Network-Based Classification and Analysis of Malware Threats in Android

Group 42: Sachin Sharma
Samyak Jain
Mohit
Harshit Garg

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Motivation

- Urgent Necessity: Our project is driven by the pressing need to safeguard Android users from the escalating threat of malicious software.
- Vast User Base: With a global user base of 3.6 billion Android users and around 140 billion annual app downloads, the potential for malware attacks is extensive.
- A Unique Approach: While many existing Android security solutions focus on device-centric strategies, we propose a network-based approach. This approach offers a lightweight, side-channel solution, eliminating the need for intricate capture of permission and API call data.

Literature Review

Paper 1 - "Research of android malware detection based on network traffic monitoring" by J. Li, L. Zhai, X. Zhang and D. Quan,2014 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, China, 2014, pp. 1739-1744, doi: 10.1109/ICIEA.2014.6931449

- Comprehensive review of Android malware detection research
- Identifies limitations and research gaps in prior work
- Highlights the necessity of their research
- Suggested Improvements: Include more recent references, offer critical analysis of previous studies, and enhance organization and clarity.

Literature Review

Paper 2 -"Applying machine learning classifiers to dynamic Android malware detection at scale" by B. Amos, H. Turner and J. White, 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), Sardinia, Italy, 2013, pp. 1666-1671, doi: 10.1109/IWCMC.2013.6583806.

- Explores Android malware growth in IoT due to Android's open-source nature.
- Segments detection techniques: signature-based, behavior-based, and taint analysis.
- Focus on machine learning classifiers, spotlighting Support Vector Machine (SVM) in mobile malware detection.
- Emphasizes the need for robust security measures in IoT and smartphones.

Dataset Description

This dataset comprises a substantial 3,55,630 entries across 85 columns and is a critical resource for network traffic dynamic analysis. It features four distinct labels: Android_Adware, Android_Scareware, Android_SMS_Malware, and Benign. We will begin by exploring the key elements of this dataset.

Some of the Key features are :-

- Source IP
- Source Port
- Destination IP
- Destination Port
- Protocol
- Timestamp
- Flow Duration
- Total Fwd Packets

Each of these features plays a pivotal role in our understanding of network communication

Data Preprocessing

Our procedure for Data Preprocessing is broadly divided into two subprocesses:-

- Data Cleaning :- To remove unwanted and invalid data from our dataset.
- Feature Selection :- To filter out most relevant features from our dataset, to reduce dimensionality and help our model work effectively.

Data Cleaning Results

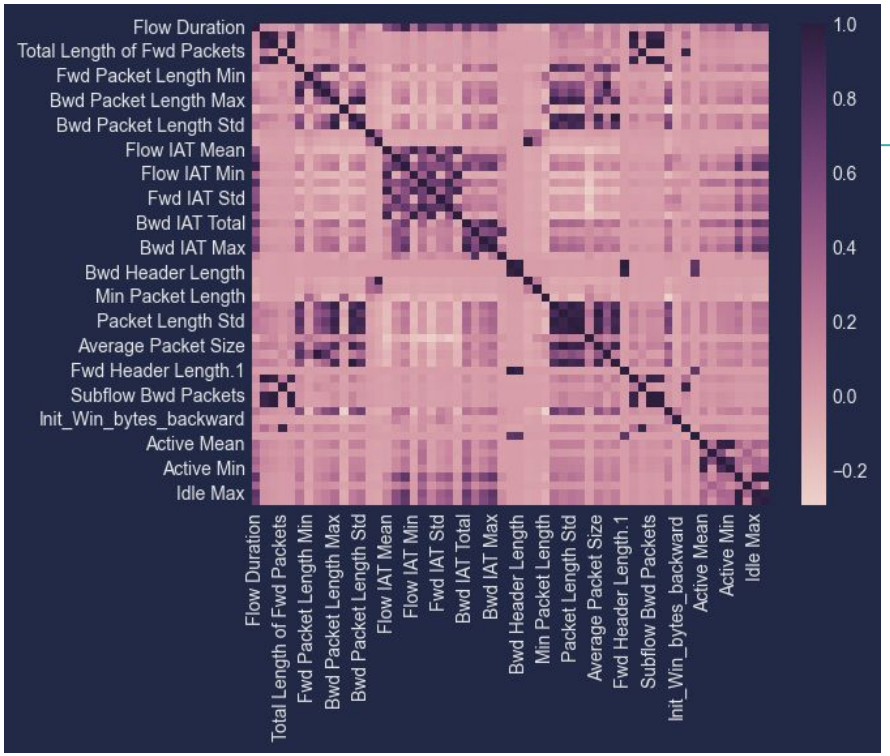
	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	...	min_seg_size_
0	10.42.0.211	50004	172.217.6.202	443.0	6.0	13/06/2017 11:52:39	37027	1	1	0.0	...	
1	10.42.0.211	35455	172.217.6.202	443.0	6.0	13/06/2017 11:52:39	36653	1	1	0.0	...	
2	10.42.0.211	51775	131.253.61.68	443.0	6.0	13/06/2017 11:52:42	534099	8	12	1011.0	...	
3	10.42.0.211	51775	131.253.61.68	443.0	6.0	13/06/2017 11:52:43	9309	3	0	0.0	...	
4	10.42.0.211	51776	131.253.61.68	443.0	6.0	13/06/2017 11:52:42	19890496	8	6	430.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
355625	172.217.7.14	80	10.42.0.211	38405.0	6.0	17/06/2017 01:29:11	126711	1	1	0.0	...	
355626	10.42.0.211	7632	10.42.0.1	53.0	17.0	17/06/2017 01:30:33	48012	1	1	30.0	...	
355627	10.42.0.211	45970	104.192.110.245	443.0	6.0	17/06/2017 01:29:45	20028018	11	8	339.0	...	
355628	10.42.0.211	51982	10.42.0.1	53.0	17.0	17/06/2017 01:29:45	347926	1	1	32.0	...	
355629	10.42.0.211	9320	10.42.0.1	53.0	17.0	17/06/2017 01:30:33	125473	1	1	30.0	...	

