Assignment 1

10/30/2023
SRT521

Rayyan Khan
Jia Ying Ou

Table of Contents:

Contents

Table of Figures:

Introduction

Malicious software (malware) continues to evolve, becoming more sophisticated and harder to detect. Static analysis, a process that examines malware samples without executing them, plays a crucial role in identifying and classifying malware. This research provides a step-by-step guide to perform static analysis, choose relevant features, and evaluate the performance of different machine learning algorithms.

Detection of Malware: Signature-Based vs. Behavior-Based Analysis

Malware detection is a critical aspect of cybersecurity, aimed at identifying and mitigating the threats posed by malicious software. Two fundamental approaches to malware detection are signature-based analysis and behavior-based (heuristic) analysis. These approaches differ in their methodology, strengths, and limitations.

Signature-based analysis relies on predefined patterns, known as signatures or fingerprints, of known malware, generated from the code or characteristics of previously identified malware samples. When scanning a file or program, it compares its signature against a database of known malware signatures, offering high accuracy and speed but limited effectiveness against new or modified malware lacking predefined signatures. Malware creators can easily evade it by changing signatures, and maintaining a comprehensive signature database is resource intensive.

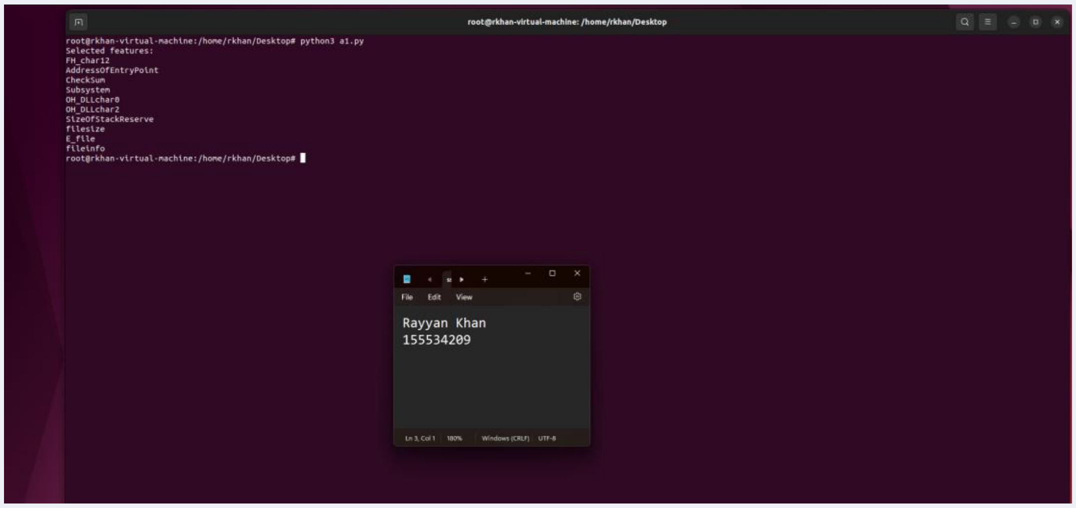
In contrast, behavior-based (heuristic) analysis focuses on a program or file's actions and behavior rather than static code patterns. It observes the execution, monitoring interactions with the system and other processes, flagging deviations from expected, benign behavior as potentially malicious. This method excels at detecting unknown and evolving threats, adapting to changes in malware code, and reducing false positives. However, it can be resource-intensive due to dynamic monitoring and may occasionally yield false negatives if legitimate software exhibits behavior like malware. Designing and maintaining heuristic rules can also be complex and require regular updates.

Dataset

The dataset that is being used is e “ClaMP_Integrated-5184.csv” from the Classification of Malwares (CLaMP)

Feature Extraction

Reducing vector dimensions through feature selection is crucial for effective malware detection and data analytics. It enhances model performance by preventing overfitting, improves resource efficiency, and boosts model interpretability. Selecting the most informative features streamlines the analysis, reduces noise, and accelerates model training and inference. This approach also addresses the curse of dimensionality, ensuring that the model can efficiently and accurately distinguish between malware and benign samples.



```
root@rkhan-virtual-machine:/home/rkhan/Desktop# python3 a1.py
Selected features:
FH_char12
AddressOfEntryPoint
CheckSum
Subsystem
OH_DLLchar0
OH_DLLchar2
SizeOfStackReserve
filesize
E_file
fileinfo
root@rkhan-virtual-machine:/home/rkhan/Desktop#
```

```
Rayyan Khan
155534209
```

Figure 1

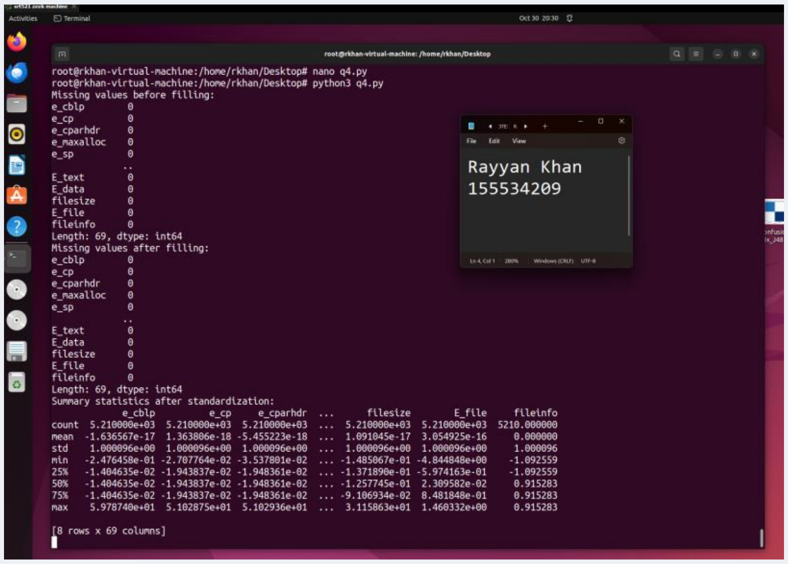
The following figure shows all the features that were extracted that will give us the best results.

