ML PROJECT REPORT 2023

Title: Android Malware Detection using App permissions
Group No.: 7

Bijendar Prasad
2019238
bijendar19238@iiitd.ac.in

Abhishek Chaturvedi
2019401
abhishek19401@iiitd.ac.in

Deepesh Kumar
2019159
deepesh19159@iiitd.ac.in

April 29, 2023

1 Abstract

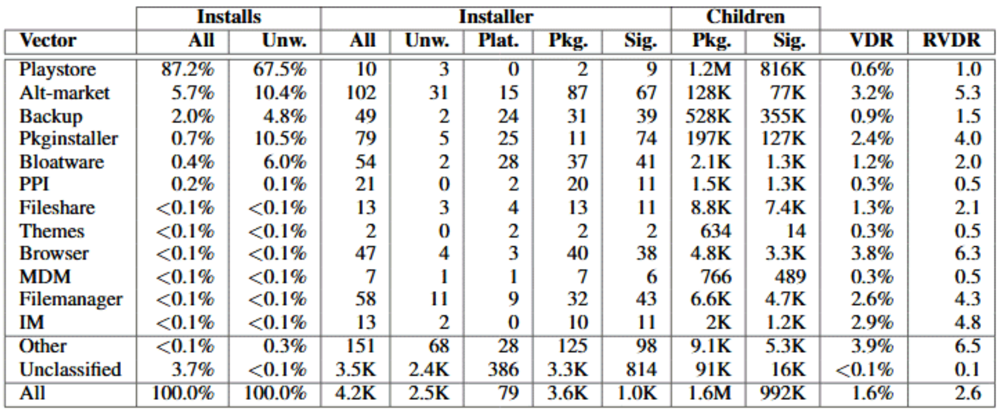
Given the increasing prevalence of malicious software targeting Android devices, Android malware detection has become a crucial area of research. The detection of Android malware is essential for safeguarding user privacy, protecting data security, and maintaining device performance. However, the task of detecting Android malware is complicated by a number of factors, including the diversity of malware families, the polymorphic nature of malware, and the evasion strategies employed by malware writers. As a result, numerous studies have highlighted the challenges associated with Android malware detection.

Keywords: Android malware, Machine learning, Feature extraction, Classification algorithms, Supervised learning and Unsupervised learning, Exploratory Data Analysis(**EDA**), Principal Component Analysis(**PCA**), API calls, Clustering, Dimensionality Reduction, Anomaly Detection, Sampling, Evaluation & Confusion Matrix, Kaggle, Matplotlib, Seaborn, **SVM, MLP, Random Forest, GaussianNB, Decision Tree, Logistic Regression**.

2 Motivation

As the android market continues to expand, so does the prevalence of malicious apps. According to ZDNet, as many as **10%-24%** of apps available on the Play store could be malicious in nature. These apps may appear innocuous at first glance, but they can wreak havoc on a user's system in a variety of harmful ways. Unfortunately, current methods for detecting malware are both resource-intensive and exhaustive, and they struggle to keep up with the rapid pace at which new malware is being developed.

TABLE VII: Summary of app distribution.



	Installs		Installer					Children			
Vector	**All**	**Unw.**	**All**	**Unw.**	**Plat.**	**Pkg.**	**Sig.**	**Pkg.**	**Sig.**	**VDR**	**RVDR**
Playstore	87.2%	67.5%	10	3	0	2	9	1.2M	816K	0.6%	1.0
Alt-market	5.7%	10.4%	102	31	15	87	67	128K	77K	3.2%	5.3
Backup	2.0%	4.8%	49	2	24	31	39	528K	355K	0.9%	1.5
Pkginstaller	0.7%	10.5%	79	5	25	11	74	197K	127K	2.4%	4.0
Bloatware	0.4%	6.0%	54	2	28	37	41	2.1K	1.3K	1.2%	2.0
PPI	0.2%	0.1%	21	0	2	20	11	1.5K	1.3K	0.3%	0.5
Fileshare	<0.1%	<0.1%	13	3	4	13	11	8.8K	7.4K	1.3%	2.1
Themes	<0.1%	<0.1%	2	0	2	2	2	634	14	0.3%	0.5
Browser	<0.1%	<0.1%	47	4	3	40	38	4.8K	3.3K	3.8%	6.3
MDM	<0.1%	<0.1%	7	1	1	7	6	766	489	0.3%	0.5
Filemanager	<0.1%	<0.1%	58	11	9	32	43	6.6K	4.7K	2.6%	4.3
IM	<0.1%	<0.1%	13	2	0	10	11	2K	1.2K	2.9%	4.8
Other	<0.1%	0.3%	151	68	28	125	98	9.1K	5.3K	3.9%	6.5
Unclassified	3.7%	<0.1%	3.5K	2.4K	386	3.3K	814	91K	16K	<0.1%	0.1
All	100.0%	100.0%	4.2K	2.5K	79	3.6K	1.0K	1.6M	992K	1.6%	2.6

What can help us to overcome these challenges ?