351513 rows × 82 columns

Data Preprocessing – Feature Selection

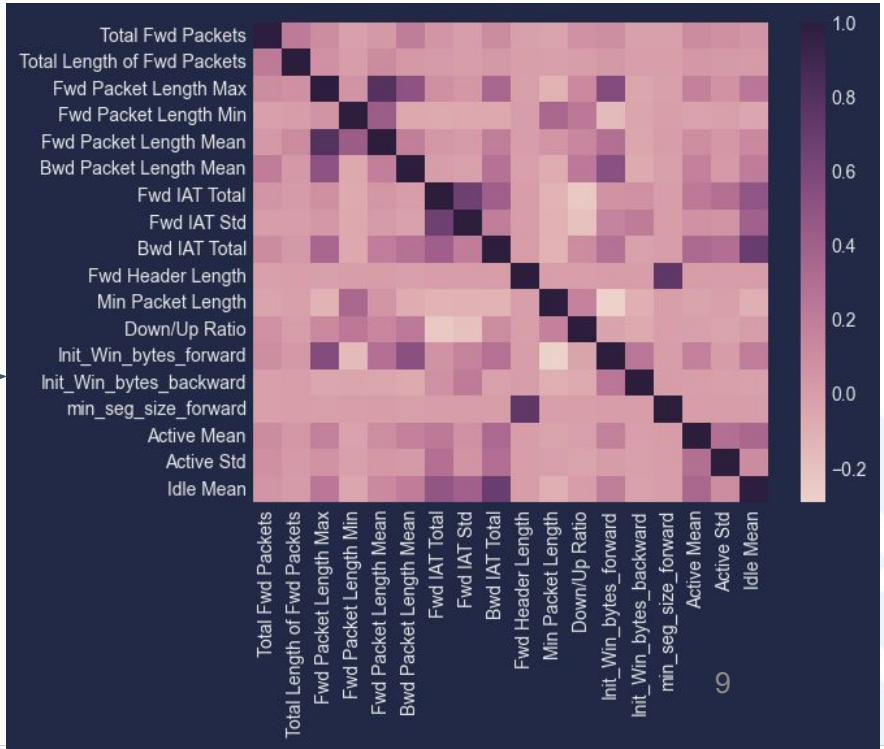
Our feature selection is done by applying the following techniques in the given order:-

- Correlation test
- Mann-Whitney-U test (for numerical features)
- Chi2 Test (for categorical features)
- Information Gain Test



- Correlation Heatmap before correlation test.

- Correlation Heatmap after correlation test.



Feature Selection – Results

From our original dataset, we were able to filter out **26** most relevant features.

1. Categorical Features = 7
2. Numerical Features = 18
3. Datetime Feature = 1

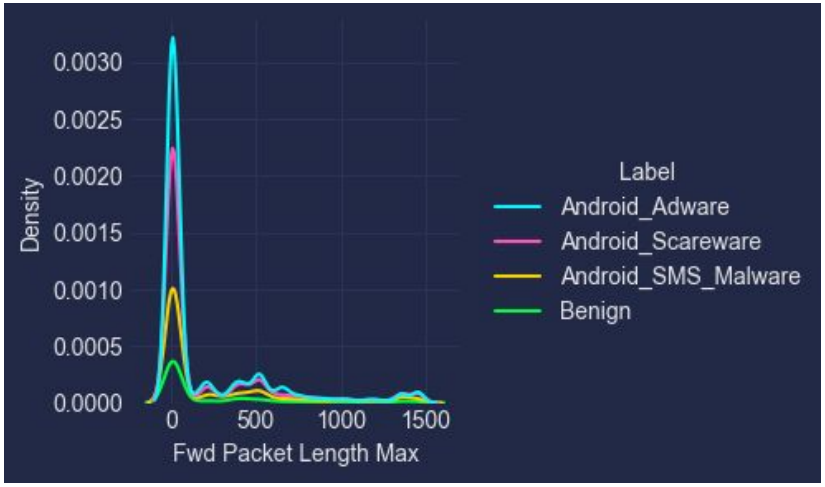
These features are also validated by finding Information Gain for each variable, which all are significant.

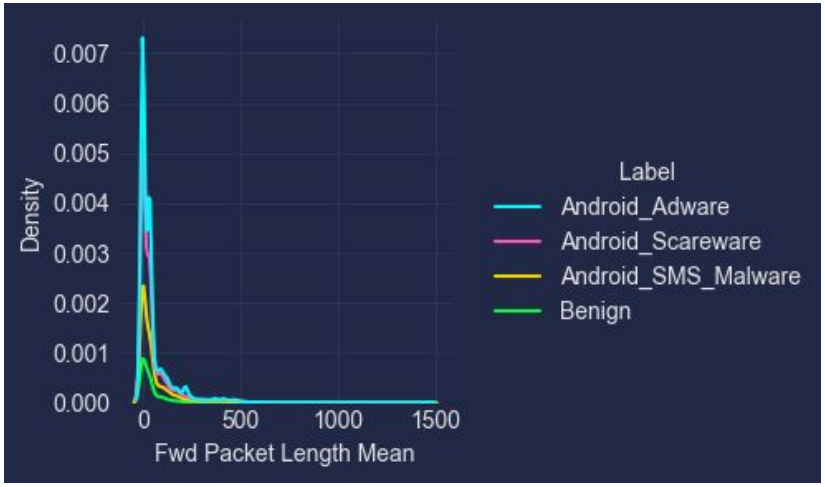
This is a great dimensionality reduction, and it will help enhance the performance of our model and reducing its complexity.

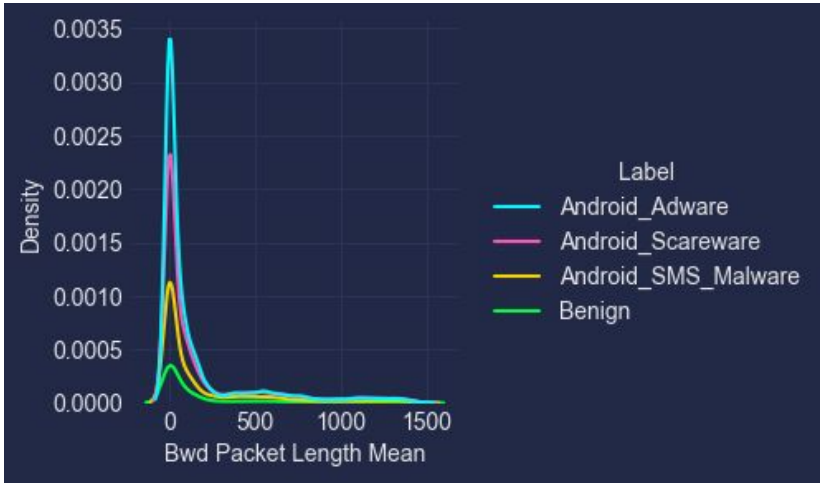
0.217023	Destination IP
0.154624	Source Port
0.112254	Source IP
0.061931	Bwd Packet Length Mean
0.043775	Fwd IAT Total

