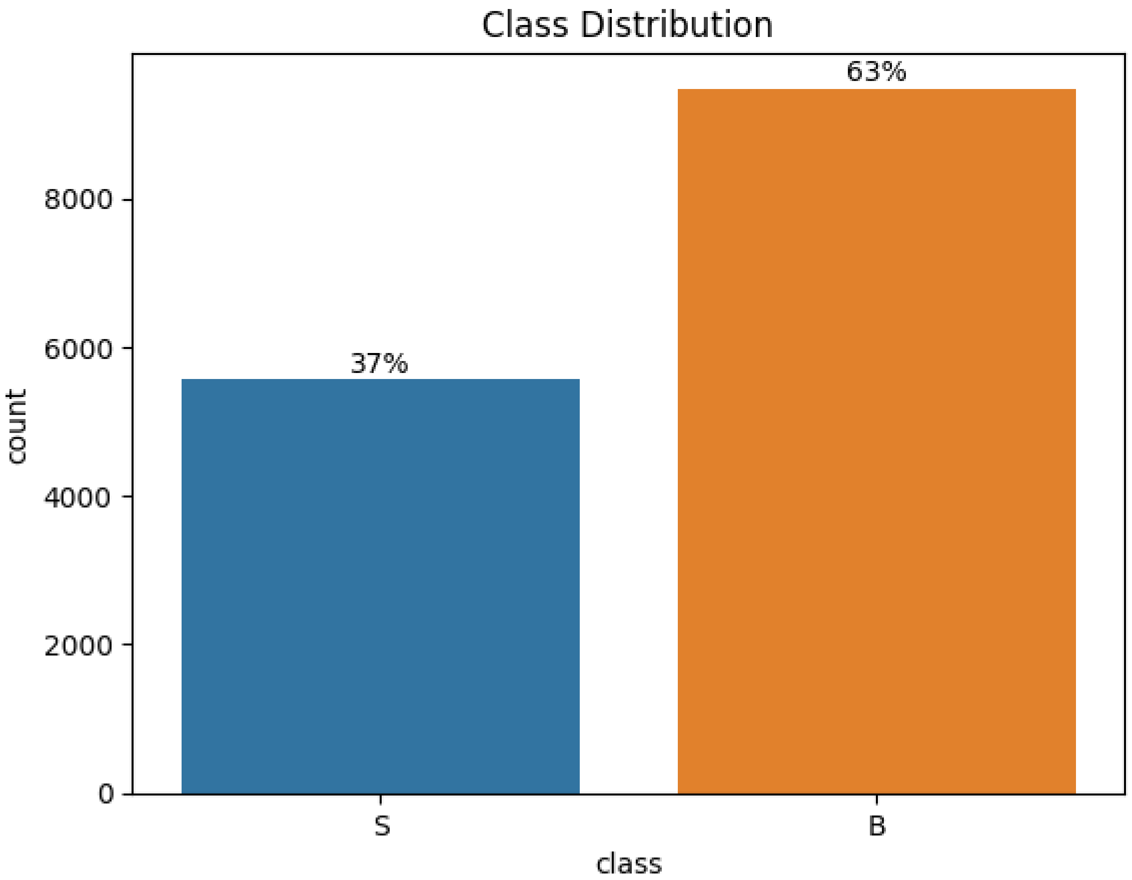
AI & ML CyberSecurity

Mageswaran Dhandapani
2022mt12028

Github: https://github.com/Mageswaran1989/aiml-bits
Notebook: https://github.com/Mageswaran1989/aiml-bits/blob/main/Exploration.ipynb
Video Link:
https://drive.google.com/drive/folders/12Th-3CBeKPSh-y7eP9XUEyUKNjEKi4uL?usp=sharing

Dataset

- Dataset contains static characteristics (features) of the malwares like : API call signature, Manifest Permission, Intent, Commands signature
- Dataset consisting of feature vectors of 215 attributes extracted from 15,036 applications (5,560 malware apps from Drebin project and 9,476 benign apps)
- File "dataset-features-categories.csv" gives information about the features and their Android function.
    - Following unique feature types are present:
        - 'API call signature', 'Manifest Permission', 'Intent', 'Commands signature'
    - There are two classes
        - B=Benign
        - S=Malware



Preprocessing

- Class values of “B” and “S” are first converted into numeric using LabelEncoder

- It is found that the data set contains characters (S , ?) by mistake which is first replaced with NaN, the number of rows affected were 5, since the number of missing rows are less, those rows are dropped

Feature Selection

- **Correlation**

  Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have high correlation, we can drop one of the two features.
- **Chi Square Test**

  Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with best Chi-square scores. It determines if the association between two categorical variables of the sample would reflect their real association in the population.

  We calculate chi-square statistics between every feature variable and the target variable and observe the existence of a relationship between the variables and the target. If the target variable is independent of the feature variable, we can discard that feature variable. If they are dependent, the feature variable is very important.

ML Algorithms

Random Forest and Support Vector Machine (SVM) for Android Malware classification for following reasons:

1. Random Forest and SVM are both powerful and widely used machine learning algorithms that have been proven to be effective in classification tasks, including malware classification.
2. Random Forest is a popular ensemble method that combines multiple decision trees to reduce overfitting and improve the generalization of the model. It is well suited for high-dimensional data and nonlinear relationships between features, which is often the case in malware classification.
3. SVM is a well-established algorithm for classification and is particularly useful when the data is separable into distinct classes. SVM has been used extensively in malware classification and is known to perform well in this domain.
4. Both Random Forest and SVM are able to handle different types of features, including binary, continuous, and categorical data. This makes them suitable for analyzing the different types of features that are typically used to classify malware.
5. Random Forest and SVM are complementary algorithms, with different strengths and weaknesses. By using both algorithms in combination, it is possible to obtain a more accurate and robust classification model.