- Developing a comprehensive strategy to assess and analyze data from confirmed malicious applications.
- Creating a model that can accurately predict the presence of malicious applications based on their permissions.
- Introducing a machine learning-based malware detection model that utilizes publicly available metadata information. This model will be evaluated to determine its effectiveness as a first-stage filter for detecting Android malware.

3 Introduction

3.1 Background

The threat of malware attacks has grown significantly as the use of mobile devices, notably Android smartphones, has increased. Apps may contain hidden malicious software that might damage the device or steal critical user data. Conventional security measures and anti-virus software have limitations when it comes to identifying and stopping these threats, which are getting more complex. Machine learning has emerged as a viable strategy for Android malware prediction in response to this difficulty. Machine learning algorithms can examine a variety of app properties and traits to spot possibly dangerous activity. This can assist in finding malware that more conventional methods would not have been able to find.

Despite the growing threat of malware, there is still no reliable and robust method for detecting malicious applications. However, with the increasing use of machine learning in various fields, we believe that this issue can be addressed through the application of machine learning techniques. Our project aims to conduct a thorough and systematic investigation into the use of machine learning for malware detection, with the ultimate goal of developing an efficient ML model capable of accurately classifying apps as either benign (0) or malware (1) based on their requested permissions. **This study Proposes:**

- Conducting an in-depth examination and evaluation of Android metadata and permissions as predictors of malware.
- Introducing a machine learning-based malware detection strategy that utilizes publicly available metadata information.
- Analyzing the effectiveness of this model and assessing its potential as a first-stage filter for detecting Android malware.

3.2 Literature Survey

Research Paper: 1 Android Malware Prediction using Machine Learning Techniques: A Review

https://www.ijrdet.com/files/Volume11Issue2/IJRDET_0222_03.pdf

The report offers a thorough analysis of numerous machine-learning methods that have been applied to foretell Android malware. Support vector machines, decision trees, naive Bayes, and random forests are the machine-learning methods most frequently used to forecast Android malware. Permissions, API requests, system calls, and network traffic are a few characteristics that have been utilized to forecast Android malware. The study highlights some of the shortcomings of previous research, such as the scarcity of diverse datasets for training and testing machine learning models and the difficulty of dealing with obfuscated and polymorphic malware. It also shows that machine learning techniques can effectively predict Android malware and that the accuracy of the prediction can be improved by using a combination of features and machine learning algorithms. The authors contend that feature selection and dimensionality reduction methods can boost Android malware prediction's effectiveness. The research emphasizes some of the issues that need to be resolved to increase the efficacy of these methods and offers helpful insights into the state-of-the-art Android malware prediction using machine learning techniques.

Research Paper: 2 An Efficient Android Malware Prediction Using Ensemble Machine Learning Algorithms

https://www.sciencedirect.com/science/article/pii/S1877050921014186

To categorize Android apps as malicious or benign, the authors combine a number of classification methods, such as decision trees, k-nearest neighbors, and support vector machines. The authors tested their classifier on a dataset of more than 17,000 Android apps in order to assess their strategy. They discovered that their ensemble approach outperformed the individual classification algorithms utilized in the ensemble and had a high accuracy of over 99 % in detecting malware. The authors also evaluated other cutting-edge machine-learning strategies for Android malware detection and discovered that their strategy outperformed these techniques as well. They come to the conclusion that their ensemble approach can be utilized to increase mobile security and is a highly successful technique for identifying malware in Android apps.

3.3 Objective

1. Provide a machine learning model with the ability to precisely identify Android malware based on its behavioural traits, coding characteristics, and other characteristics.
2. For the purpose of training and assessing the machine learning model, collect and prepare a sizable dataset of Android malware samples and good apps.
3. Preprocess the data and perform **EDA** on the dataset. **API calls**, permissions, network traffic, system calls, and other pertinent information that can distinguish between malicious and benign apps should be chosen and extracted from the dataset.
4. Use relevant metrics to assess the machine learning model's performance, including **precision**, **recall**, **F1 score**, and **Accuracy**.
5. Improve the accuracy, speed, and durability of the machine learning model while minimising overfitting and bias.
6. Test the machine learning model's efficacy using actual Android malware samples, and evaluate how well it performs in comparison to other cutting-edge malware detection methods.

3.4 Scope

1. **Goals:**
   - To build a machine learning model that can accurately predict whether an Android app is malware or not.
   - To provide a useful tool for security researchers and developers to analyze and classify Android apps quickly and efficiently.
2. **Deliverables:**
   - A trained machine learning model that can classify Android apps as malware or benign with high accuracy.
   - A report and a presentation documenting the methodology used to build and evaluate the model.

