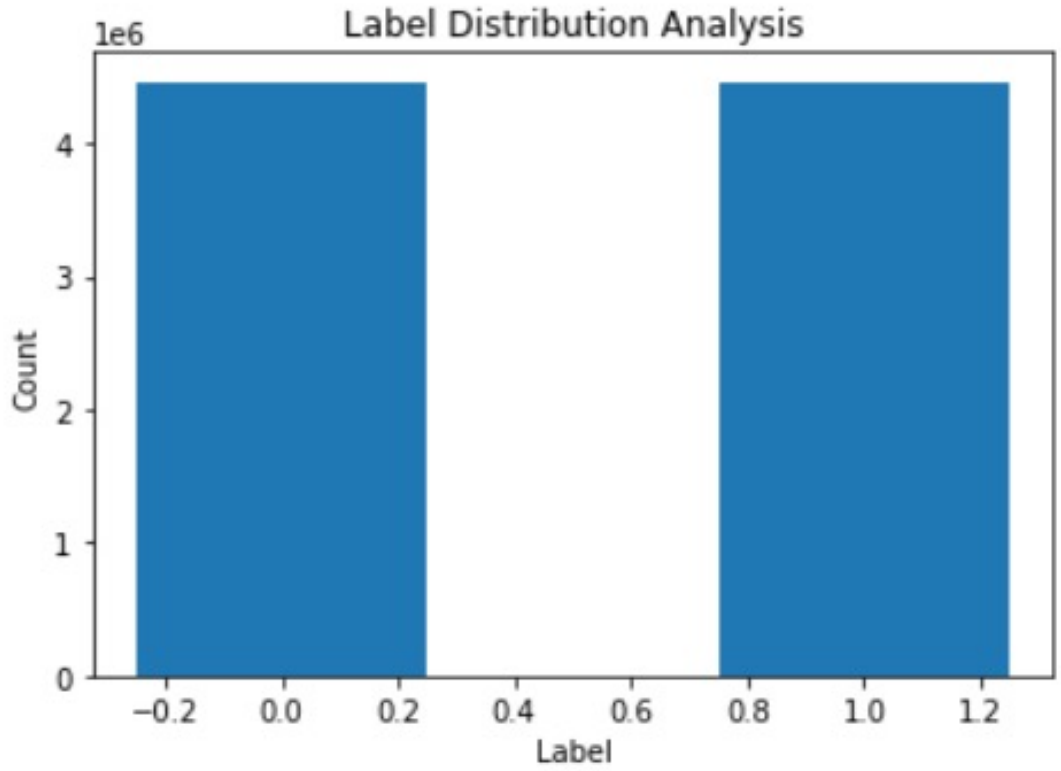
- **Malware:** MALicious softWARE including, computer viruses, ransomware, computer worms, keyloggers, Trojan horses, spywares, etc.
- **Binary Classification:** Task of classifying the elements of a set into two groups based on classification rule(s).
- **Cardinality:** Number of unique items in the feature column
- **Correlation:** A statistical term describing the degree to which two variables move in coordination with one another.
  - The relationship could be causal or not
- **Feature Importance:** Technique that calculates a score for all the input features for a given model.

6) Exploratory Data Analysis (EDA)

- The telemetry dataset provided by Microsoft consists of:
  - 8,921,483 rows
  - 82 columns (or features)
  - 1 target column (or label) having binary values as ground truth
    - 0 denotes Malware was not detected
    - 1 denotes Malware was detected
- We perform the following major analysis to get an idea of the exhaustive dataset:
  - Target label distribution
  - Feature data type distribution