6. Both algorithms have a well-established track record in the field of malware classification, with numerous studies demonstrating their effectiveness. This provides confidence that the chosen algorithms are likely to perform well in this context.

Overall, choosing Random Forest and SVM for Android Malware classification is a good decision due to their effectiveness, ability to handle different types of features, and complementary strengths. By using both algorithms in combination, it is possible to build a more accurate and robust classification model.

Random Forest

It belongs to the family of ensemble methods, which combine multiple models to improve the performance and reduce overfitting.

Random Forest builds a collection of decision trees, where each tree is trained on a random subset of the data and a random subset of the features. This process is called bootstrap aggregating or bagging. By training each tree on a different subset of the data and features, the trees are able to capture different aspects of the data and reduce the variance of the model.

Strengths:

- Random Forest is an ensemble learning method that builds multiple decision trees and aggregates their outputs. It is robust to overfitting and can handle high-dimensional data.
- It can handle missing values and can be used for both classification and regression problems.
- Random Forest can provide feature importance rankings that can help in feature selection.

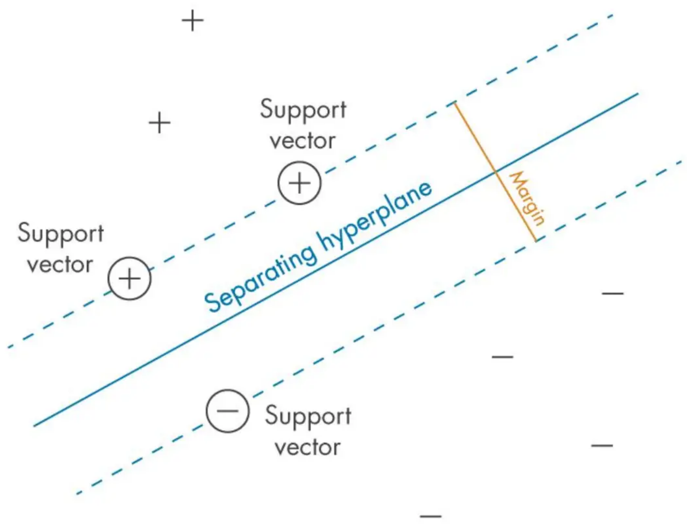
Weaknesses:

- Random Forest can be computationally expensive and may require more time for training compared to other algorithms.
- It can also be difficult to interpret and visualize the individual trees within the ensemble.

Support Vector Classifier

- SVM for classification is termed as support vector classification (SVC)

- SVM classifier is also termed as maximum margin classifier, meaning that it finds the line or hyperplane that has the largest distance to the nearest training data points (support vectors) of any class.



- The data points that are closest to the hyperplane are called support vectors
- When the data is not linearly separable, SVMs use a technique called kernel trick, which maps the original features into a higher-dimensional feature space where the data is linearly separable.
- **Hyperparameters**
  - **C** is typically used as a regularization parameter to control overfitting. C controls the trade-off between maximizing the margin and minimizing training error.
    - C inversely proportional to margin size
    - Larger value of C will cause model overfitting
  - The **gamma** hyperparameter can take on two special values: **'scale'**, **'auto'** or **'float'**. When gamma is set to 'scale', it means that the value of gamma is calculated as 1 / (number of features * variance in data). This ensures that all features are given equal importance in the model and produces consistent results no matter how many features are used.
  - **Kernel** : 'linear', 'poly', 'rbf' and 'sigmoid'.
    - When the kernel is set to **'linear'**, it means that the model will use a linear boundary for classification and regression. This is the simplest type of SVM and works best when data are linearly separable.
    - When the kernel is set to **'poly'**, it means that the model will use polynomial functions of degree higher than 1 for classification. This type of SVM is more suitable for complex non-linear datasets.
    - When the kernel is set to **'rbf'**, it means that the model will use radial basis functions for classification or regression. RBF kernels are capable of dealing with complex multi-class datasets and have good generalization performance with noisy data points.

- When the kernel is set to **‘sigmoid’**, it means that the model will apply sigmoid functions instead of RBFs for classification or regression tasks. Sigmoid kernels tend to be less sensitive than RBFs with respect to outliers in data but may not generalize as well unless their parameters are tuned properly.

Strengths:

- SVM is a powerful algorithm for classification that can handle non-linear decision boundaries and high-dimensional data.
- It is effective in dealing with noisy data and can avoid overfitting by using regularization.
- SVM can be used with different kernel functions that can capture complex relationships between features.

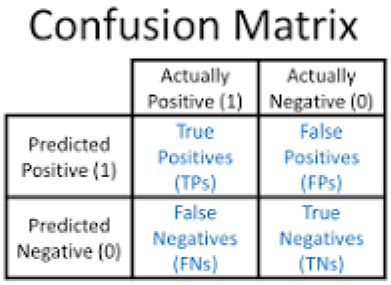
Weaknesses:

- SVM may require a careful selection of the kernel function and its parameters, which can be time-consuming and may require expert knowledge.
- SVM is sensitive to the choice of the kernel function and can be prone to overfitting if not properly tuned.

Model Performance Measurement

Performance measurement of the classification process is first done using the confusion matrix.

Confusion Matrix



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)

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm.