3.5 Impact

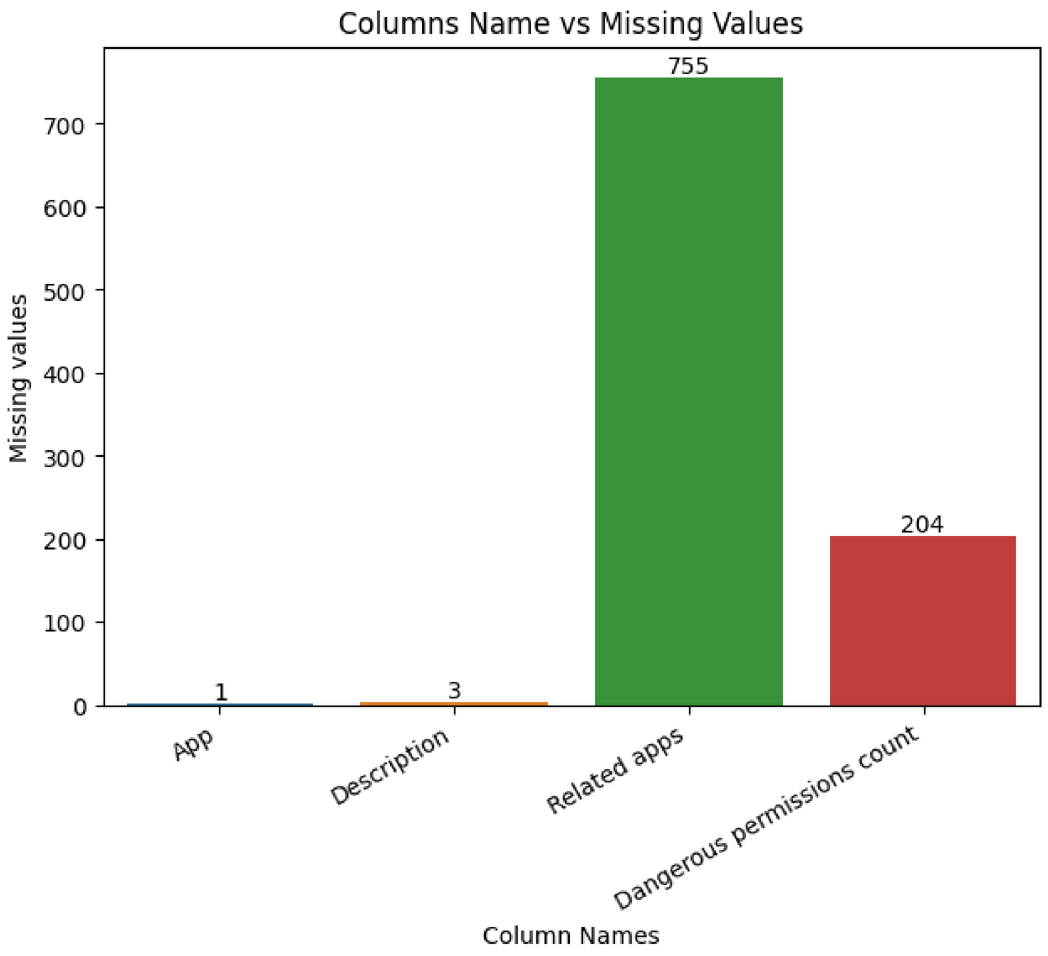
1. **Better Detection:** The speed and accuracy of detecting Android malware can be increased by using machine learning methods. To identify patterns and traits that point to the existence of malware, machine learning models can be trained on a huge dataset of known malware.
2. **Real-time protection:** A machine learning model is able to track and detect harmful behavior on Android devices in real time. This can lessen the risk of malware spreading and harming users' devices.
3. **Fewer False Positives:** A large percentage of false positives frequently occurs when using traditional malware detection techniques. On the other hand, machine learning models can be trained to distinguish minute details between legitimate and malicious activities, lowering the incidence of false positives and improving the precision of detection.
4. **Better User Experience:** Machine learning models can enhance the overall user experience by detecting and blocking malware in real time. Users will be less likely to experience problems brought on by malware and will feel more secure using their devices.
5. **Improved Security:** Machine learning models can swiftly update their detection skills and respond to new threats. This lowers the overall risk of malware assaults and enables the implementation of more effective and efficient protection solutions.
6. **Future Research and Development:** This project can act as a stepping stone for future research and can be integrated into software for the benefits described above.