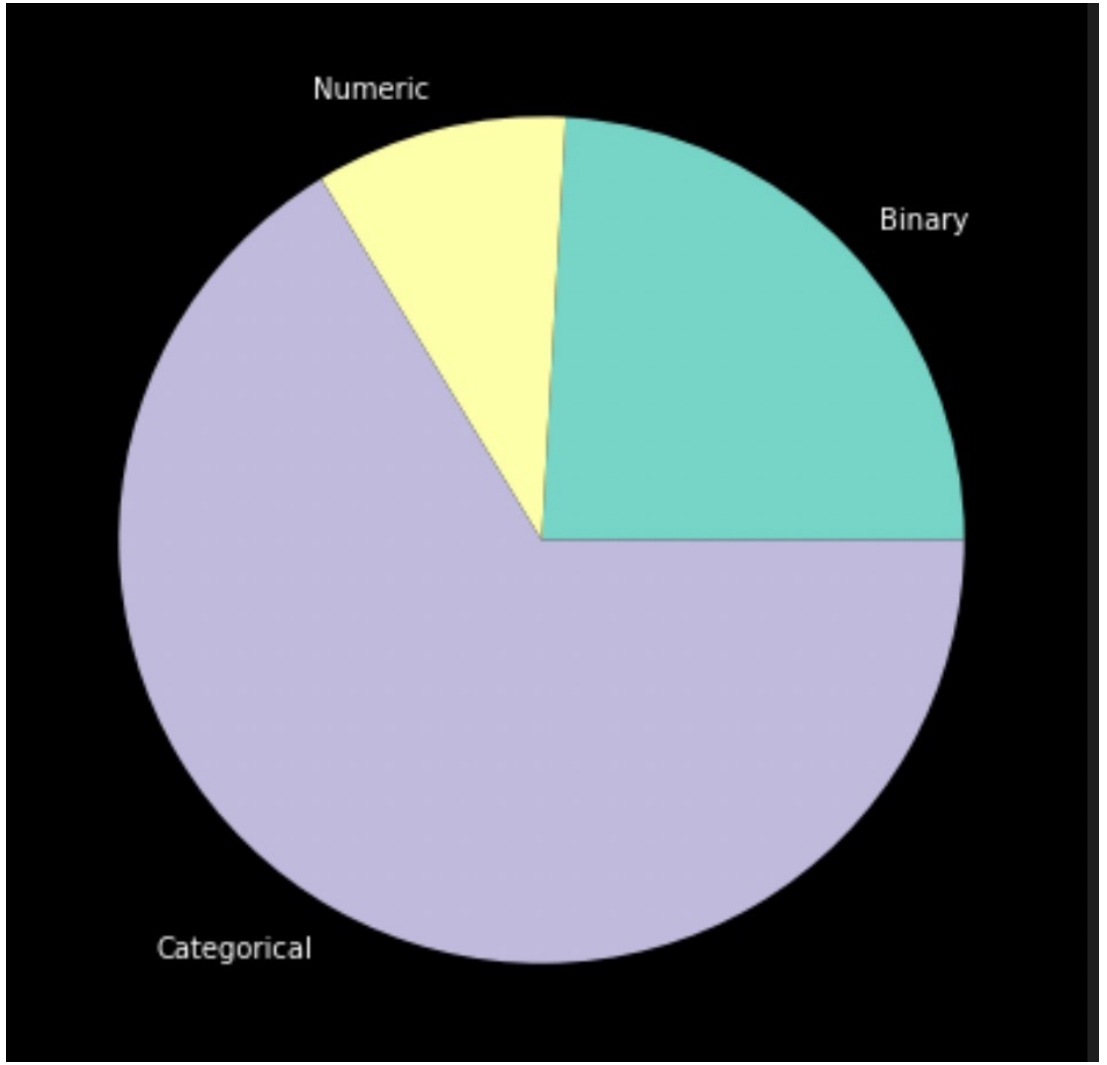
6.a) Target Label Distribution

- Upon analyzing the target label distribution, it was found to be uniform
  - A uniform and in-sync class label distribution with validation/test datasets is desired for better training of ML models

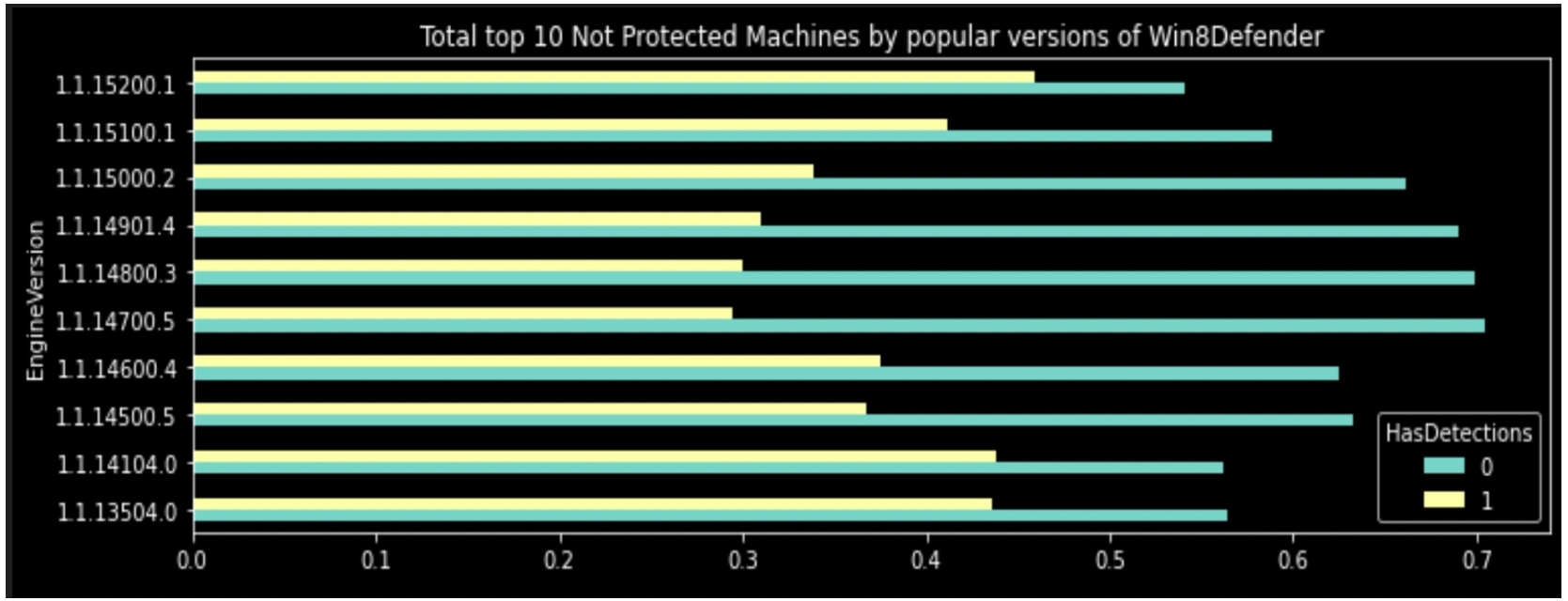


6.b) Feature Data Type Distribution

- We observe the following distribution upon analyzing the feature data types:
  - 21 Binary Features
  - 54 Categorical Features
  - 8 Numeric Features



