Malware Prediction
Imbalanced Dataset

S Venkata Sai Gopal
IMT2018083
IIIT Bangalore
Venkatasai.gopal@iiitb.org

A Venkata Karthik Reddy
IMT2018011
IIIT Bangalore
venkatakarthik.reddy@iiitb.org

C Suchith Kumar
IMT2018019
IIIT Bangalore
suchith.kumar@iiitb.org

***Abstract*—Windows machines are getting infected by various types of malware, based on different properties of the machine. These attacks are done via Malware, and have resulted in huge financial damage.**

The problem statement comes under classification, i.e predicting whether computer given with some features will get infected with malware or not. The main problem here is that the data we have is imbalanced i.e dataset contains more data on the computers which are not infected by malware compared to computers infected by malware. In order to overcome this we apply sampling techniques and some boosting models.

***Keywords*— label encoder, AUROC score, Over sampling, Undersampling. SMOTE, Adaboost, Xgboost, Lgbm, Logistic regression, Catboost**

INTRODUCTION

Imbalanced classes are a common problem in machine learning classification where there are a disproportionate ratio of observations in each class.

The rest of the paper proceeds as follows: Sec.1 Handling Imbalanced datasets Sec.2 describes dataset, Sec.3 basic EDA, Sec.4 preprocessing, Sec.5 covers model building Sec.6 conclusion.

I. HANDLING IMBALANCED DATASETS

Challenge here is to deal with an imbalanced data set – a scenario where the number of observations belonging to one class is significantly higher than those belonging to the other class.

A. Problems caused due to imbalance:

Choosing the evaluation metrics becomes quite challenging when dealing with imbalanced data sets. Accuracy is not the right metric as the accuracy of the model which predicts everything as majority class can be more than 80%.

In situations where anomaly detection is essential, conventional models could be biased and inaccurate. A simple reason could be that generally ML algorithms just try to improve accuracy by reducing error. The class balance, proportion and distributions are not taken into account.

B. Handling Imbalanced data set:

To solve the first problem of choice of evaluation metrics, the concept of confusion matrix becomes helpful.

Confusion Matrix

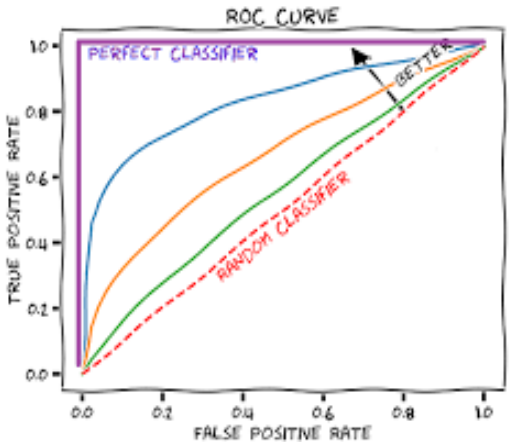
	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Accuracy = (TP+TN) / (TP+TN+FP+FN)
Recall = TP / (TP + FN)
Precision = TP/(TP + FP)
F1 Score = 2(Precision * Recall)/(Precision+Recall)

So, a possible way of evaluation could be that not only accuracy but also recall ,precision and f1 scores should be good.

There are also ROC AUC scores which are the most used evaluation metrics while dealing with imbalanced data sets. Though F1 scores seem good, ROC and AUC(area under ROC curve) curves are metrics for when we ask the model to predict probabilities and want the flexibility of creating our own thresholds for prediction.

To summarize it very briefly, accuracy is good when there is very little to no imbalance in classes, F1 score is used when there is heavy imbalance in classes and you care only about the positive class, and AUROC curve is used when the imbalance is present but you care about both positive and negative classes, and want to rank the predictions in order of their probabilities instead of having one absolute value.