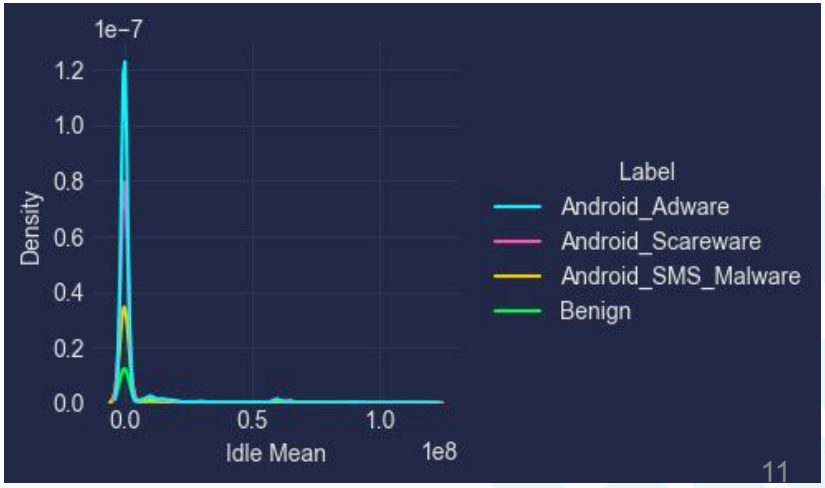
Visualization

Distribution Plots for Numeric variables





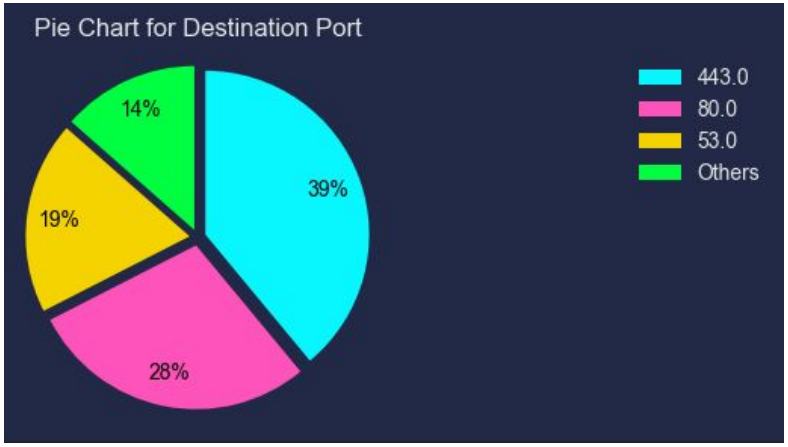




Visualization

Statistics for Categorical Variables

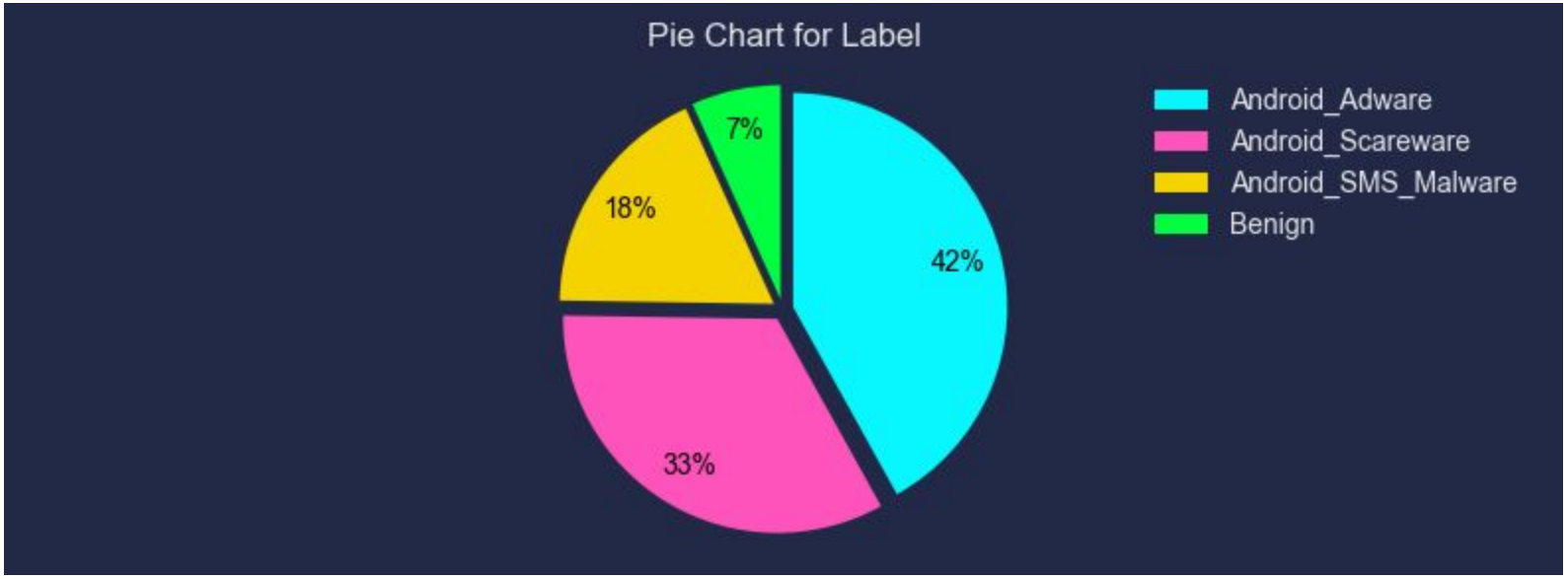
	Source IP	Destination IP	Source Port	Destination Port	Protocol	PSH Flag Count	ACK Flag Count
count	351513	351513	351513	351513.0	351513.0	351513.0	351513.0
unique	2533	4743	51341	19407.0	3.0	2.0	2.0
top	170524883	170524673	443	443.0	6.0	0.0	0.0
freq	177261	66792	19601	137034.0	273223.0	235194.0	200248.0




```
Source IP:

Android_Adware: 170524883 has max prob = 0.6211824230380554
Android_Scareware: 170524883 has max prob = 0.43453860543781875
Android_SMS_Malware: 170524883 has max prob = 0.3720316622691293
Benign: 170524883 has max prob = 0.4746920870592205
=> All labels have the same most used Source IP = 170524883
```