Accuracy

Model accuracy is defined as the number of classifications a model correctly predicts divided by the total number of predictions made

Precision

- Measures how many of the applications the model classifies as malware are true malware
- Precision measures how frequently a sample app is malware when the model predicts that it is malware. In other words, it is the True Positive Rate of the model. It is calculated by dividing the True Positives by the total number of Positive predictions.

Recall

- Measures how many of the malware samples in the dataset are correctly classified as malware
- Recall measures how much of the actual malware in the data is correctly predicted as malware by the model. This measure helps in ensuring that the model accurately detects as much malware as possible to prevent harm. It is calculated by dividing the True Positive by the total number of actual Positive samples.

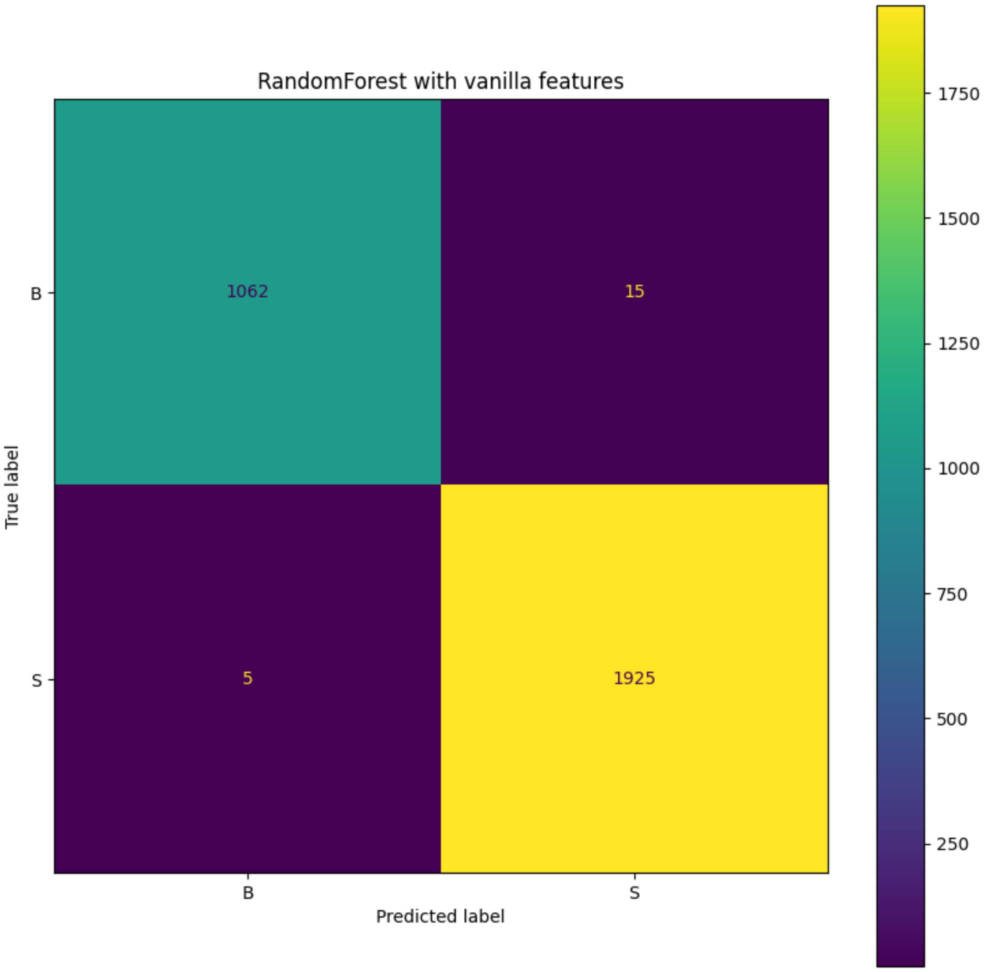
F1-score

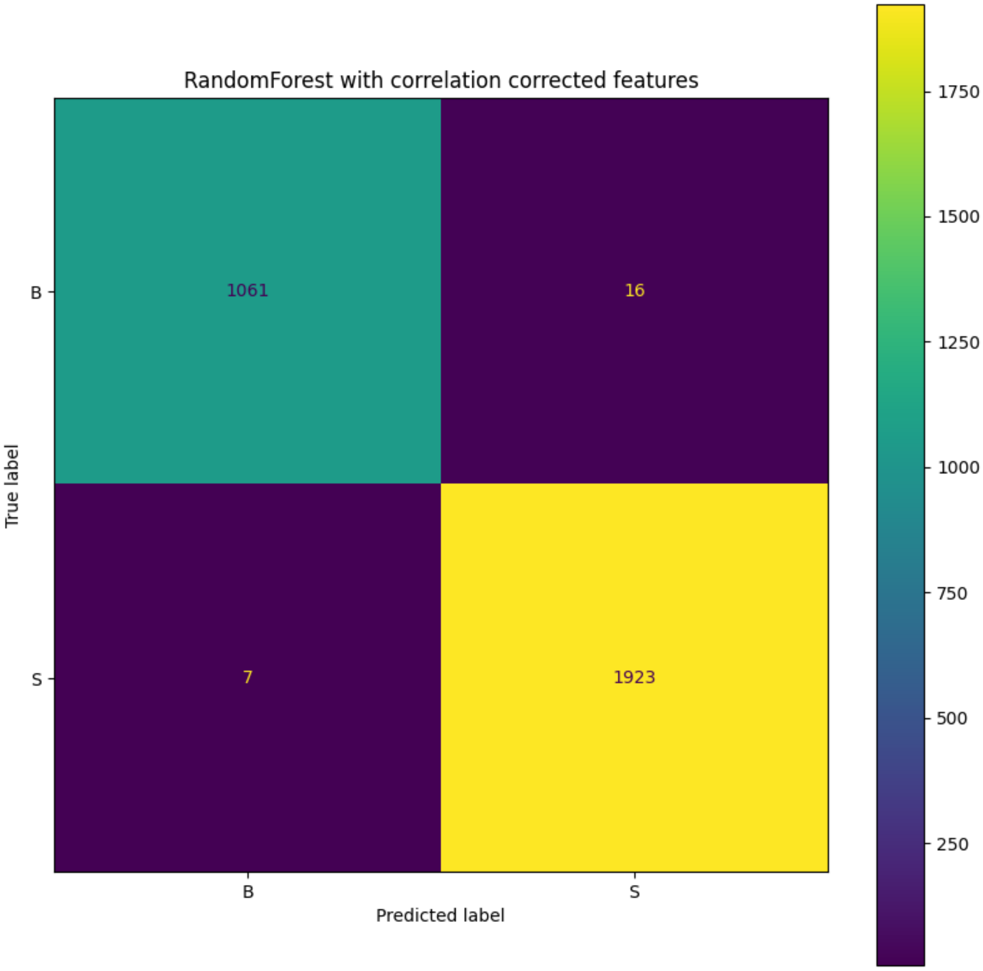
- It is the harmonic mean between precision and recall and measures the overall accuracy of the model – it should be as high as possible.

