

Early-Stage Autism Spectrum Disorder Detection using Machine Learning

ABSTRACT

ABSTRACT

The paper proposes a machine learning framework for early-stage detection of Autism Spectrum Disorder (ASD) using four different feature scaling strategies and eight ML algorithms. The proposed framework employs four different Feature Scaling (FS) strategies i.e., Quantile Transformer (QT), Power Transformer (PT), Normalizer, and Max Abs Scaler (MAS). Then, the feature-scaled datasets are classified through eight simple but effective ML algorithms like Ada Boost (AB), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), Gaussian Naïve Bayes (GNB), Logistic Regression (LR), Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA). The experiments are performed on four standard ASD datasets (Toddlers, Adolescents, Children, and Adults). The best-performing classification methods, and the best FS techniques for each ASD dataset are identified. The highest accuracies are achieved while scaling Toddlers and Children with normalizer FS and Adolescents and Adults with the QT FS method. The ASD risk factors are calculated, and the most important attributes are ranked according to their importance values using four different Feature Selection Techniques (FSTs) i.e., Info Gain Attribute Evaluator (IGAE), Gain Ratio Attribute Evaluator (GRAE), Relief F Attribute Evaluator (RFAE), and Correlation Attribute Evaluator (CAE). The proposed framework has achieved promising results compared to existing approaches for the early detection of ASD.

TABLE OF CONTENTS

S.NO	CONTENT	PGNO
1	Introduction	1
	1.1 Objective	3
	1.2 Problem Statement	3
	1.3 Software requirements	3
	1.4 Hardware requirements	5
2	Feasibility study	7
	2.1 Economic feasibility	8
	2.2 Technical feasibility	9
	2.3 Social Feasibility	9
3	Literature survey	10
4	System analysis	31
	4.1 Existing system	32
	4.1.1 Disadvantages of existing system	32
	4.2 Proposed system	32
	4.2.1 Advantages of proposed system	33
	4.3 Functional requirements	33
	4.4 Non-Functional requirements	34
5	System design	35
	5.1 System architecture	36
	5.2 UML diagrams	39
6	Implementation	47
	6.1 Modules	48
	6.2 Sample code	51
7	Software environment	55
8	System testing	75
	8.1 Testing strategies	76
	8.2 Test cases	79
9	Screens	80

10	Conclusion	96
11	References	98

LIST OF FIGURES

Fig No.	Fig Name	Page No.
Fig 5.1.	System Architecture	36
Fig 5.1.1	Data Flow Diagram	38
Fig 5.2.1	Use Case Diagram	41
Fig 5.2.2	Class Diagram	42
Fig 5.2.3	Activity Diagram	43
Fig 5.2.4	Sequence Diagram	44
Fig 5.2.5	Collaboration Diagram	45
Fig 5.2.6	Component Diagram	46
Fig 5.2.7	Deployment Diagram	46
Fig 9.1	Accuracy Comparison Graphs - Quantile Transformer	83
Fig 9.2	Precision Comparison Graphs - Quantile Transformer	83
Fig 9.3	Recall Comparison Graphs - Quantile Transformer	84
Fig 9.4	F1 Score Comparison Graphs - Quantile Transformer	84
Fig 9.5	Accuracy Comparison Graphs - Power Transformer	85
Fig 9.6	Precision Comparison Graphs - Power Transformer	85
Fig 9.7	Recall Comparison Graphs - Power Transformer	86
Fig 9.8	F1 Score Comparison Graphs - Power Transformer	86
Fig 9.9	Accuracy Comparison Graphs - MAXABSSCALER	87
Fig 9.10	Precision Comparison Graphs - MAXABSSCALER	87

Fig 9.11	Recall Comparison Graphs - MAXABSSCALER	88
Fig 9.12	F1 Score Comparison Graphs - MAXABSSCALER	88
Fig 9.13	Accuracy Comparison Graphs - Normalizer	89
Fig 9.14	Precision Comparison Graphs - Normalizer	89
Fig 9.15	Recall Comparison Graphs - Normalizer	90
Fig 9.16	F1 Score Comparison Graphs - Normalizer	90
Fig 9.17	Home Page	91
Fig 9.18	Registration Page	91
Fig 9.19	Login Page	92
Fig 9.20	Click on Adolescent	92
Fig 9.21	Upload Input Data	93.
Fig 9.22	Output Screen	93
Fig 9.23	Click on Adults Screening	94
Fig 9.24	Upload Input Data	95
Fig 9.25	Output Screen	95

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
8.2	Test Cases	79
9.1	Performance Evaluation – Toddler Dataset	81
9.2	Performance Evaluation – Children Dataset	81
9.3	Performance Evaluation – Adults Dataset	82
9.4	Performance Evaluation – Adolescent Dataset	82

INTRODUCTION

1. INTRODUCTION

Autism Spectrum Disorder (ASD) is a neurodevelopment condition associated with brain development that starts early stage of life, impacting a person's social relationships and interaction issues. ASD has restricted and repeated behavioral patterns, and the word spectrum encompasses a wide range of symptoms and intensity . Even so, the identification and diagnosis of ASD are really difficult and sophisticated, using traditional behavioral science. Usually, Autism is most commonly diagnosed at about two years of age and can also be diagnosed later, based on its severity. A variety of treatment strategies are available to detect ASD as quickly as possible. These diagnostic procedures aren't always widely used in practice until a severe chance of developing ASD. Over the past few years, several studies have been conducted incorporating various Machine Learning (ML) approaches to analyze and diagnose ASD and also other diseases, such as diabetes, stroke, and heart failure prediction as quickly as possible. So, This paper proposes an effective framework for the evaluation of various Machine Learning (ML) techniques for the early detection of ASD.

This study assembles four standard ASD datasets (Babies, Kids, Youths, and Grown-ups) and at first preprocesses the datasets (control of missing qualities and encoding). Then, at that point, four Element Scaling (FS) strategies including Quantile Transformer (QT), Power Transformer (PT), Normalizer, and Max Abs Scaler (MAS) are embraced to plan the datasets into a proper arrangement for additional appraisals. Thereafter, the element scaled datasets are arranged by eight straightforward be that as it may, viable arrangement draws near (Stomach muscle, RF, DT KNN, Gaussian Credulous Bayes (GNB), LR, SVM, and LDA), and the best characterization models are recognized. In the mean time, we likewise investigate the meaning of the FS strategies on each dataset by investigating the

exploratory results of the transformed datasets. Thereafter, four Element Determination Techniques (FST) i.e., Data Gain Trait Evaluator (IGAE), Gain Proportion Characteristic Evaluator (GRAE), Help F Quality Evaluator (RFAE), and Connection Trait Evaluator (CAE) are carried out to work out the gamble variables of ASD and rank the main highlights of these element scaled Babies, Kids, Youths and Grown-ups datasets.

1.1 OBJECTIVE

The main aims to create an effective prediction model using different types of ML methods to detect autism in people of different ages.

1.2 PROBLEM STATEMENT

The proposed Project will allow healthcare practitioners to take into account the most important features while screening ASD cases. The limitation of our research work is that the amount of data was not sufficient enough to build a generalized model for people of all stages.

1.3 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Platform – In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a

computer's architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

APIs and drivers – Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

Web browser – Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

- 1) Software : Anaconda**
- 2) Primary Language : Python**
- 3) Frontend Framework : Flask**
- 4) Back-end Framework : Jupyter Notebook**
- 5) Database : Sqlite3**
- 6) Front-End Technologies : HTML,CSS,JavaScript and Bootstrap4**

1.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Architecture – All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power – The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

Memory – All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the

application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary storage – Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display adapter – Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

Peripherals – Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

1) Operating System : Windows Only

2) Processor : i5 and above

3) Ram : 8gb and above

4) Hard Disk : 25 GB in local drive

FEASIBILITY STUDY

2. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was

achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

LITERATURE SURVEY

3. LITERATURE SURVEY

3.1 Efficient Machine Learning Models for Early Stage Detection of Autism Spectrum Disorder:

<https://www.mdpi.com/1999-4893/15/5/166>

ABSTRACT: Autism spectrum disorder (ASD) is a neurodevelopment disorder that severely impairs an individual's cognitive, linguistic, object recognition, communication, and social abilities. This situation is not treatable, although early detection of ASD can assist to diagnose and take proper steps for mitigating its effect. Using various artificial intelligence (AI) techniques, ASD can be detected an at earlier stage than with traditional methods. The aim of this study was to propose a machine learning model that investigates ASD data of different age levels and to identify ASD more accurately. In this work, we gathered ASD datasets of toddlers, children, adolescents, and adults and used several feature selection techniques. Then, different classifiers were applied into these datasets, and we assessed their performance with evaluation metrics including predictive accuracy, kappa statistics, the f1-measure, and AUROC. In addition, we analyzed the performance of individual classifiers using a non-parametric statistical significant test. For the toddler, child, adolescent, and adult datasets, we found that Support Vector Machine (SVM) performed better than other classifiers where we gained 97.82% accuracy for the RIPPER-based toddler subset; 99.61% accuracy for the Correlation-based feature selection (CFS) and Boruta CFS intersect (BIC) method-based child subset; 95.87% accuracy for the Boruta-based adolescent subset; and 96.82% accuracy for the CFS-based adult subset. Then, we applied the Shapley Additive Explanations (SHAP) method into different feature subsets, which gained the highest accuracy and ranked their features based on the analysis.

3.2 A Deep Learning Approach to Predict Autism Spectrum Disorder Using Multisite Resting-State fMRI:

<https://www.mdpi.com/2076-3417/11/8/3636>

ABSTRACT: Autism spectrum disorder (ASD) is a complex and degenerative neuro-developmental disorder. Most of the existing methods utilize functional magnetic resonance imaging (fMRI) to detect ASD with a very limited dataset which provides high accuracy but results in poor generalization. To overcome this limitation and to enhance the performance of the automated autism diagnosis model, in this paper, we propose an ASD detection model using functional connectivity features of resting-state fMRI data. Our proposed model utilizes two commonly used brain atlases, Craddock 200 (CC200) and Automated Anatomical Labelling (AAL), and two rarely used atlases Bootstrap Analysis of Stable Clusters (BASC) and Power. A deep neural network (DNN) classifier is used to perform the classification task. Simulation results indicate that the proposed model outperforms state-of-the-art methods in terms of accuracy. The mean accuracy of the proposed model was 88%, whereas the mean accuracy of the state-of-the-art methods ranged from 67% to 85%. The sensitivity, F1-score, and area under receiver operating characteristic curve (AUC) score of the proposed model were 90%, 87%, and 96%, respectively. Comparative analysis on various scoring strategies shows the superiority of BASC atlas over other aforementioned atlases in classifying ASD and control.

3.3 A New Machine Learning Model Based On Induction of Rules For Autism Detection:

<https://pubmed.ncbi.nlm.nih.gov/30693818/>

ABSTRACT: Autism spectrum disorder is a developmental disorder that describes certain challenges associated with communication (verbal and non-verbal), social skills, and repetitive behaviors. Typically, autism spectrum disorder is diagnosed in a clinical environment by licensed specialists using procedures which can be lengthy and cost-ineffective. Therefore, scholars in the medical, psychology, and applied behavioral science fields have in recent decades developed screening methods such as the Autism Spectrum Quotient and Modified Checklist for Autism in Toddlers for diagnosing autism and other pervasive developmental disorders. The accuracy and efficiency of these screening methods rely primarily on the experience and knowledge of the user, as well as the items designed in the screening method. One promising direction to improve the accuracy and efficiency of autism spectrum disorder detection is to build classification systems using intelligent technologies such as machine learning. Machine learning offers advanced techniques that construct automated classifiers that can be exploited by users and clinicians to significantly improve sensitivity, specificity, accuracy, and efficiency in diagnostic discovery. This article proposes a new machine learning method called Rules-Machine Learning that not only detects autistic traits of cases and controls but also offers users knowledge bases (rules) that can be utilized by domain experts in understanding the reasons behind the classification. Empirical results on three data sets related to children, adolescents, and adults show that Rules-Machine Learning offers classifiers with higher predictive accuracy, sensitivity, harmonic mean, and specificity than those of other machine learning approaches such as Boosting, Bagging, decision trees, and rule induction.

3.4 Spatial Modeling of Gully Erosion Using Linear and Quadratic Discriminant Analyses in GIS and R:

ABSTRACT: Gully erosion is one of the most important types of water erosion that causes the destruction of agricultural and pasture lands in arid and semi-arid areas. The main purpose of this study is to produce gully erosion susceptibility maps using R-based data mining linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) models and comparison of their performances in Shahroud Watershed, Semnan Province, Iran. The important input parameters for gully erosion susceptibility assessment were obtained from different sources. Firstly, 172 gully erosion locations were obtained using Google Earth images and extensive field surveys. Then, the gully inventory was randomly classified into two datasets: 70 % (121 gullies location) for training the models and 30 % (51 gullies location) for validation purpose. Secondly, 12 gully erosion conditioning factors including, elevation, slope degree, slope aspect, plan curvature, distance from river, drainage density, convergence index, topography wetness index (TWI), distance from road, LU/LC, NDVI, and lithology were selected. Subsequently, gully erosion susceptibility maps created using LDA and QDA models in R statistical software and divided into four classes including low, moderate, high, and very high. Finally, the validation dataset, which was not used in the modeling process, was considered to validate gully erosion susceptibility maps using the receiver operating characteristics (ROC) curve. Results of validation showed that LDA and QDA models with the area under the curve (AUC) values of 0.875, 0.8620 are good predictors for gully erosion susceptibility mapping. Also, results indicate that in LDA and QDA models, 13.44% and 22.61% of total area located in very high susceptibility class to soil erosion. Outcome of this research could represent a fundamental tool for a sustainable land use planning, protect the land

from the water related soil erosion processes, and gully erosion hazard mitigation in the study area.

3.5 Machine learning in autistic spectrum disorder behavioral research: A review and ways forward:

<https://pubmed.ncbi.nlm.nih.gov/29436887/>

ABSTRACT: Autistic Spectrum Disorder (ASD) is a mental disorder that retards acquisition of linguistic, communication, cognitive, and social skills and abilities. Despite being diagnosed with ASD, some individuals exhibit outstanding scholastic, non-academic, and artistic capabilities, in such cases posing a challenging task for scientists to provide answers. In the last few years, ASD has been investigated by social and computational intelligence scientists utilizing advanced technologies such as machine learning to improve diagnostic timing, precision, and quality. Machine learning is a multidisciplinary research topic that employs intelligent techniques to discover useful concealed patterns, which are utilized in prediction to improve decision making. Machine learning techniques such as support vector machines, decision trees, logistic regressions, and others, have been applied to datasets related to autism in order to construct predictive models. These models claim to enhance the ability of clinicians to provide robust diagnoses and prognoses of ASD. However, studies concerning the use of machine learning in ASD diagnosis and treatment suffer from conceptual, implementation, and data issues such as the way diagnostic codes are used, the type of feature selection employed, the evaluation measures chosen, and class imbalances in data among others. A more serious claim in recent studies is the development of a new method for ASD diagnoses based on machine learning. This article critically analyses these recent investigative studies on autism, not only articulating the

aforementioned issues in these studies but also recommending paths forward that enhance machine learning use in ASD with respect to conceptualization, implementation, and data. Future studies concerning machine learning in autism research are greatly benefitted by such proposals.