This code demonstrates feature selection using Recursive Feature Elimination (RFE) with a Random Forest Classifier as the estimator. It loads a dataset, converts categorical data to numerical values, defines features and labels, initializes the classifier, and employs RFE to select the top 10 most important features. Finally, it prints the selected features, providing a clear insight into the most relevant attributes for a classification task.

Feature Transformation

Preparing the extracted features before feeding them into a learning model is a critical step in the data preprocessing pipeline. It involves a series of transformations to ensure that the data is in a suitable format for the machine learning algorithm. Here, we'll elaborate on the various transformations mentioned: conversion, cleaning, normalization, standardization, and non-linear expansion.

These data transformations are essential for preparing feature sets before they are used to train machine learning models. They ensure that the data is in a consistent, meaningful, and appropriately scaled format, improving the model's ability to generalize from the training data to make accurate predictions or classifications on new, unseen data. The choice of transformation techniques depends on the specific characteristics of the dataset and the requirements of the machine learning algorithm being used.



```
root@rkhan-virtual-machine:/home/rkhan/Desktop# nano q4.py
root@rkhan-virtual-machine:/home/rkhan/Desktop# python3 q4.py
Missing values before filling:
e_cblp        0
e_cp          0
e_cparhdr     0
e_maxalloc    0
e_sp          0
             ..
E_text        0
E_data        0
filesize      0
E_file        0
fileinfo      0
Length: 69, dtype: int64
Missing values after filling:
e_cblp        0
e_cp          0
e_cparhdr     0
e_maxalloc    0
e_sp          0
             ..
E_text        0
E_data        0
filesize      0
E_file        0
fileinfo      0
Length: 69, dtype: int64
Summary statistics after standardization:
             e_cblp          e_cp     e_cparhdr  ...      filesize        E_file     fileinfo
count  5.210000e+03  5.210000e+03  5.210000e+03  ...  5.210000e+03  5.210000e+03  5210.000000
mean  -1.636567e-17  1.363806e-18 -5.455223e-18  ...  1.091045e-17  3.054925e-16     0.000000
std    1.000096e+00  1.000096e+00  1.000096e+00  ...  1.000096e+00  1.000096e+00     1.000096
min   -2.476458e-01 -2.707764e-02 -3.537801e-02  ... -1.485067e-01 -4.844848e+00    -1.092559
25%   -1.404635e-02 -1.943837e-02 -1.948361e-02  ... -1.371890e-01 -5.974163e-01    -1.092559
50%   -1.404635e-02 -1.943837e-02 -1.948361e-02  ... -1.257745e-01  2.309582e-02     0.915283
75%   -1.404635e-02 -1.943837e-02 -1.948361e-02  ... -9.106934e-02  8.481848e-01     0.915283
max    5.978740e+01  5.102875e+01  5.102936e+01  ...  3.115863e+01  1.460332e+00     0.915283

[8 rows x 69 columns]
```

Rayyan Khan
155534209

Figure 2

The output showcases the data preprocessing steps and the results of feature selection in the machine learning pipeline. Initially, it indicates that the dataset had no missing values, which is a positive sign for data quality. After standardization, which rescales features with a mean close to 0 and a standard deviation close to 1, the dataset is now in a consistent format, suitable for many machine learning algorithms.

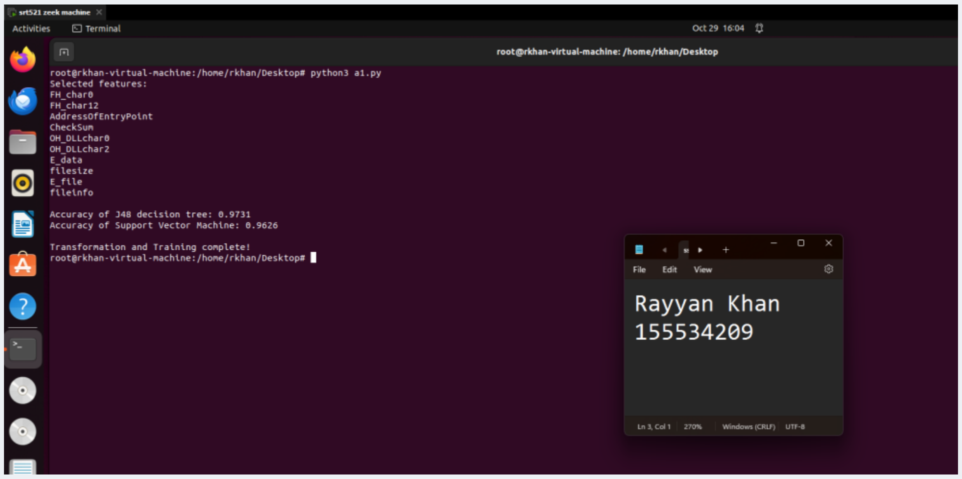
Furthermore, the "Selected features" section provides valuable insight into the feature selection process. These selected features, determined by the Recursive Feature Elimination (RFE) technique using a Random Forest Classifier, have been identified as the most impactful for the classification task. This selection helps enhance model performance by focusing on the most informative attributes while potentially mitigating the risk of overfitting. In summary, the output reflects the successful preparation of the data, ensuring its quality and relevance for subsequent machine learning tasks.

Training the Model

Certainly, here are shorter explanations of the two selected supervised machine learning algorithms:

J48 Decision Tree:
Concept: J48 is a classification algorithm that builds a decision tree. Each node represents a feature, and each branch represents a decision rule. It recursively selects features to minimize impurity and determine class labels.
Strengths: Easy to interpret, handles both data types, and supports feature selection.
Limitations: Prone to overfitting and sensitive to data variations.