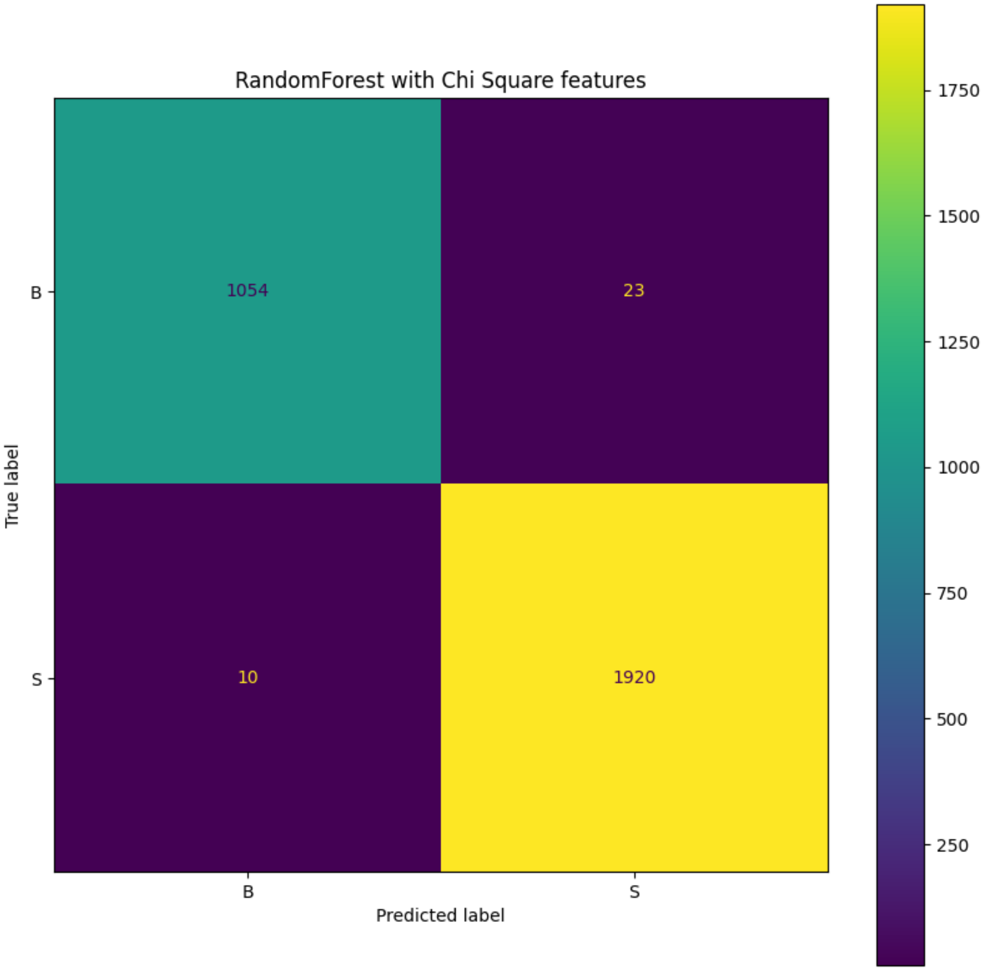
Two models and three feature sets are used totaling to 6 experiments.