Tabular format of Literature Survey

Sl.NO .	Title & Authors	Methodology	Proposed System	Cons	Conclusion
1.	Title: Eye Tracking-Based Diagnosis and Early Detection of Autism Spectrum Disorder Using Machine Learning and Deep Learning Techniques, (PDF) Eye Tracking-	The study utilized eye tracking to assess visual attention in children with autism spectrum disorder (ASD). Three artificial intelligence techniques were developed: neural networks	The proposed system integrates eye tracking technology with artificial intelligence techniques, including neural networks and deep learning (CNN models), for early diagnosis of autism	Ethical concerns: Invasive monitoring of children's eye movements and privacy issues related to data collection without informed consent may arise,	In conclusion, the study demonstrated the effectiveness of eye tracking in early ASD diagnosis. Three AI techniques, including neural networks, deep learning

<p><u>Based</u> <u>Diagnosis</u> <u>and Early</u> <u>Detection of</u> <u>Autism</u> <u>Spectrum</u> <u>Disorder</u> <u>Using</u> <u>Machine</u> <u>Learning and</u> <u>Deep</u> <u>Learning</u> <u>Techniques</u> <u>(researchgate.net)</u>,</p> <p>Author: Ibrahim Abdulrab Ahmed, Ebrahim Mohammed Senan, Taha H. Rassem., 2022</p>	<p>(FFNNs and ANNs) based on hybrid feature classification (LBP and GLCM) achieving 99.8% accuracy, pre- trained CNN models (GoogleNet and ResNet-18) with accuracy of 93.6% and 97.6%, and a hybrid approach (GoogleNet + SVM and ResNet-18 + SVM) with accuracies of 95.5% and 94.5%,</p>	<p>spectrum disorder (ASD). It offers highly accurate and efficient assessment of children's visual behavior to aid in early detection.</p>	<p>raising ethical questions about the implementa- tion of this system. Limited generalizati- on: The system's accuracy may vary across different cultural and demographi- c groups, potentially leading to misdiagnos- es or underdiagn- oses in some populations</p>	<p>(CNN models), and hybrid methods, yielded high accuracy rates, highlightin- g their potential for enhancing early and accurate autism diagnosis.</p>
--	--	--	---	---

		respectively.		.	
	<p>Title: A new machine learning model based on induction of rules for autism detection, A new machine learning model based on induction of rules for autism detection - PubMed (nih.gov),</p> <p>Author: Fadi Thabtah, David Peebles, 2020</p>	<p>The methodology involved developing a new machine learning method, Rules-Machine Learning, for autism spectrum disorder detection. It utilized three datasets and compared its performance with existing methods through metrics like accuracy, sensitivity, specificity, and harmonic mean.</p>	<p>The proposed system, Rules-Machine Learning, aims to enhance autism spectrum detection by utilizing machine learning techniques. It not only identifies autistic traits but also provides interpretable knowledge rules for clinicians, yielding</p>	<p>Complexity : Integrating machine learning with rule-based systems can be complex and require substantial computational resources. Data Bias: ML models may perpetuate biases present in training data, potentially</p>	<p>In conclusion, leveraging machine learning, specifically the Rules-Machine Learning method, shows promise in enhancing the accuracy and efficiency of autism spectrum disorder detection, offering valuable insights for clinicians</p>

			<p>improved accuracy and sensitivity across age groups.</p>	<p>leading to biased results in autism detection.</p> <p>Interpretability Trade-off: Balancing interpretability with ML's complexity can be challenging , making it difficult for clinicians to fully understand and trust the system's decisions.</p>	<p>and users to better understand diagnostic outcomes.</p>
	Title:	The study	Proposed	Complexity	In

<p>Exploring the pattern of Emotion in children with ASD as an early biomarker through Recurring-Convolution Neural Network (R-CNN),</p> <p>[PDF]</p> <p>Exploring the pattern of Emotion in children with ASD as an early biomarker through Recurring-Convolution Neural Network (R-</p>	<p>employs a time-variant approach to identify and analyze emotions in autistic children by utilizing a CNN for facial expression recognition with 68 facial landmark points and an RNN-based RCNN-FER system for improved accuracy and reduced time complexity compared to traditional machine learning</p>	<p>System: The study introduces an RCNN-FER system that utilizes CNN-based facial expression recognition with 68 facial landmarks to analyze emotions in autistic children, aiming for early intervention and improved accuracy compared to traditional machine learning models.</p>	<p>: RCNN-FER systems can be computationally intensive, potentially limiting real-time applications and increasing hardware requirements. Data Dependence: Effective performance relies heavily on a large and diverse dataset, which can be</p>	<p>conclusion, the study highlights the significance of early identification of Autism Spectrum Disorder (ASD) through facial expressions and emotions. The RCNN-FER system demonstrates superior accuracy and efficiency,</p>
---	--	--	--	--

	CNN) Semantic Scholar, Author: S. Abirami, G. Kousalya, R. Karthick, 2021	models.		challenging to acquire for autistic children. Et hical Concerns: Privacy and consent issues arise when dealing with sensitive data, especially in the context of minors, necessitating stringent ethical considerations	offering promise for improved support in the autistic community .
	Title: A new computational	Methodology: Utilized Variable	Proposed System: Develop a	Creating a Variable Analysis	In conclusion, the

	<p>intelligence approach to detect autistic features for autism screening, https://pubmed.ncbi.nlm.nih.gov/30032959/, Author: Fadi Thabtah, Firuz Kamalov, Khairan Rajab, 2018</p>	<p>Analysis (Va) to identify influential features from ASD screening methods, reducing feature-to-feature correlations. Verified results with two machine learning algorithms, assessing specificity, sensitivity, PPVs, NPVs, and predictive accuracy. Compared with other filtering methods.</p>	<p>computational intelligence tool called Variable Analysis (Va) to reduce features in ASD screening methods, verifying its efficacy using two machine learning algorithms. Enhance early ASD detection.</p>	<p>tool for ASD screening may inadvertently oversimplify the complex diagnostic process, potentially leading to missed cases and reduced accuracy in early ASD detection.</p>	<p>Variable Analysis (Va) approach shows promise in efficiently identifying influential features for ASD screening tools, potentially aiding early detection and treatment while maintaining competitive predictive accuracy and</p>
--	---	--	--	---	--

					sensitivity.
	<p>Title:</p> <p>Machine learning approach for early detection of autism by combining questionnaire and home video screening.</p> <p>https://pubmed.ncbi.nlm.nih.gov/29741630/,</p> <p>Author:</p> <p>Halim Abbas, Ford Garberson, Eric Glover, Dennis P Wall, 2018</p>	<p>Methodology:</p> <p>Two ML algorithms are trained using parent-reported questionnaires and home videos to detect autism. Novel techniques address data challenges. A multi-center clinical study with 162 children validates algorithm performance, showing improved accuracy.</p>	<p>Proposed System:</p> <p>Develop two machine learning algorithms utilizing parent-reported questionnaires and home videos to screen for autism, followed by a combination algorithm. Address data challenges with novel techniques. Validate performance through a multi-center</p>	<p>Cons of the proposed system include potential privacy concerns with home videos, resource-intensive multi-center studies, and uncertainty in generalizing results to diverse populations.</p> <p>Additionally, algorithm complexity</p>	<p>Conclusion : Machine learning-based autism screening tools, utilizing parent questionnaires and home videos, offer a promising, cost-effective approach with improved accuracy. Further studies are needed to validate</p>

			clinical study of 162 children, demonstrating improved accuracy over existing tools.	may hinder adoption.	and expand these findings.
	<p>Classification of Adults with Autism Spectrum Disorder using Deep Neural Network, Classification of Adults with Autism Spectrum Disorder using Deep Neural Network Semantic Scholar,</p>	<p>Methodology: Two adult ASD screening datasets were utilized. Deep Neural Network (DNN) and Support Vector Machine (SVM) models were employed for classification, with accuracy compared to assess DNN's performance.</p>	<p>Proposed System: Utilizing Deep Neural Network (DNN) and Support Vector Machine (SVM) models to analyze two adult ASD screening datasets, aiming to enhance ASD diagnosis accuracy</p>	<p>One drawback of utilizing both Deep Neural Network (DNN) and Support Vector Machine (SVM) models for analyzing adult ASD screening datasets is the increased complexity</p>	<p>Conclusion : The Deep Neural Network (DNN) outperformed Support Vector Machine (SVM) in diagnosing ASD using adult screening data, achieving higher classification</p>

	<p>Author: M. F. Misman, A. A. Samah, Farah Aqilah Ezudin, Hairuddin Abu Majid, Z. A. Shah, H. Hashim, Muhamad Farhin Harun, 2019</p>		<p>through machine learning techniques.</p>	<p>and computational resources required for training and deploying two distinct machine learning models, potentially leading to higher resource utilization and maintenance costs.</p>	<p>accuracy. DNN's effectiveness suggests its potential for improving ASD diagnosis.</p>
	<p>Title: Machine Learning-Based Models for Early Stage</p>	<p>We collected ASD datasets for various age groups, applied feature transformations</p>	<p>Proposed System: We aim to develop a machine learning-</p>	<p>One potential drawback of this proposed system is</p>	<p>In conclusion, our study demonstrates the potential of</p>

	<p>Detection of Autism Spectrum Disorders, https://www.researchgate.net/publication/337367854_Machine_Learning-Based_Models_for_Early_Stage_Detection_of_Autism_Spectrum_Disorders, Author: Tania Akter, Md. Shahriare Satu, Md. Imran Khan, Mohammad Hanif Ali, 2019</p>	<p>, tested classifiers, and identified significant risk factors using machine learning methods.</p>	<p>based ASD detection system utilizing optimized classifiers and feature transformations. Early intervention strategies can then be applied for early ASD detection.</p>	<p>the risk of overreliance on machine learning, potentially neglecting the importance of human clinical expertise in ASD diagnosis and intervention.</p>	<p>machine learning in predicting ASD status, particularly in early stages. SVM, Adaboost, and feature transformations like sine and Z-score showed promise, offering hope for early intervention strategies.</p>
	Title: Aarya	Methodology:	Proposed	However,	Conclusion

	<p>- A Kinesthetic companion for children with Autism Spectrum Disorder, https://www.semanticscholar.org/paper/Aarya-A-Kinesthetic-companion-for-children-with-Sreedasyam-Rao/e745a9fb70813d8abce0a0efce3aa845f7361476, Author: Rachita Sreedasyam, Aishwarya Rao, Nidhi</p>	<p>Aarya employs a gesture-based Microsoft Kinect virtual environment to aid children with ASD in facing real-world situations, enhancing confidence, and fostering social and communication skills. Continuous refinement and expert input drive tool improvement.</p>	<p>System: Aarya, a gesture-based virtual companion using Microsoft Kinect, aids children with ASD in real-world interaction, enhancing confidence, social skills, and communication. Continuous refinement and expert guidance drive system improvement.</p>	<p>potential drawbacks include privacy concerns as Kinect collects personal data, limited accessibility for those without the necessary technology, and reliance on technology rather than human interaction for social skill development.</p>	<p>: Aarya, utilizing Microsoft Kinect, offers a promising approach to help children with ASD overcome social and communication challenges. Continuous refinement and expert insights hold the potential for significant enhancements in supporting</p>
--	---	---	---	--	---

	Sachidanandan, Nalini Sampath, S. K. Vasudevan, 2017				their growth and interaction abilities.
	Title: Typically developed adults and adults with autism spectrum disorder classification using centre of pressure measurements, (PDF) Typically developed adults and adults with autism spectrum disorder	In this study, 19 typically developed adults and 11 adults with high-functioning autism or Asperger's syndrome participated. Force plate measurements of center of pressure were analyzed using a correlation-based feature selection algorithm,	The proposed system utilizes force plate measurements of center of pressure to classify adults with ASD and typically developed adults, employing a correlation-based feature selection algorithm for attribute evaluation, achieving a	One limitation of this proposed system is its reliance on force plate measurements, which may not be readily available or feasible for widespread implementation in clinical or real-world settings.	In conclusion, this study successfully differentiated adults with ASD from typically developed adults using center of pressure measurements. The correlation-based feature

classification using centre of pressure measurements (researchgate.net) , Author: Kwang Leng Goh, Susan Morris, Simon Rosalie, Chris Foster, Torbjorn Falkmer, 2016	achieving a high 0.976 classification accuracy of up to 97.6%.	high 0.976 classification accuracy.	Additionally, the high classification accuracy may not generalize well to diverse populations.	selection algorithm achieved high accuracy, highlighting its potential for clinical applications in assessing postural control in ASD..
Title: Use of machine learning for behavioral distinction of autism and ADHD, Use of machine	Using forward feature selection, undersampling, and 10-fold cross-validation, we trained six machine	We propose a streamlined system employing machine learning models to assess ASD and ADHD	However, implementing such a system may face challenges, including potential over-	In conclusion, our study demonstrates the potential of machine learning to effectively

	learning for behavioral distinction of autism and ADHD - PubMed (nih.gov) , Authors: M Duda, R Ma, N Haber, D P Wall, 2016.	learning models on Social Responsiveness Scale data from 2925 individuals with ASD or ADHD.	risk using a 65-item Social Responsiveness Scale, enabling quick, accurate, and electronically administered preliminary evaluations for expedited diagnosis.	reliance on automated assessments, limited consideration of individual nuances, and concerns about data privacy and accuracy.	differentiate between ASD and ADHD using a minimal set of behavioral indicators. This approach holds promise for expediting diagnosis and facilitating early intervention for these prevalent developmental disorders.
--	--	---	--	---	--

SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

In existing work they analyzed the ASD attributes utilizing Rule-based ML (RML) techniques and confirmed that RML helps classification models boost classification accuracy. Another study combined the Random Forest (RF) along with Iterative Dichotomiser 3 (ID3) algorithms and produced predictive models for children, adolescents, and adults. Another study conducted by demonstrates a feature-to-class and feature-to-feature correlation value utilizing cognitive computing and implemented Support Vector Machines (SVM), Decision Tree (DT), Logistic Regression (LR) as ASD diagnostic and prognosis classifiers.

4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

1. Previous studies used either specific or only three datasets or a specific age group for their study.
2. Previous studies have used only limited feature selection techniques to identify the most important attributes for the classification of ASD.
3. They don't used any feature scaling techniques to preprocess the data.

4.2 Proposed System:

The proposed work of this paper is to develop an effective framework for the early detection of Autism Spectrum Disorder (ASD) using various machine learning (ML) techniques. The paper aims to evaluate the performance of different ML algorithms like Ada Boost (AB), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), Gaussian Naïve Bayes (GNB), Logistic Regression (LR), Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA).

And different feature scaling techniques on four standard ASD datasets (Toddlers, Adolescents, Children, and Adults). The paper also uses four different Feature Selection Techniques (FSTs) like Info Gain Attribute Evaluator (IGAE), Gain Ratio Attribute Evaluator (GRAE), Relief F Attribute Evaluator (RFAE), and Correlation Attribute Evaluator (CAE) to rank the most important attributes.