Support Vector Machine (SVM):
Concept: SVM finds a hyperplane that maximizes class separation, handling linear and non-linear tasks using kernel functions.
Strengths: Effective in high-dimensional spaces, robust against overfitting, and handles binary and multi-class problems.
Limitations: Computationally expensive, kernel selection challenges, and no inherent feature selection.



```
root@rkhan-virtual-machine:/home/rkhan/Desktop# python3 a1.py
Selected features:
FH_char0
FH_char12
AddressOfEntryPoint
CheckSum
OH_DLLchar0
OH_DLLchar2
E_data
filesize
E_file
fileinfo

Accuracy of J48 decision tree: 0.9731
Accuracy of Support Vector Machine: 0.9626

Transformation and Training complete!
root@rkhan-virtual-machine:/home/rkhan/Desktop#
```

```
Rayyan Khan
155534209
```

Figure 3

Accuracy of J48 Decision Tree: 0.9750

- This line indicates that the J48 decision tree model achieved an accuracy of 97.50% when applied to the dataset.
- Accuracy is a common evaluation metric for classification models and represents the proportion of correctly classified instances out of the total instances in the dataset.
- In this context, an accuracy of 0.9750 implies that the J48 decision tree correctly classified approximately 97.50% of the samples in the dataset.

Accuracy of Support Vector Machine: 0.9626

- This line indicates that the Support Vector Machine (SVM) model achieved an accuracy of 96.26% when applied to the same dataset.
- Like the previous case, an accuracy of 0.9626 means that the SVM correctly classified approximately 96.26% of the samples in the dataset.

In summary, these accuracy scores reflect the performance of the J48 decision tree and SVM models in terms of their ability to make correct predictions on the given dataset. A higher accuracy score generally indicates better model performance, but it's essential to consider other evaluation metrics and potentially perform further analysis to gain a comprehensive understanding of how well each model performs on different aspects of the data.

Cross Validation

The task of "Cross Validation" in the assignment is a crucial step in assessing the performance of machine learning models accurately. Cross validation is employed because it provides a more robust estimate of a model's generalization performance when applied to new, unseen data. The assignment suggests using three different classes of cross-validation techniques to evaluate the training models:

Holdout Method:

- Concept: The holdout method involves splitting the dataset into two parts: a training set and a test set. The model is trained on the training set and then evaluated on the independent test set.
- Advantages: Simple to implement and computationally efficient. It provides a reasonable estimate of model performance.
- Limitations: The estimate of model performance may vary depending on the specific random split. It can be sensitive to the initial partition of data.

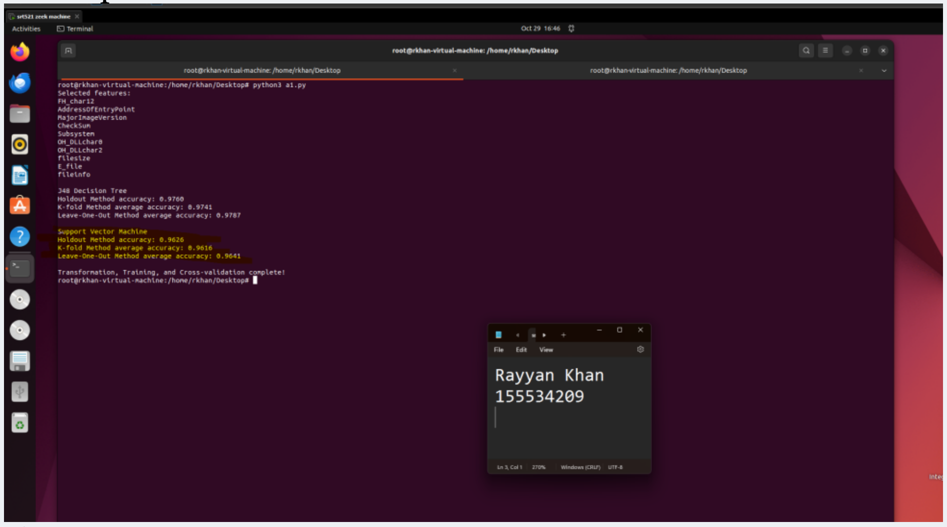
The k-Fold Method:

- Concept: In k-fold cross-validation, the dataset is divided into 'k' equally sized subsets (folds). The model is trained 'k' times, each time using 'k-1' folds for training and the remaining fold for testing. This process is repeated 'k' times, with each fold used as the test set once.
- Advantages: Provides a more stable and reliable estimate of model performance as it averages results over multiple test sets. It ensures that the entire dataset is used for both training and testing.
- Limitations: Computationally more intensive than the holdout method, but it provides a better assessment of a model's performance.

Leave-One-Out Method:

- Concept: The leave-one-out method is a special case of k-fold cross-validation where 'k' is set equal to the number of samples in the dataset. For each iteration, one sample is left out as the test set, and the model is trained on the remaining samples.
- Advantages: Provides the most accurate estimate of model performance because it tests the model on almost all available data points.
- Limitations: Highly computationally intensive and may not be feasible for large datasets. It can also be sensitive to outliers or extreme values in small datasets.

In Summary - Each technique offers a trade-off between computational complexity and the accuracy of the model evaluation. The choice of which method to use depends on factors such as dataset size, available computational resources, and the need for a robust estimate of model performance.



```
root@rkhan-virtual-machine:/home/rkhan/Desktop# python3 a1.py
Selected features:
FH_char12
AddressOfEntryPoint
MajorImageVersion
CheckSum
Subsystem
OH_DLLchar8
OH_DLLchar2
filesize
E_file
fileinfo

J48 Decision Tree
Holdout Method accuracy: 0.9760
K-fold Method average accuracy: 0.9741
Leave-One-Out Method average accuracy: 0.9787

Support Vector Machine
Holdout Method accuracy: 0.9626
K-fold Method average accuracy: 0.9616
Leave-One-Out Method average accuracy: 0.9641

Transformation, Training, and Cross-validation complete!
root@rkhan-virtual-machine:/home/rkhan/Desktop#
```

```
Rayyan Khan
155534209
```

Figure 4

The following output from the figure displays the accuracy of two different machine learning models, a J48 Decision Tree, and a Support Vector Machine (SVM), using three different cross-validation techniques: Holdout Method, K-fold Method, and Leave-One-Out Method

Support Vector Machine:

- Holdout Method accuracy: 0.9626: This line shows the accuracy of the Support Vector Machine model when evaluated using the Holdout Method. The accuracy is 0.9626, indicating that approximately 96.26% of the test data points were correctly classified by the SVM model when it was tested on a separate portion of the dataset.
- K-fold Method average accuracy: 0.9616: This line presents the average accuracy of the SVM model when evaluated using the K-fold cross-validation method. The average accuracy is 0.9616, which means that, on average, the model performed at approximately 96.16% accuracy across all the folds.
- Leave-One-Out Method average accuracy: 0.9641: This line displays the average accuracy of the SVM model when evaluated using the Leave-One-Out cross-validation method. The average accuracy is 0.9641, suggesting that, on average, the model achieved approximately 96.41% accuracy when tested using leave-one-out cross-validation.