C. Approach to handle Imbalanced datasets:

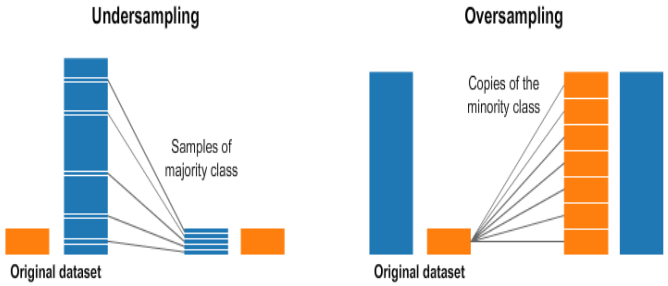
Coming to the problem of handling imbalanced data set, there are two main approaches :

1. Balancing the classes present in the training data before providing the data as input to the machine learning algorithm.

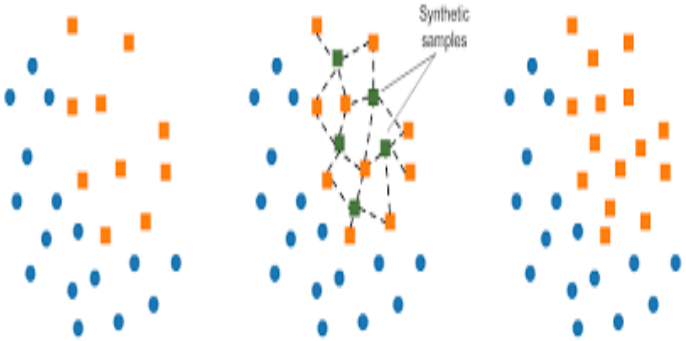
Basic ways to do so are

undersampling - randomly eliminating majority class samples until both classes are balanced out. The main disadvantage is that the more imbalanced the dataset the more samples will be discarded when undersampling, therefore throwing away potentially useful information.

over sampling - increases the number of instances in the minority class by randomly replicating them. The main disadvantage is it increases the chances of overfitting since it replicates the minority class samples.



SMOTE(Synthetic Minority Oversampling Technique) - is also a popular way. It is basically over sampling but instead of adding exact replicas, new synthetic similar instances are created and added. Unlike undersampling there is no loss of information, unlike oversampling it avoids overfitting the data.



2. Improving classification algorithms.

The classification algorithms are improved by using ensemble techniques.

The main objective of ensemble methodology is to improve the performance of single classifiers. The approach involves constructing several two stage classifiers from the original data and then aggregate their predictions.

Boosting:

Boosting is a technique that combines weak learners to create a strong learner that can make accurate predictions.

Ada Boost:

Adaptive boosting, also known as Ada boost, after each round, gives more focus to examples that are harder to classify. The quantity of focus is measured by a weight, which initially is equal for all instances. After each iteration, the weights of misclassified instances are increased and the weights of correctly classified instances are decreased.

In each iteration, these updated weighted observations are fed to the weak classifier to improve its performance. This process continues till the misclassification rate significantly decreases thereby resulting in a strong classifier.

Gradient Tree Boosting:

In Gradient Boosting many models are trained sequentially. It is a numerical optimization algorithm where each model minimizes the loss function, using the Gradient Descent Method. The loss here is the difference between real value and output of the previous learners.

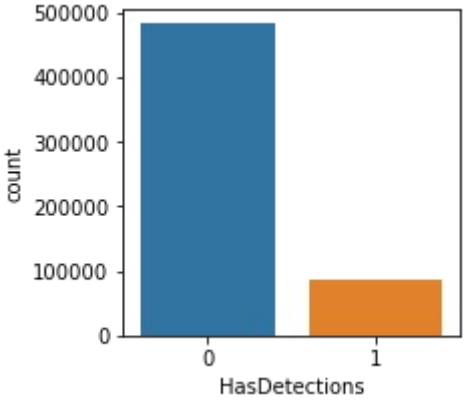
At every step, the residual of the loss function is calculated using the Gradient Descent Method and the new residual becomes a target variable for the subsequent iteration. The algorithm internally calculates the loss function, updates the target at every stage and comes up with an improved classifier as compared to the initial classifier.

