National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cyber Security

Abdur Razzaq Shaik
Student ID: X22178333

School of Computing
National College of Ireland

Supervisor: Arghir Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Abdur Razzaq Shaik
……

Student ID: ……22178333
……..……

Programme: ……MSc in Cyber Security…… **Year:** ……2024……..

Module: ……MSc Research Project……
……..……

Lecturer: ……Arghir Nicolae Moldovan……
……..……

Submission Due Date: ……06-03-2024……..
……..……

Project Title: ……Enhancing Efficiency of Machine Learning Techniques with Feature Selection and Hyper Parameter Tuning for Intrusion and Detection Towards Leveraging Cyber Security
……..……

Word Count: ……2120…… **Page Count ……** 11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Abdur Razzaq Shaik
……

Date: ……05-03-2024
……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Abdur Razzaq Shaik
Student ID: X22178333

1 Introduction

1.1. Overview

Cybersecurity enhancement is a continuous process due to increasing cyber-attacks of late. Traditional security mechanisms were based on certain heuristics that could detect intrusions based on their detection process. However, with the emergence of Artificial Intelligence (AI) methods such as machine learning (ML) approaches, learning based models are found efficient due to their ability to learn from labelled data continuously. It is found in the literature that ML models based on supervised learning show deteriorated intrusion detection performance when training samples are not with designed quantity and quality. The system is evaluated using CICIDS2017 dataset. Intrusion detection system is implemented using binomial classification and also multi-class classification

1.2 System Environment

Hardware:
Base Memory:4608 MB
Processor: 4
Storage: 25 GB of free disk space
Network: Intel Pro/1000 MT Desktop (Nat Network, 'NatNetwork')

Software Dependencies:
Anaconda, python

Software Dependencies:
Below are the software dependencies required to run the code in this notebook:

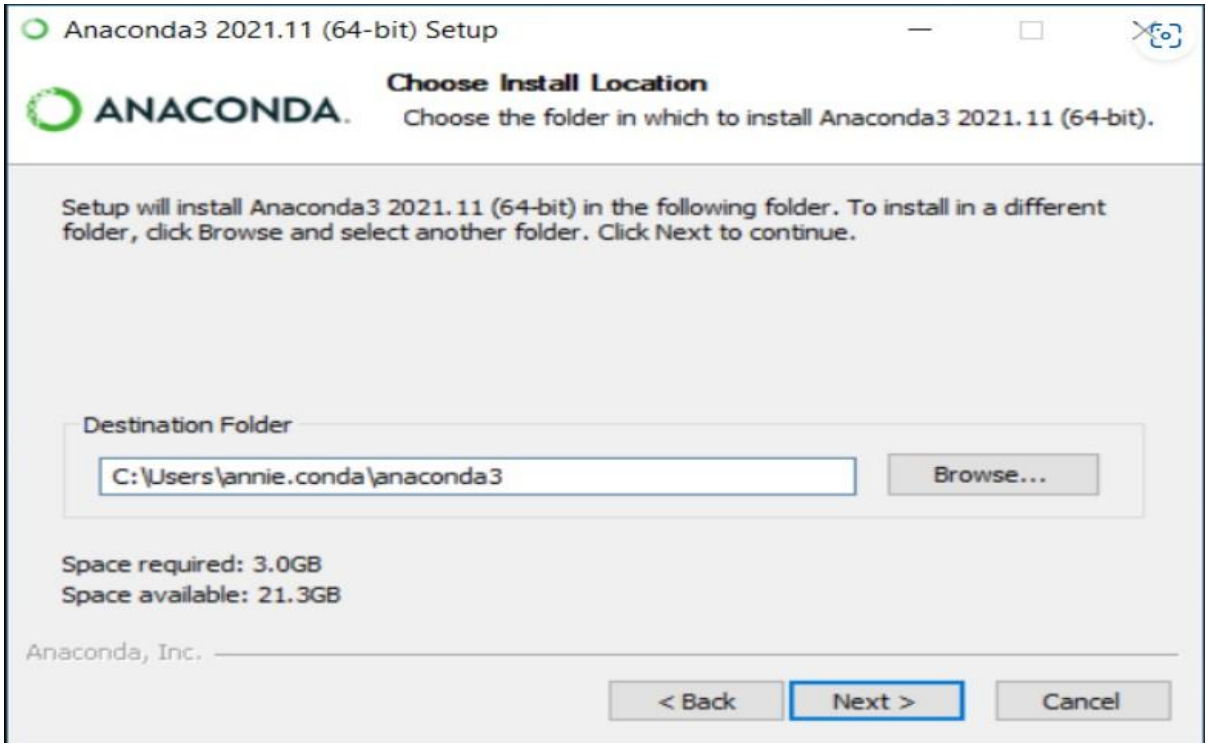
- **NumPy**: A fundamental package for scientific computing with Python.
- **Pandas**: A powerful data analysis and manipulation library.
- **Matplotlib**: A plotting library for creating static, interactive, and animated visualizations.
- **Seaborn**: A statistical data visualization library based on Matplotlib.
- **Scikit-learn**: A machine learning library that provides simple and efficient tools for data mining and data analysis.
- **XGBoost**: An optimized gradient boosting library designed for speed and performance.
- To Quickly train large datasets.
- **Decision Trees**: Basic algorithm for classification and regression tasks
- **Random Forests**: Ensemble of Decision trees using bagging

- Reduces overfitting compared to a single decision tree
- **Extra Trees:** like random forests but with more randomness in splits.
- To improve predictive accuracy and control overfitting
- Faster to train and can be more robust.