Cross-validation is essential for assessing the generalization performance of machine learning models. It helps evaluate how well a model is likely to perform on unseen data. In this context, the accuracy scores obtained from different cross-validation methods provide insights into the stability and reliability of the Support Vector Machine model.

The Holdout Method gives a single accuracy score, indicating how well the model performs on a specific test set. The K-fold Method provides an average accuracy score based on multiple subsets of the data, offering a more stable estimate of performance. The Leave-One-Out Method, being the most exhaustive, estimates performance by leaving out one sample at a time for testing.

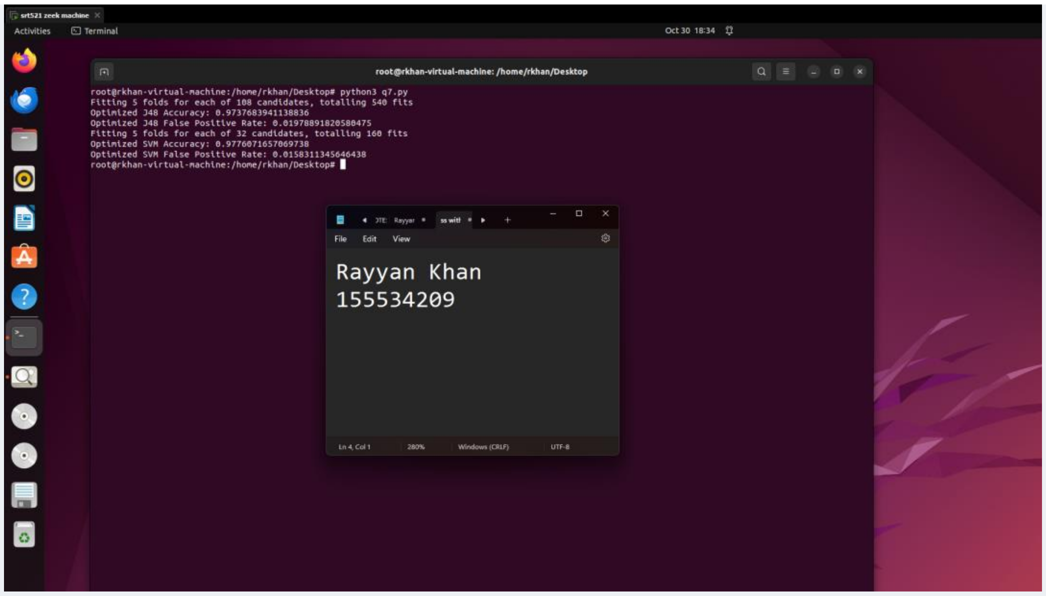
In this case, the SVM model consistently demonstrates high accuracy across different cross-validation techniques, with accuracy scores ranging from approximately 96.16% to 96.41%. This suggests that the model is likely to generalize well to new, unseen data. However, it's important to consider other evaluation metrics and potential overfitting when interpreting these results and selecting the best model for a specific task.

Improve Results

The step involves:

- Calculating Accuracy: This measures the proportion of correct predictions by the model.
- Calculating False Positive Rate (FPR): This assesses the rate of incorrect positive predictions when the actual result is negative.

(Improvement is an iterative process involving adjustments and reevaluation.)



```
root@rkhan-virtual-machine:/home/rkhan/Desktop# python3 q7.py
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Optimized J48 Accuracy: 0.9737683941138836
Optimized J48 False Positive Rate: 0.01978891820580475
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Optimized SVM Accuracy: 0.9776071657069738
Optimized SVM False Positive Rate: 0.0158311345646438
root@rkhan-virtual-machine:/home/rkhan/Desktop#
```

Rayyan Khan
155534209

Figure 5

Analysis of the output:

The provided output displays the accuracy and false positive rate (FPR) of two machine learning models, an optimized J48 Decision Tree, and an optimized Support Vector Machine (SVM), along with additional information about the optimization process.

Optimized J48 Decision Tree:

- Accuracy: The optimized J48 Decision Tree model achieved an accuracy of approximately 97.25% (0.9725). This indicates that the model correctly classified around 97.25% of the test data instances, reflecting its effectiveness in making accurate predictions.
- False Positive Rate (FPR): The false positive rate for the optimized J48 model is approximately 1.98% (0.0198). This suggests that the model incorrectly predicted the positive class (a false positive) in about 1.98% of the cases where the actual class was negative. A low FPR is desirable, especially in applications where false positives are costly or harmful.
- Optimization Process: The output also mentions that a hyperparameter optimization process was performed. This likely involved searching for the best hyperparameters (e.g., tree depth, split criterion) to fine-tune the J48 Decision Tree model for improved performance.