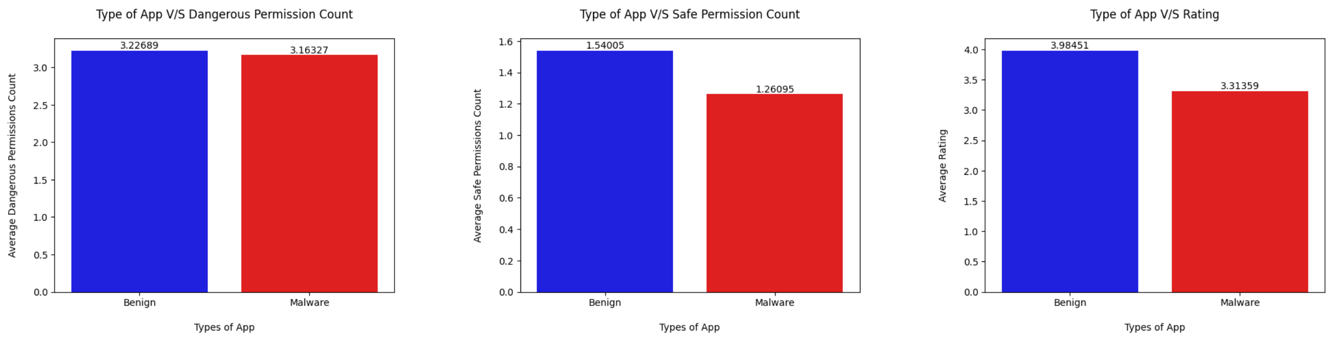
4 Materials and Methods

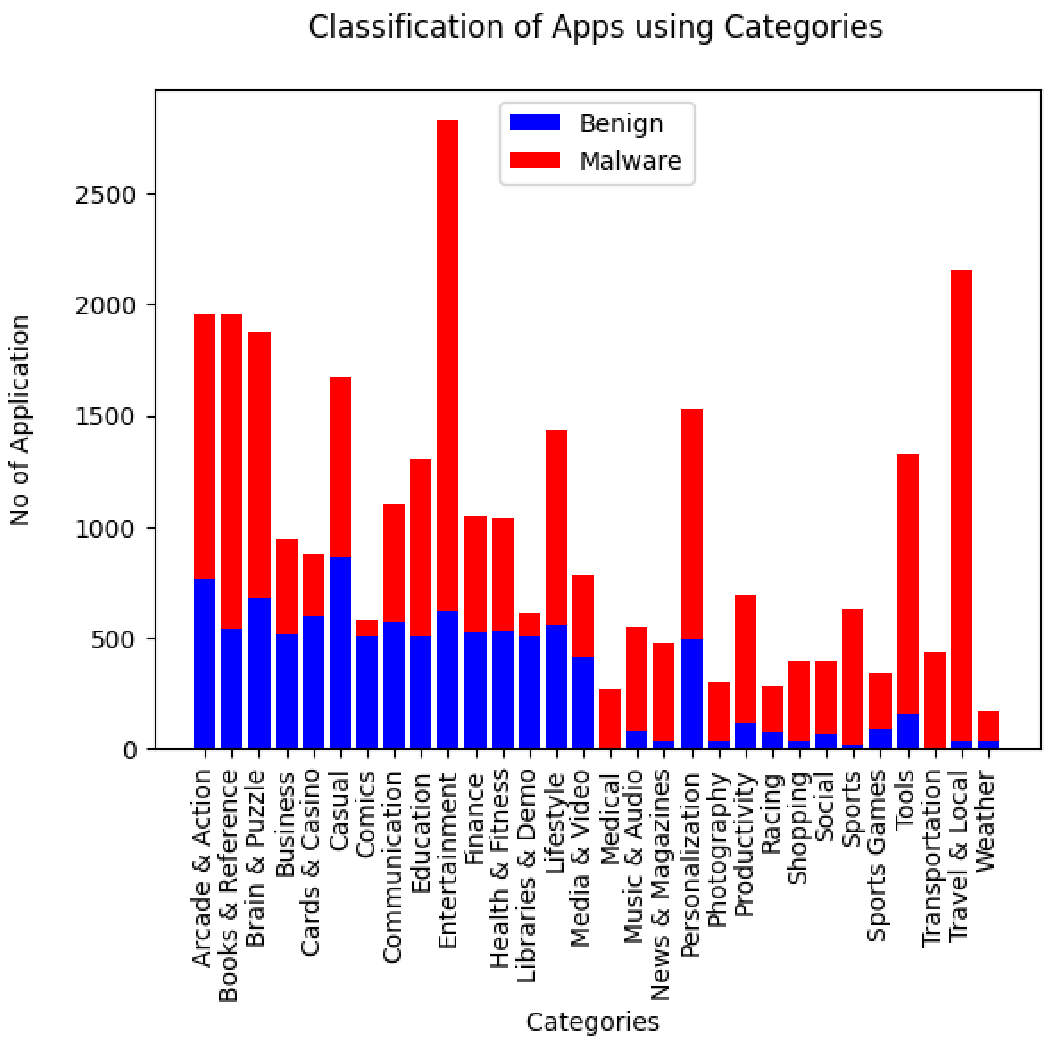
4.1 Dataset

1. Kaggle data was used for the dataset.
2. Data includes information on the permissions for nearly 30k apps.
3. The dataset contains 183 characteristics, such as Dangerous Permissions Count, Default: Access DRM material, Default: Transfer Application Resource, etc.
4. There is a single binary target class named "Class" that designates benign (binary 0) and malicious (binary 1) apps.
5. There are 29,999 records, 9,999 good programmes, and 20,000 malwares.