XG Boost:

XGBoost-Extreme Gradient Boosting is an advanced and more efficient implementation of Gradient Boosting Algorithm. It implements parallel processing. It is highly flexible as users can define custom optimization objectives and evaluation criteria. It has an inbuilt mechanism to handle missing values. It also has a better way of dealing with negative loss when compared to gradient boosting.

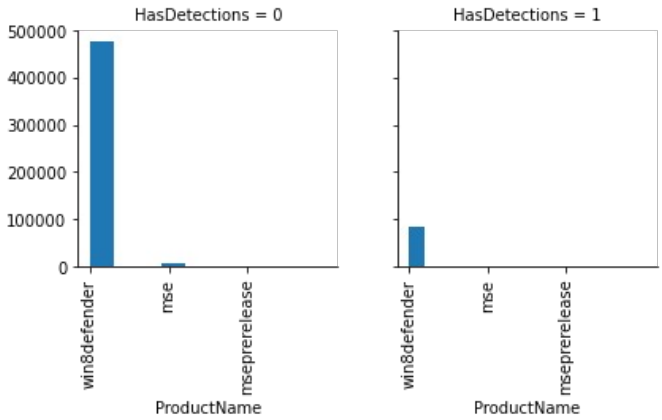
II. DATASET EXPLORATION

Dataset used in this assignment is taken from Microsoft Malware Prediction hosted on kaggle. Coming to the dataset, it contains 83 columns with target variable as 'HasDetections' which tells whether the computer is infected with malware or not(0 : no malware, 1: malware). Remaining 82 columns are features of the computer.

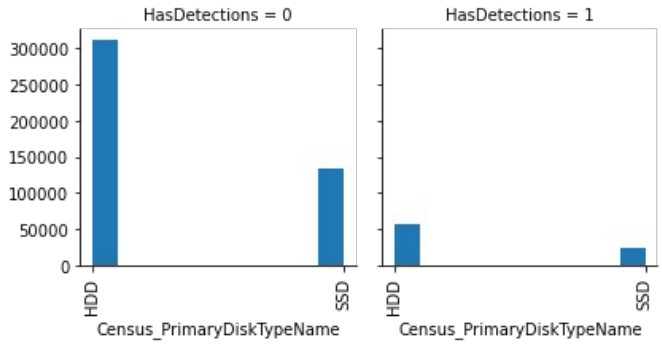


This is the data distribution of the target variable, contains more information on computers which are not infected with malware.

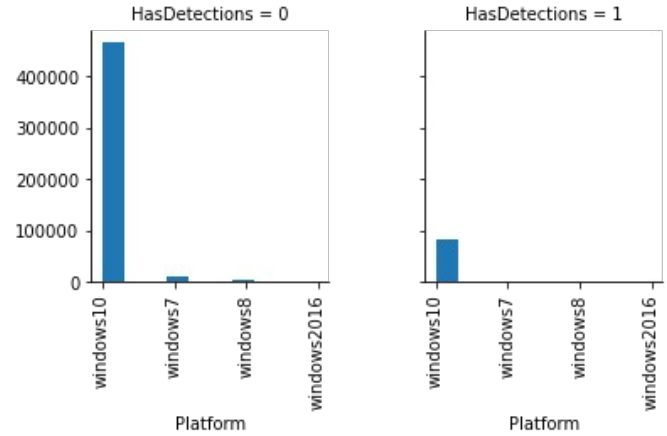
III. BASIC EDA



From this we can say that win8defender has more chances of getting infected with malware cause they are high in number.



From the plot we can say that computers with primary disk HDD are infected more cause they are high in number.



Here similar to productname, platform also has windows10 computers in huge amounts. So, they are infected more compared to the remaining windows versions.