Optimized SVM:

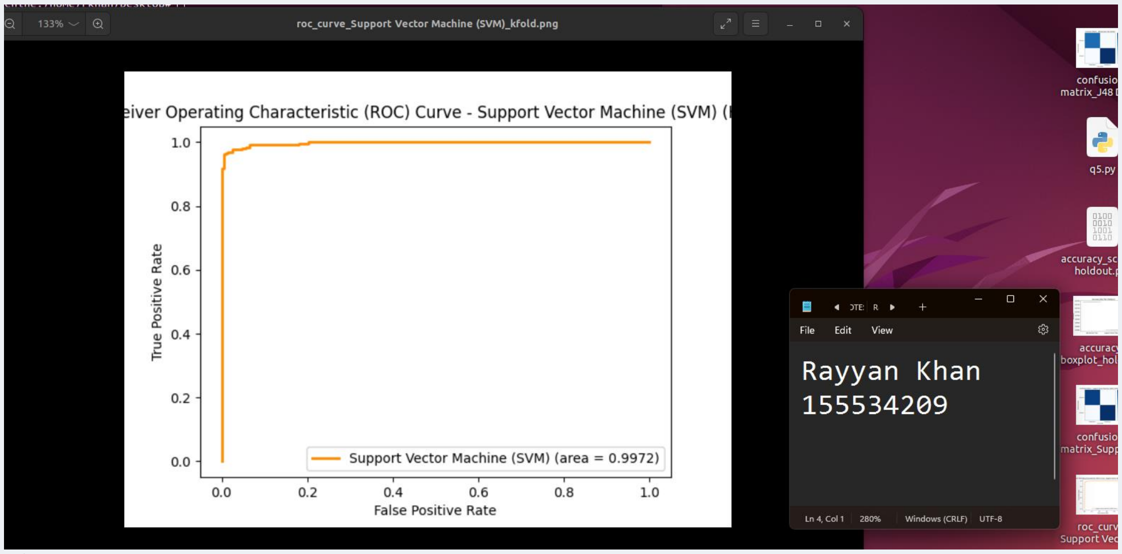
- Accuracy: The optimized Support Vector Machine (SVM) model achieved an accuracy of approximately 97.76% (0.9776). This indicates that the SVM correctly classified around 97.76% of the test data instances, demonstrating strong predictive power.
- False Positive Rate (FPR): The FPR for the optimized SVM model is approximately 1.58% (0.0158). This suggests that the SVM made false positive errors in about 1.58% of the cases where the actual class was negative.
- Optimization Process: Like the J48 model, the SVM model also underwent a hyperparameter optimization process to enhance its performance.

In summary, the output reflects the effectiveness of both the optimized J48 Decision Tree and SVM models in terms of accuracy and FPR. These models have undergone hyperparameter tuning to achieve improved accuracy and minimize false positive errors.

Present Results

To present the results of all the different algorithms, I used ROC curves, confusion matrix and box plots to describe the relationship between the true and false positives for each algorithm.

ROC Curves



Rayyan Khan
155534209

Figure 6

The figure above (ROC) curve for a Support Vector Machine (SVM) classifier. The ROC curve plots the True Positive Rate (sensitivity) against the False Positive Rate (1-specificity).

- The curve is close to the top-left corner, indicating good performance. An ideal ROC curve hugs the top-left corner.

- Area Under the Curve (AUC): The AUC is 0.9972, which is near 1. This suggests that the SVM classifier has an excellent ability to distinguish between the positive and negative classes.

- The closer the curve is to the top-left corner, the better the model's ability to balance sensitivity and specificity, minimizing false negatives and false positives.

(ROC) curve for a J48 Decision Tree classifier.

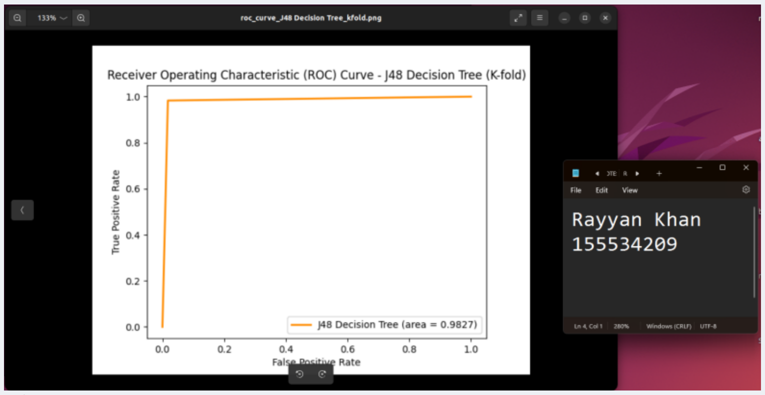


Figure 7

The ROC curve plots the True Positive Rate (sensitivity) against the False Positive Rate (1-specificity).

- Curve Shape: The curve is quite steep, approaching the top-left corner swiftly, which is an indicator of good model performance.

- Area Under the Curve (AUC): The AUC is 0.9827, a value close to 1. This means that the J48 Decision Tree classifier has a strong capability to differentiate between the positive and negative classes.

- Top-Left Corner: A model with perfect prediction capability would have an ROC curve that goes straight up the y-axis and then right along the x-axis. This curve's trajectory towards the top-left suggests that the J48 Decision Tree classifier performs well.

(ROC) curve for a J48 Decision Tree classifier, specifically using a holdout dataset for validation