4.2.1 Advantages of proposed system:

1. The paper used four standard datasets or age groups such as Toddlers, Adolescents, Children, and Adults.
2. The ASD risk factors are calculated, and the most important attributes are ranked according to their importance values using four different Feature Selection Techniques (FSTs).
3. This study proposes a machine learning framework that employs multiple feature scaling strategies, and classification algorithms.
4. The detailed feature importance analysis provided in the paper can guide the decision-making of healthcare practitioners while screening ASD cases.
5. The best-performing classification methods and the best feature scaling techniques for each ASD dataset are identified based on the experimental outcomes.

4.3 FUNCTIONAL REQUIREMENTS

1. Data Collection
2. Data Pre-processing
3. Training and Testing
4. Modeling
5. Predicting

4.4 NON FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non-functional Requirements allow you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

SYSTEM DESIGN

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

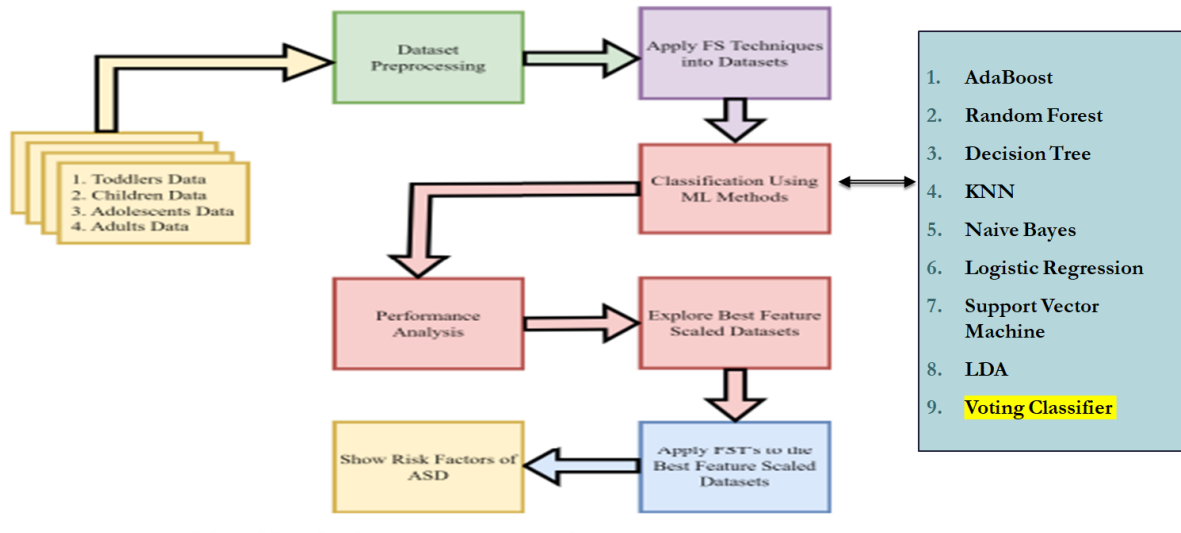


Fig.5.1 System architecture

The image outlining the steps involved in a Machine learning project. The specific project appears to be focused on Early Satge Detection using the Autism datasets.

Here's a breakdown of the workflow depicted in the image:

1. **Data Exploration:** This step involves loading the data into the system and understanding its structure, features, and distributions. It's essential for gaining insights into the data before proceeding further.

2. **Processing:** After data exploration, the next step is processing the data, which typically involves tasks such as cleaning, handling missing values, feature engineering, and possibly scaling or transforming features to prepare them for modeling.

3. Splitting Data into Train & Test: This step involves dividing the dataset into two parts: one for training the model and the other for evaluating its performance. The train set is used to train the model, while the test set is used to assess how well the model generalizes to unseen data.

4. Model Generation: This is the core step where machine learning models are built and trained on the training data. Based on your description, it seems like you're considering various algorithms such as AdaBoost, Random Forest, Decision Tree, KNN, Naive Bayes, Logistic Regression, Support Vector Machine, LDA, and ensemble methods like Voting Classifier.

It appears you're planning to test these models with different feature transformations/scalings such as Quantile Transformer, Power Transformer, MaxAbsScaler, and Normalizer. Each of these techniques has its own way of transforming the features, which can affect model performance.

Algorithms usage in project described below:

AdaBoost

AdaBoost, short for Adaptive Boosting, is an ensemble learning technique that combines the predictions of weak classifiers to create a strong classifier. It assigns weights to misclassified instances, allowing subsequent weak learners to focus more on the difficult cases. AdaBoost is effective for handling imbalanced datasets and improving the overall classification performance. Its adaptability makes it suitable for capturing complex relationships within ASD datasets.

Random Forest

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the Classes or mean prediction (regression) of the individual trees. Random Forest is robust, handles high dimensional data well, and is less prone to overfitting. Its ability to capture feature importance makes it valuable for identifying relevant factors in ASD detection.

Decision Tree

A Decision Tree is a tree-like model where an internal node represents a test on an attribute, each branch represents the outcome of the test, And each leaf node represents a class label. Decision Trees provide a clear visualization of decision-making processes. They are Interpretable and can reveal key factors contributing to ASD prediction, aiding in the identification of crucial features.

KNN

K-Nearest Neighbors is a non parametric algorithm that classifies a data point based on the majority class of its k-nearest neighbors in the feature space. KNN is valuable for identifying patterns in data without assuming a specific functional form. It can capture local relationships within ASD datasets that might not be evident globally.

Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' theorem with the assumption of independence between features. Naive Bayes is computationally efficient and works well with high dimensional datasets. Its simplicity and speed make it suitable for the initial exploration of ASD data.

Logistic Regression

Logistic Regression is a linear model for binary classification that predicts the probability of an instance belonging to a particular class using the logistic function. Logistic Regression is interpretable and provides insights into the relationship between features and the likelihood of ASD. It serves as a baseline model for binary classification tasks.

SVM

Support Vector Machine is a supervised learning algorithm that finds the hyperplane that best separates classes in a high dimensional space. SVM is effective in handling complex decision boundaries. It can capture non linear relationships in ASD datasets, potentially improving classification accuracy.

LDA

Linear Discriminate Analysis is dimensionality reduction and classification technique that finds linear combinations of features that best separate classes. LDA is useful for reducing dimensionality and highlighting discriminating features. It can enhance interpretability and may aid in Identifying critical factors in ASD detection.

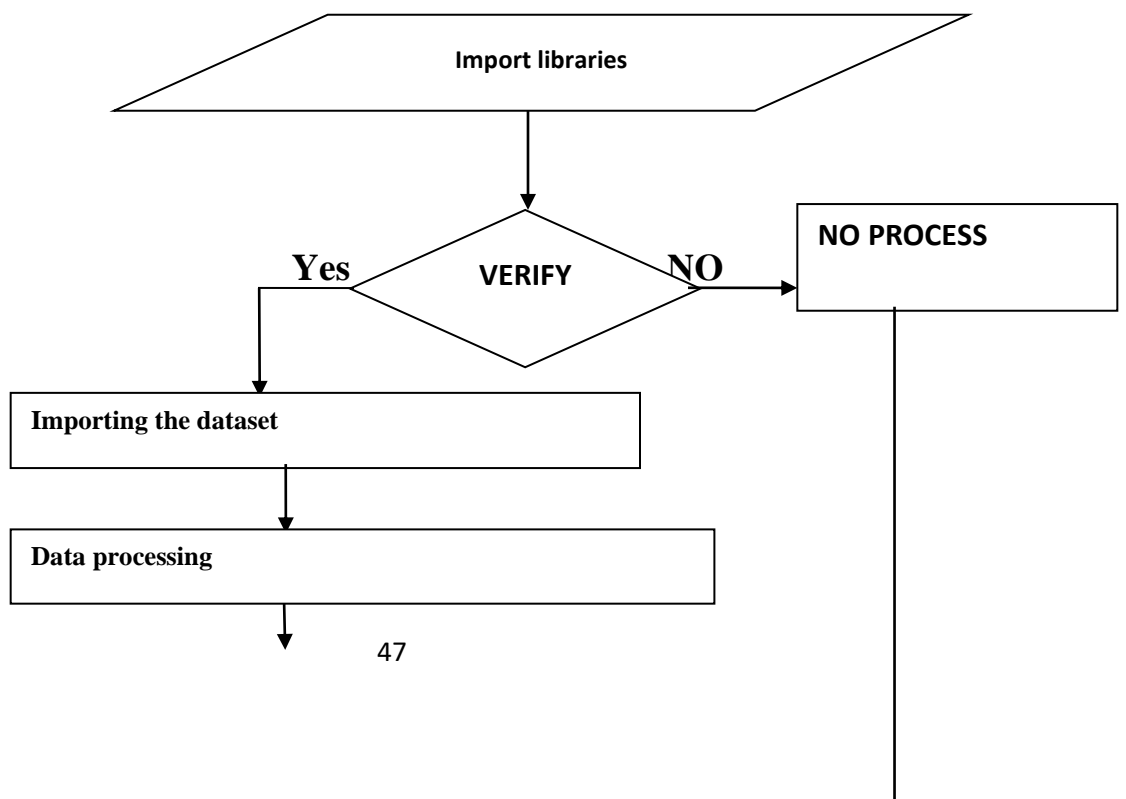
Voting Classifier

A Voting Classifier, combining is a form of ensemble learning where multiple individual classifiers are trained, and their predictions are combined to make a final prediction. In this project, we have chosen AdaBoost and Random Forest as the base classifiers.

5. Performance Evaluation: Once the models are trained, they need to be evaluated to assess their performance and determine which one performs best for the given task. Common evaluation metrics include accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC), among others.

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



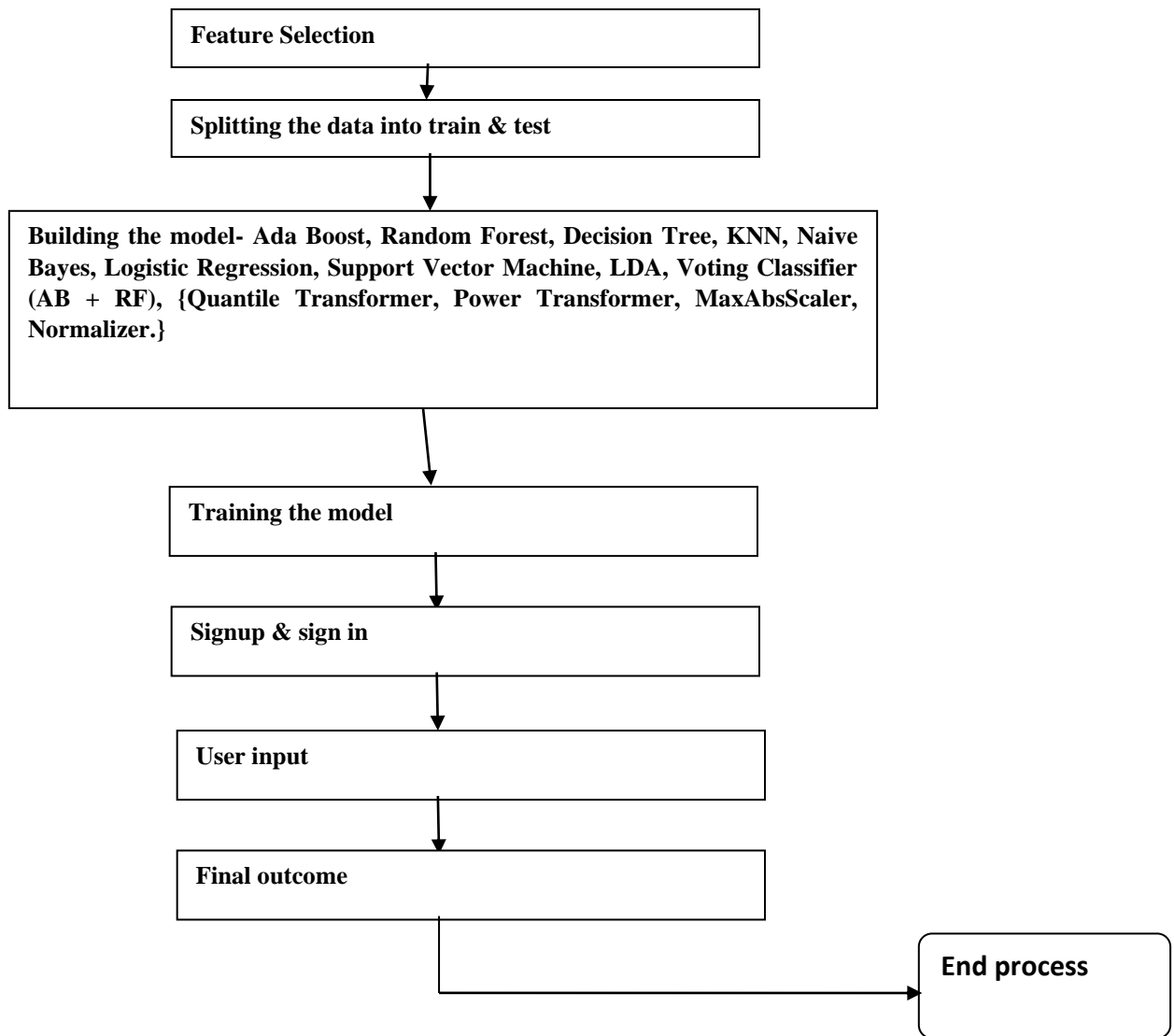


Fig 5.1.1 Data Flow Diagram

5.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Use case diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