IV. PREPROCESSING

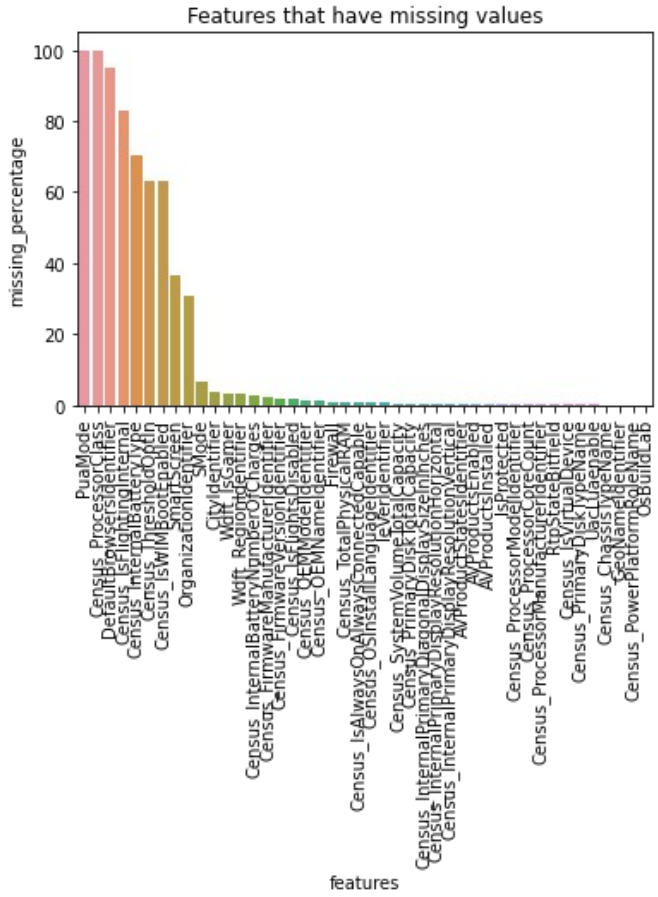
Preprocessing includes removing redundant features, adding new features, converting features into suitable form of modelling.

First we divided the columns into three categories numerical, binary, categorical columns. Out of 82 columns, 4 are numerical, 19 are binary, 59 are categorical columns. Now we checked the missing values i.e no of nans present in the data. We removed those columns which have nan percentage more than 80. We checked all the columns and removed those which have only one unique value or the unique value with a very high percentage. As they don't contribute to the model at all.

After doing basic data cleaning we added new features by splitting the features. 'Census_OSVersion', 'OsVer', 'EngineVersion', 'AppVersion', 'AvSigVersion' these features have data in x.y.z.w format so we divided them and named them accordingly like 'OsVer' is now converted to 'OsVer0', 'OsVer1', 'OsVer2', 'OsVer3'. From these new features we removed 'EngineVersion0','EngineVersion1', 'AvSigVersion3', 'AppVersion0' as they contain only one unique value.

We cleaned data in some features like 'SmartScreen' by replacing {"OFF": "Off", "requireadmin" : "RequireAdmin", "on": "On", "off": "Off"} and in columns 'Census_PowerPlatformRoleName', 'Census_PrimaryDiskTypeName' we transformed "UNKNOWN" and "Unspecified" to nan. In Census_ChassisTypeName we transformed "Unknown" to nan

we converted 'Census_SystemVolumeTotalCapacity', 'Census_PrimaryDiskTotalCapacity','Census_TotalPhysicalRAM' (which has numerical data with units in mb) these feature values from mb to gb. In 'Census_InternalPrimaryDisplayResolutionHorizontal','Census_InternalPrimaryDisplayResolutionVertical'(which has numerical data with units in pixels) these feature values from pixels to mm. And we transformed the data in 'Census_InternalBatteryType' as lithium with 0 and non lithium with 1.



We label encoded all the categorical and binary columns. We now replaced nan values in binary columns with mode of that respective column. We used iterative imputer in order to replace the nan values in the categorical columns, numerical columns. Iterative imputer replaces nan values by taking consideration of other columns.

V. MODEL BUILDING AND SELECTION

Coming to the model, we have tried both the methods of dealing with imbalance i.e Applying model on sampled dataset & Applying boosting models.

We first divided the given train data into two parts with 80% on training and 20% on validation. We tried logistic regression, Random forest, Easy ensembler with base estimate as random forests, Adaboost with base estimator as random forest, Xgboost, Catboost, Light Gradient Boosting Machine.