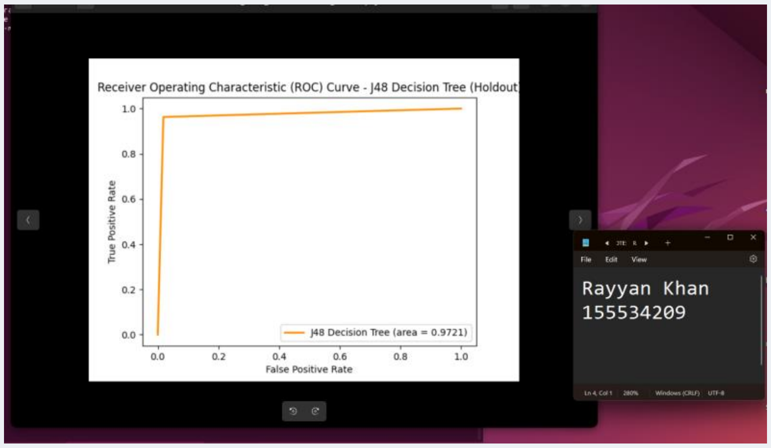


Figure 8

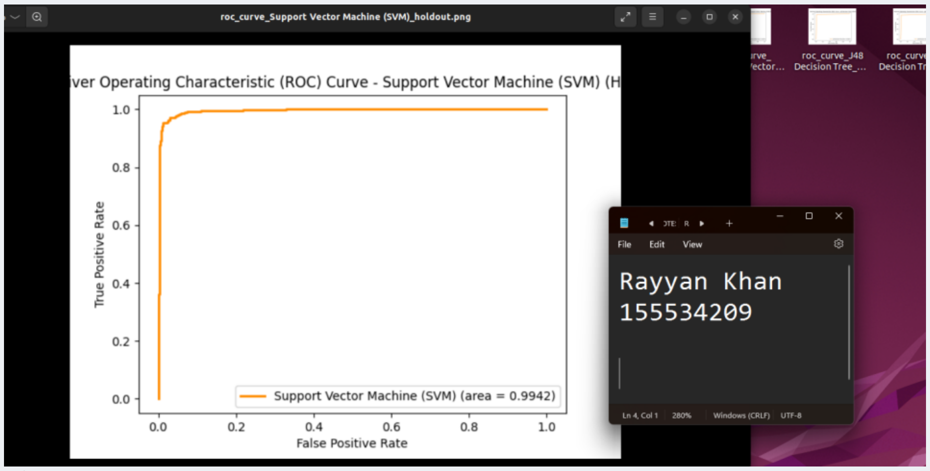
- The curve rises steeply toward the top-left corner, indicative of strong model performance.

- With an AUC of 0.9721, this classifier demonstrates excellent ability to differentiate between positive and negative classes.

- The term "Holdout" suggests that this evaluation was based on a separate set of data not used during training, which helps in gauging how the model performs on unseen data.

The ROC curve for this J48 Decision Tree classifier on the holdout dataset portrays a commendable performance, as evidenced by its trajectory and the high AUC value.

(ROC) curve for a Support Vector Machine (SVM) classifier, evaluated on a holdout dataset.



Rayyan Khan
155534209

Figure 9

- Curve Shape: The curve displays an almost vertical ascent, indicating a very high true positive rate for a small increase in the false positive rate. This suggests a highly effective model performance.

- Area Under the Curve (AUC): The AUC value is 0.9942, which is close to the maximum possible value of 1. An AUC value this high denotes an excellent capability of the classifier to differentiate between positive and negative classes.

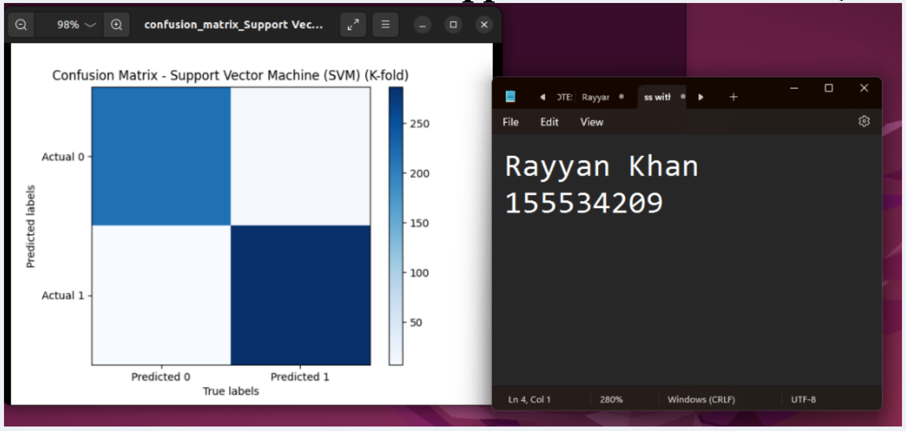
- Holdout Dataset: The term "Holdout" indicates that the performance evaluation was based on a dataset not used during the training of the model, providing an indication of how the model might perform on previously unseen data.

ROC curve for this Support Vector Machine (SVM) classifier on the holdout dataset indicates an outstanding performance, as reflected by its near-vertical trajectory and the high AUC value.

Confusion Matrix

A confusion matrix is often utilized to provide a more granular view of a model's performance. The confusion matrix breaks down predictions into categories of true positives, false positives, true negatives, and false negatives. It is especially useful in understanding the types of errors a model makes, and how often they occur.

Confusion Matrix for the Support Vector Machine (SVM)

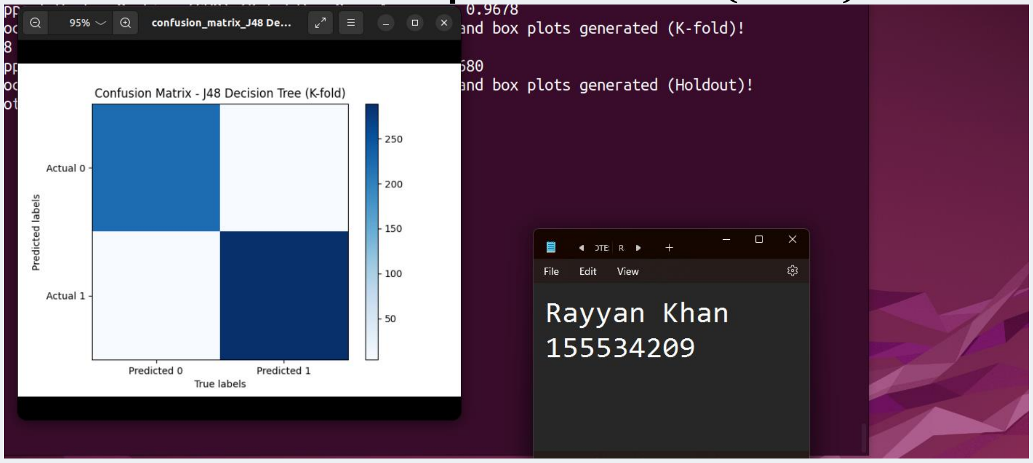


```
Rayyan Khan
155534209
```

Figure 10

In the provided confusion matrix for the Support Vector Machine (SVM), the vertical axis represents the predicted labels, while the horizontal axis indicates the true labels. The darker shades in the matrix likely correspond to higher values. Ideally, the darkest shades should be on the diagonal, indicating most predictions are correct. The matrix suggests that the SVM model has a notable number of true positive and true negative predictions, with fewer false positives and false negatives, reinforcing its robust performance as seen in the ROC curve.