Preprocessing, Visualization and Analysis: The data is first imported from a **CSV file** and loaded into a dataframe for ease of use. The necessary attributes are then extracted from the dataset. To gain a better understanding of the data, several plots are generated. The data is checked for **null** or **missing values**, and any such values are replaced with the mean of the corresponding column. The distribution of malware and benign applications across various settings is then analyzed, and the results are visualized through a series of plots created using **Matplotlib** and **Seaborn**.

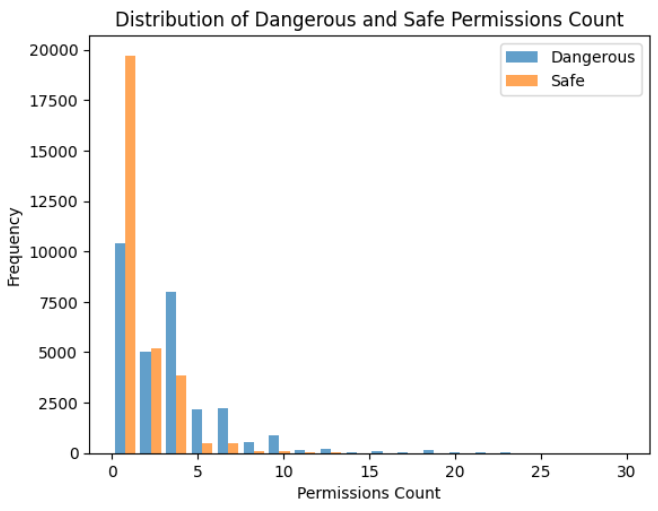


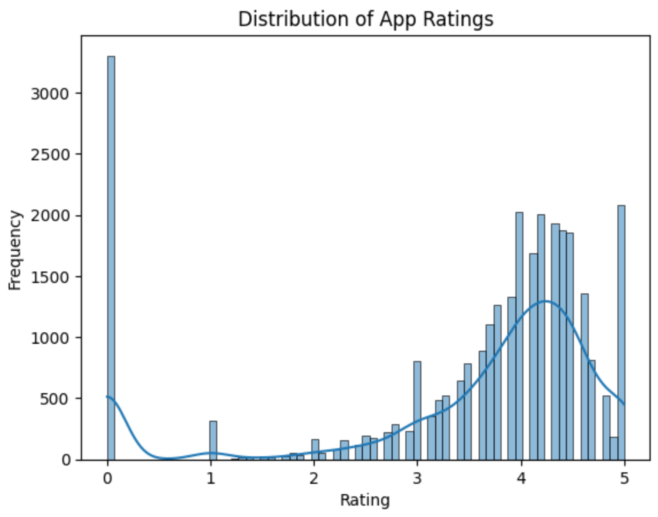


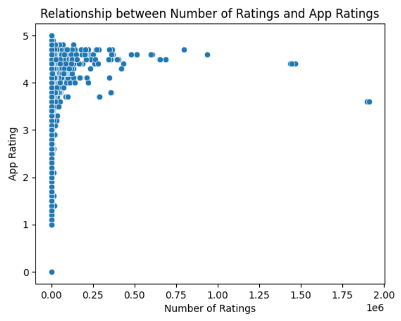


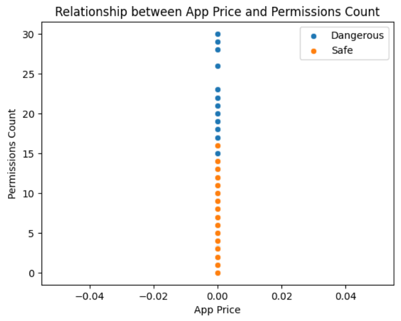
4.2 Exploratory Data Analysis(EDA):

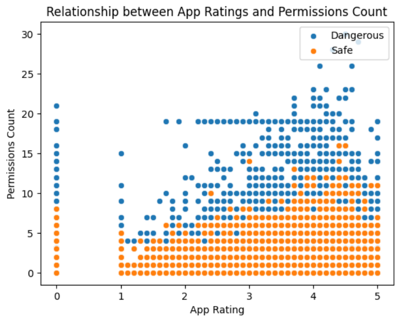
The **EDA** for the Android Permission Dataset provided valuable insights into the relationships between different features in the dataset and helped us identify the most important features for predicting the app rating. It also provided a foundation for further analysis using machine learning techniques.