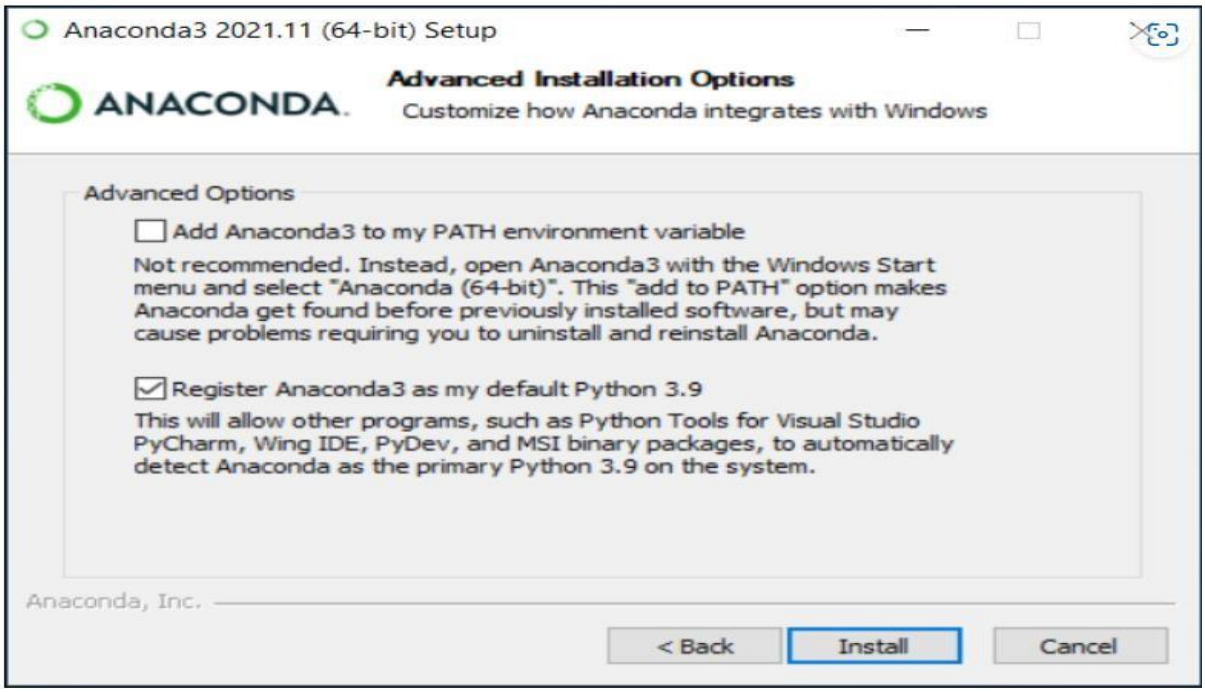
Windows 11 :
Hardware:
Base Memory:3658 MB
Processor: 4
Storage: 30 GB of free disk space
Network: Intel Pro/1000 MT Desktop (Nat Network, 'NatNetwork')

2 Installation

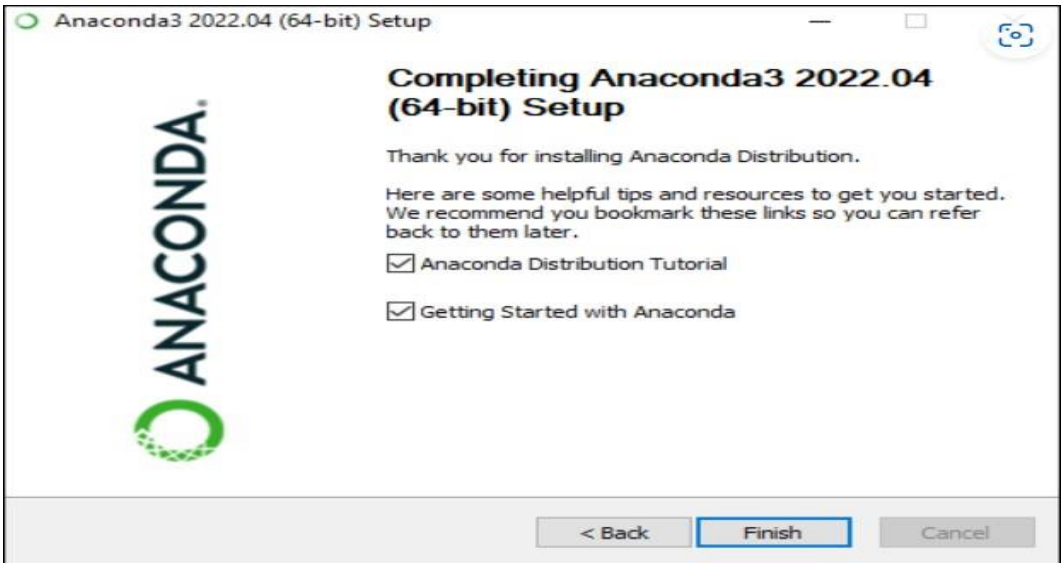
1. Download the Anaconda installer.
2. Go to your Downloads folder and double-click the installer to launch. To prevent permission errors, do not launch the installer from the Favorites folder.
3. Review the license agreement and click **I Agree** option
4. It is recommended that you install for **Just Me**, option which will install Anaconda Distribution for the current user account. Select the **AllUsers** option if you need to install for all users' accounts on the computer (which requires Windows Administrator privileges).
5. Choose a destination folder to install Anaconda and click **Next**. Install Anaconda to a directory path that does not contain spaces or Unicode characters.