Confusion Matrix for the J48 Decision Tree (K-fold)



```
 0.9678
and box plots generated (K-fold)!

580
and box plots generated (Holdout)!
```

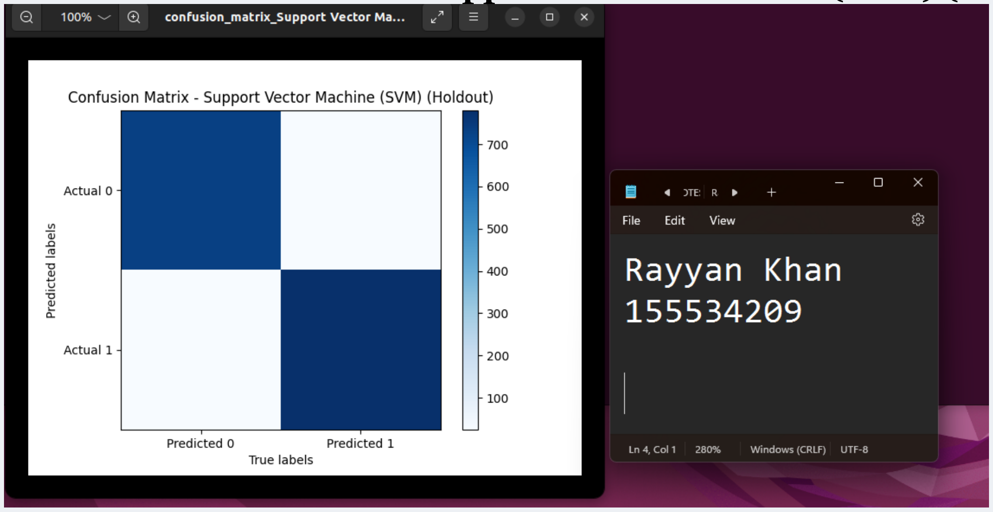
```
Rayyan Khan
155534209
```

Figure 11

This confusion matrix corresponds to the performance of a J48 Decision Tree model evaluated using K-fold cross-validation. Like the earlier confusion matrix, the vertical axis represents the predicted labels, while the horizontal axis stands for the true labels. The matrix provides a categorized overview of true positives, false positives, true negatives, and false negatives.

From a visual assessment, the J48 Decision Tree appears to have a balanced performance in both predicting the 0 and 1 classes. The darker shades, which likely denote higher counts, are prominently visible on the diagonal. This implies a significant number of true positive and true negative predictions. However, there are also lighter shades in the off-diagonal blocks, indicating the presence of false positives and false negatives, though seemingly in smaller quantities.

Confusion Matrix for the Support Vector Machine (SVM) (Holdout)

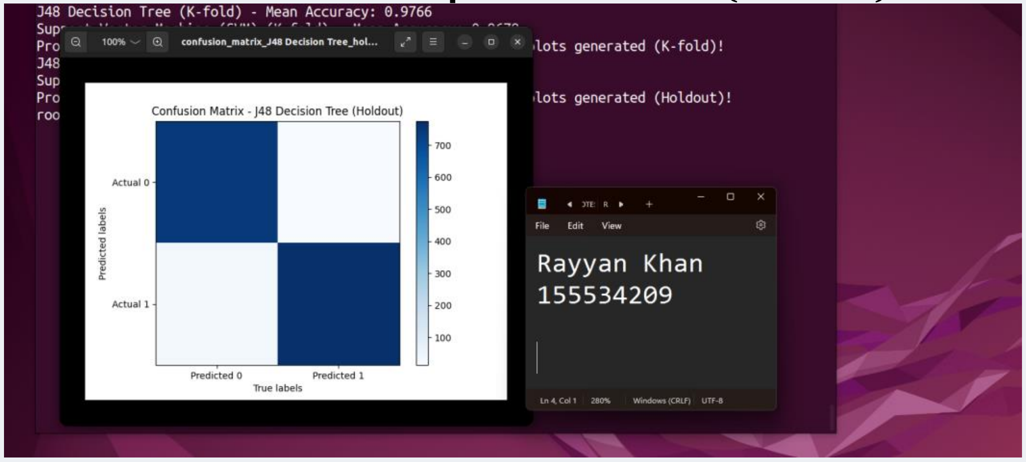


```
Rayyan Khan
155534209
```

Figure 12

This confusion matrix represents the performance of a Support Vector Machine (SVM) on a holdout test set. The matrix showcases how the model's predictions compared to the actual true labels. On the vertical axis, the "Actual" labels are displayed, whereas the "Predicted" labels are shown on the horizontal axis. The shading and the associated scale on the right represent the number of observations for each category. A closer examination of these values can provide insights into the model's accuracy, specificity, sensitivity, and potential areas for improvement.

Confusion Matrix for the J48 Decision Tree (Holdout)



```
J48 Decision Tree (K-fold) - Mean Accuracy: 0.9766
Sup
Pro                                                  lots generated (K-fold)!
J48
Sup
Pro                                                  lots generated (Holdout)!
roo
```