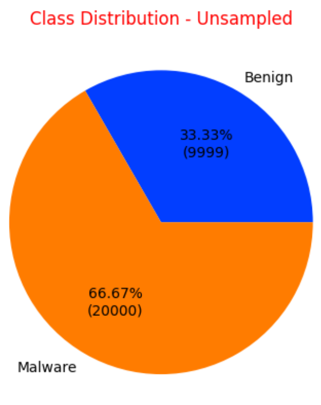


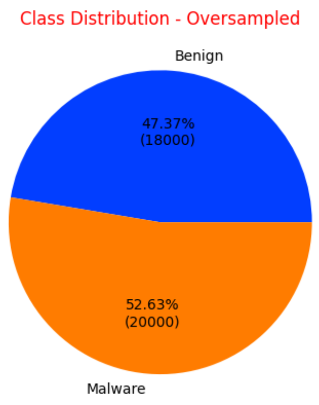


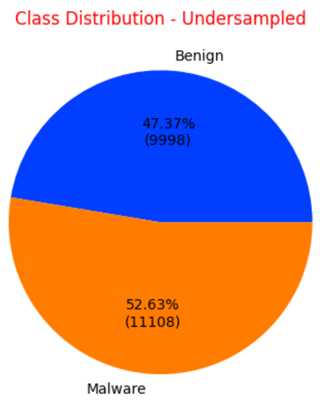


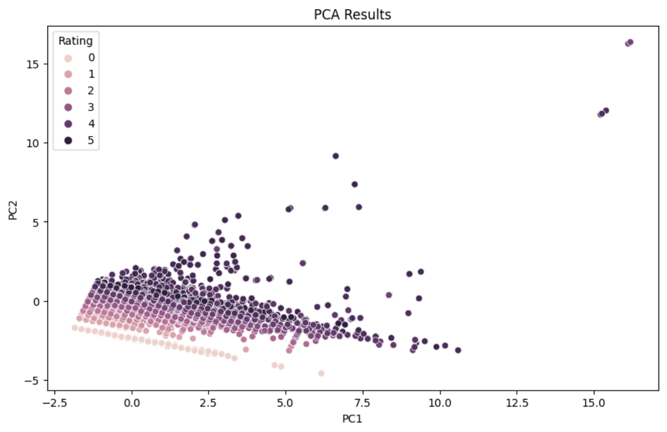
4.3 Methodology

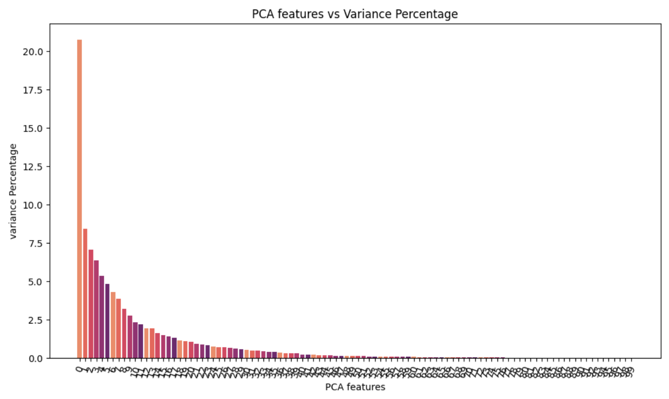
After preprocessing the data, it is split into testing and training sets at an 8:2 ratio. We attempted both under and oversampling techniques on the dataset, but the results were not promising. We then applied various classifiers, including logistic regression, decision trees, and Naive Bayes, but the outcomes were unsatisfactory. Upon further inspection of the dataset, we discovered that it contained several multivariate data tables, which required us to apply **PCA** to each dataset. We plotted the variance percentage after using PCA and chose to use the inverse transform. We then applied **Random Forest** to the dataset, which resulted in a significant improvement in accuracy. We then used the boosting approach to further increase prediction accuracy, both on an unsampled dataset and on one with reliable features selected. The results showed that the model was improving. Finally, we applied **SVM** and **MLP** to the final dataset and achieved our best results. When comparing the results obtained after feature selection and boosting, we can see that we have made significant progress and achieved our final **accuracy**.