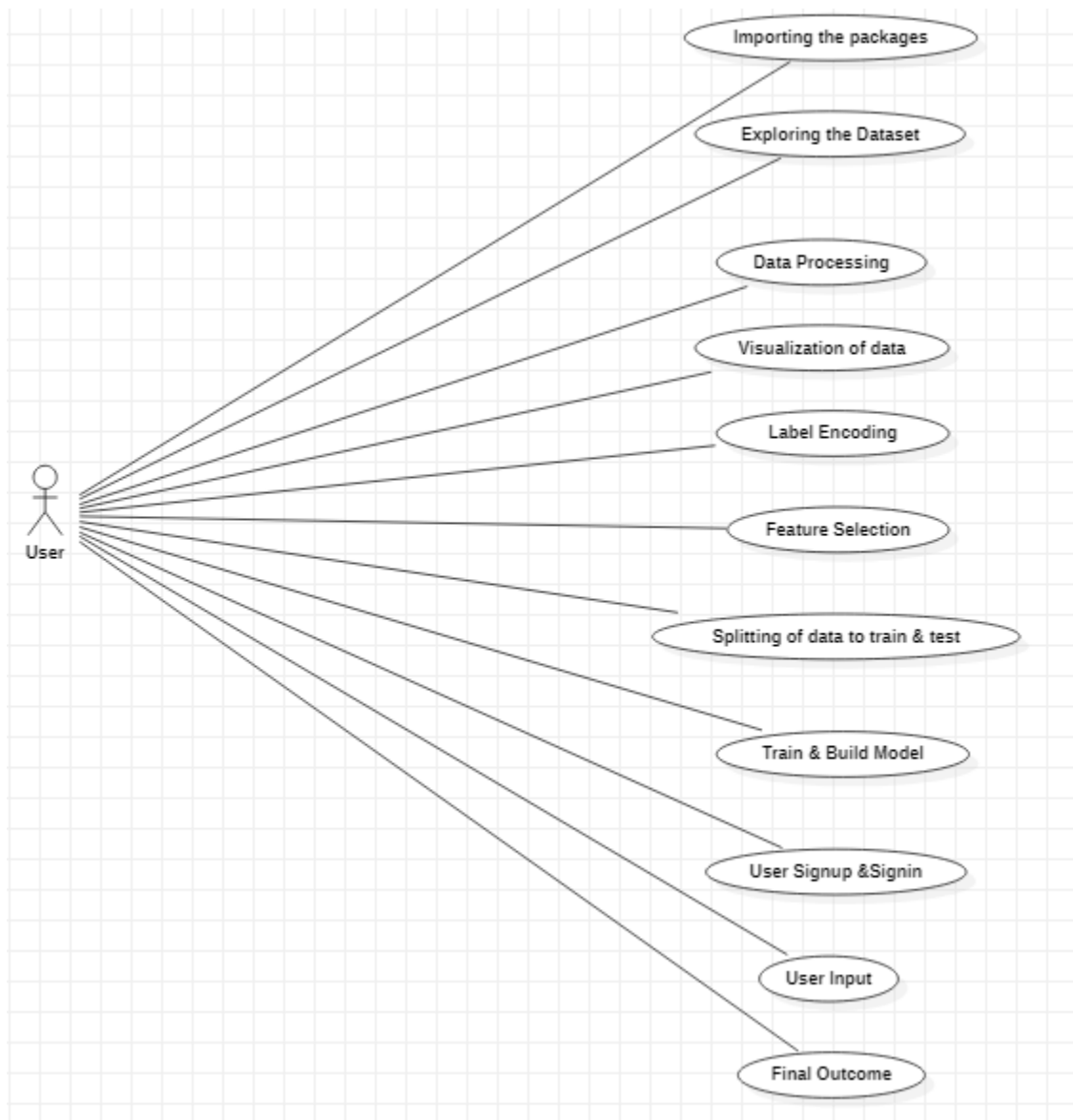


Fig 5.2.1 Use Case Diagram

Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

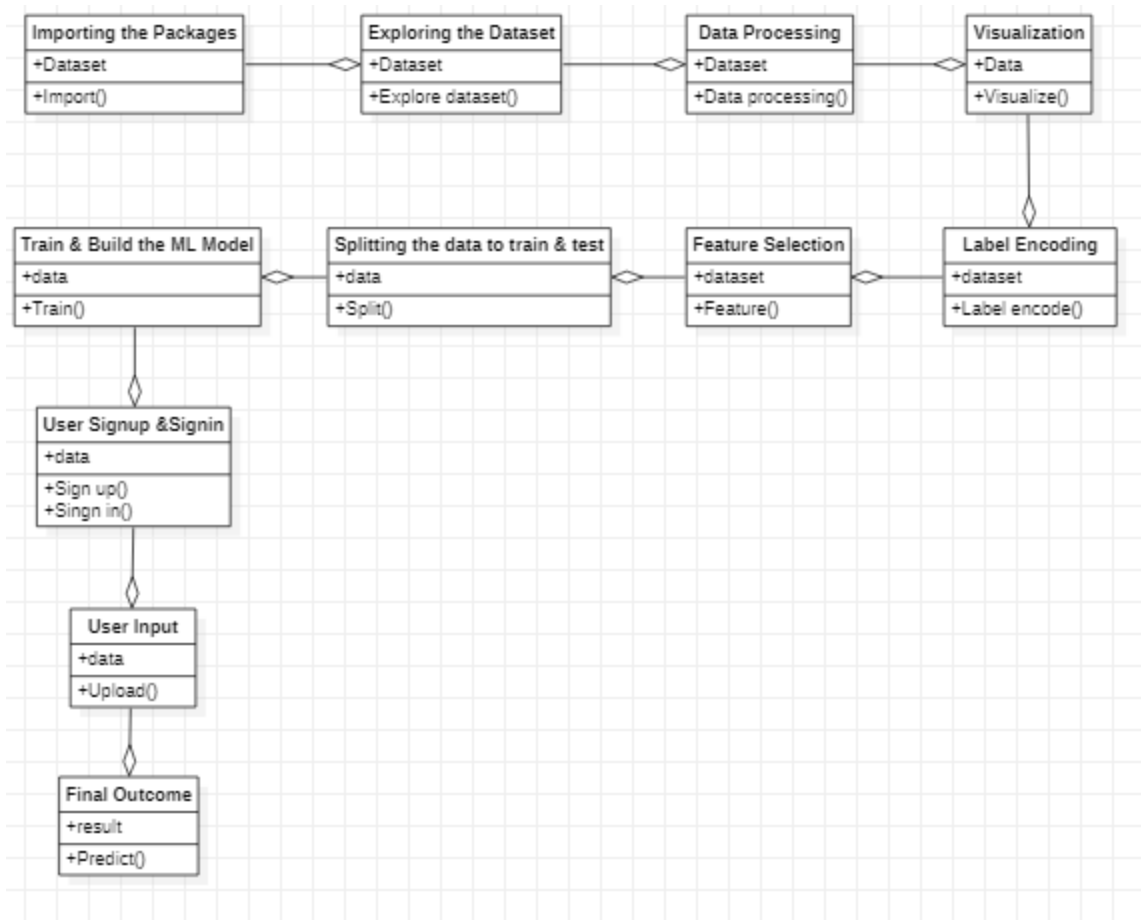


Fig 5.2.2 Class Diagram

Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

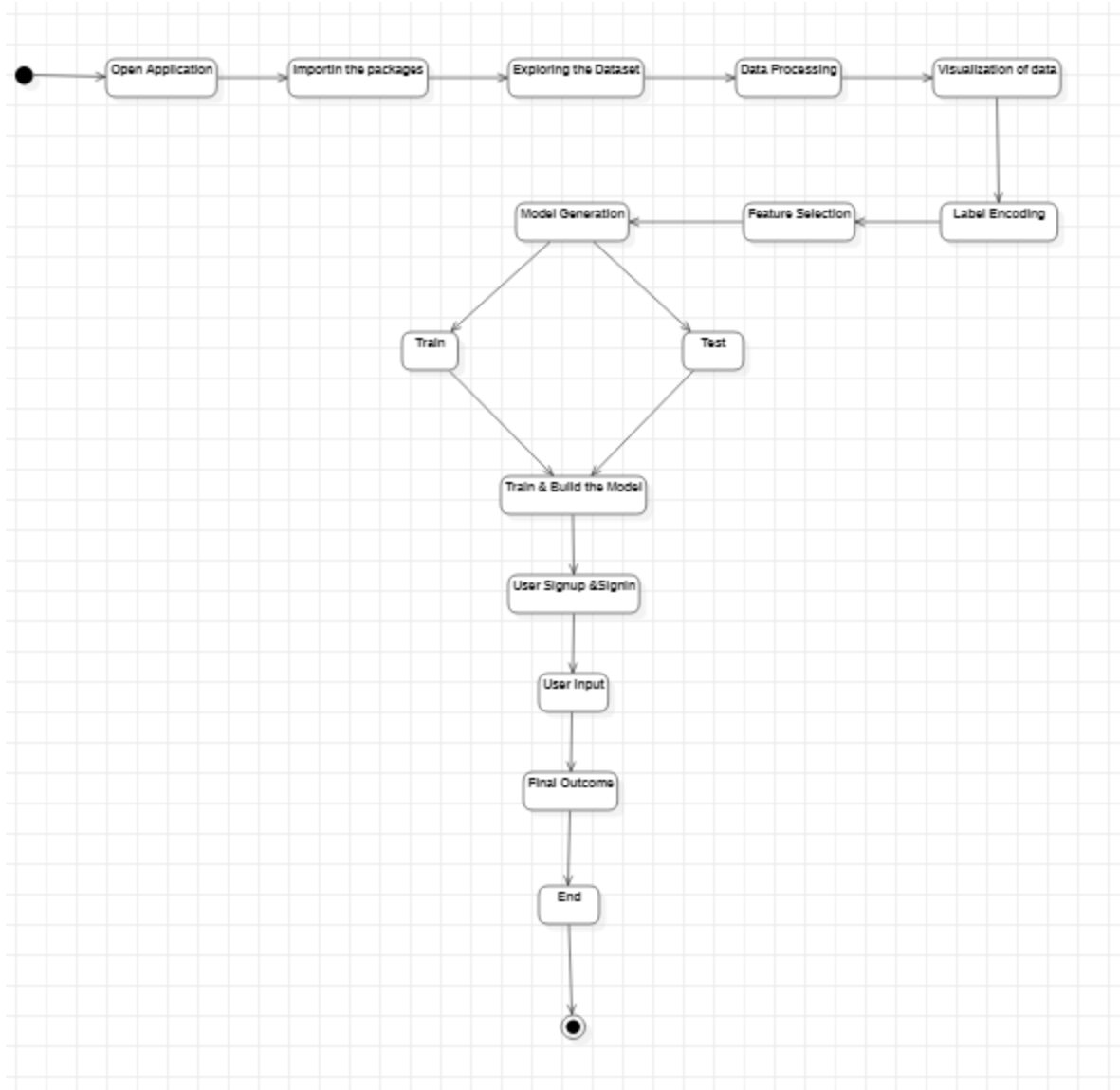


Fig 5.2.3 Activity Diagram

Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

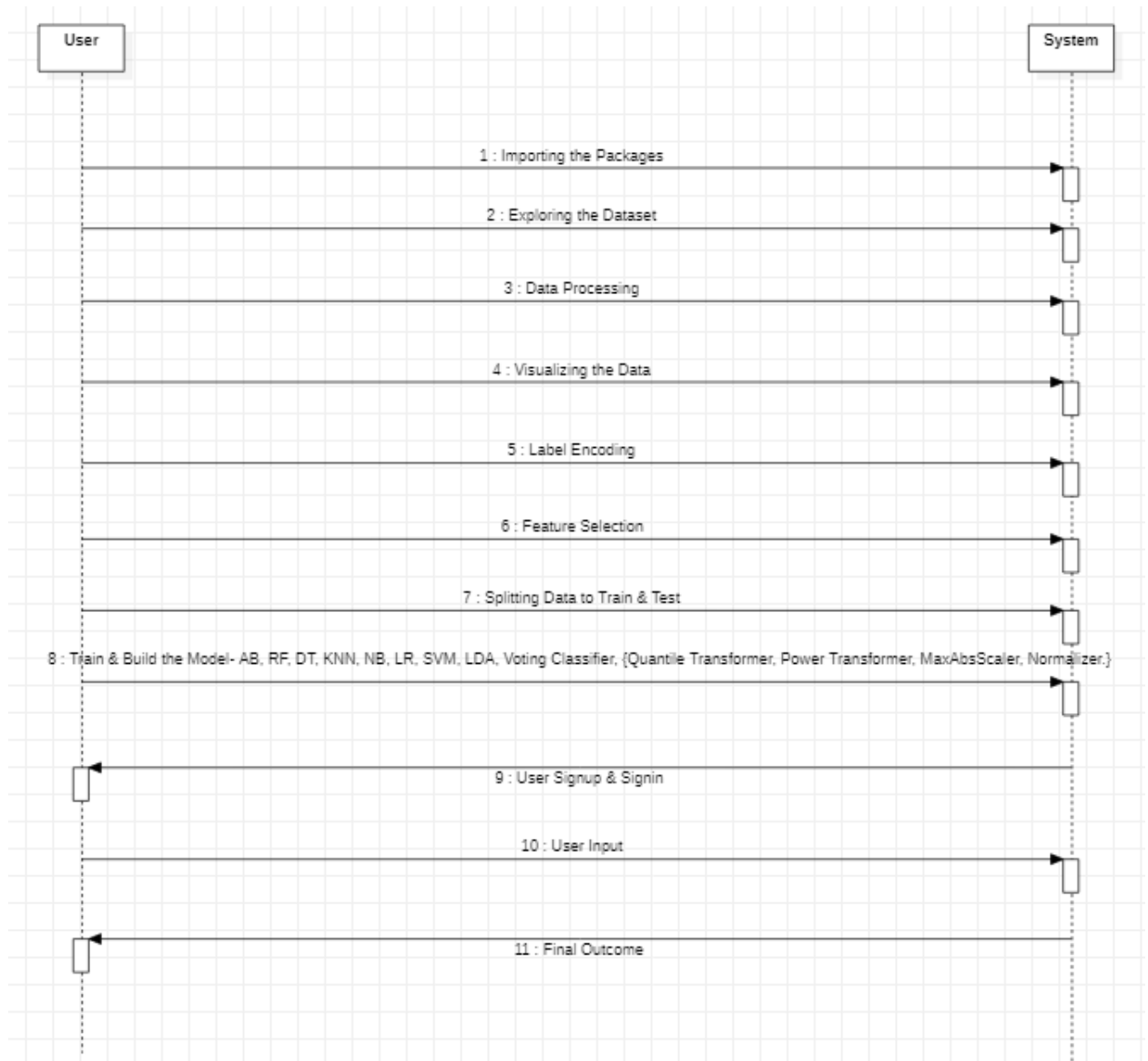


Fig 5.2.4 Sequence Diagram

Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

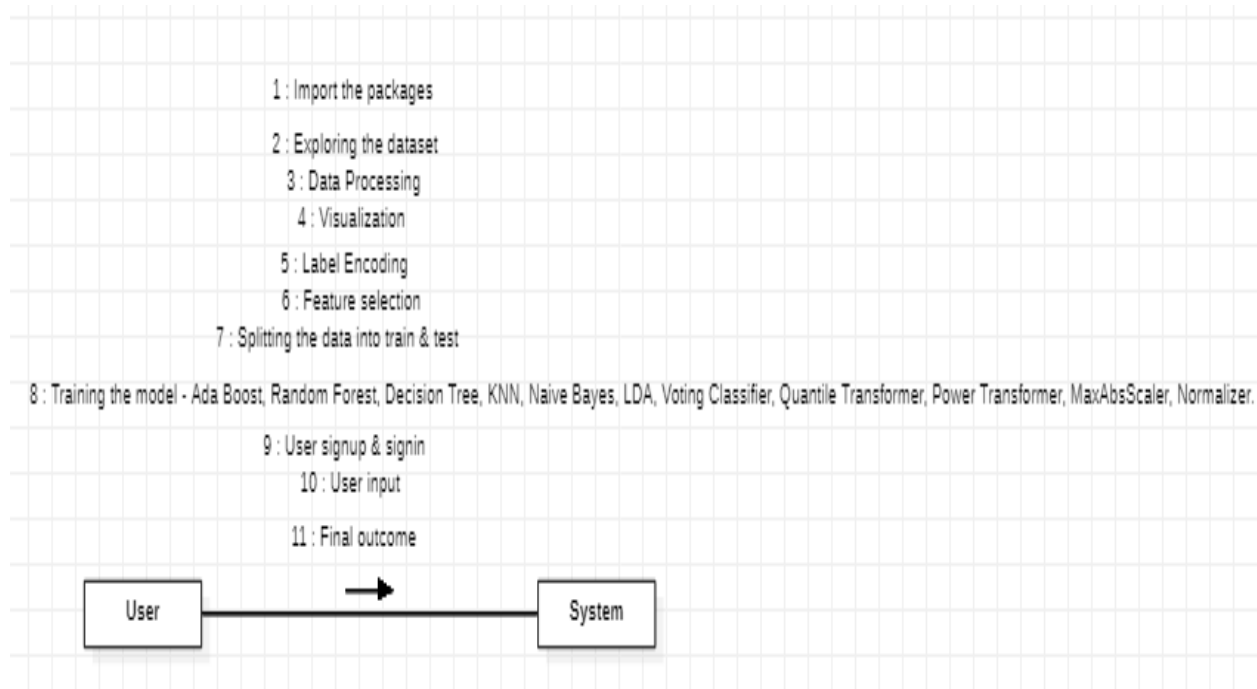


Fig 5.2.5 Collaboration Diagram

Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

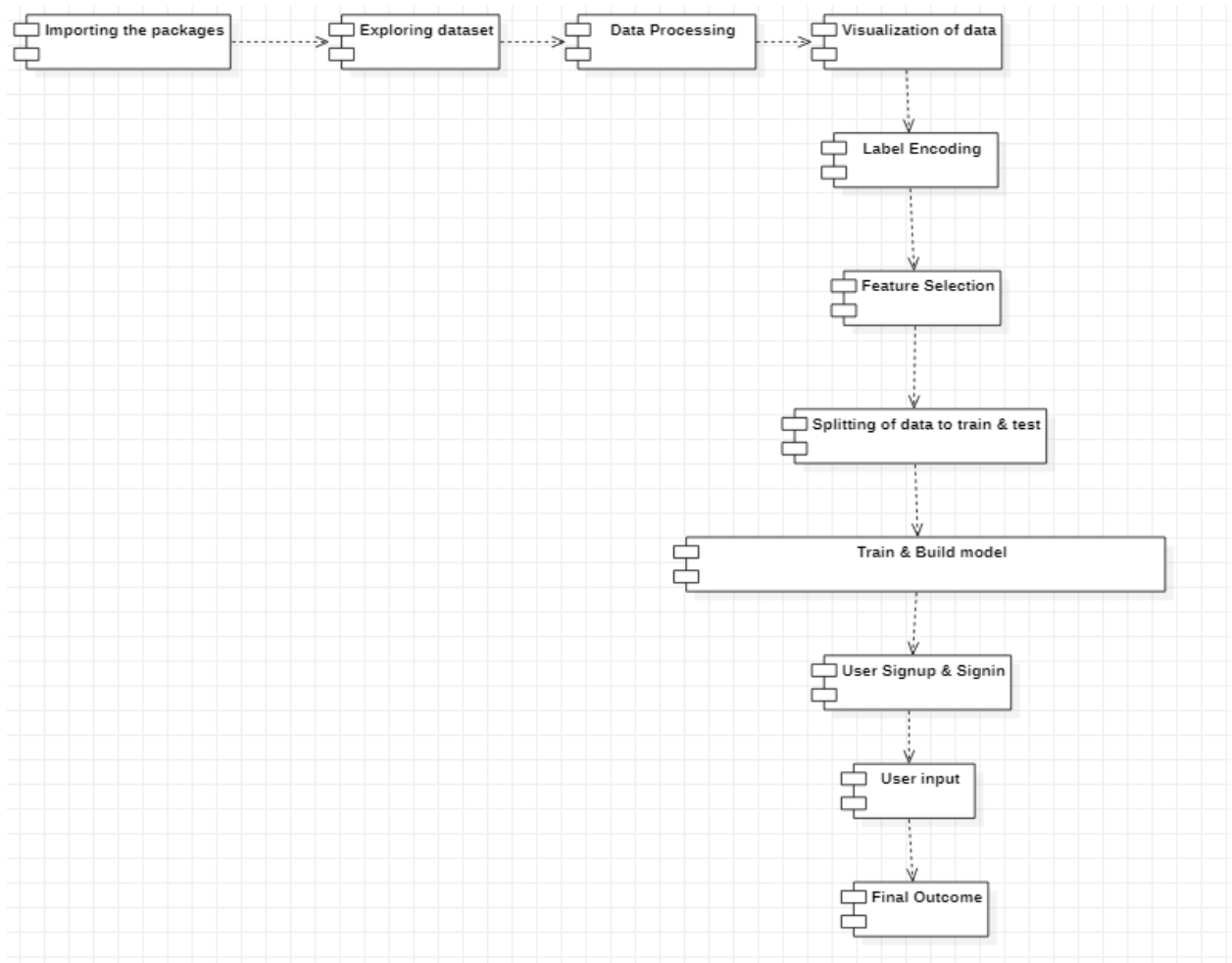


Fig 5.2.6 Component Diagram

Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.



Fig 5.2.7 Deployment Diagram

IMPLEMENTATION

6. IMPLEMENTATION

MODULES:

- Data exploration: using this module we will load data into system
- Processing: Using the module we will read data for processing
- Splitting data into train & test: using this module data will be divided into train & test
- Model generation: Model building
 - Quantile Transformer:- AdaBoost, Random Forest, Decision Tree, KNN, Naive Bayes, Logistic Regression, Support Vector Machine, LDA, Voting Classifier (AB + RF).
 - Power Transformer:- AdaBoost, Random Forest, Decision Tree, KNN, Naive Bayes, Logistic Regression, Support Vector Machine, LDA, Voting Classifier (AB + RF).
 - MaxAbsScaler:- AdaBoost, Random Forest, Decision Tree, KNN, Naive Bayes, Logistic Regression, Support Vector Machine, LDA, Voting Classifier (AB + RF).
 - Normalizer:- Ada Boost, Random Forest, Decision Tree, KNN, Naive Bayes, Logistic Regression, Support Vector Machine, LDA, Voting Classifier (AB + RF).
- User signup & login: Using this module will get registration and login
- User input: Using this module will give input for prediction
- Prediction: final predicted displayed

Note: As an extension we applied an ensemble method combining the predictions of multiple individual models to produce a more robust and accurate final prediction.

However, we can further enhance the performance by exploring other ensemble techniques such as Voting Classifier with RF + Ada boost which got 100% accuracy for QT.

Algorithms:

Quantile Transformer: This method transforms the features to follow a uniform or a normal distribution. Therefore, for a given feature, this transformation tends to spread out the most frequent values. It also reduces the impact of (marginal) outliers: this is therefore a robust preprocessing scheme.

Power Transformer: Power transforms are a family of parametric, monotonic transformations that are applied to make data more Gaussian-like. This is useful for modeling issues related to heteroscedasticity (non-constant variance), or other situations where normality is desired.

Max Abs Scaler: Maximum absolute scaling scales the data to its maximum value; that is, it divides every observation by the maximum value of the variable: The result of the preceding transformation is a distribution in which the values vary approximately within the range of -1 to 1.

Normalizer: The normalizer of S in G is the set of elements of G that satisfy the weaker condition of leaving the set. fixed under conjugation. The centralizer and normalizer of S are subgroups of G . Many techniques in group theory are based on studying the centralizers and normalizers of suitable subsets S .

Ada Boost: AdaBoost, also called Adaptive Boosting, is a technique in Machine Learning used as an Ensemble Method. The most common estimator used with

AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called Decision Stumps.

Random Forest: Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

Decision Tree: Decision Trees. A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

KNN: K-Nearest Neighbors Algorithm. The k-nearest neighbors' algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Naive Bayes: Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the “naive” assumption of conditional independence between every pair of features given the value of the class variable.

Logistic Regression: Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

Support Vector Machine: SVM is a powerful supervised algorithm that works best on smaller datasets but on complex ones. Support Vector Machine, abbreviated as

SVM can be used for both regression and classification tasks, but generally, they work best in classification problems.

LDA: Linear Discriminant Analysis (LDA) is a supervised learning algorithm used for classification tasks in machine learning. It is a technique used to find a linear combination of features that best separates the classes in a dataset.