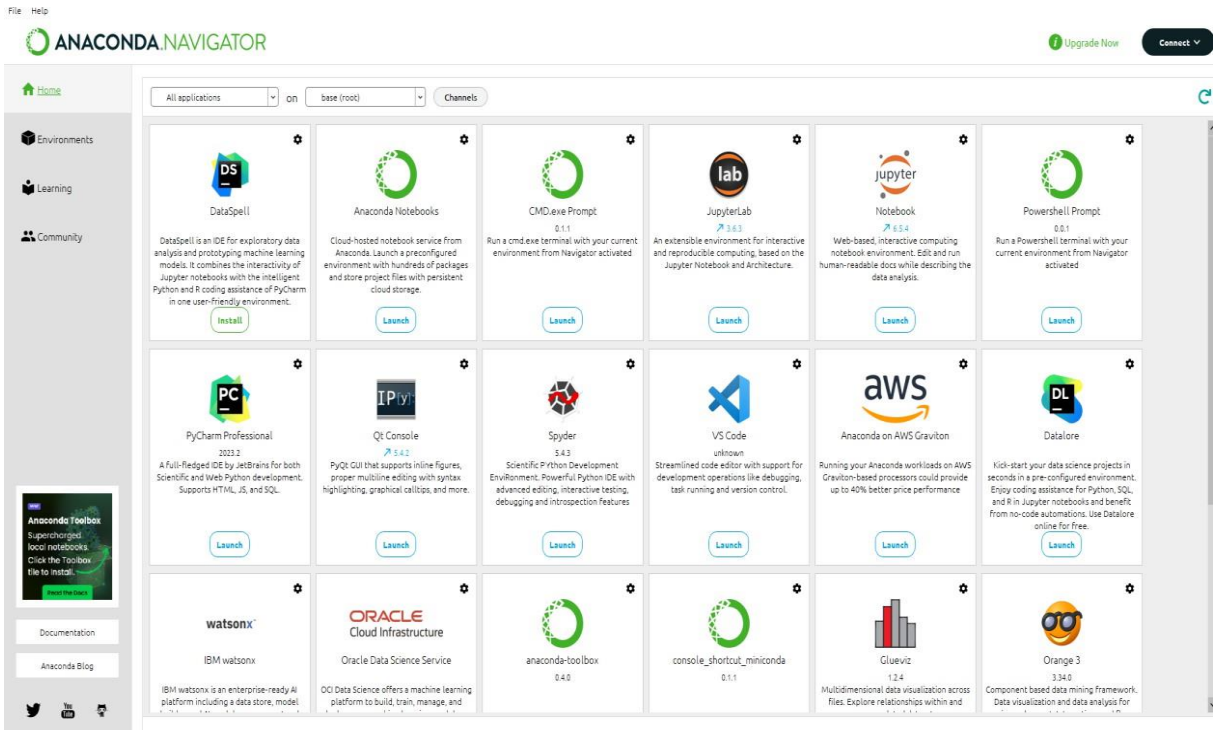
6. Choose whether to add Anaconda to your PATH environment variable or register Anaconda as your default Python. We **don't recommend** adding Anaconda to your PATH environment variable, since this can interfere with other software. Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, it is advisable to accept the default setting and leave this box checked



7. Click **Install**. If you want to monitor the packages Anaconda is installing, you can click on Show Details.
8. Once the installation is completed you will see the "Thanks for installing Anaconda" dialog box