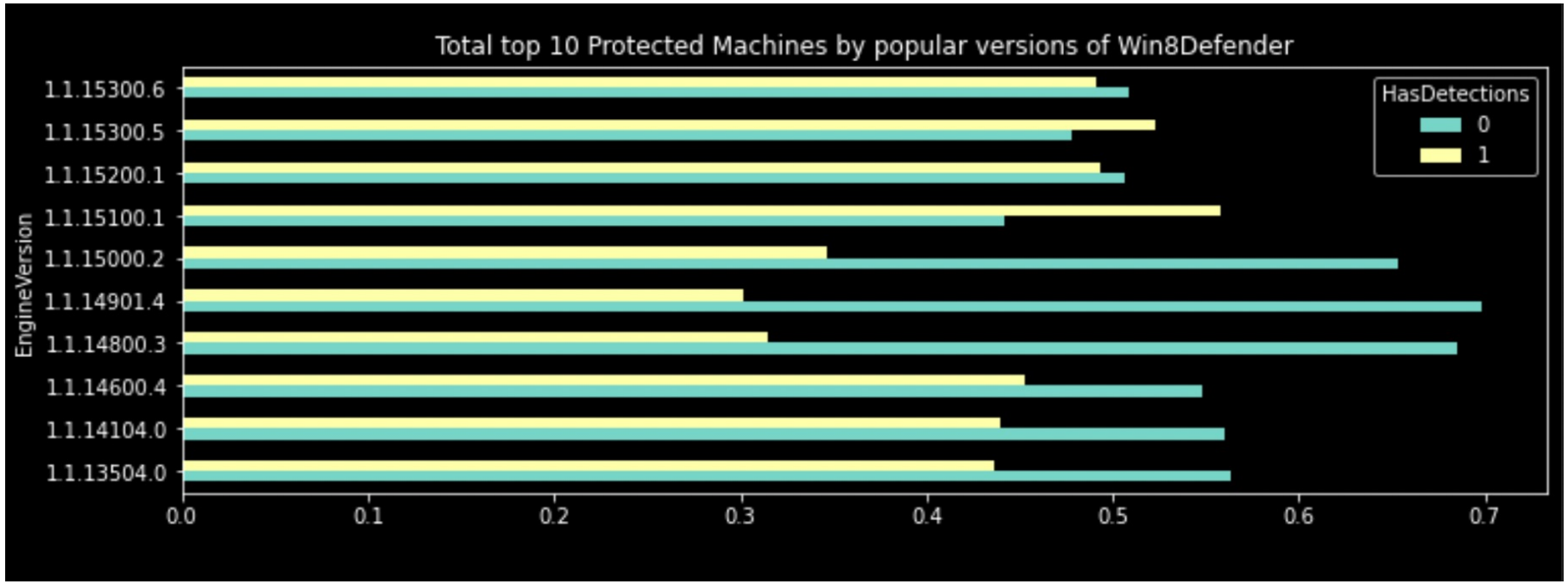
7) Inferences from Dataset



Here we can see that the machines running on version 1.1.151000.2, 1.1.14901.4, 1.1.14800.3 and 1.1.14700.5 had performed well as the large % of the machines hadn't detected any malware/failure in their respective cases.

However, the performance of machine with much updated ver. 1.1.15200.1 is not as great as their predecessors as there is a small gap between percent of machines that had either detected or not detected malwares.

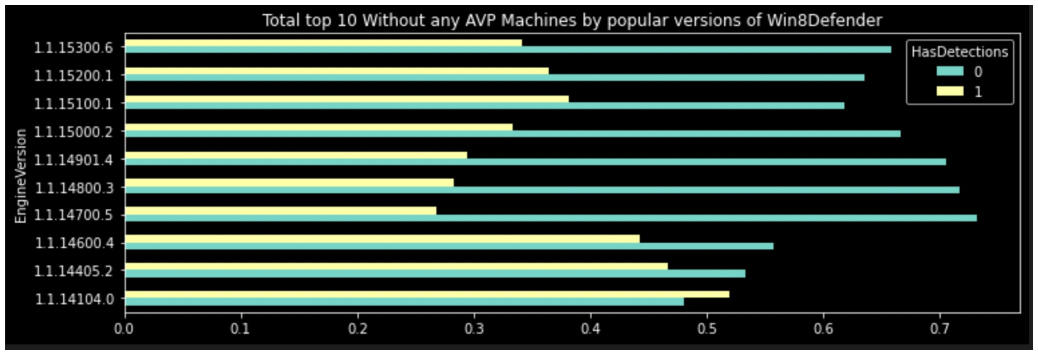
7) Inferences from Dataset (contd.)



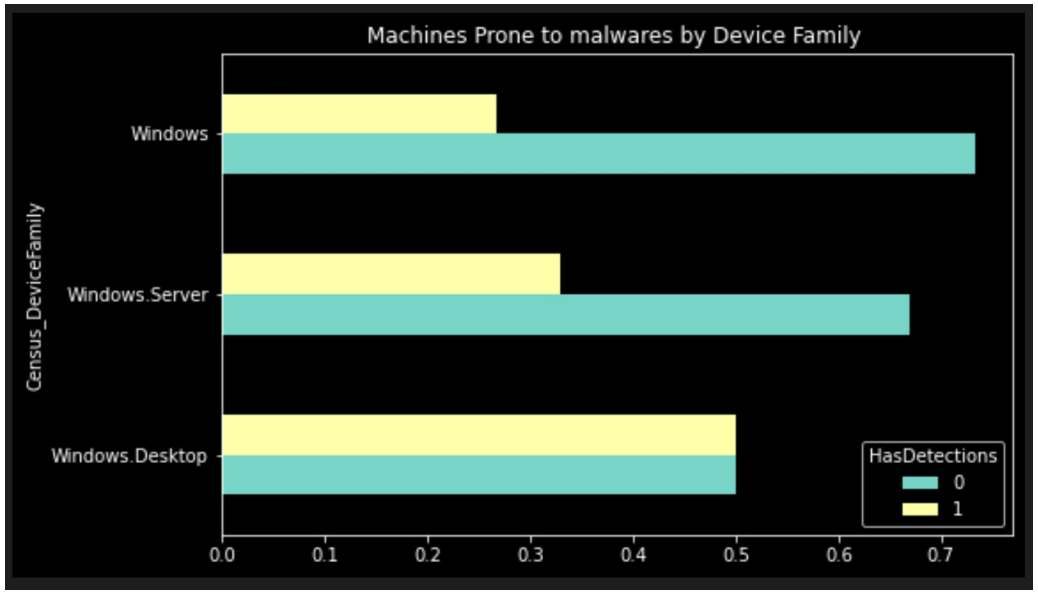
The graph depicts that as the version of Win8Defender gets updated after ver. 1.1.15000.2, the performance of machines degraded as the gap between the percentage of malware infected and not-infected machines decreases.

One more interesting fact that one can infer by combining this graph's result with previous graph's result is that there had been a significant decrease in the machine's performance running version 1.1.15100.1 as more systems had detected malwares as compared to the previous scenario.