Voting Classifier (AB + RF): A voting classifier is a machine learning estimator that trains various base models or estimators and predicts on the basis of aggregating the findings of each base estimator. The aggregating criteria can be combined decision of voting for each estimator output.

6.2 SAMPLE CODE:

```
# This Python 3 environment comes with many helpful analytics libraries installed
```

```
# It is defined by the kaggle/python Docker image:
```

```
https://github.com/kaggle/docker-python
```

```
# For example, here's several helpful packages to load
```

```
import numpy as np # linear algebra
```

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
# Input data files are available in the read-only "../input/" directory
```

```
# For example, running this (by clicking run or pressing Shift+Enter) will list all  
files under the input directory
```

```
import os
```

```
for dirname, _, filenames in os.walk('/kaggle/input'):
```

```
    for filename in filenames:
```

```
print(os.path.join(dirname, filename))
```

```
# You can write up to 20GB to the current directory (/kaggle/working/) that gets  
preserved as output when you create a version using "Save & Run All"
```

```
# You can also write temporary files to /kaggle/temp/, but they won't be saved  
outside of the current session
```

```
df = pd.read_csv("/kaggle/input/autism-screening-on-  
adults/autism_screening.csv")
```

```
df.head()
```

```
df.shape
```

```
df.info()
```

```
df["ethnicity"].value_counts()
```

```
df["ethnicity"] = df["ethnicity"].str.replace("others", "Others")
```

```
df["ethnicity"] = df["ethnicity"].str.replace("?", "Others")
```

```
df["jundice"] = df["jundice"].map({"no": 0, "yes": 1})
```

```
df["austim"] = df["austim"].map({"no": 0, "yes": 1})
```

```
df["used_app_before"] = df["used_app_before"].map({"no": 0, "yes": 1})
```

```
df["gender"] = df["gender"].map({"f": 0, "m": 1})
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
plt.figure(figsize=(15, 5))
```

```
sns.countplot(x=df["ethnicity"])
```

```
plt.show()
```

```
plt.figure(figsize=(16, 12))
sns.countplot(x=df["gender"], hue=df["Class/ASD"])
plt.show()
```

```
from sklearn.preprocessing import LabelEncoder

def label_encoder(column):
    le = LabelEncoder().fit(column)
    print(column.name, le.classes_)
    return le.transform(column)
```

```
ndf["age"] = ndf["age"].astype("int64")
ndf["result"] = ndf["result"].astype("int64")

plt.figure(figsize=(10, 10))
sns.heatmap(ndf.corr() > 0.8, annot=True, cbar=False)
plt.show()
```

```
X = ndf.drop(["Class/ASD"], axis=1)
y = ndf["Class/ASD"]
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

In [23]:
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score
```

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)
r2_score(y_test, y_pred_logreg)
```

```
from xgboost import XGBClassifier
```

```
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)
r2_score(y_test, y_pred_xgb)
```

```
from sklearn.svm import SVC
```

```
svc = SVC()
svc.fit(X_train, y_train)
y_pred_svc = svc.predict(X_test)
r2_score(y_test, y_pred_svc)
```

```
test = np.array([[1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 32, 1, 9, 0, 0, 0, 6, 4]])
logreg.predict(test), xgb.predict(test), svc.predict(test)
```


SOFTWARE ENVIRONMENT

7. SOFTWARE ENVIRONMENT

MACHINE LEARNING:

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Challenges in Machines Learning:-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of over fitting & under fitting – If the model is over fitting or under fitting, it cannot be represented well for the problem.

Curse of dimensionality – another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment – Complexity of the ML model makes it quite difficult to be deployed in real life.

DEEP LEARNING

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on

artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.

What is Anaconda for Python?

Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, and Linux.

People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.

What is Anaconda for Python?

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks. This is why it is perfect for beginners who want to learn Python.

Now that you know what Anaconda Python is, let's look at how to install it.

How to install Anaconda for Python?



To install Anaconda, just head to the [Anaconda Documentation](#) website and follow the instructions to download the installer for your operating system. Once the installer successfully downloads, double-click on it to start the installation process.

Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select "yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to find the files it needs.

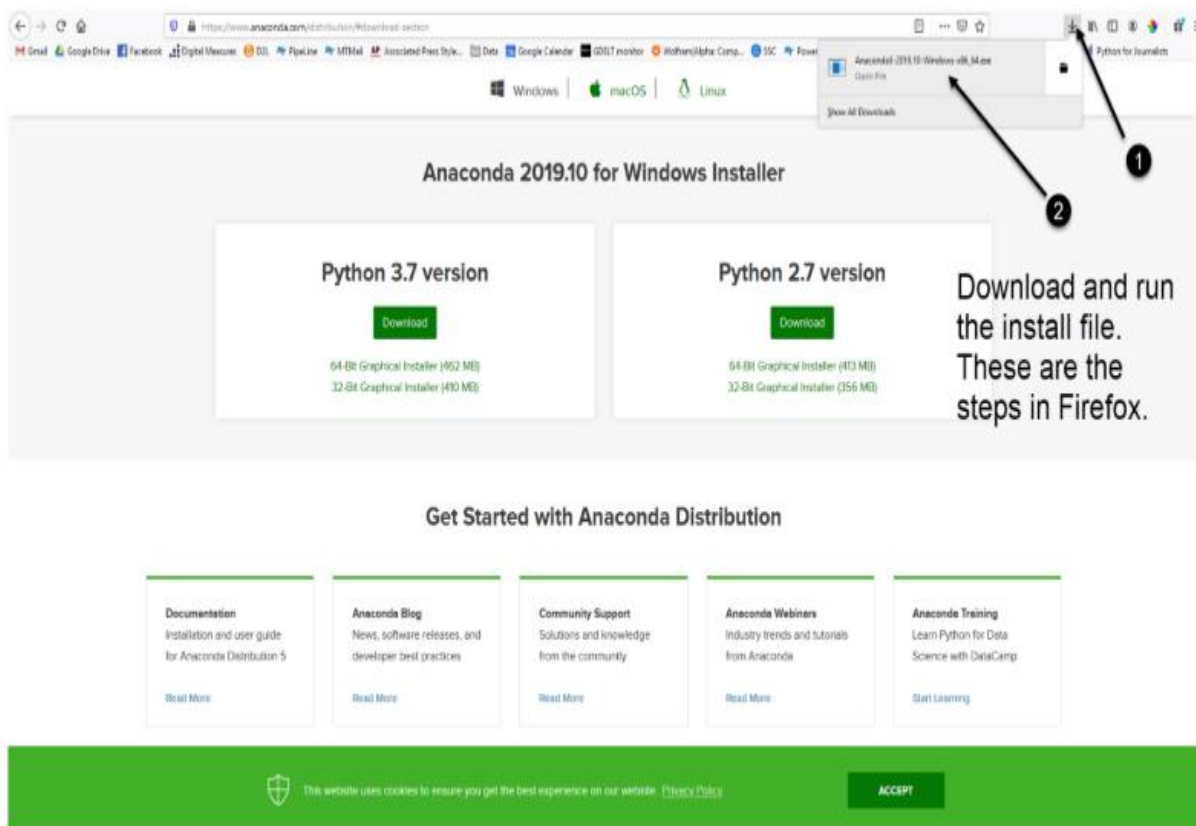
Once the installation is complete, you will be asked if you want to "enable Anaconda as my default Python." We recommend selecting "yes" to use Anaconda as your default Python interpreter.