We applied these models in two ways first applied SMOTE to the divided 80% training data and second way is without applying SMOTE. We took the AUROC scores for both ways on all models, we came to know that for boosting algorithms SMOTE didn't work well. Basic models like logistic regression, random forest were giving AUROC score under 0.7 but boosting algorithms Adaboost, Lgbm, Catboost were giving AUROC score above 0.7. So we thought to fine tune Adaboost and Catboost models.

Model's metric scores on Validation Data with SMOTE

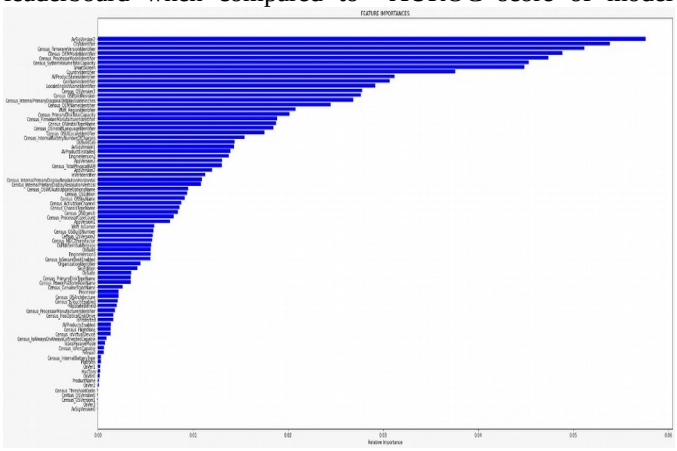
Model	**AUROC Score**	F1 Score	Accuracy
Logistic	**0.504**	0.220	0.568
Random Forest	**0.642**	0.213	0.829
Easy ensemble	**0.645**	0.226	0.834
Adaboost	**0.705**	0.177	0.857
Xgboost	**0.699**	0.197	0.857
Catboost	**0.704**	0.193	0.858
Lgbm	**0.692**	0.184	0.856

Model's metric scores on Validation Data without SMOTE

Model	**AUROC Score**	F1 Score	Accuracy
Logistic	**0.499**	0	0.849
Random Forest	**0.693**	0.024	0.850
Easy ensemble	**0.694**	0.338	0.593
Adaboost	**0.714**	0.17	0.857
Xgboost	**0.645**	0.194	0.858
Catboost	**0.710**	0.193	0.850
Lgbm	**0.708**	0.179	0.858

After fine tuning, Adaboost is working well with base estimator as Random forest.

Now for final submission instead of testing the tuned Adaboost model which trained on 80% data on test data, we trained the model with entire train data and submitted on kaggle. This increased the AUROC score in the kaggle leaderboard when compared to AUROC score of model trained on 80% data.



Feature Importance by the Adaboost model

VI. CONCLUSIONS

In this assignment, the dataset we had to work on was the Microsoft Malware Prediction Dataset. It was an imbalanced dataset and so we used techniques like Smote, ensemble boosting to deal with it. We used Area Under the ROC Curve (AUC) Score as the main defining metric for our experimentation but also took into consideration accuracy and F1 scores. We applied seven different algorithms - Logistic regression, Random forest, Easy ensemble, AdaBoost, CatBoost, LightGBM and XgBoost. AdaBoost algorithm showed the best results as it obtained the highest AUC Score, had great efficiency, good accuracy and took a comparatively low amount of time to run. We would like to conclude that we were able to fairly deal with the imbalance in the dataset and come up with a decent model for Malware Prediction.

REFERENCES

->https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
->https://imbalanced-learn.readthedocs.io/en/stable/api.html#module-imblearn.ensemble
->https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#:~:text=An%20AdaBoost%20%5B1%5D%20classifier%20is,focus%20more%20on%20difficult%20cases.
->https://catboost.ai/docs/concepts/python-reference_catboostclassifier.html
->https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28
->https://www.analyticsvidhya.com/blog/2017/03/imbalanced-data-classification/
->https://medium.com/@itbodhi/handling-imbalanced-data-sets-in-machine-learning-5e5f33c70163