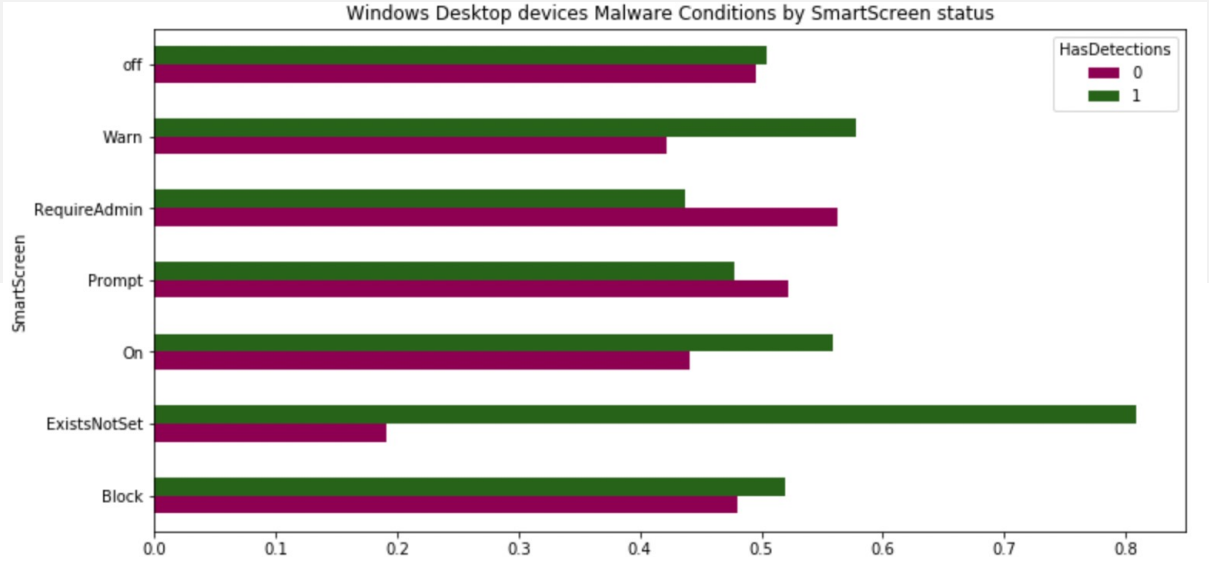
7) Inferences from Dataset (contd.)



- Here we can observe that machine solely running version 1.1.14104.0 gets more prone to malwares as compared to the two previous cases.



7) Inferences from Dataset (contd.)

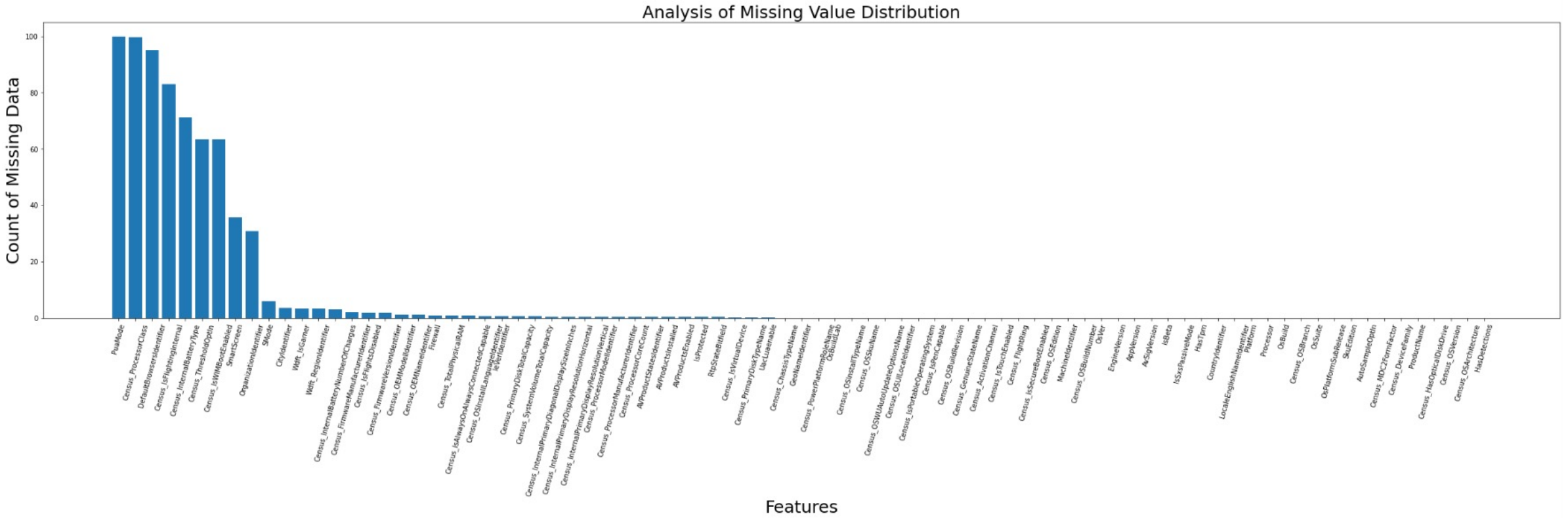


- This graph shows that SmartScreen does not play much significant role in preventing malwares in case of Windows Desktop Machines.
- No matter whether a Smart Screen is ON, OFF, Blocked or Not set, desktop mostly get attacked by malwares.

8) Data Wrangling

- Based on the above EDA, we get the intuition for performing the data pre-processing tasks before passing the input data to the ML models for learning.
- The data pre-processing tasks include:
  - Missing value treatment
  - Removal of features having unusually high cardinality
  - Analyzing the correlation of each feature with the target class label