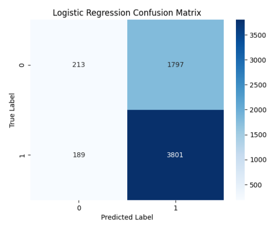


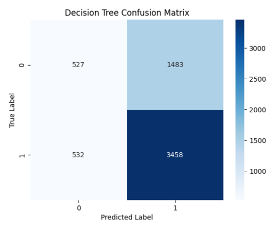


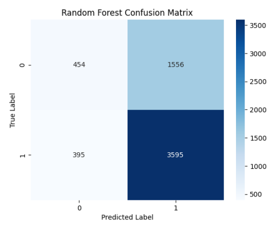


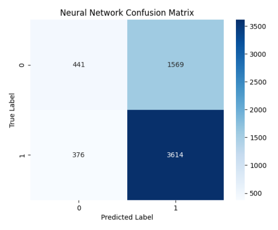


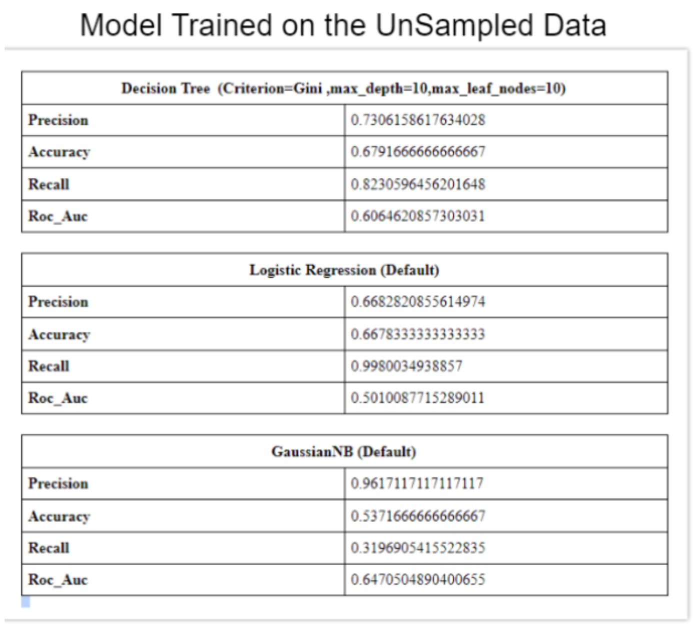
4.4 Results and Analysis









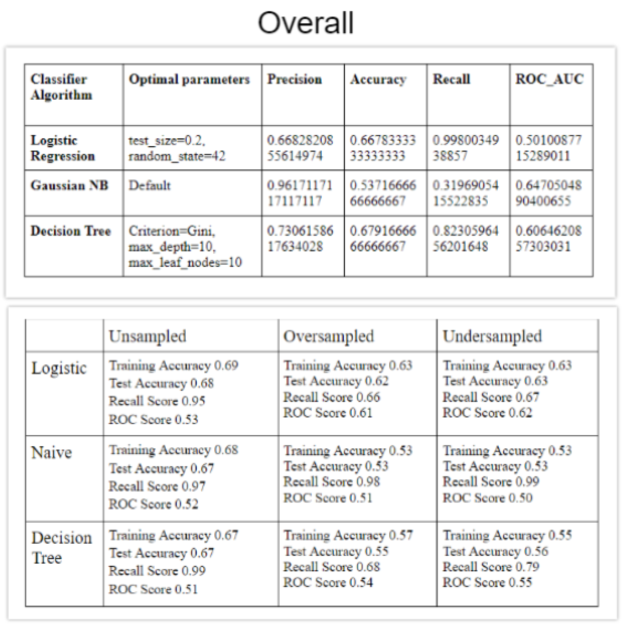


Model Trained on the UnSampled Data

Decision Tree (Criterion=Gini ,max_depth=10,max_leaf_nodes=10)	
Precision	0.7306158617634028
Accuracy	0.6791666666666667
Recall	0.8230596456201648
Roc_Auc	0.6064620857303031

Logistic Regression (Default)	
Precision	0.6682820855614974
Accuracy	0.6678333333333333
Recall	0.9980034938857
Roc_Auc	0.5010087715289011

GaussianNB (Default)	
Precision	0.9617117117117117
Accuracy	0.5371666666666667
Recall	0.3196905415522835
Roc_Auc	0.6470504890400655

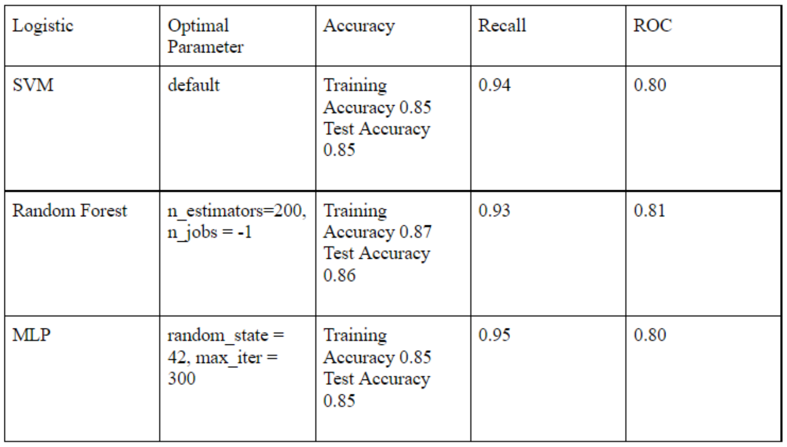


Overall

Classifier Algorithm	Optimal parameters	Precision	Accuracy	Recall	ROC_AUC
Logistic Regression	test_size=0.2, random_state=42	0.66828208 55614974	0.66783333 33333333	0.99800349 38857	0.50100877 15289011
Gaussian NB	Default	0.96171171 17117117	0.53716666 66666667	0.31969054 15522835	0.64705048 90400655
Decision Tree	Criterion=Gini, max_depth=10, max_leaf_nodes=10	0.73061586 17634028	0.67916666 66666667	0.82305964 56201648	0.60646208 57303031