Python Anaconda Installation

Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

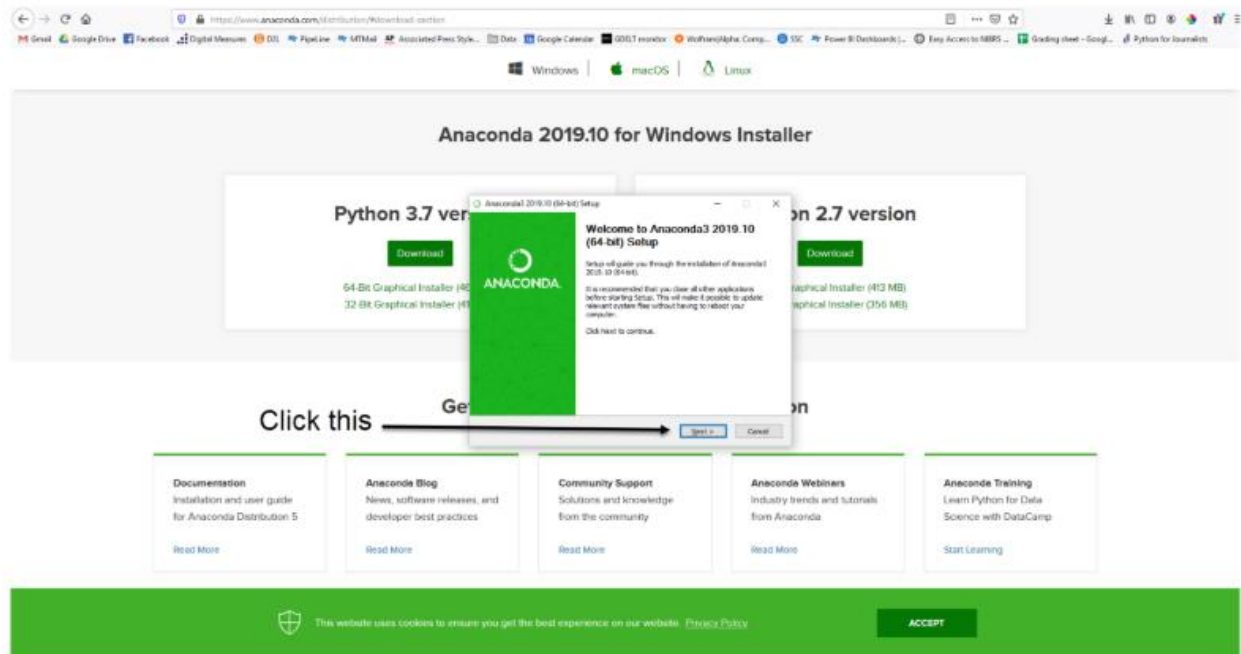
1. Go to this link and download Anaconda for Windows, Mac, or Linux: –

[Download anaconda](https://www.anaconda.com/distribution/#download-section)

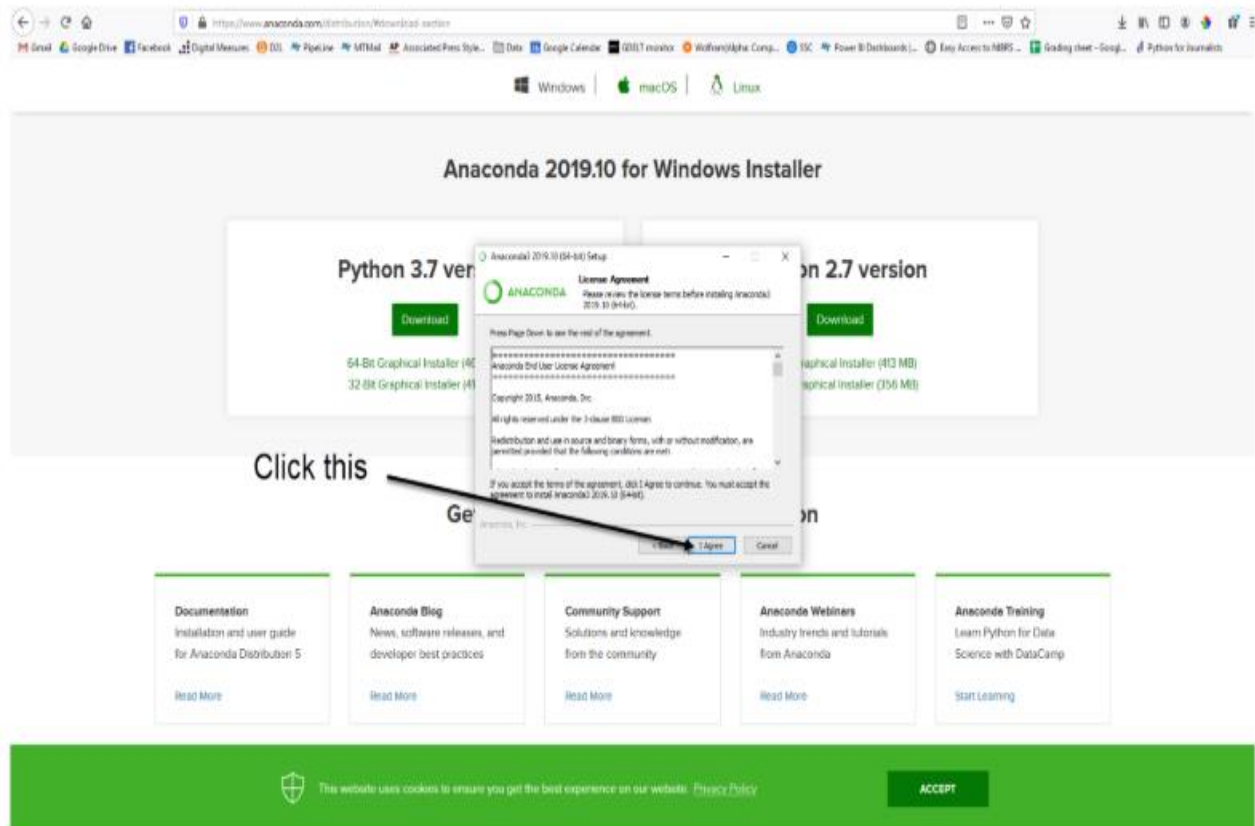


You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

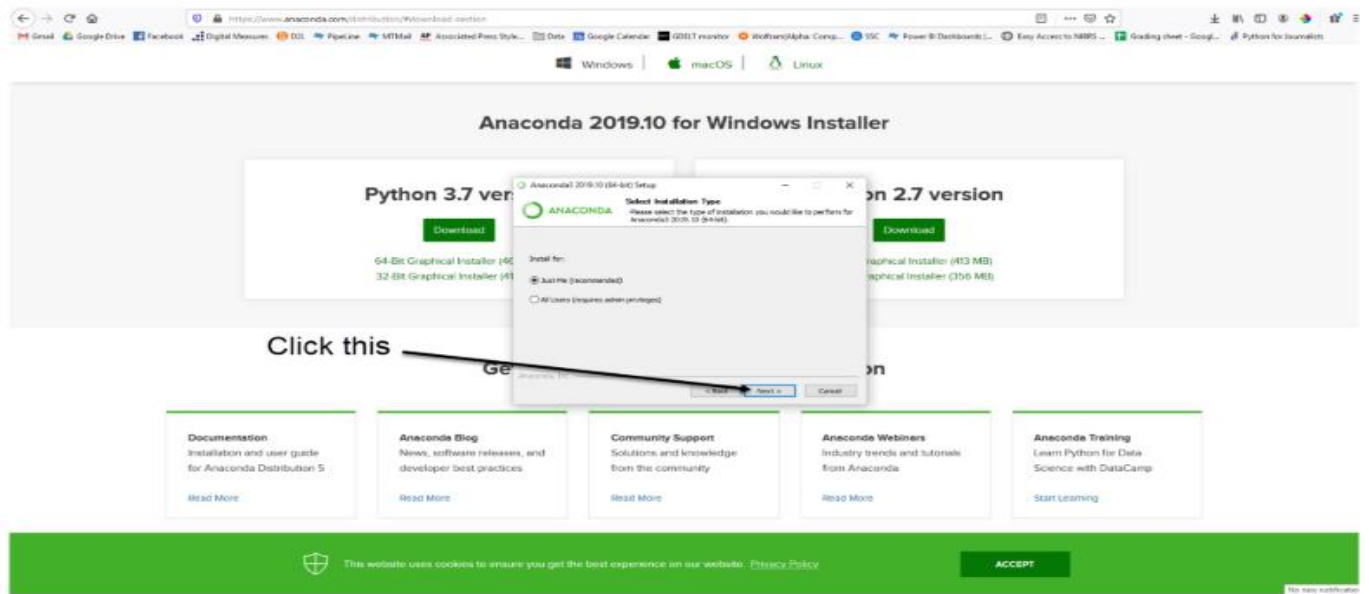
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



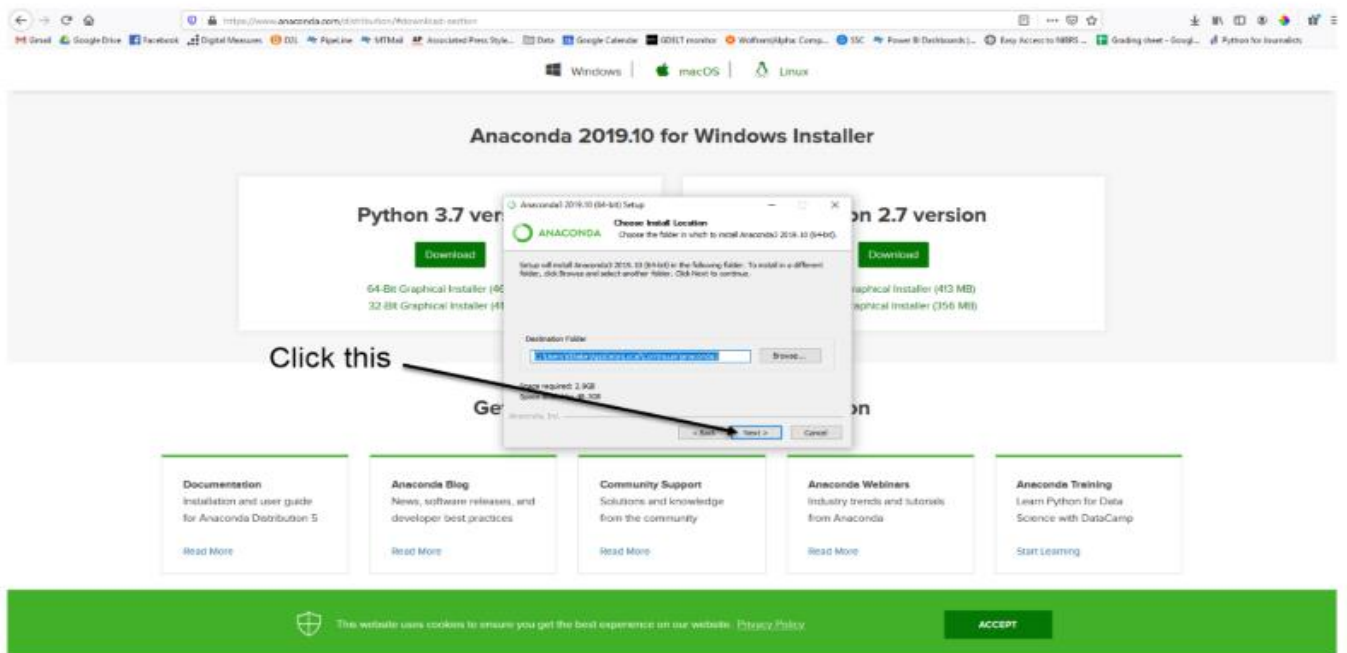
3. Now, you'll see the license agreement. Click on 'I Agree'.



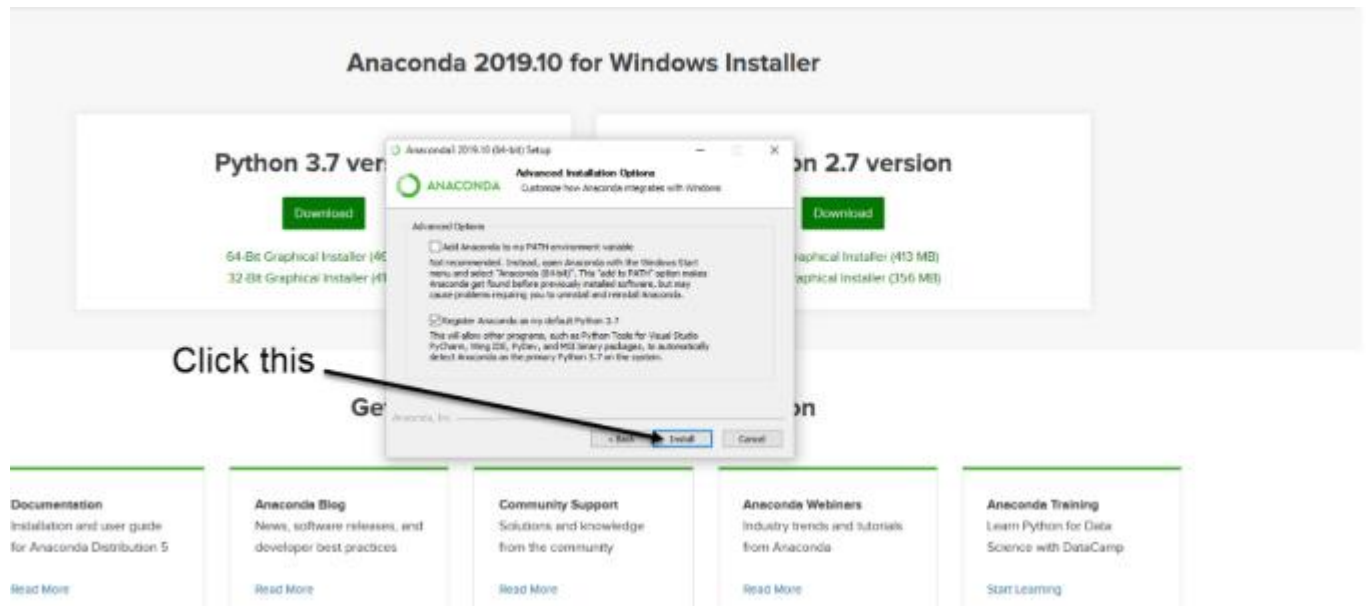
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