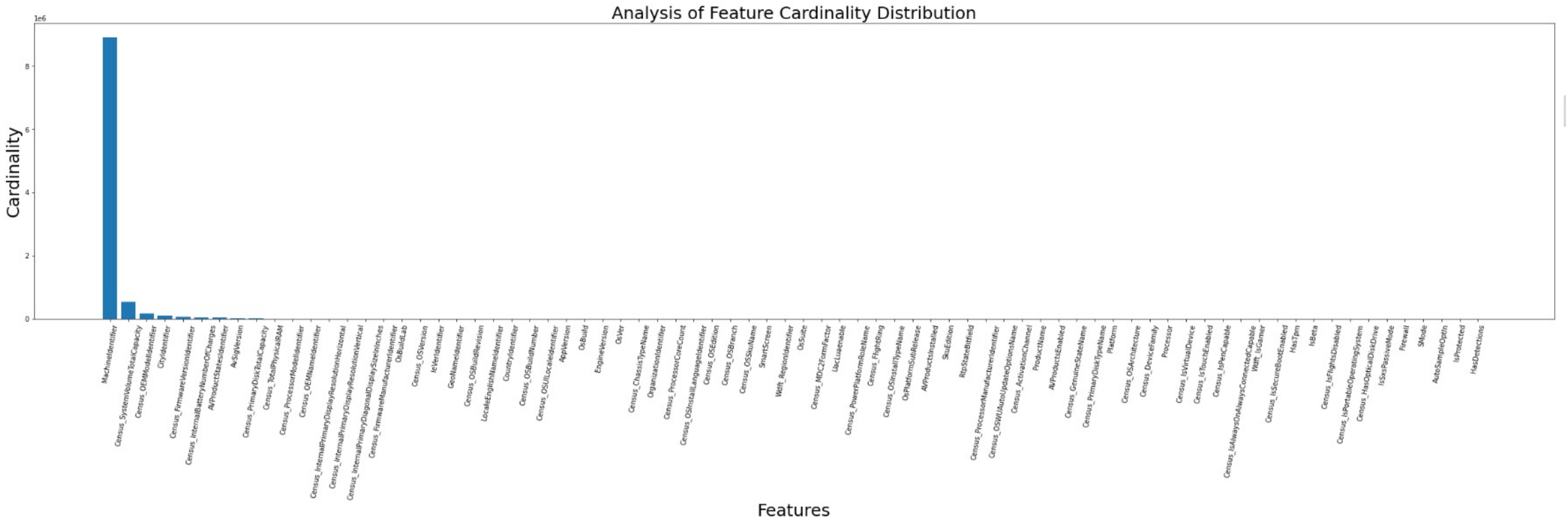
8.a) Missing Value Treatment



8.a) Missing Value Treatment

- The following features were removed due to significantly high percentage of missing values (>=40% as threshold):
  - *PuaMode*: Pua Enabled mode from the service(Potentially Unwanted Application)
  - *Census_ProcessorClass*: A classification of processors into high/medium/low.
  - *DefaultBrowsersIdentifier*: ID for the machine's default browser
  - *Census_IsFlightingInternal*: NA
  - *Census_InternalBatteryType*: NA
  - *Census_ThresholdOptIn*: NA
  - *Census_IsWIMBootEnabled*: NA

8.b) High Cardinality Removal



8.b) High Cardinality Removal

- The following features were removed due to unusually high cardinality(threshold>=500):
  - *MachineIdentifier*: Individual machine ID
  - *Census_SystemVolumeTotalCapacity*: The size of the partition that the System volume is installed on in MB
  - *CityIdentifier*: ID for the city the machine is in
  - *AvSigVersion*: Defender state information e.g. 1.217.1014.0
  - *Census_TotalPhysicalRAM*: Retrieves the physical RAM in MB
  - *Census_InternalPrimaryDiagonalDisplaySizeInInches*: Retrieves the physical diagonal length in inches of the primary display
  - *Census_FirmwareVersionIdentifier*, and a few more

8.c) Correlation Analysis

- Correlation explains how one or more variables are related to each other.
- These variables can be input data features which have been used to forecast our target variable.
- The correlation coefficient is used to represent the strength of relationship between an input feature and the given target class label.
- The following features were removed due to negligible correlation coefficient:
  - *AppVersion*: Defender state information e.g. 4.9.10586.0