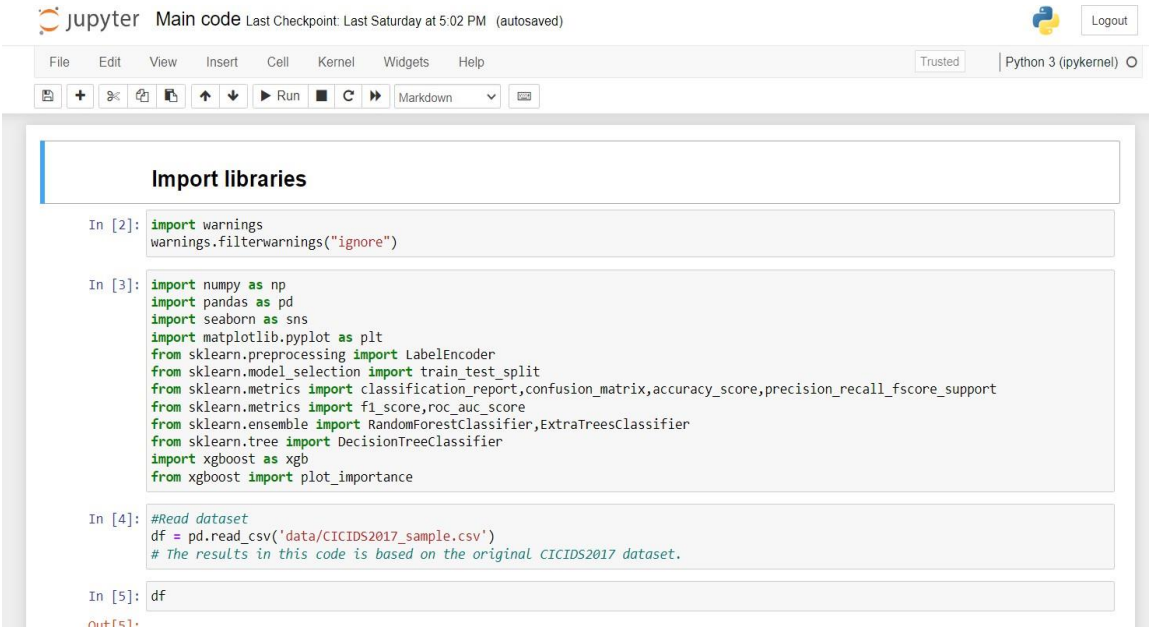


3 Implementation

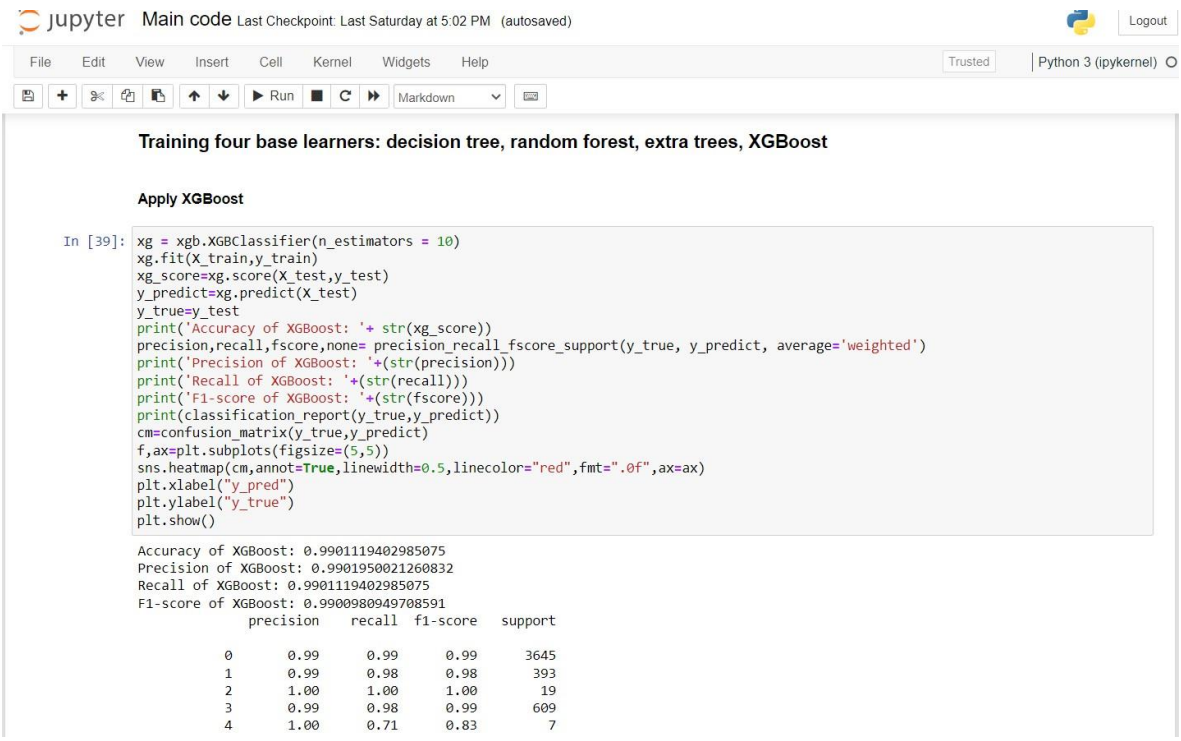


On Windows, you can launch Jupyter via the shortcut added by the Anaconda to your start menu, which will open a new tab in your default web browser that should look something like the following screenshot

In this notebook, we'll explore a classification task using various machine learning algorithms. We'll utilize popular libraries such as NumPy, Pandas, Seaborn, Matplotlib, Scikit-learn, and XGBoost.



In this section, I have downloaded and imported the essential libraries required for our classification task.
Using Pip install.
These libraries provide powerful tools for data analysis, visualization, preprocessing, modeling, and evaluation. We'll utilize them throughout the notebook to perform various tasks and analyze the results.



In this notebook, we'll explore the application of various machine learning models for a classification task. We'll utilize popular algorithms, including Random Forest, Extra Trees, Decision Tree, and XGBoost, each renowned for its unique characteristics and efficacy.

1. Random Forest Classifier

Random Forest is a powerful ensemble learning method that constructs multiple decision trees during training. It leverages the collective wisdom of these trees by outputting the mode of the classes for classification tasks or the mean prediction for regression tasks. Renowned for its robustness, scalability, and consistently high accuracy, Random Forest is a popular choice for a wide range of machine learning tasks.

2. Extra Trees Classifier

Extra Trees, also known as Extremely Randomized Trees, represents another ensemble learning approach. Similar to Random Forest, it builds a forest of uncorrelated decision trees. However, unlike Random Forest, Extra Trees selects feature thresholds randomly, making it faster at the expense of potentially lower accuracy. Extra Trees is particularly useful for reducing overfitting and variance in models.