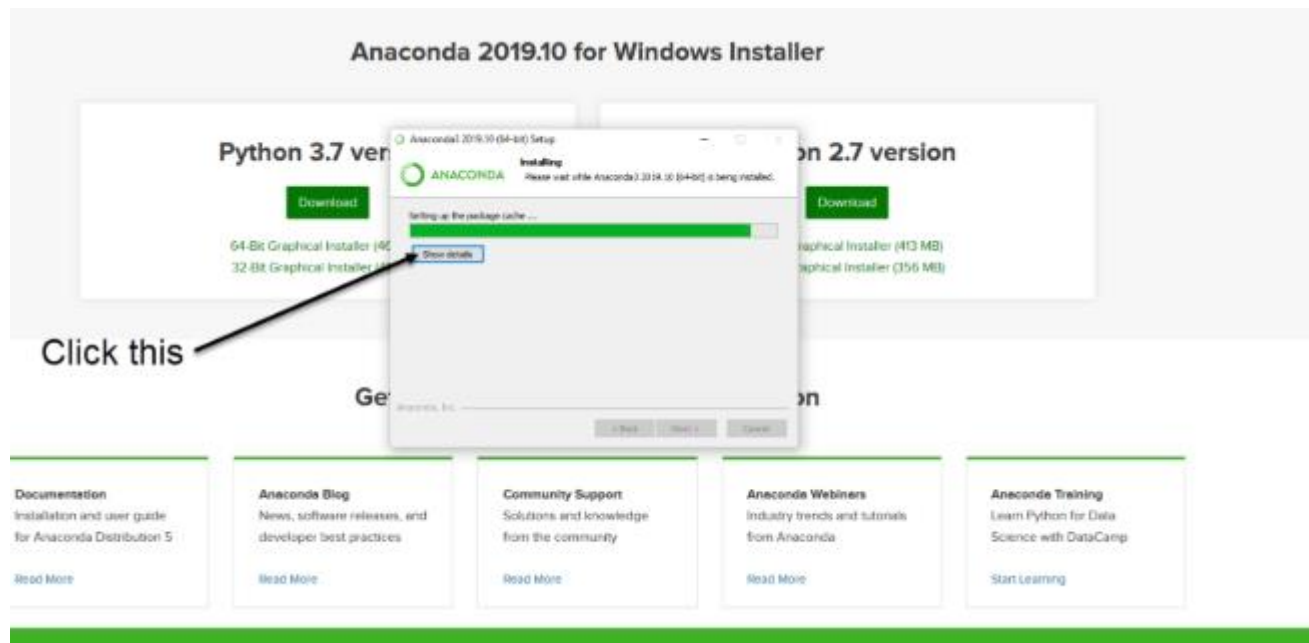
5. Choose where you want to install it. Here, you can see the available space and how much you need.



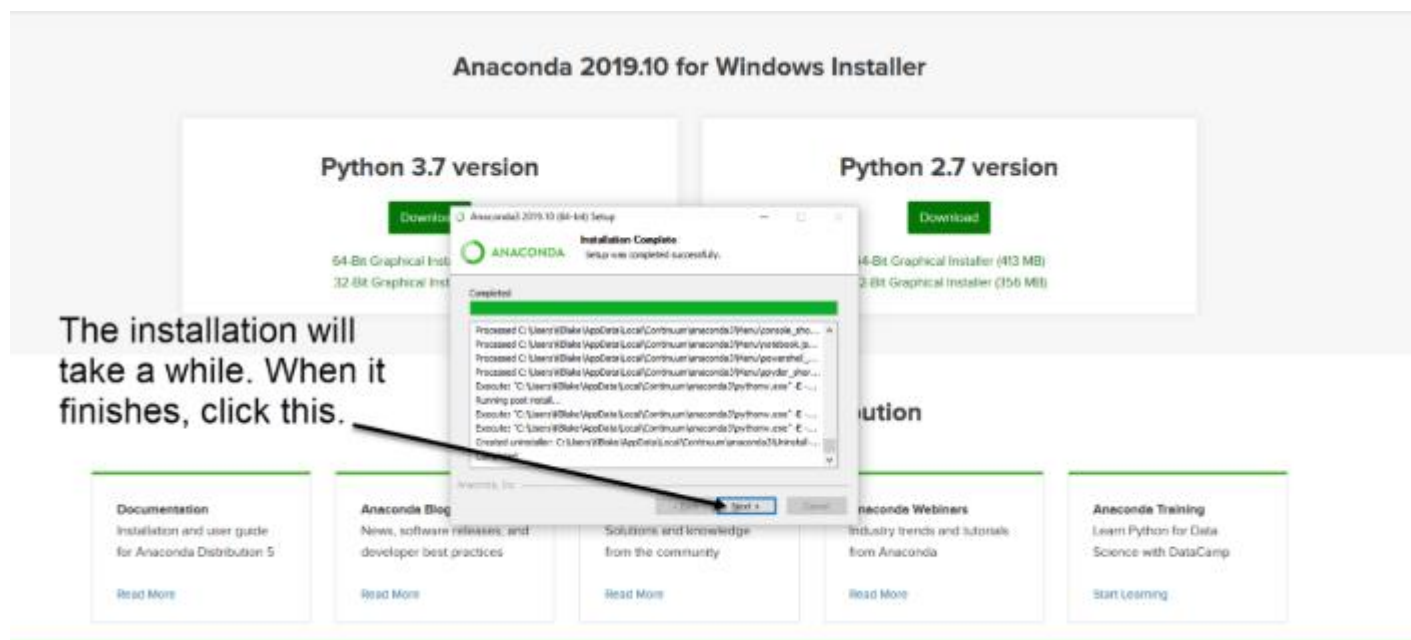
6. Now, you'll get some advanced options. You can add Anaconda to your system's PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on 'Install'.



7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



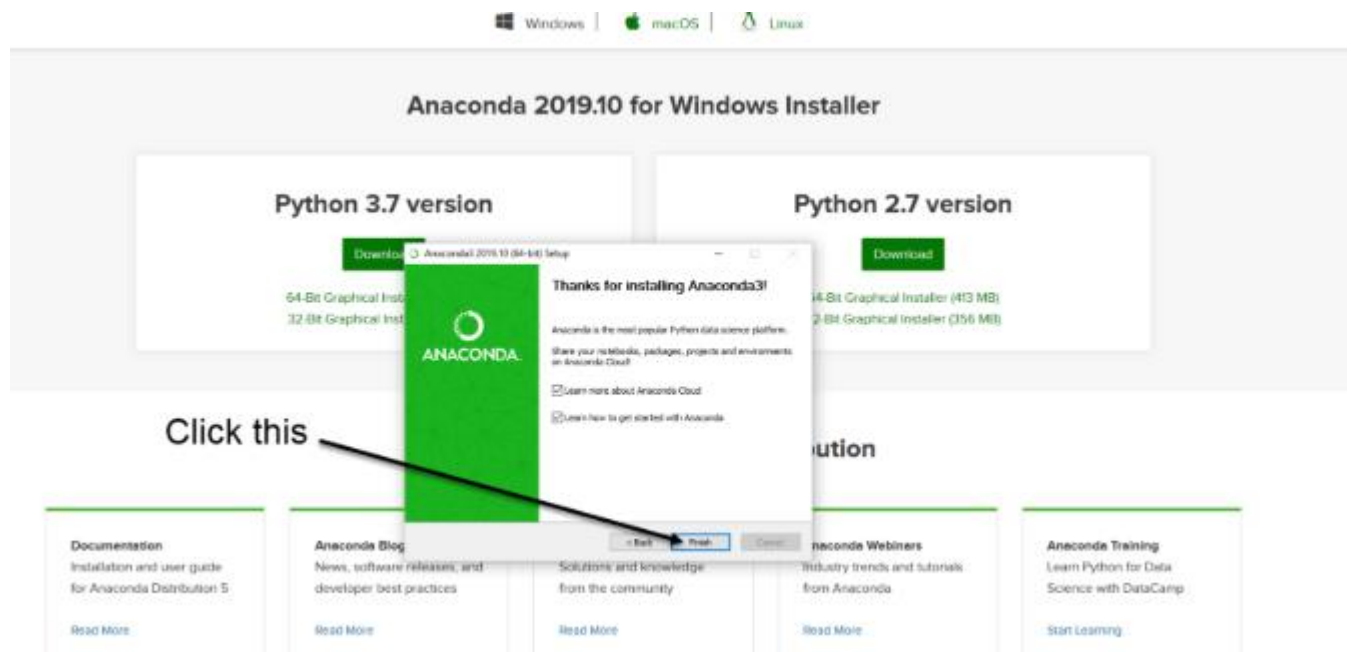
8. The installation is complete. Click Next.



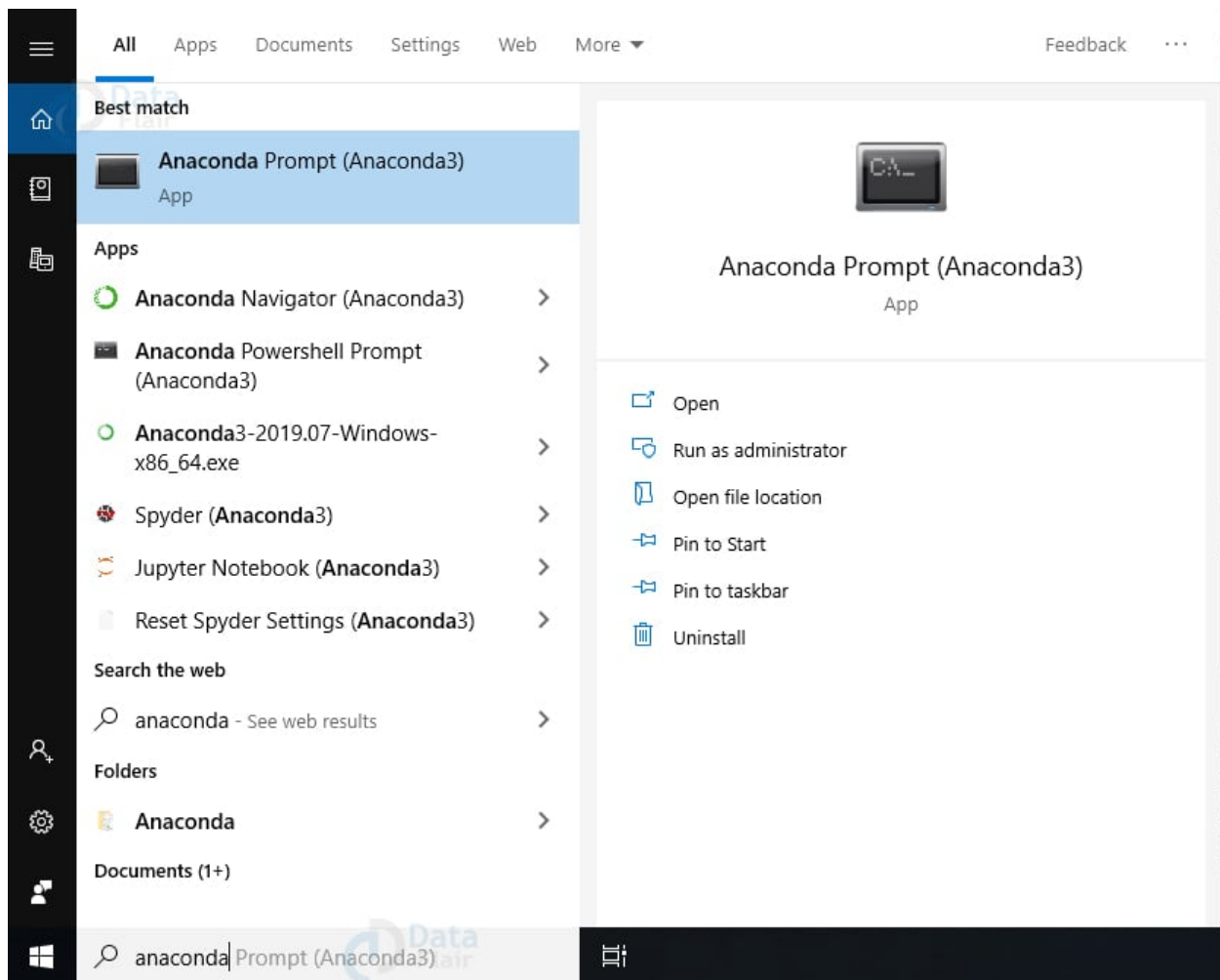
9. This screen will inform you about PyCharm. Click Next.



10. The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.



11. If you search for Anaconda now, you will see the following options:



PYTHON LANGUAGE:

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form

without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language. For example, `x = 10` Here, x can be anything such as String, int, etc.

Features in Python:

There are many features in Python, some of which are discussed below as follows:

1. Free and Open Source

Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that

source code is also available to the public. So you can download it, use it as well as share it.

2. Easy to code

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python's syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

4. Object-Oriented Language

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

5. GUI Programming Support

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

6. High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

7. Extensible feature

Python is an Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.

9. Python is a Portable language

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

10. Python is an integrated language

Python is also an integrated language because we can easily integrate Python with other languages like C, C++, etc.

11. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called byte code.

12. Large Standard Library

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

13. Dynamically Typed Language

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

14. Frontend and backend development

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like JavaScript. Backend is the strong forte of Python it's extensively used for this work cause of its frameworks like Django and Flask.

15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write `int y = 18` if the integer value 15 is set to y. You may just type `y=18`.