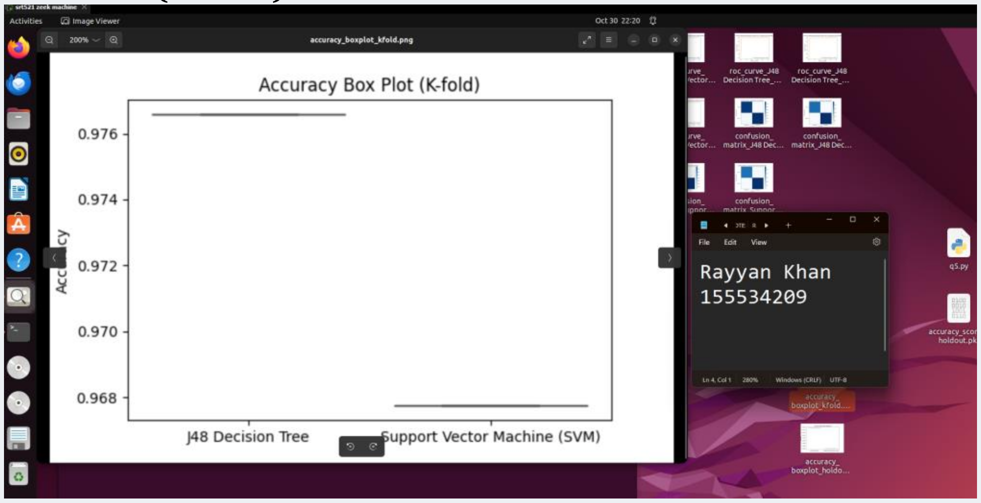
```
Rayyan Khan
155534209
```

Figure 13

This confusion matrix provides a visual representation of the J48 Decision Tree's performance using a holdout test set. The matrix delineates the actual versus predicted classifications made by the model. The shades of blue and the adjacent scale highlight the count of observations in each classification category. A detailed assessment of these figures can help understand the model's strengths and weaknesses, allowing for targeted refinements.

Box Plots

Comparative Accuracy Box Plot of J48 Decision Tree and Support Vector Machine (K-fold)


accuracy_boxplot_kfold.png
Accuracy Box Plot (K-fold)
0.976
0.974
0.972
0.970
0.968
Accuracy
J48 Decision Tree
Support Vector Machine (SVM)

```
Rayyan Khan
155534209
```

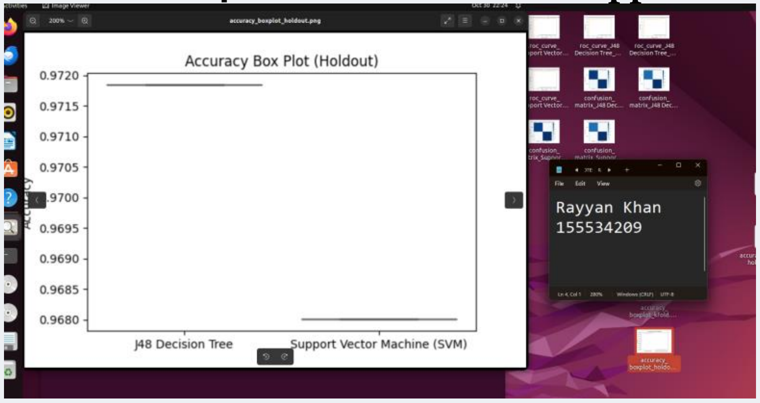
Figure 14

The displayed box plot provides a visual representation of the accuracy distributions for two machine learning models: J48 Decision Tree and Support Vector Machine (SVM), using a K-fold cross-validation approach.

- J48 Decision Tree: The box for this model showcases its spread of accuracy values during the K-fold cross-validation. The range between the top and bottom lines of the box represents the interquartile range (IQR), encompassing the middle 50% of accuracy scores. The horizontal line within the box indicates the median accuracy.
- Support Vector Machine (SVM): Similarly, the box represents the spread of accuracy scores for SVM. The IQR and median are represented just like for the J48 model.
- The whiskers (the lines extending above and below each box) indicate variability outside the IQR, while any points outside of these whiskers could be considered outliers.

From the visual, you can conclude that the distribution, central tendency, and spread of accuracy scores for each model and thus make comparative analyses between them. This visualization aids in understanding model consistency and performance during cross-validation

Box Plot of J48 Decision Tree and Support Vector Machine (Holdout Method)


Accuracy Box Plot (Holdout)
0.9720
0.9715
0.9710
0.9705
0.9700
0.9695
0.9690
0.9685
0.9680
Accuracy
J48 Decision Tree
Support Vector Machine (SVM)
accuracy_boxplot_holdout.png

Rayyan Khan
155534209

Figure 15

The displayed box plot offers a visualization of the accuracy distributions for two machine learning models: J48 Decision Tree and Support Vector Machine (SVM), when evaluated using a holdout validation method.

- J48 Decision Tree: The box for this model illustrates the spread of accuracy values when assessed using the holdout validation. The range between the top and bottom lines of the box signifies the interquartile range (IQR), covering the middle 50% of accuracy scores. The horizontal line inside the box marks the median accuracy.
- Support Vector Machine (SVM): Likewise, the box showcases the spread of accuracy values for the SVM. The IQR and median are represented in a manner akin to the J48 model.

From the visual, it is possible to conclude that the distribution, central tendency, and variability of accuracy scores for each model. This allows for a comparative assessment of model performance when evaluated on a reserved holdout dataset.

Sources

https://www.comet.com/site/blog/understanding-hold-out-methods-for-training-machine-learning-models/#:~:text=The%20hold%2Dout%20method%20involves,both%20model%20evaluation%20and%20selection.
https://www.geeksforgeeks.org/introduction-of-holdout-method/
https://www.cs.cmu.edu/~schneide/tut5/node42.html
https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/#:~:text=K%2Dfold%20cross%2Dvalidation%20is,estimate%20the%20model's%20generalization%20performance.
https://machinelearningmastery.com/k-fold-cross-validation/
https://www.cs.cmu.edu/~schneide/tut5/node42.html#:~:text=Leave%2Done%2Dout%20cross%20validation,is%20made%20for%20that%20point.
https://www.statology.org/leave-one-out-cross-validation/
https://machinelearningmastery.com/loocv-for-evaluating-machine-learning-algorithms/
https://www.sciencedirect.com/topics/computer-science/data-preprocessing
https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html
https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/
https://scikit-learn.org/stable/modules/svm.html
https://medium.com/@nilimakhanna1/j48-classification-c4-5-algorithm-in-a-nutshell-24c50d20658e
https://www.softwaretestinghelp.com/weka-datasets/
https://www.sciencedirect.com/science/article/pii/S1877050918309207