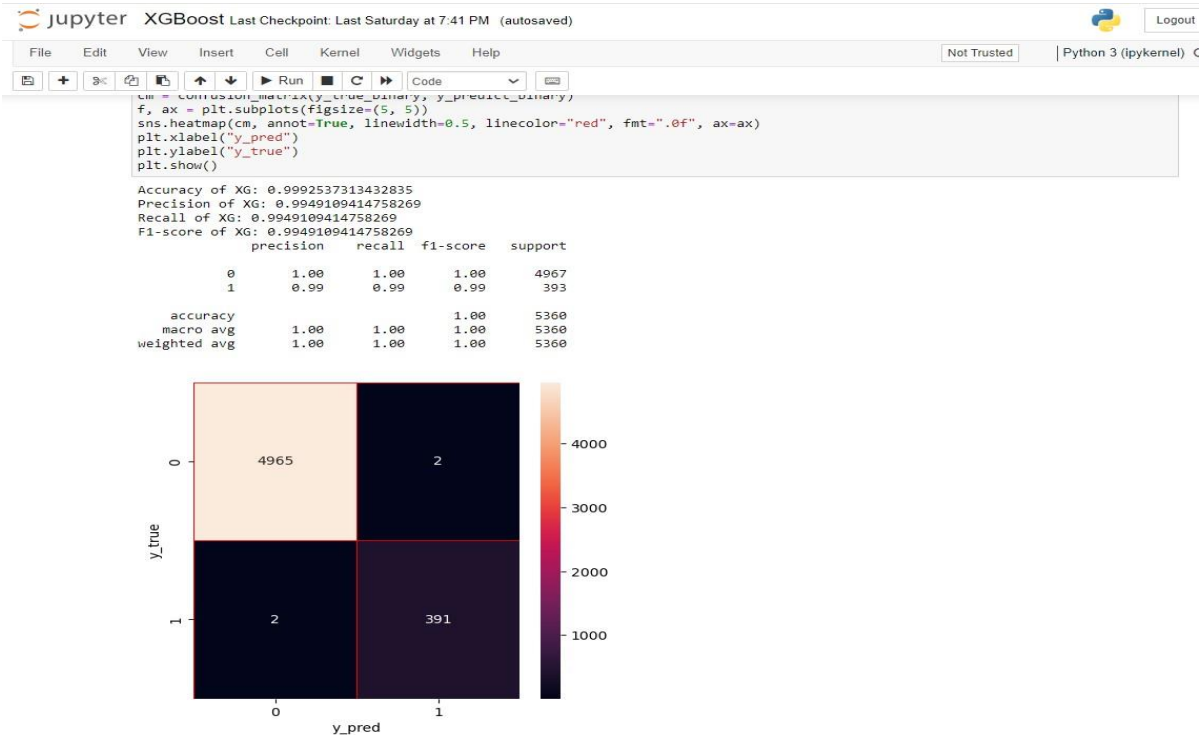
3. Decision Tree Classifier

Decision Tree is a versatile and intuitive non-parametric supervised learning algorithm used for both classification and regression tasks. It partitions the feature space into regions and predicts the target variable based on the majority class or mean value of instances within each region. Despite its simplicity, Decision Tree can be highly effective, especially for capturing complex relationships in the data.

4. XGBoost Classifier

XGBoost, short for Extreme Gradient Boosting, stands out as a cutting-edge gradient boosting library designed for exceptional speed and performance. Operating within the gradient boosting framework, XGBoost sequentially builds an ensemble of weak learners, typically decision trees, and combines their predictions to enhance overall performance. Recognized for its high accuracy, efficiency, and adaptability, XGBoost is a favored choice in various machine learning competitions and real-world applications.

Binomial class classification



jupyter RandomForest Last Checkpoint: Last Saturday at 5:12 PM (autosaved)

Logout

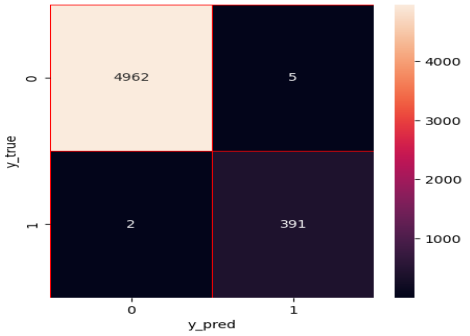
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

```
plt.xlabel("y_pred")
plt.ylabel("y_true")
plt.show()
```

```
Accuracy of RF: 0.9986940298507463
Precision of RF: 0.9873737373737373
Recall of RF: 0.9949109414758269
F1-score of RF: 0.991128010139417
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4967
           1       0.99      0.99      0.99       393

    accuracy                           1.00      5360
   macro avg       0.99      1.00      1.00      5360
weighted avg       1.00      1.00      1.00      5360
```



In [ ]:

jupyter ExtraTrees Last Checkpoint: Last Saturday at 5:50 PM (autosaved)

Logout

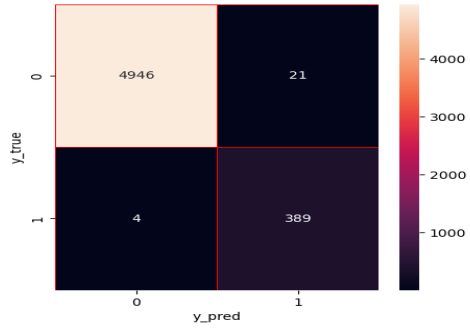
File Edit View Insert Cell Kernel Widgets Help