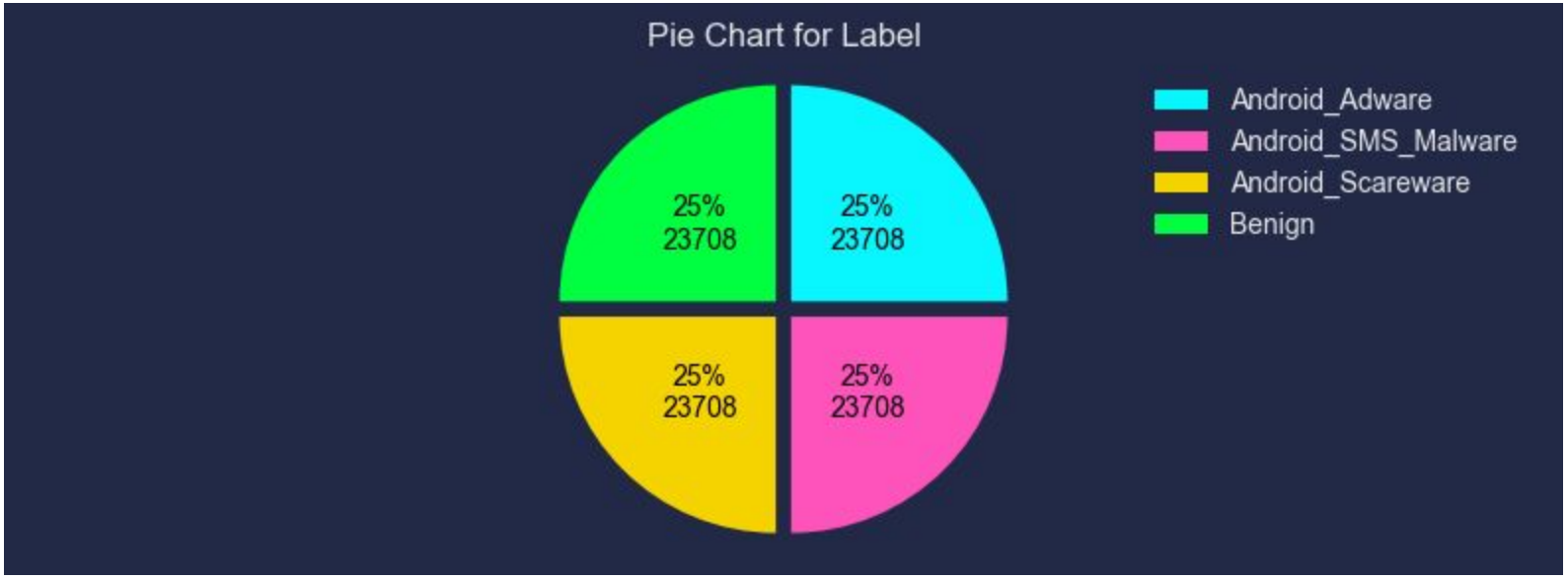
Visualization

Pie Chart for target variable Before UnderSampling



Visualization

Pie Chart for target variable After UnderSampling



Model Training – Baseline

We have used 3 distinct category models to create a suitable baseline.

1. **Logistic Regression:** Linear model
2. **Random Forests:** Tree based ensemble model
3. **Naive Bayes:** Probability based model Independent of params

The idea is to find which type of model can best fit the dataset. It can help comment about the separability of the dataset.

Baseline – Preprocessing

We have used **Label Encoding** to encode the categorical features.

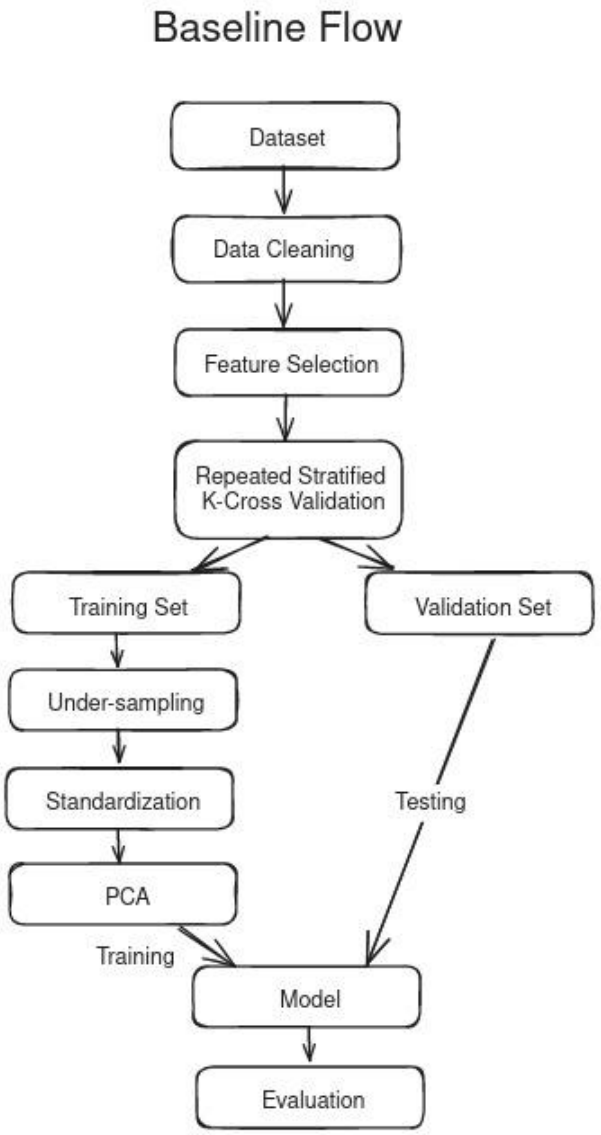
Since the number of unique values in the columns is quite large (~6000 per column), One Hot Encoding is not appropriate as it will make the dataset very sparse.

For interpretability of datetime feature **'Timestamp'**, we extracted day, month, hour, minute etc variables out of it.

Finally we **standardized** the dataset using StandardScaler in Python.

→ **PCA :** We have applied PCA for reducing the Dimensionality, so that the model can easily identify the underlying pattern.

Baseline Flow



Baseline- 2 datasets

We have created baseline models on both the datasets, the one before feature selection and the one after it.

This will help us validate that feature selection and preprocessing was useful.

We have created two models on both the datasets one with PCA and one without PCA.