  - *Census_DeviceFamily*: AKA DeviceClass. Indicates the type of device that an edition of the OS is intended for. Example values: Windows.Desktop, Windows.Mobile, and iOS.Phone
  - *IsBeta*: Defender state information e.g. false, and a few more

9) Setup for Classification

- The dataset is split in 3 parts:
  - Training (60%)
  - Validation (20%), and
  - Test 20(%)

Note: A sample of the resulting dataset, having 5% of the rows, is used where necessary to save runtime and overcome RAM limitations.

Random Classifier

Here we randomly generate the labels without any model fitting or training.

Accuracy achieved over Training data is 50.05%

Accuracy achieved over Validation data is 49.98%

Accuracy achieved over Test data is 49.97%

From this we can conclude achieving 50% accuracy on classification problems especially bi-classification is quite easy.

Decision Tree Classifier

We applied grid search here on Decision Tree Criterion and Max Depth to find most optimal hyperparameters.

Optimal parameters achieved {'criterion': 'entropy', 'max_depth': 100}.

Accuracy achieved over Training data is 72.18%

Accuracy achieved over Validation data is 63.58%

Accuracy achieved over Test data is 63.56%

Decision trees are not largely influenced by outliers or missing values, and it can handle both numerical and categorical variables.

A small change in the data tends to cause a big difference in the tree structure, which causes instability.

Random Forest Classifier

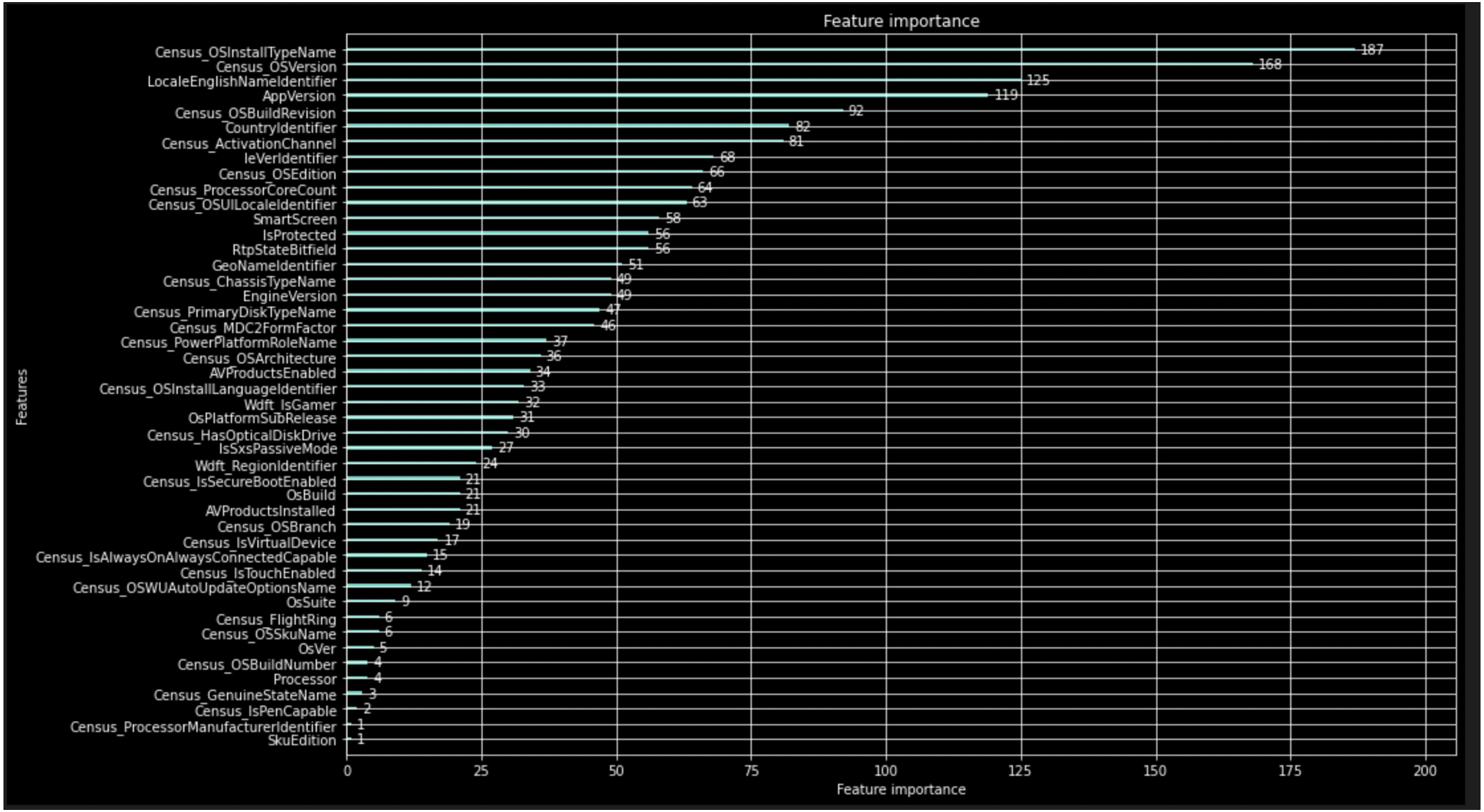
- We have used same optimal parameters here also as in case of Decision tree.
- Accuracy achieved over Training data is 77.32%
- Accuracy achieved over Validation data is 64.05%
- Accuracy achieved over Test data is 63.91%
- Slight improvement in accuracy, however, takes more time in training, compared to decision tree.
- Random forest algorithm avoids and prevents overfitting by using multiple trees.