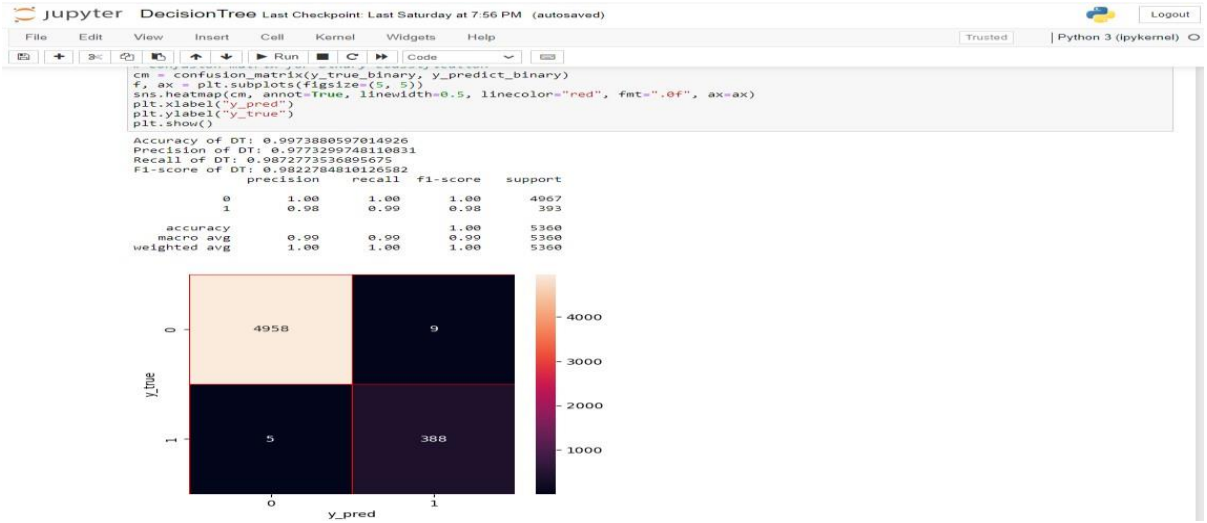
Not Trusted | Python 3 (ipykernel)

Run Code

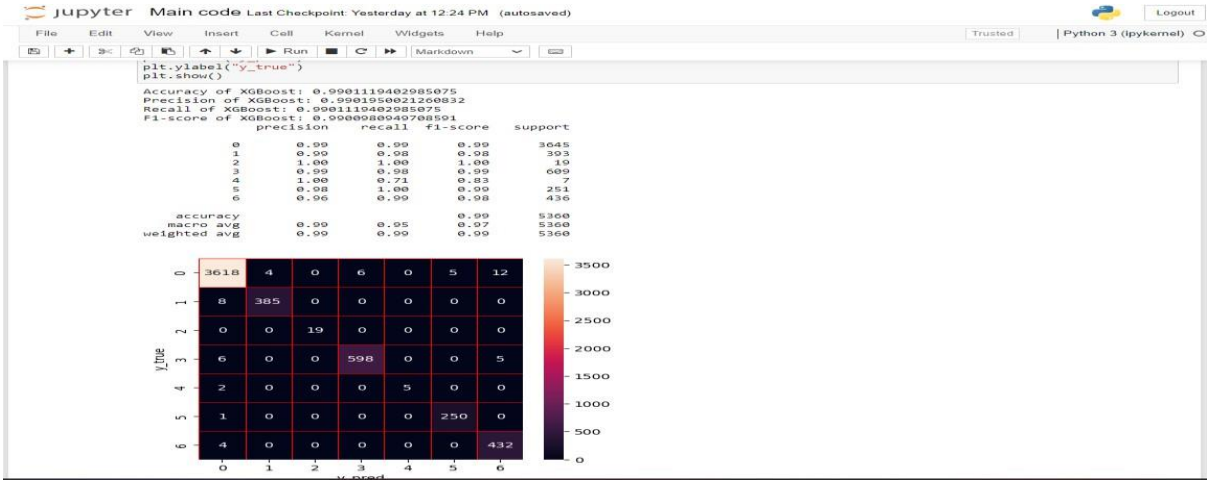
```
plt.ylabel("y_true")
plt.show()
```

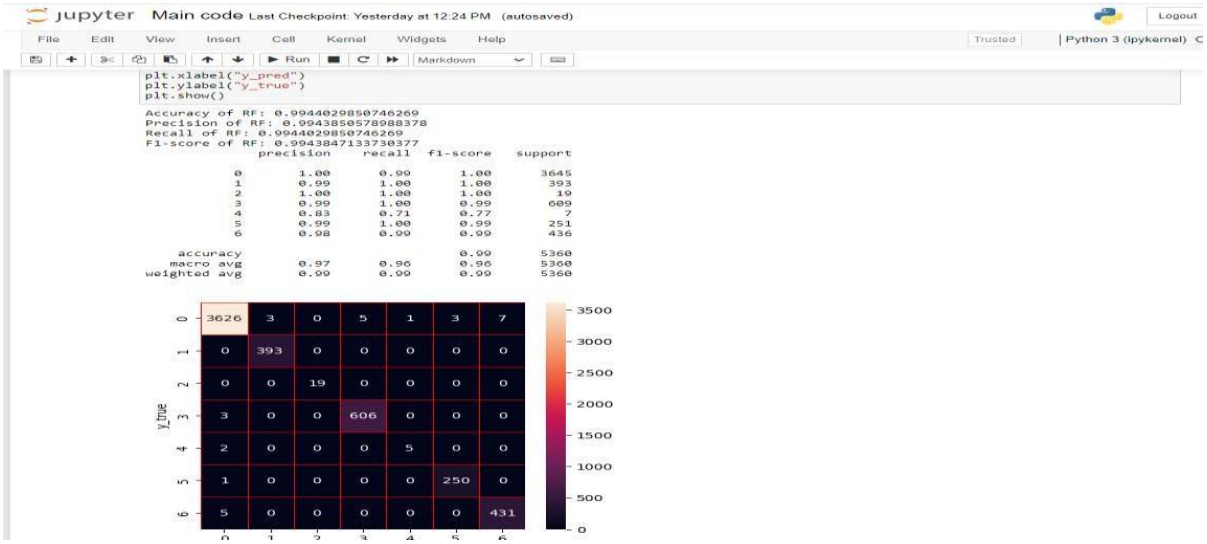
```
Accuracy of ET: 0.9953358208955224
Precision of ET: 0.948780487804878
Recall of ET: 0.989821882951654
F1-score of ET: 0.9688667496886675
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4967
           1       0.95      0.99      0.97       393

    accuracy                           1.00      5360
   macro avg       0.97      0.99      0.98      5360
weighted avg       1.00      1.00      1.00      5360
```