Columns before reduction: 83

Columns after reduction: 27

Baseline- Cross Validation

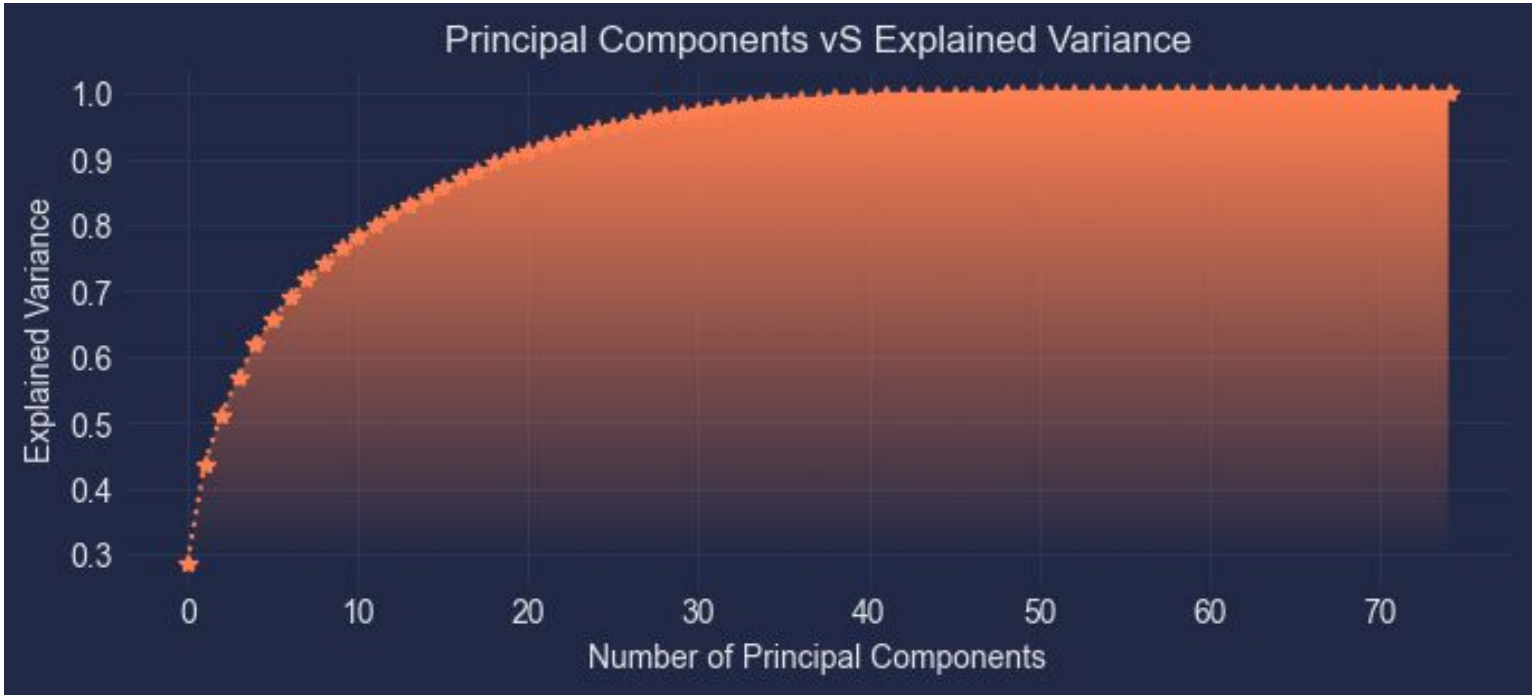
We have used Repeated Stratified K-Cross Validation Technique.

It ensures that every fold has similar data distribution as in the entire dataset, thus, giving a consistent model performance in every iteration.

Since, the dataset is quite large, we have used K=3.

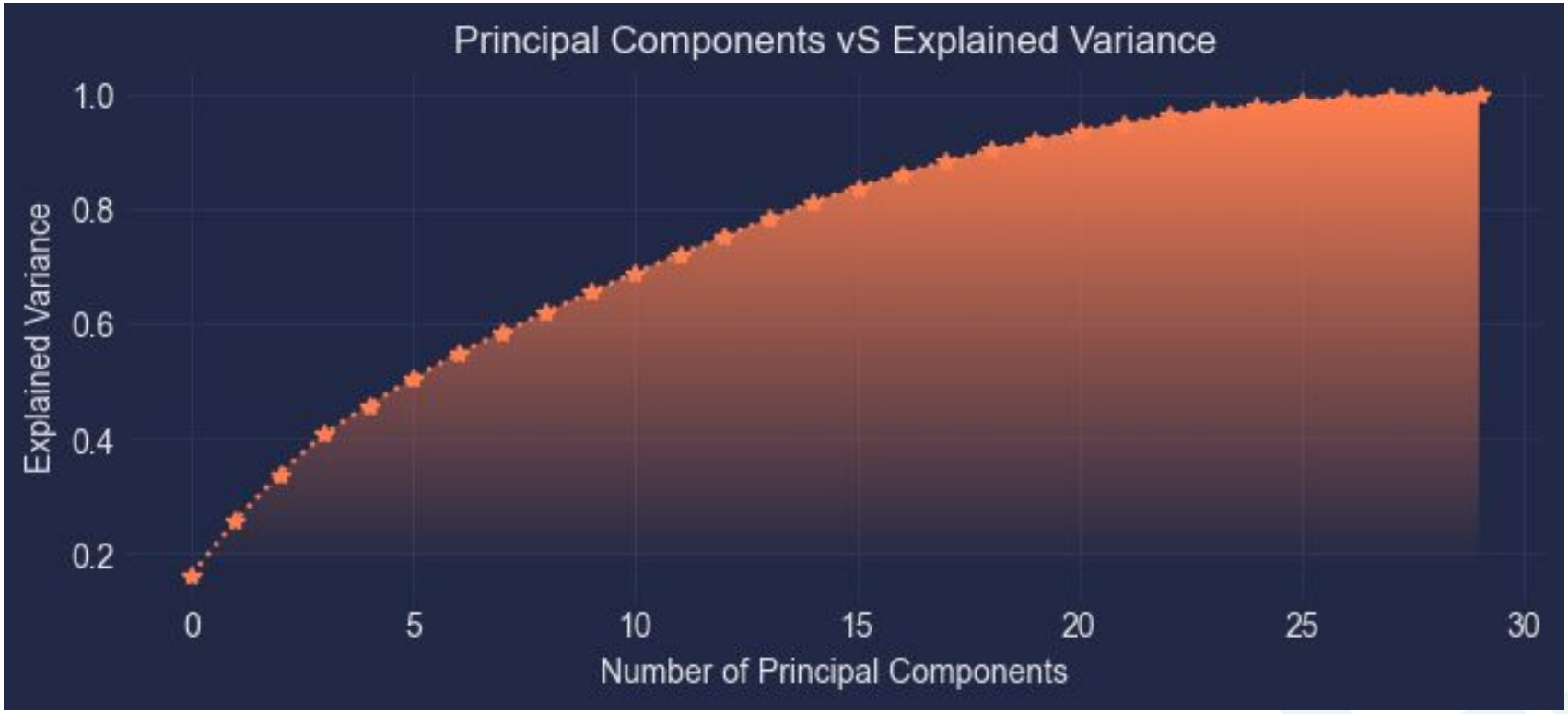
Finally we have trained the models and evaluated their performance using multiclass metrics like accuracy, precision, recall and F1 Score.