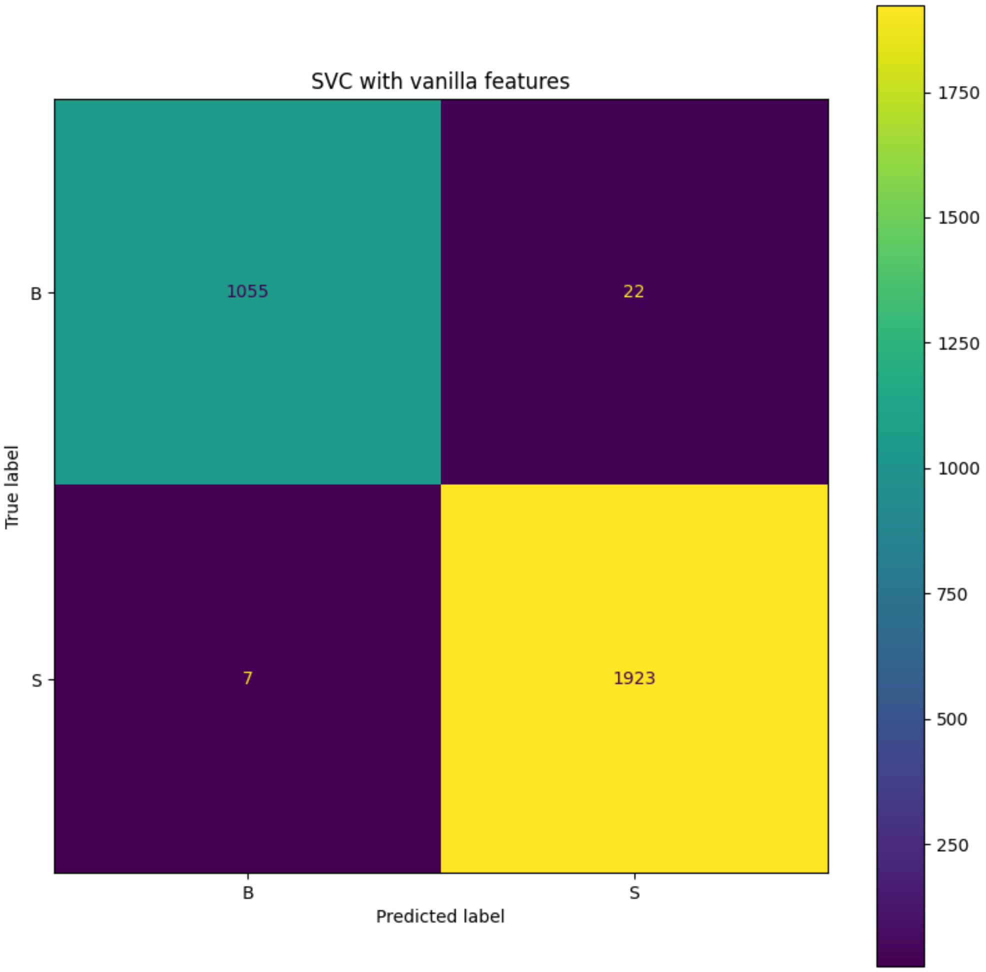
Here we use the metrics **accuracy** and **recall**. In a malware detection problem, it is more important to identify all the true malware correctly than to ensure that all the malware identified is actually true malware. Therefore, recall is more important than precision.

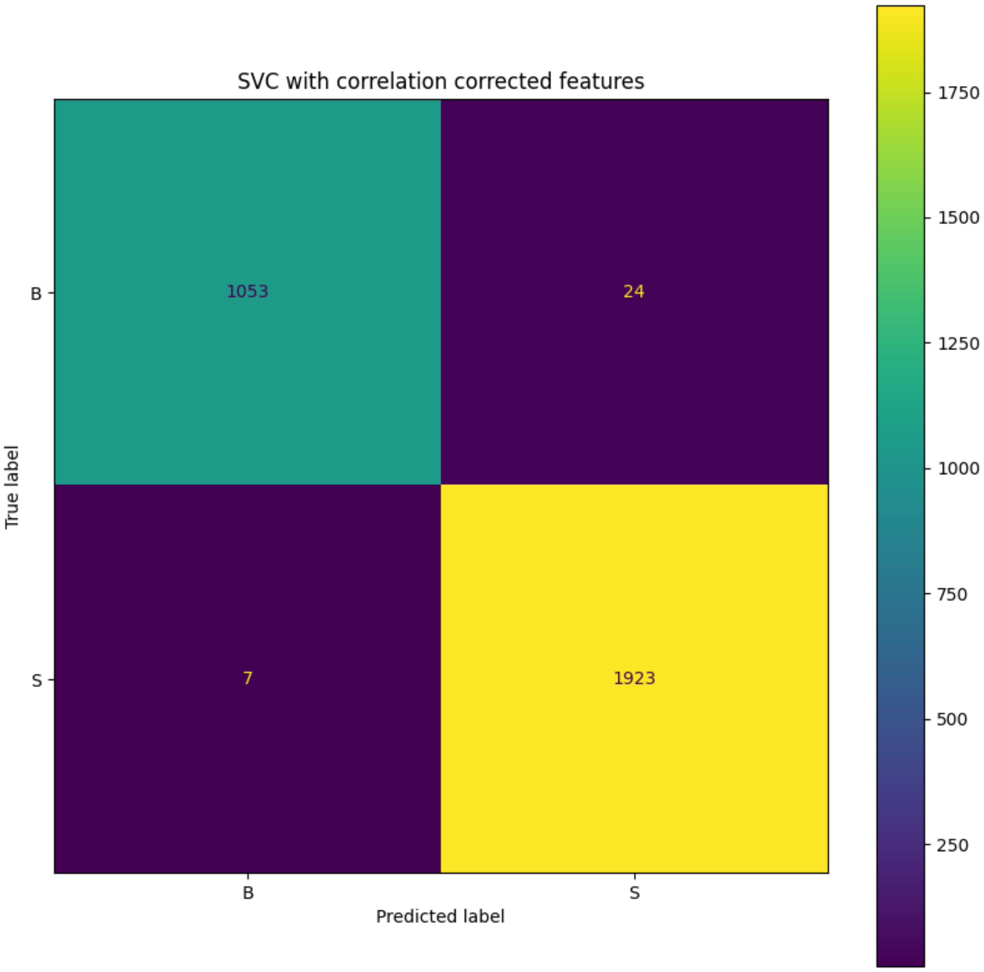
Here are the confusion matrix screenshots for each:

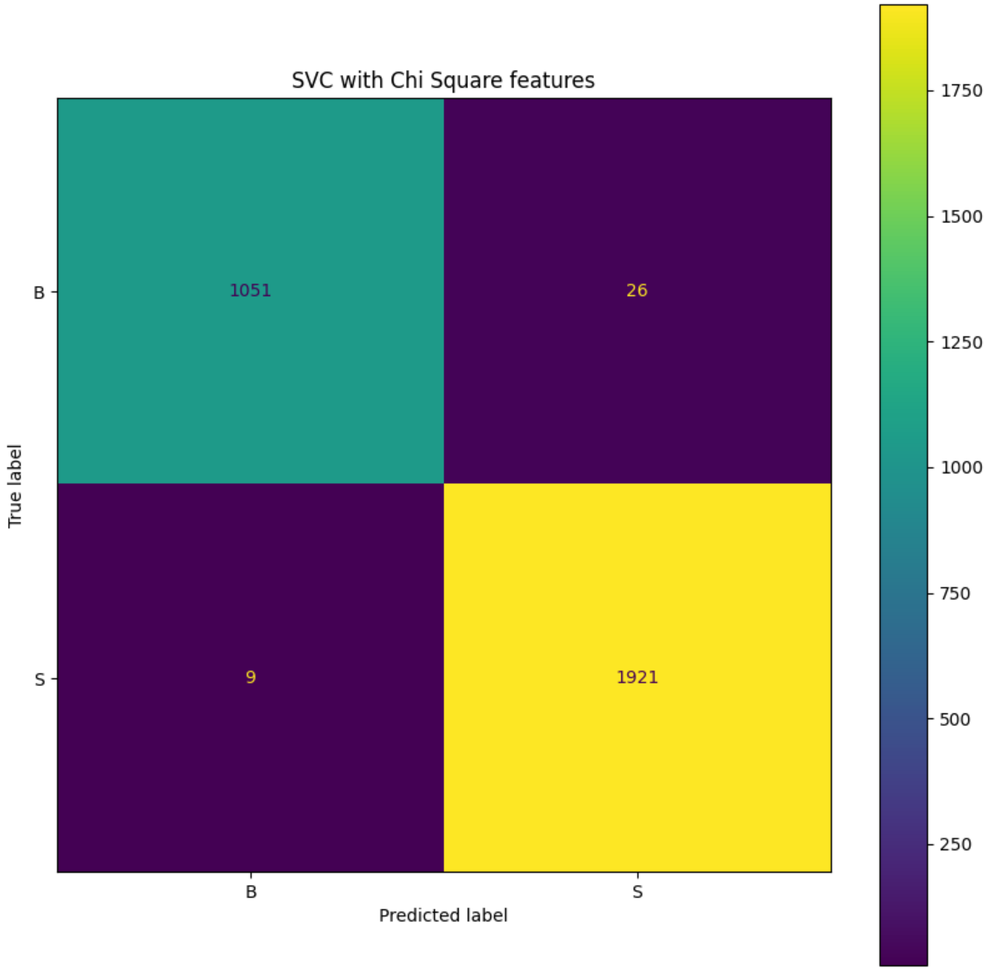

RandomForest with vanilla features
True label
B
S
1062
15
5
1925
B
S
Predicted label
1750
1500
1250
1000
750
500
250


RandomForest with correlation corrected features
True label
B
S
1061
16
7
1923
B
S
Predicted label
1750
1500
1250
1000
750
500
250

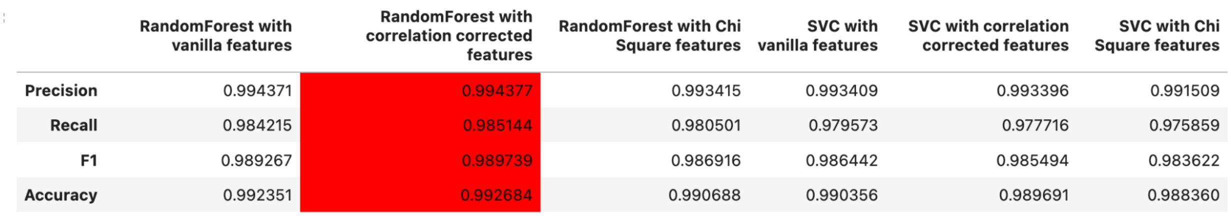

RandomForest with Chi Square features
True label
B
S
Predicted label
B
S
1054
23
10
1920
1750
1500
1250
1000
750
500
250


SVC with vanilla features
True label
B
S
1055
22
7
1923
B
S
Predicted label
1750
1500
1250
1000
750
500
250


SVC with correlation corrected features
True label
B
S
Predicted label
B
S
1053
24
7
1923
1750
1500
1250
1000
750
500
250