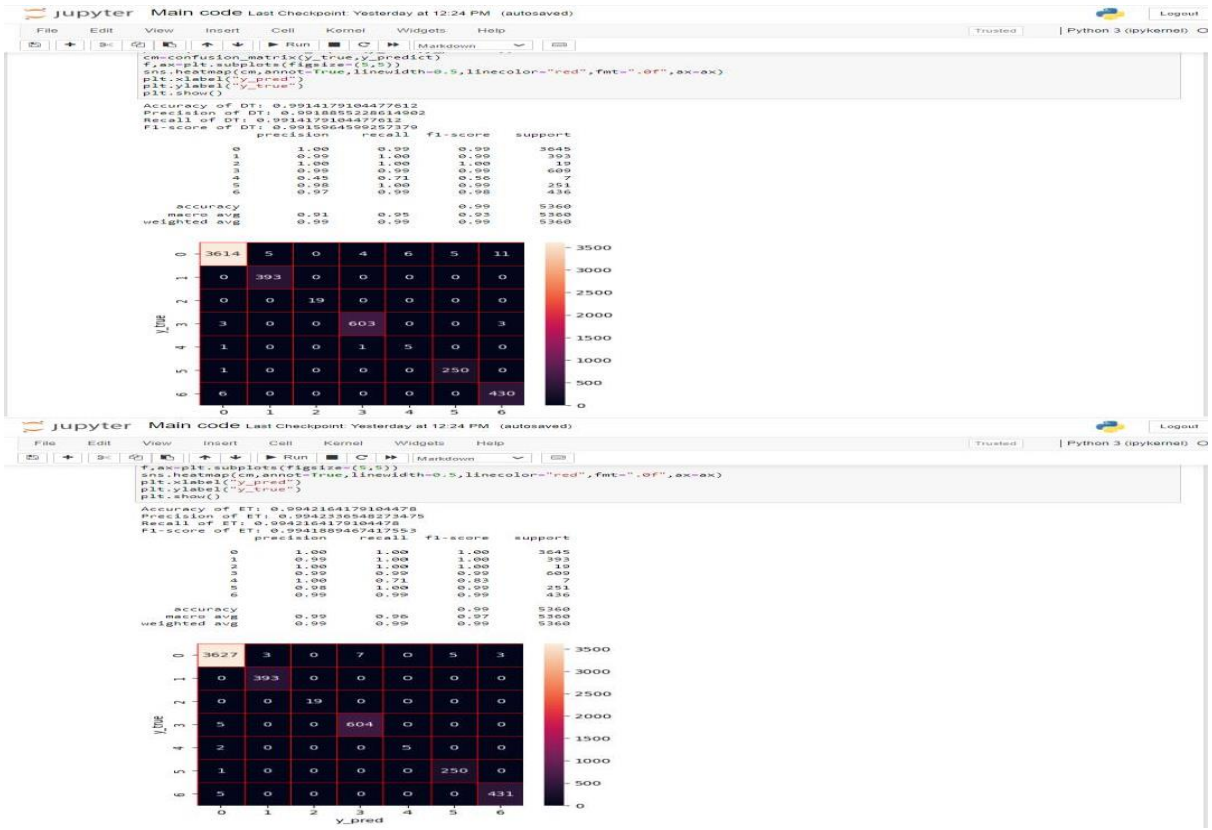




Multi class classification







4. Conclusion

This research is aimed at designing and implementing an intrusion detection system based on ML models. The system is implemented using four ML models with two optimizations such as feature selection and hyperparameter optimization. These research objectives are met as evident in the results of the research. The first objective is to investigate on the existing ML methods and related works used to realize intrusion detection systems. It is achieved and outcome is presented in related works. The second objective is to propose a ML based framework with its mechanisms and optimizations for efficient intrusion detection. Finally, conclusions are drawn as found in this section and also future work possibilities are provided. With regard to research questions, the first research question is "Can machine learning models be used for realizing an intrusion detection system?". This research question is found affirmative answer as the ML models were found suitable for intrusion detection. The second research question is "Can optimizations like feature selection and hyperparameter optimization have impact on intrusion detection performance of ML models?". This research question also found affirmative answer in this research and in literature also. However, in this research the improvement of accuracy when optimizations are made to ML models is relatively less. In other words, there is no significant improvement in the accuracy when optimizations are applied.