PCA on Original Dataset



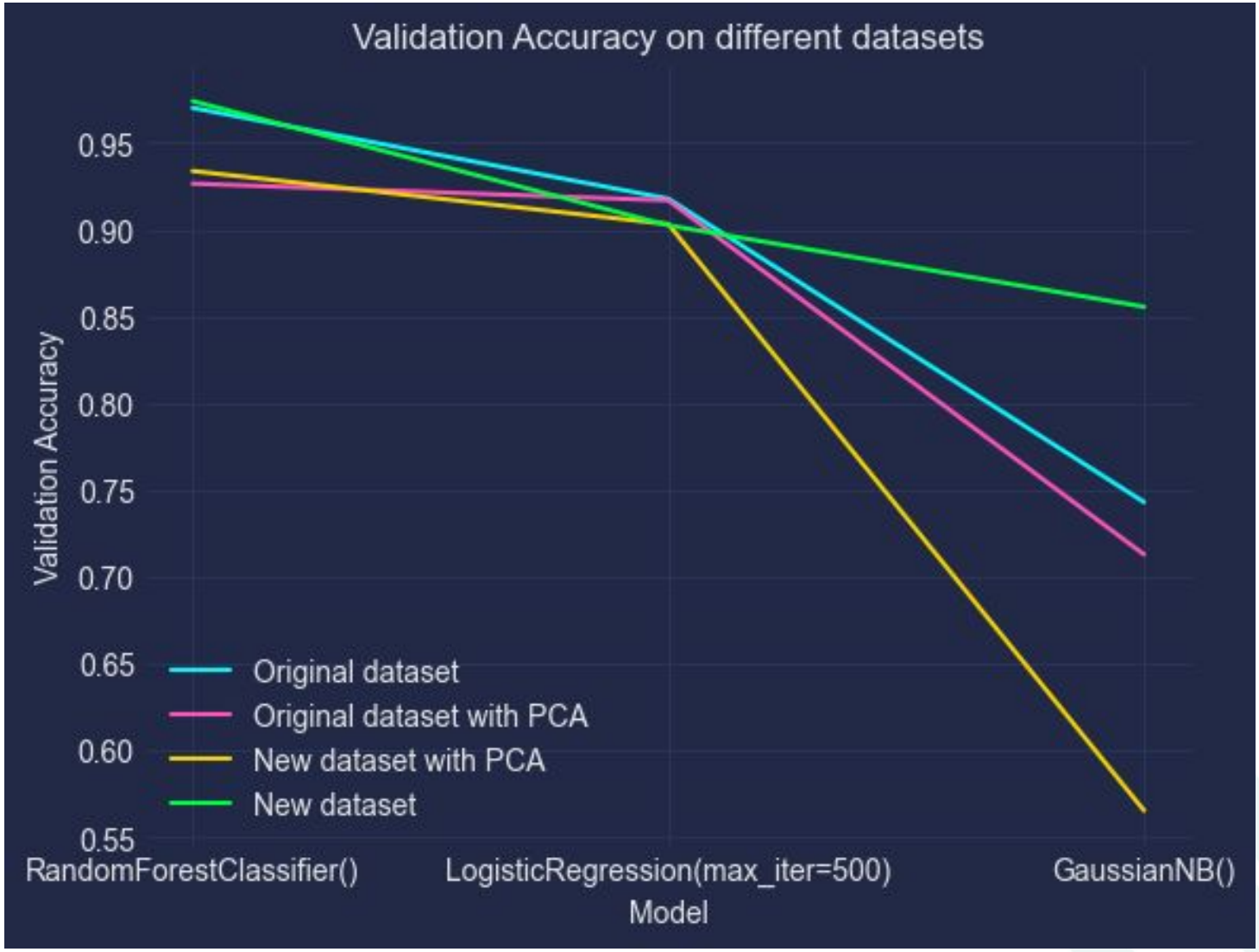
Optimal no of PCA comps = 26 with explained variance of 95%

PCA on Reduced Dataset



Optimal no of PCA comps = 22 with explained variance of 95%

Accuracies on Baseline Model



Results – Reduced Dataset (before Mid Evaluation)

Training Scores

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.986099	0.986083	0.986099	0.986088
LogisticRegression()	0.887590	0.900218	0.887590	0.874707
GaussianNB()	0.716279	0.827792	0.716279	0.732041

Validation Scores

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.964377	0.964326	0.964377	0.964348
LogisticRegression()	0.887560	0.900158	0.887560	0.874681
GaussianNB()	0.716534	0.828115	0.716534	0.732355

Here too, RF performs the best, and the difference between training and testing accuracy is less than in case of original dataset. Thus, **less variance.**

Results – Analysis (Baseline)

From the bias-variance analysis, we can infer that the feature selection and preprocessing is useful.

It is evident that the Random Forest classifier performs better than the others.

Comment of separability and linearity:

As Logistic regression doesn’t perform that well. This indicates that the data is not perfectly linearly separable, thus, linear models are less likely to perform well on it.

Random Forest performed better because it doesn’t assume any distribution. Also, due to ensemble learning involved, reduces variance.

1. PCA decreases the accuracy on the new dataset. This is because dataset is already minimized after feature reduction, thus, applying PCA with even lesser components leads to significant information loss => Don't apply PCA

2. RandomForest performs best among all the baseline classifiers

Changes after Mid-Evaluation

It also indicated that tree based and non linear models can perform better on this dataset. Thus, we created the following models after mid evaluation:

1. Gradient Boost
2. SVM (with kernel trick)

We also improved our preprocessing by applying:

1. Clustering (as an additional feature)
2. Outlier Removal using LOF
3. Random Under sampling

SVM

Objective: SVM finds the optimal hyperplane in high-dimensional space for effective classification or regression.

Margin Maximization: It maximizes the margin, the distance between the hyperplane and the nearest data points of different classes.

Flexibility: SVM handles both linear and non-linear relationships using various kernels.

Applications: Widely used in diverse fields like image recognition, text classification, and bioinformatics.

Advantages: SVM is robust against overfitting, effective in high-dimensional spaces, and known for its accuracy and efficiency in handling various types of data.