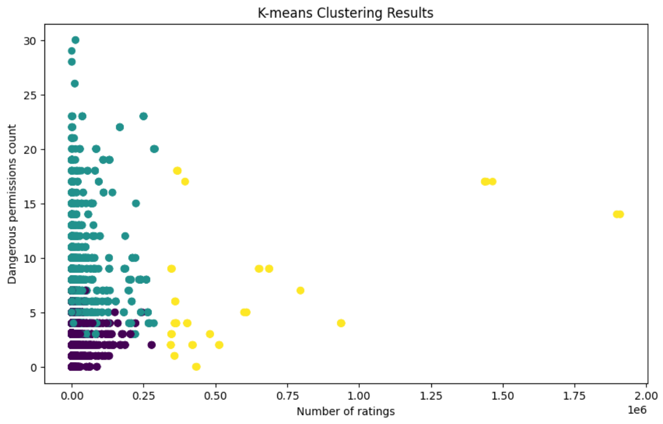
	Unsampled	Oversampled	Undersampled
Logistic	Training Accuracy 0.69 Test Accuracy 0.68 Recall Score 0.95 ROC Score 0.53	Training Accuracy 0.63 Test Accuracy 0.62 Recall Score 0.66 ROC Score 0.61	Training Accuracy 0.63 Test Accuracy 0.63 Recall Score 0.67 ROC Score 0.62
Naive	Training Accuracy 0.68 Test Accuracy 0.67 Recall Score 0.97 ROC Score 0.52	Training Accuracy 0.53 Test Accuracy 0.53 Recall Score 0.98 ROC Score 0.51	Training Accuracy 0.53 Test Accuracy 0.53 Recall Score 0.99 ROC Score 0.50
Decision Tree	Training Accuracy 0.67 Test Accuracy 0.67 Recall Score 0.99 ROC Score 0.51	Training Accuracy 0.57 Test Accuracy 0.55 Recall Score 0.68 ROC Score 0.54	Training Accuracy 0.55 Test Accuracy 0.56 Recall Score 0.79 ROC Score 0.55

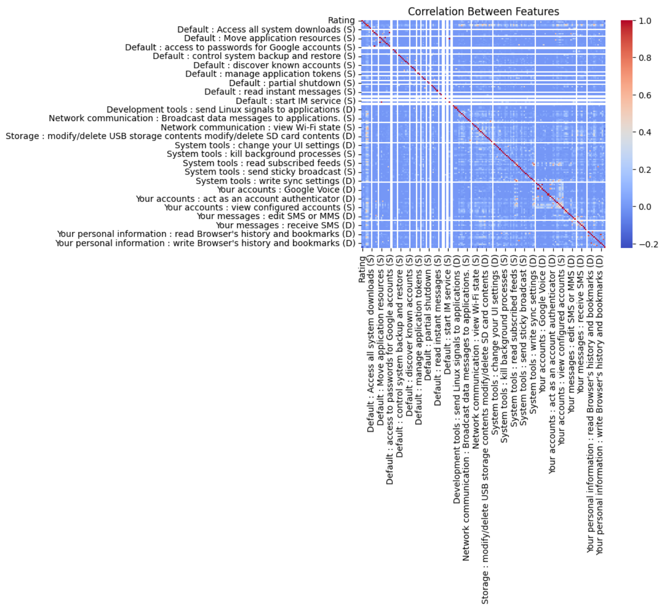
As we seen in the Tabulation that, Accuracy follows the order as follow:
Decision Tree > **Logistic Regression** > **Gaussian Naive Bayes**. This were the results before we had done sampling and under-sampling.



Logistic	Optimal Parameter	Accuracy	Recall	ROC
SVM	default	Training Accuracy 0.85 Test Accuracy 0.85	0.94	0.80
Random Forest	n_estimators=200, n_jobs = -1	Training Accuracy 0.87 Test Accuracy 0.86	0.93	0.81
MLP	random_state = 42, max_iter = 300	Training Accuracy 0.85 Test Accuracy 0.85	0.95	0.80

By looking at the result we can say that Random Forest perform best among all the classifiers with Accuracy of 86%. As we seen in the Tabulation that, Accuracy follows the order as follow: **Random Forest > MLP > SVM**





5 Conclusion

- **Learning-** Different ways to visualize the data for better understanding of features. Machine Learning models like Logistic Regression, Naive Bayes and Decision Tree to model the problem. How to use platforms like Kaggle and Google Colab. How to work and collaborate in teams.
- **Work Left-** We conducted study, but there is always opportunity for some improvement. We hope that our project will be useful in future research.

6 Distribution of work among group members

- **Bijendar Prasad:** Preprocessing, Training Model, Data Visualization, Exploratory Data Analysis, Principal Component Analysis, Feature Selection, Result Analysis
- **Abhishek Chaturvedi:** Data Preprocessing, Data Visualisation, Model Testing, Report Writing, Model Tuning
- **Deepesh Kumar:** Model Testing, Data Preprocessing, Data Visualisation, Report Writing

References

[1] Hareram Kumar (2022) *The Research Paper*, Android Malware Prediction using Machine Learning Techniques: A Review.

[2] Neamat Al Sarah (2021) *Online Paper*, An Efficient Android Malware Prediction Using Ensemble machine learning algorithms

[3] *Machine Learning for Android Malware Detection Using Permission and API Calls*

[4] *Dynamic Permissions based Android Malware Detection using Machine Learning Techniques*

[5] *Android Permission Dataset*