References

Konstantin Grotov., Sergey Titov., Vladimir Sotnikov.,Yaroslav Golubev., Timofey. (2022). A Large-Scale Comparison of Python Code in Jupyter Notebooks and Scripts. *Washington, DC*, pp.01-12. https://arxiv.org/pdf/2203.16718.pdf

Jiawei Wang.,Li Li.,Andreas Zeller. (2020). Restoring Execution Environments of Jupyter Notebooks. *Proceedings of the ACM/IEEE 42nd international.*, pp.1-12. https://arxiv.org/ftp/arxiv/papers/2103/2103.02959.pdf

Konstantin Grotov., Sergey Titov., Vladimir Sotnikov. (2019). A Large-Scale Comparison of Python Code in Jupyter Notebooks and Scripts. *IEEE*, pp.353-364. https://doi.org/10.1145/3524842.3528447

Brian E. Granger., Fernando Perez. (2021). Jupyter: Thinking and Storytelling With Code and Data. IEEE Computer Society., pp.7-14. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9387490

N Braun., T Hauth., C Pulvermacher., M Ritter. (2017). An Interactive and Comprehensive Working Environment for High-Energy Physics Software with Python and Jupyter Notebooks. IOP Conf. Series: Journal of Physics: Conf. Series. 898, pp.1-8. https://iopscience.iop.org/article/10.1088/1742-6596/898/7/072020/pdf

Sheeba Samuel.,Birgitta König-Ries. (2019). ReproduceMeGit: A Visualization Tool for Analyzing Reproducibility of Jupyter Notebooks. *Springer*.pp.1-2. https://arxiv.org/pdf/2006.12110.pdf

Mohammad Gharehyazie, Baishakhi Ray, Mehdi Keshani, Masoumeh Soleimani Zavosht, Abbas Heydarnoori, and Vladimir Filkov. “Cross-project code clones in GitHub”. In: Empirical Software Engineering 24.3 (2019), pages 1538–1573. doi: 10.1007/s10664-018-9648-z

Mary B. Kery, Marissa Radensky, Mahima Arya, Bonnie E. John, and Brad A. Myers. “The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool”. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI ’18. ACM Press, 2018, pages 1–11. doi: 10.1145/3173574.3173748.

Andreas Koenzen, Neil Ernst, and Margaret-Anne Storey. “Code Duplication and Reuse in Jupyter Notebooks”. In: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) 2020. 2020, pages 1–9. doi: 10.1109/VL/ HCC50065.2020.9127202. arXiv: 2005.13709

Diamantopoulos, T., Karagiannopoulos, G., and Symeonidis, A. CodeCatch: Extracting Source Code Snippets from Online Sources. In 2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE) (May 2018), pp. 21–27.

Ichinco, M., and Kelleher, C. Exploring novice programmer example use. In 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (Oct. 2015), pp. 63–71.

Rong, X., Yan, S., Oney, S., Dontcheva, M., and Adar, E. Codemend: Assisting interactive programming with bimodal embezdding. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (New York, NY, USA, 2016), UIST ’16, ACM, pp. 247–258.

Wightman, D., Ye, Z., Brandt, J., and Vertegaal, R. Snipmatch: Using source code context to enhance snippet retrieval and parameterization. pp. 219–228.

Sebastian Baltes and Paul Ralph. 2020. Sampling in Software Engineering Research: A Critical Review and Guidelines. arXiv preprint arXiv:2002.07764 (2020).

Jonathan Bell, Owolabi Legunsen, Michael Hilton, Lamyaa Eloussi, Tifany Yung, and Darko Marinov. 2018. DeFlaker: Automatically detecting flaky tests. In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 433–444.

Adam Brinckman, Kyle Chard, Niall Gaffney, Mihael Hategan, Matthew B Jones, Kacper Kowalik, Sivakumar Kulasekaran, Bertram Ludäscher, Bryce D Mecum, Jarek Nabrzyski, et al. 2019. Computing environments for reproducibility: Capturing the "Whole Tale". Future Generation Computer Systems 94 (2019), 854–867.

freezegun. 2019. Let your Python tests travel through time. (2019). Retrieved August 23, 2019 from https://pypi.org/project/freezegun/0.1.11/

Pingfan Kong, Li Li, Jun Gao, Tegawendé F Bissyandé, and Jacques Klein. 2019. Mining Android Crash Fixes in the Absence of Issue- and Change-Tracking Systems. In The 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019).

Li Li, Tegawendé F Bissyandé, Haoyu Wang, and Jacques Klein. 2018. CiD: Automating the Detection of API-related Compatibility Issues in Android Apps. In The ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2018).

Jeffrey M. Perkel. 2018. Why Jupyter is data scientists' computational notebook of choice. Nature news 563 (2018), 145–146.

Adam Rule, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, Mai H Nguyen, Sara Brin Rosenthal, Fernando Pérez, et al. 2019. Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. (2019).

Jiawei Wang, Li Li, and Andreas Zeller. 2020. Better Code, Better Sharing: On the Need of Analyzing Jupyter Notebooks. In The 42nd International Conference on Software Engineering, NIER Track (ICSE 2020).

Ziv Yaniv, Bradley C Lowekamp, Hans J Johnson, and Richard Beare. 2018. SimpleITK image-analysis notebooks: a collaborative environment for education and reproducible research. Journal of digital imaging 31, 3 (2018), 290–303.