→**We have used SVM to improve Hyperplane Based Models.**

GRADIENT BOOSTING

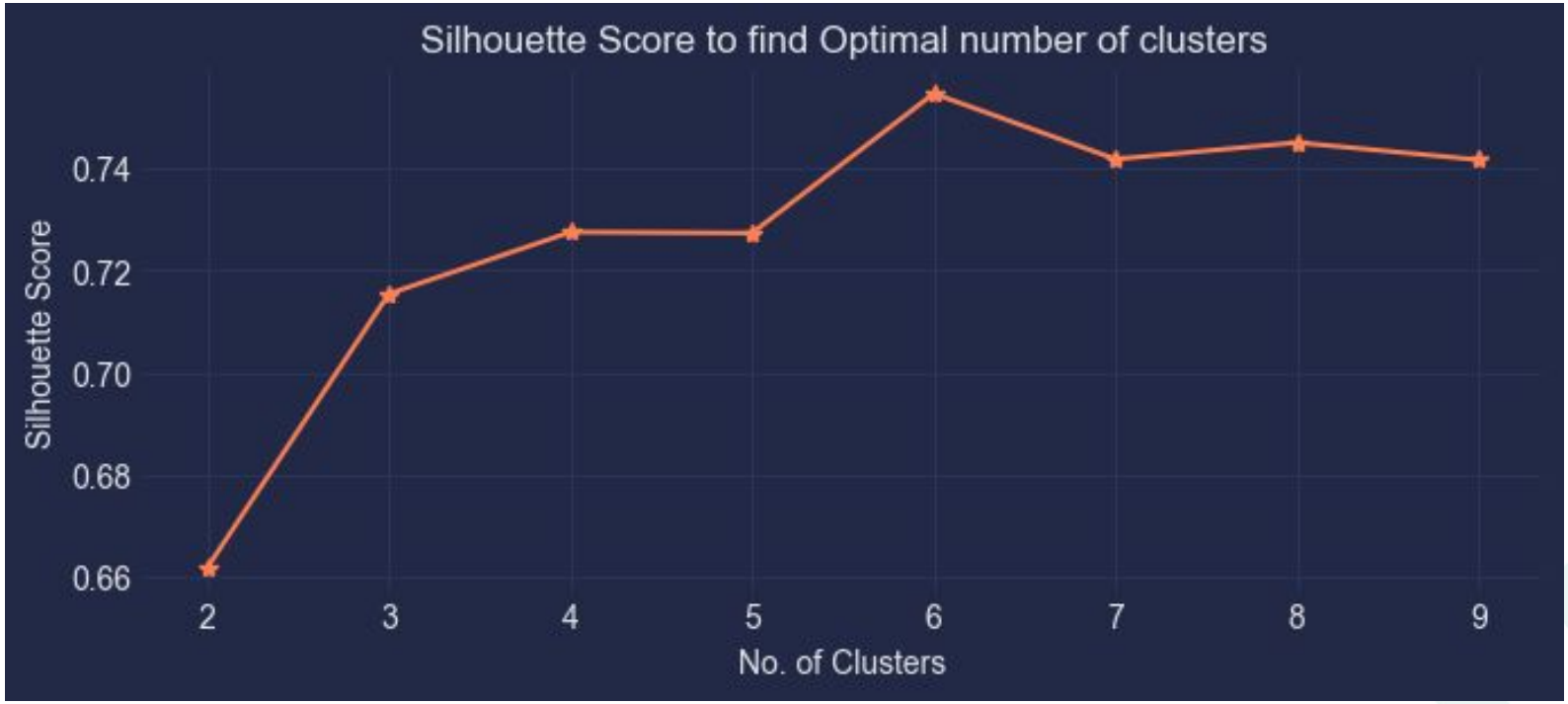
Gradient Boosting is an ensemble learning algorithm that combines the predictive power of multiple learners, which is Decision Tree in our case.

The algorithm works iteratively with each new tree focusing on the errors made by the combination of existing trees. In each iteration, the algorithm calculates the gradient of the loss function with respect to the predicted values, and the new tree is built to minimize these errors.

→ **In our project, as Random Forest worked out as best baseline model, the Gradient Boosting was expected to get a better performance than the Random Forest.**

FEATURE ENGINEERING

We clustered the dataset using K-means by finding the optimal K-Value, after this we assigned a cluster label to all data points , and appended this column in the dataset.

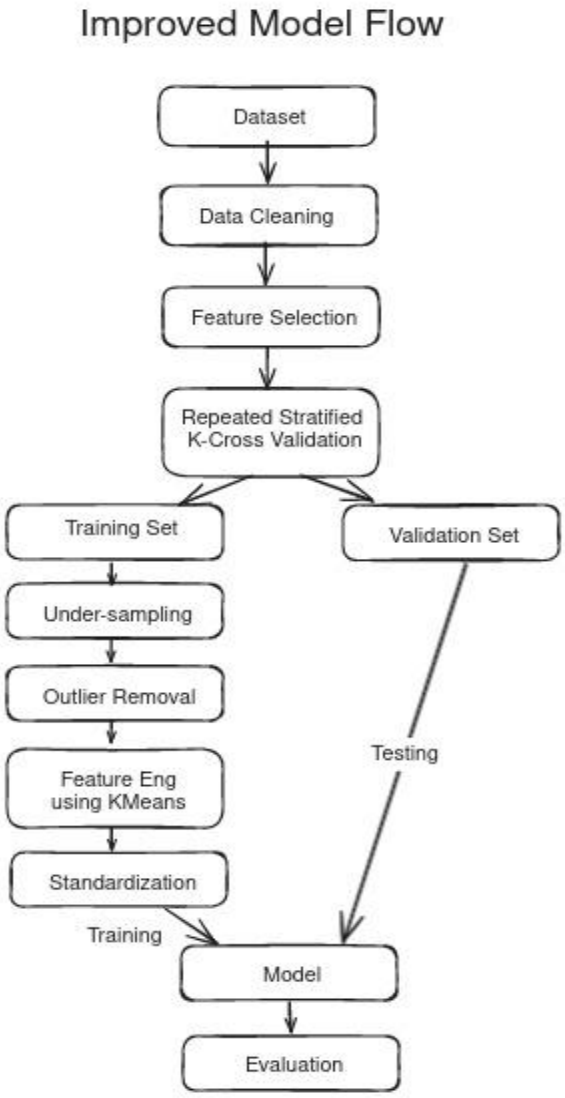


Optimal Number of Clusters = 6 using Silhouette score analysis, with Silhouette Score = 0.76

LOF

- LOF (Local Outlier Factor): A machine learning algorithm for outlier detection.
- Local Density Comparison: Calculates the local density of data points by considering their neighbors.
- Relative Density Measure: Computes the LOF for each point based on its local density compared to neighbors.
- Outlier Identification: Points with significantly lower density, indicating isolation, are labeled as outliers.
- Context-Sensitive Detection: LOF adapts to varying density regions, offering robust outlier detection in complex datasets.

Model Training – Improved



After outlier removal using LOF

```
X inliers

Training Scores
```

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.995991	0.995997	0.995991	0.995991
GradientBoostingClassifier()	0.982249	0.982927	0.982249	0.982237
SVC()	0.943899	0.944068	0.943899	0.943889
SVC(kernel='poly')	0.915842	0.917106	0.915842	0.915809

```
Validation Scores
```

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.974196	0.975157	0.974196	0.974391
GradientBoostingClassifier()	0.976903	0.979006	0.976903	0.977174
SVC()	0.927278	0.932464	0.927278	0.928439
SVC(kernel='poly')	0.893384	0.909019	0.893384	0.897279