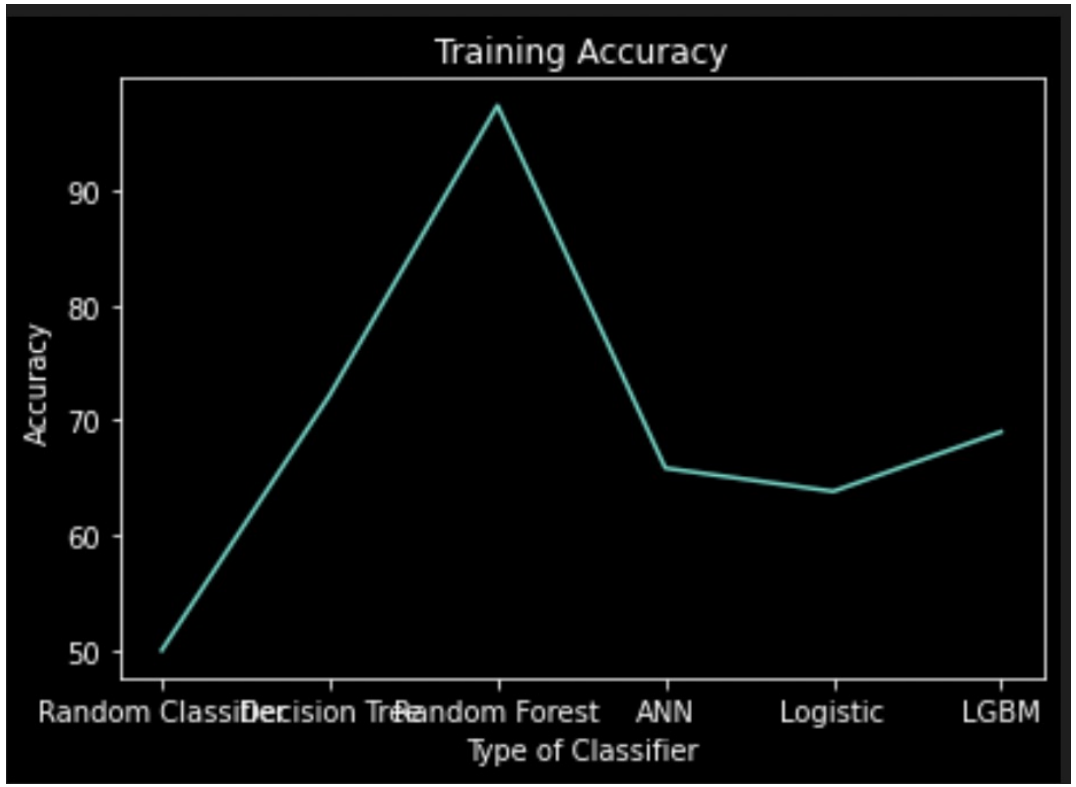
ANN & Logistic

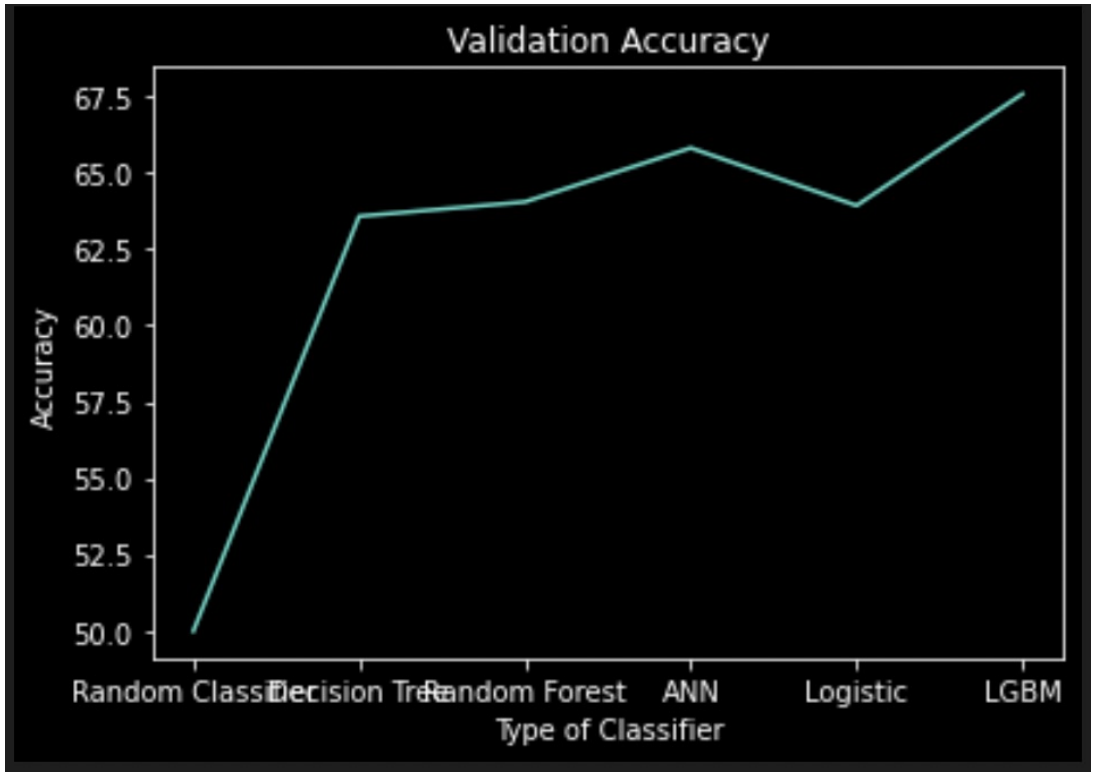
- Accuracy achieved over Training data is 69.32%
- Accuracy achieved over Validation data is 65.05%
- Accuracy achieved over Test data is 65.91%
- ANN provides better performance compared to previous classifiers that we have discussed.
- Logistic Regression takes lesser time than Random forest in model training but gives almost same performance on test and validation data
- Used SK-Learn pipeline class which is **a useful tool for encapsulating multiple different transformers alongside an estimator into one object**, so that you only have to call your important methods such as (fit() , predict()) once.

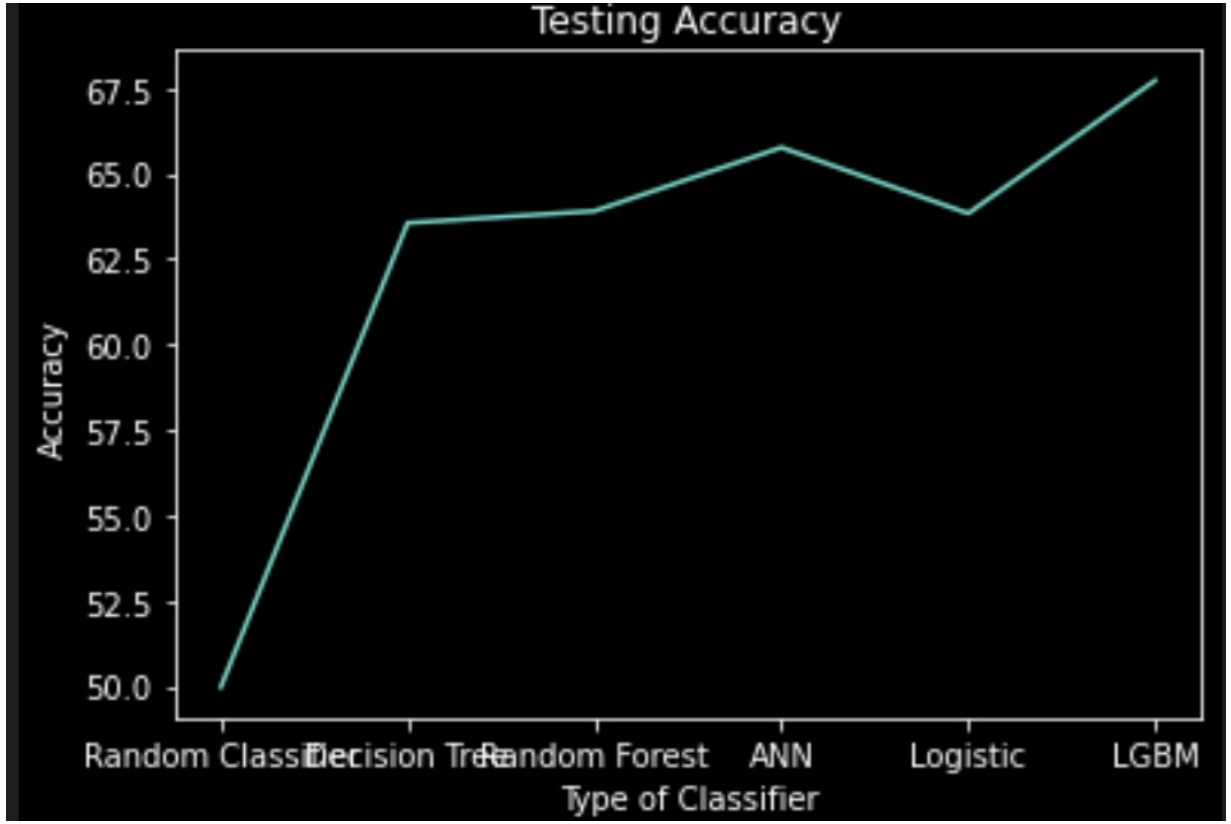
LightGBM Classifier with StratifiedKFold