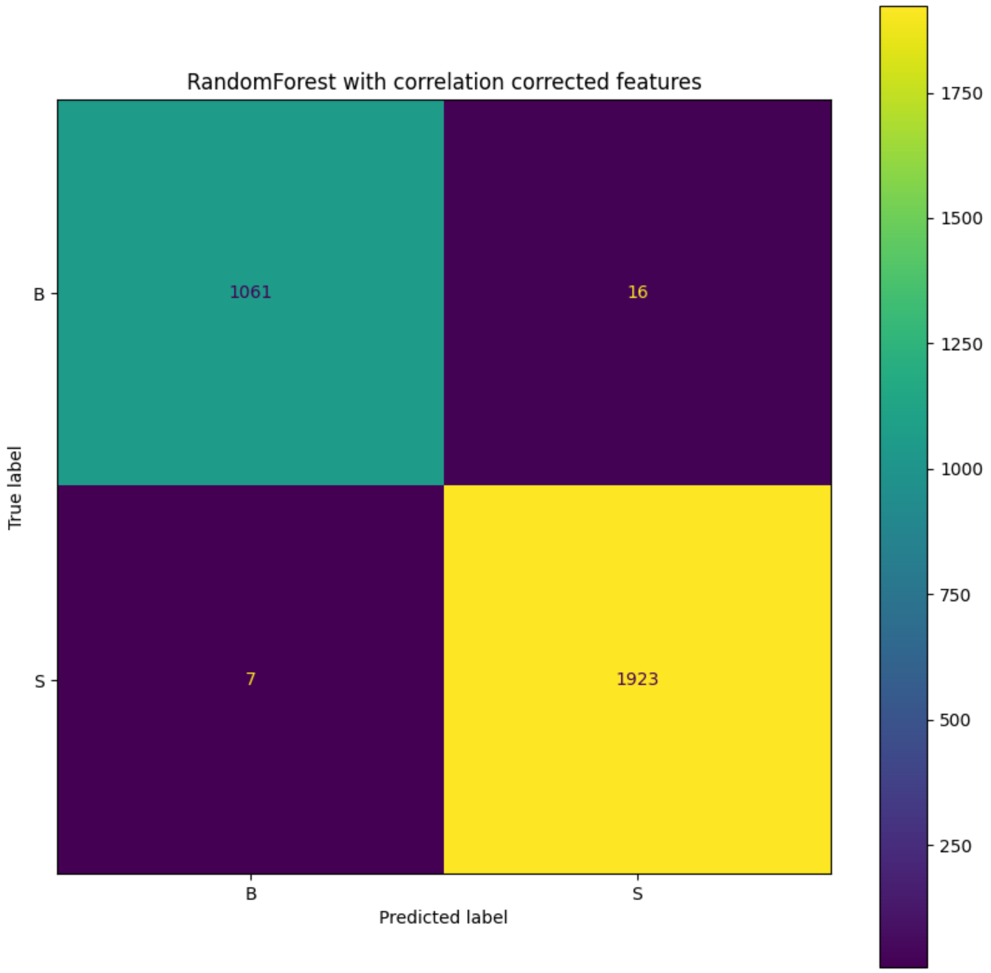
Metrics Collected for each experiment:



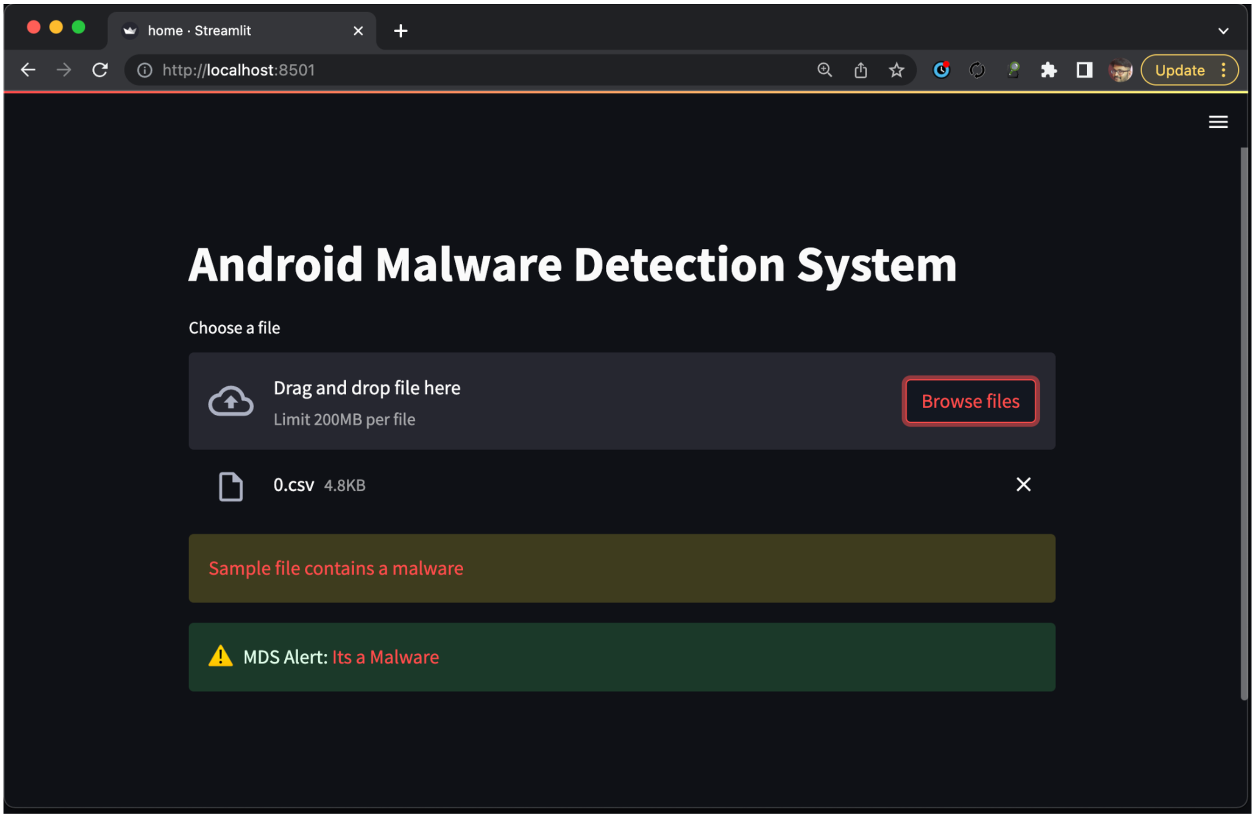
	RandomForest with vanilla features	RandomForest with correlation corrected features	RandomForest with Chi Square features	SVC with vanilla features	SVC with correlation corrected features	SVC with Chi Square features
Precision	0.994371	0.994377	0.993415	0.993409	0.993396	0.991509
Recall	0.984215	0.985144	0.980501	0.979573	0.977716	0.975859
F1	0.989267	0.989739	0.986916	0.986442	0.985494	0.983622
Accuracy	0.992351	0.992684	0.990688	0.990356	0.989691	0.988360