Results After Feature Engineering

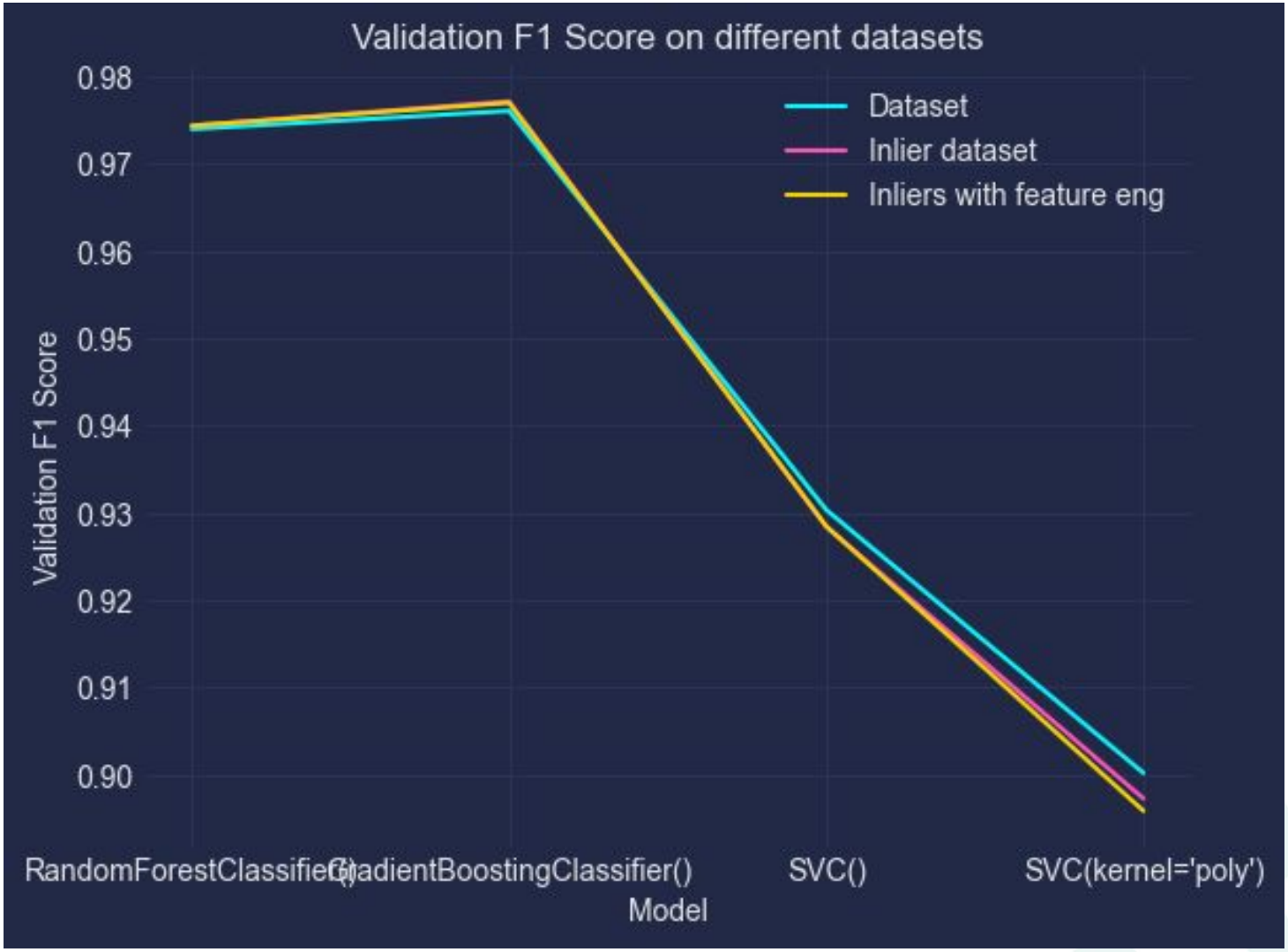
```
X inliers with feature engineering

Training Scores
```

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.996158	0.996161	0.996158	0.996158
GradientBoostingClassifier()	0.981651	0.982370	0.981651	0.981637
SVC()	0.942425	0.942612	0.942425	0.942416
SVC(kernel='poly')	0.914047	0.915252	0.914047	0.914015

```
Validation Scores
```

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.974210	0.975052	0.974210	0.974390
GradientBoostingClassifier()	0.976743	0.978839	0.976743	0.977016
SVC()	0.927319	0.932604	0.927319	0.928488
SVC(kernel='poly')	0.891983	0.907468	0.891983	0.895870

Accuracies on improved Models



Results of Final Model

```
X inliers with feature engineering

Training Scores
```

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.996158	0.996161	0.996158	0.996158
GradientBoostingClassifier()	0.981651	0.982370	0.981651	0.981637
SVC()	0.942425	0.942612	0.942425	0.942416
SVC(kernel='poly')	0.914047	0.915252	0.914047	0.914015

```
Validation Scores
```

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.974210	0.975052	0.974210	0.974390
GradientBoostingClassifier()	0.976743	0.978839	0.976743	0.977016
SVC()	0.927319	0.932604	0.927319	0.928488
SVC(kernel='poly')	0.891983	0.907468	0.891983	0.895870

Results – Analysis (Improved Models)

1. All the datasets perform similarly. But dataset with outliers removed and feature engineering has slightly better performance in case of Gradient Boosting Model.

2. out of all the models, Gradient Boosting performs the best and should be selected as the final model.

→Tree based ensemble models like gradient boosting are very flexible in capturing non linearity in data.

→Though SVM with kernels can also handle non linear data, but this highly depends on the choice of the kernel and the type of non-linearity in the data. If data is too complex then kernel might not be able to identify its pattern , that is why, here, gradient boosting is outperforming kernel based SVM.

Individual Contribution

- Data Analysis and Literature Review - Mohit
- Data Cleaning and Preprocessing - Samyak Jain
- Visualisations and Analysis of Results - Harshit Garg
- Model Training and Validation - Sachin Sharma

Collaboration: All team members contributed equally towards performing experiments and analyzing data and results.

Project Timeline

Timeline Followed :-

- Data Analysis and Cleaning (2 weeks)
- Data Preprocessing and Feature Selection (2 weeks)
- Model Training, Hyperparameter Tuning, and Validation (3 weeks)
- Model Testing and Analysis of Predictions (3 weeks)
- Writing Report and Presentation (1 week)
- Validation (1 week)
- Model Analysis (1 week)
- Results and Prediction Analysis (1 week)

References

[1] J. Li, L. Zhai, X. Zhang and D. Quan, "Research of android malware detection based on network traffic monitoring," 2014 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, China, 2014, pp. 1739-1744, doi: 10.1109/ICIEA.2014.6931449.

[2]B. Amos, H. Turner and J. White, "Applying machine learning classifiers to dynamic Android malware detection at scale," 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), Sardinia, Italy, 2013, pp. 1666-1671, doi: 10.1109/IWCMC.2013.6583806.

[3]S. Y. Yerima, S. Sezer, G. McWilliams and I. Muttik, "A New Android Malware Detection Approach Using Bayesian Classification," 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), Barcelona, Spain, 2013, pp. 121-128, doi: 10.1109/AINA.2013.88.

[4] Ham, Hyo-Sik & Kim, Hwan-Hee & Kim, Myung-Sup & Choi, Mi-Jung. (2014). Linear SVM-Based Android Malware Detection for Reliable IoT Services. Journal of Applied Mathematics. 2014. 1-10. 10.1155/2014/594501.