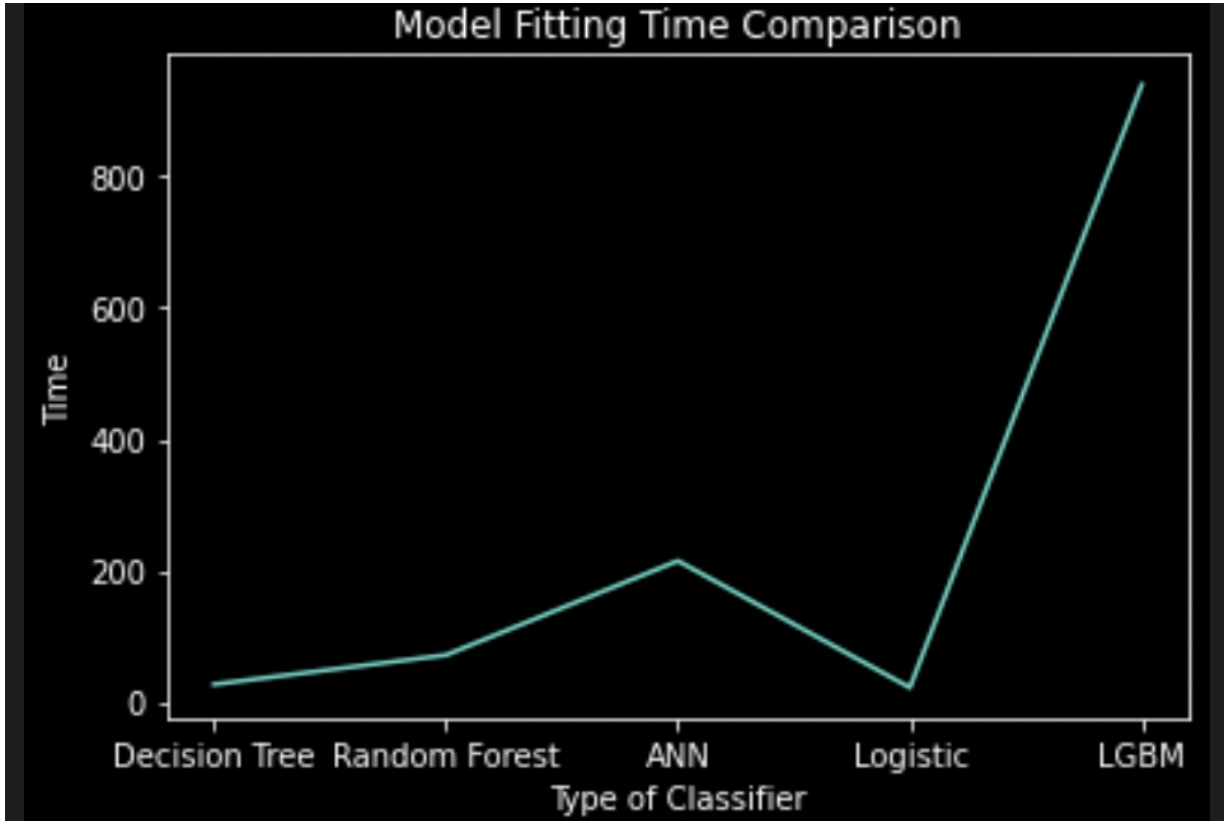
- LightGBM gives best performance compared to all classifiers that we have used so far.
- Accuracy achieved over Training data is 69.01%
- Accuracy achieved over Validation data is 67.59%
- Accuracy achieved over Test data is 67.73%
- Used StratifiedKFold which ensures that each fold of dataset has the same proportion of observations with a given label, which is not the case with K-fold.


Feature importance
Census_OSInstallTypeName 187
Census_OSVersion 168
LocaleEnglishNameIdentifier 125
AppVersion 119
Census_OSBuildRevision 92
CountryIdentifier 82
Census_ActivationChannel 81
IeVerIdentifier 68
Census_OSEdition 66
Census_ProcessorCoreCount 64
Census_OSUILocaleIdentifier 63
SmartScreen 58
IsProtected 56
RtpStateBitfield 56
GeoNameIdentifier 51
Census_ChassisTypeName 49
EngineVersion 49
Census_PrimaryDiskTypeName 47
Census_MDC2FormFactor 46
Census_PowerPlatformRoleName 37
Census_OSArchitecture 36
AVProductsEnabled 34
Census_OSInstallLanguageIdentifier 33
Wdft_IsGamer 32
OsPlatformSubRelease 31
Census_HasOpticalDiskDrive 30
IsSxsPassiveMode 27
Wdft_RegionIdentifier 24
Census_IsSecureBootEnabled 21
OsBuild 21
AVProductsInstalled 21
Census_OSBranch 19
Census_IsVirtualDevice 17
Census_IsAlwaysOnAlwaysConnectedCapable 15
Census_IsTouchEnabled 14
Census_OSWUAutoUpdateOptionsName 12
OsSuite 9
Census_FlightRing 6
Census_OSSkuName 6
OsVer 5
Census_OSBuildNumber 4
Processor 4
Census_GenuineStateName 3
Census_IsPenCapable 2
Census_ProcessorManufacturerIdentifier 1
SkuEdition 1
Features
0
25
50
75
100
125
150
175
200
Feature importance


Training Accuracy
Accuracy
90
80
70
60
50
Random Classifier
Decision Tree
Random Forest
ANN
Logistic
LGBM
Type of Classifier


Validation Accuracy
Accuracy
67.5
65.0
62.5
60.0
57.5
55.0
52.5
50.0
Random Classifier
Decision Tree
Random Forest
ANN
Logistic
LGBM
Type of Classifier


Testing Accuracy
67.5
65.0
62.5
60.0
57.5
55.0
52.5
50.0
Accuracy
Random Classifier
Decision Tree
Random Forest
ANN
Logistic
LGBM
Type of Classifier


Model Fitting Time Comparison
800
600
400
200
0
Time
Decision Tree
Random Forest
ANN
Logistic
LGBM
Type of Classifier

11) Conclusion

- Real-world datasets require complex models to be build to give out predictions with utmost accuracy. However, they do not end up being highly accurate on the unseen data.
- Increasing the complexity even further generally results in losing the ability to explain the model to humans.
- We use several pre-processing and classification techniques on the given dataset to train a commendable model and found the LightGBM technique to give the highest accuracy on unseen data.
- LightGBM, originally developed by Microsoft, uses decision-tree based learning algorithms. It has the following advantages:
  - Highly efficient
  - Lower memory usage
  - Better accuracy
  - Support of parallel, distributed, and GPU learning
  - Capable of handling large-scale data

12) Future Work

- Defining appropriate benchmarks for malware analysis is a priority of the whole research area.
- The discussion on malware analysis issues can provide further ideas worth to be explored.
- The novel concept of malware analysis economics can encourage further research directions
- Appropriate tuning strategies can be provided to balance competing metrics (e.g., accuracy and cost) when designing a malware analysis environment.

13) References

- *A Review of Android Malware Detection Approaches Based on Machine Learning*, 2020, by Kaijun Liu, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, and Haifeng Liu
- *The rise of machine learning for detection and classification of malware: Research developments*, trends and challenges, 2020, Daniel Gibert, Carles Mateu, Jordi Planes
- *Survey of Machine Learning Techniques for Malware Analysis*, 2019, by Daniele Uccia, Leonardo Aniellob , Roberto Baldonia
- *A Machine Learning Approach to Android Malware Detection*, 2012, by Justin Sahs and Latifur Khan

Thank You!