Conclusion:

It is evident that the experiment with RandomForest as classifier and correlation feature selection using has best performance compared to all other experiments carried out.



Hence the pipeline “RandomForest with correlation corrected features” is stored and used for model serving.



Android Malware Detection System

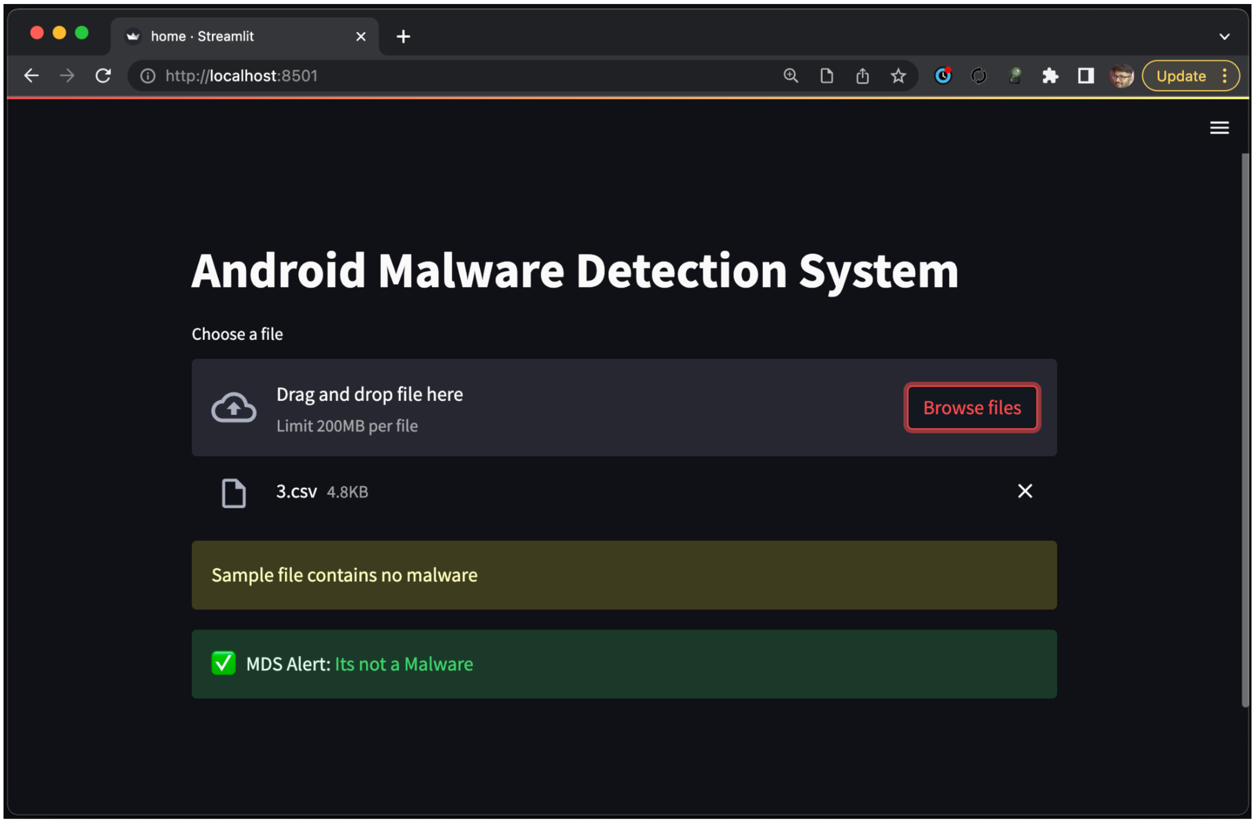
Choose a file

Drag and drop file here
Limit 200MB per file

0.csv 4.8KB

Sample file contains a malware

⚠ MDS Alert: Its a Malware



Android Malware Detection System

Choose a file

Drag and drop file here
Limit 200MB per file

3.csv 4.8KB

Sample file contains no malware

✅ MDS Alert: Its not a Malware

References:

- https://www.kaggle.com/datasets/shashwatwork/android-malware-dataset-for-machine-learning
- https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/5854653
- https://www.simplilearn.com/tutorials/machine-learning-tutorial/feature-selection-in-machine-learning
- https://vitalflux.com/classification-model-svm-classifier-python-example/
- https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/
-