LIBRARIES/PACKGES:-

Tensor flow

Tensor Flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full

control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

SYSTEM TESTING

8. SYSTEM TESTING

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

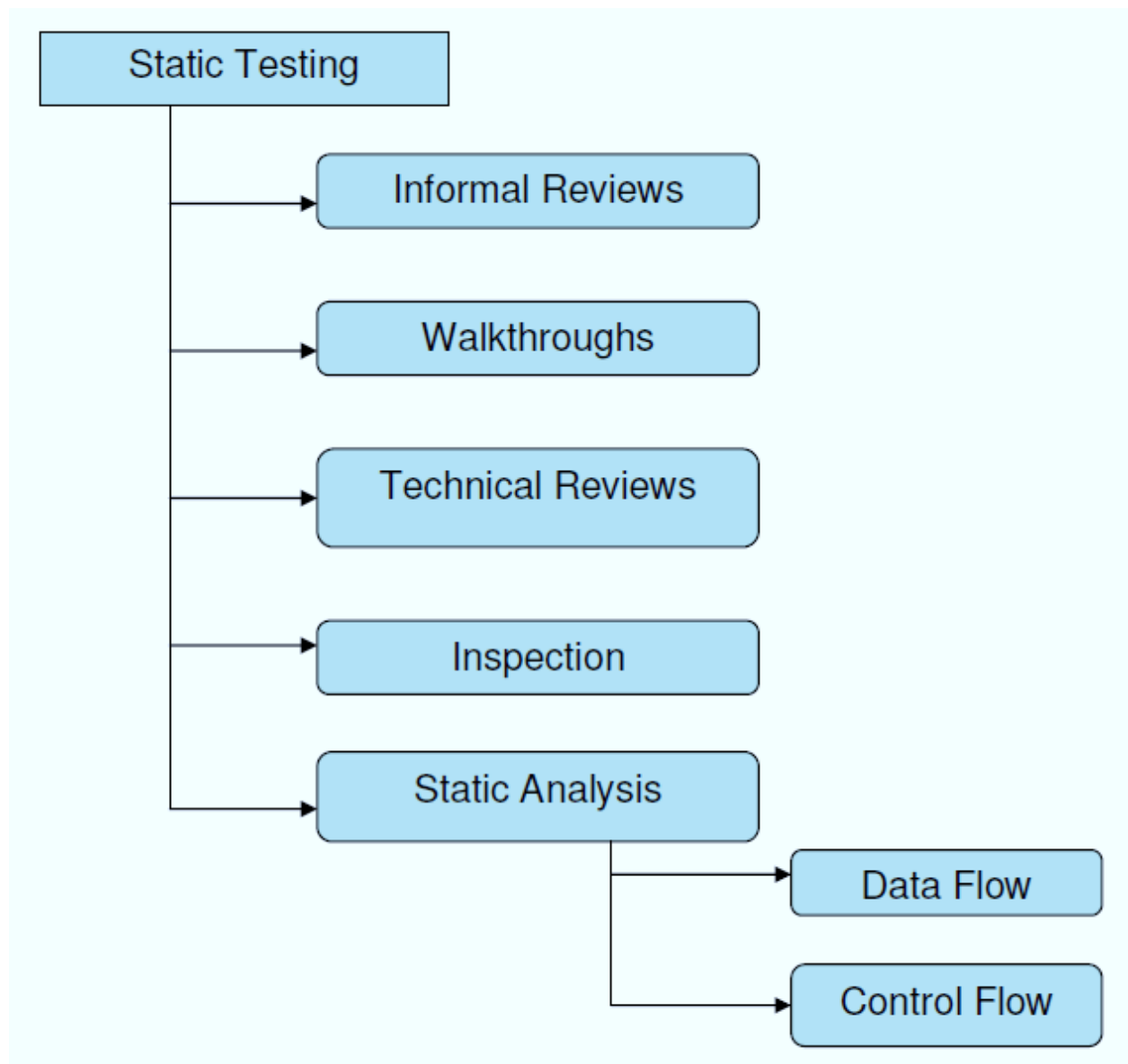
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

8.1 Software Testing Strategies:

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

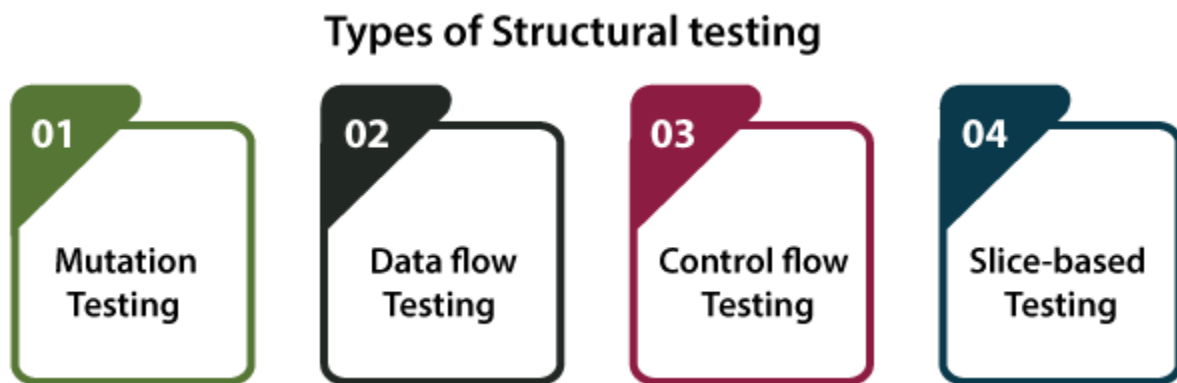
Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



Structural Testing:

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).

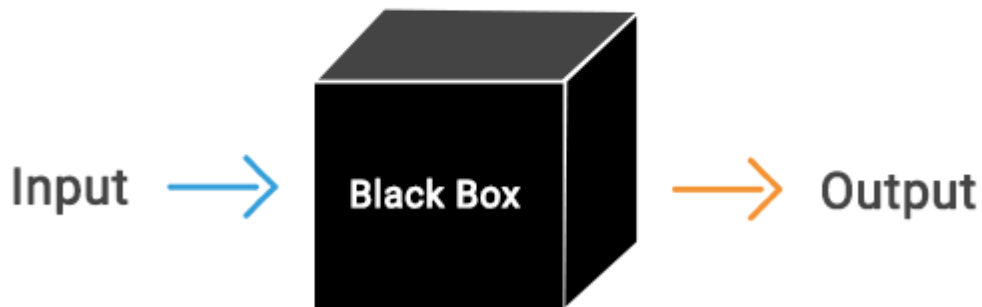


Behavioral Testing:

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as

regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.

Black Box Testing



8.2 TEST CASES:

S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User sign in	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

SCREENS

7. SCREENSHOTS

ML Model	Accuracy	Precision	Recall	F1-Score
AdaBoost	0.966	1.00	0.909	0.952
Random Forest	0.915	0.95	0.826	0.884
Decision Tree	0.780	0.75	0.652	0.698
KNN	0.678	0.95	0.514	0.667
Naive Bayes	0.610	0.45	0.429	0.439
Logistic Regression	0.576	0.65	0.419	0.510
SVC	0.661	0.25	0.500	0.333
LDA	0.797	0.70	0.700	0.700
Voting Classifier	0.864	0.90	0.750	0.818

Table 9.1 Performance Evaluation – Toddler Dataset

ML Model	Accuracy	Precision	Recall	F1-Score
AdaBoost	0.995	0.993	1.000	0.996
Random Forest	0.995	1.000	0.993	0.996
Decision Tree	0.991	0.993	0.993	0.993
KNN	0.981	0.979	0.993	0.986
Naive Bayes	0.905	0.901	0.955	0.928
Logistic Regression	0.953	1.000	0.934	0.966
SVC	0.967	0.958	0.993	0.975
LDA	0.976	0.986	0.979	0.982
Voting Classifier	0.991	0.993	0.993	0.993

Table 9.2 Performance Evaluation – Children Dataset

ML Model	Accuracy	Precision	Recall	F1-Score
AdaBoost	1.000	1.00	1.000	1.000
Random Forest	1.000	1.00	1.000	1.000
Decision Tree	1.000	1.00	1.000	1.000
KNN	0.943	0.94	0.904	0.922
Naive Bayes	0.979	0.96	0.980	0.970
Logistic Regression	0.993	0.98	1.000	0.990
SVC	0.993	1.00	0.980	0.990
LDA	0.936	0.88	0.936	0.907
Voting Classifier	1.000	1.00	1.000	1.000

Table 9.3 Performance Evaluation – Adults Dataset

ML Model	Accuracy	Precision	Recall	F1-Score
AdaBoost	0.810	0.769	0.909	0.833
Random Forest	0.857	0.846	0.917	0.880
Decision Tree	0.810	0.769	0.909	0.833
KNN	0.857	1.000	0.812	0.897
Naive Bayes	0.762	0.769	0.833	0.800
Logistic Regression	1.000	1.000	1.000	1.000
SVC	1.000	1.000	1.000	1.000
LDA	0.857	0.846	0.917	0.880
Voting Classifier	0.952	0.923	1.000	0.960

Table 9.4 Performance Evaluation – Adolescent Dataset

SCREENS:

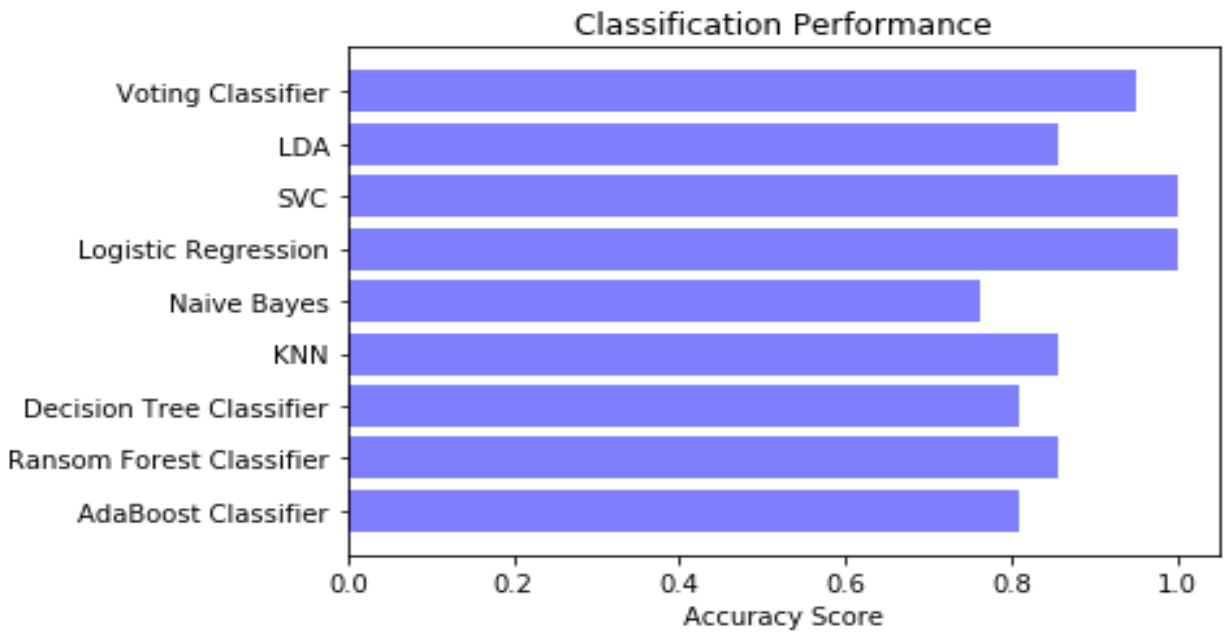


Fig 9.1 Accuracy Comparison Graphs - Quantile Transformer

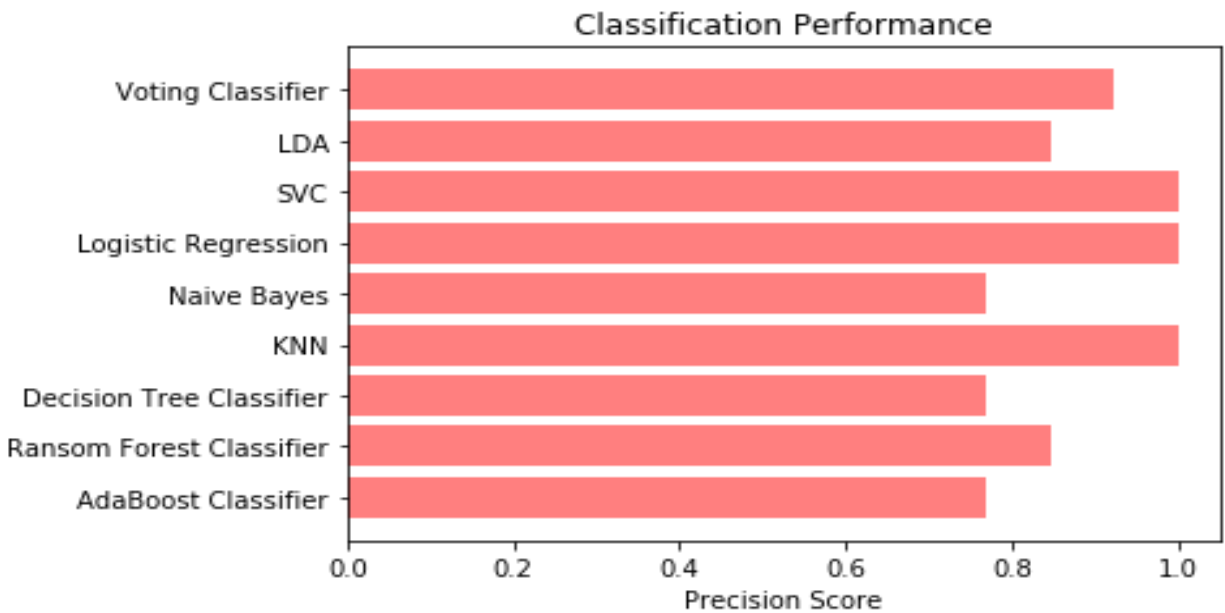


Fig 9.2 Precision Comparison Graphs - Quantile Transformer

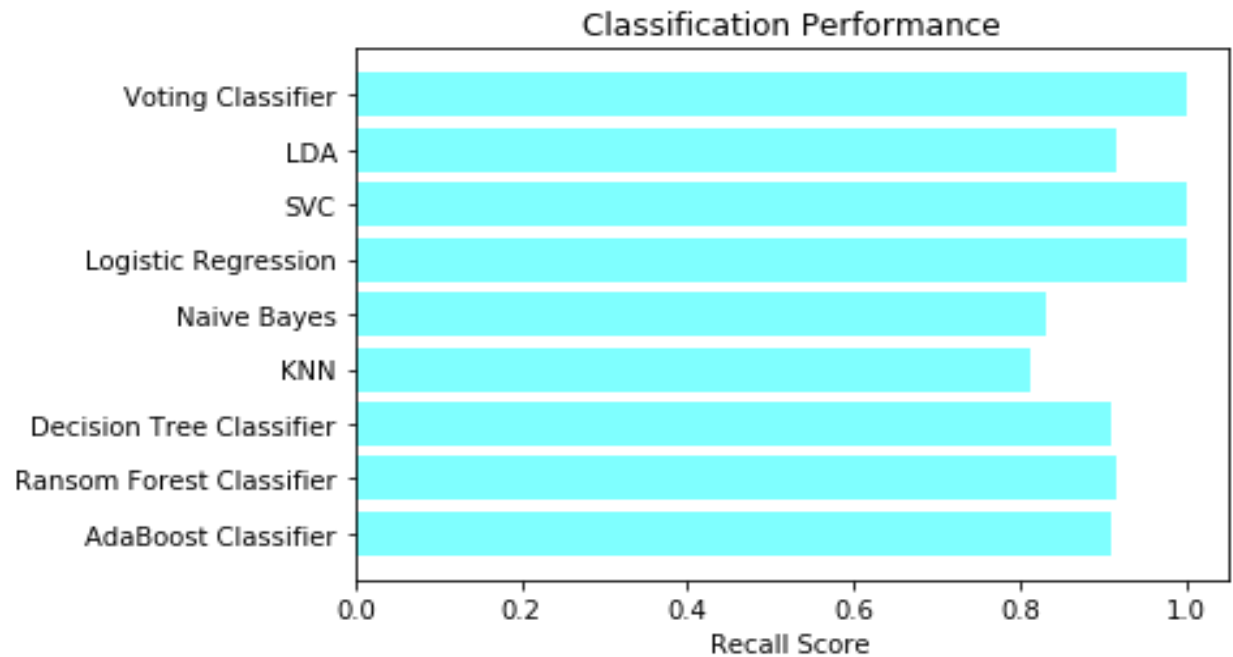


Fig 9.3 Recall Comparison Graphs - Quantile Transformer

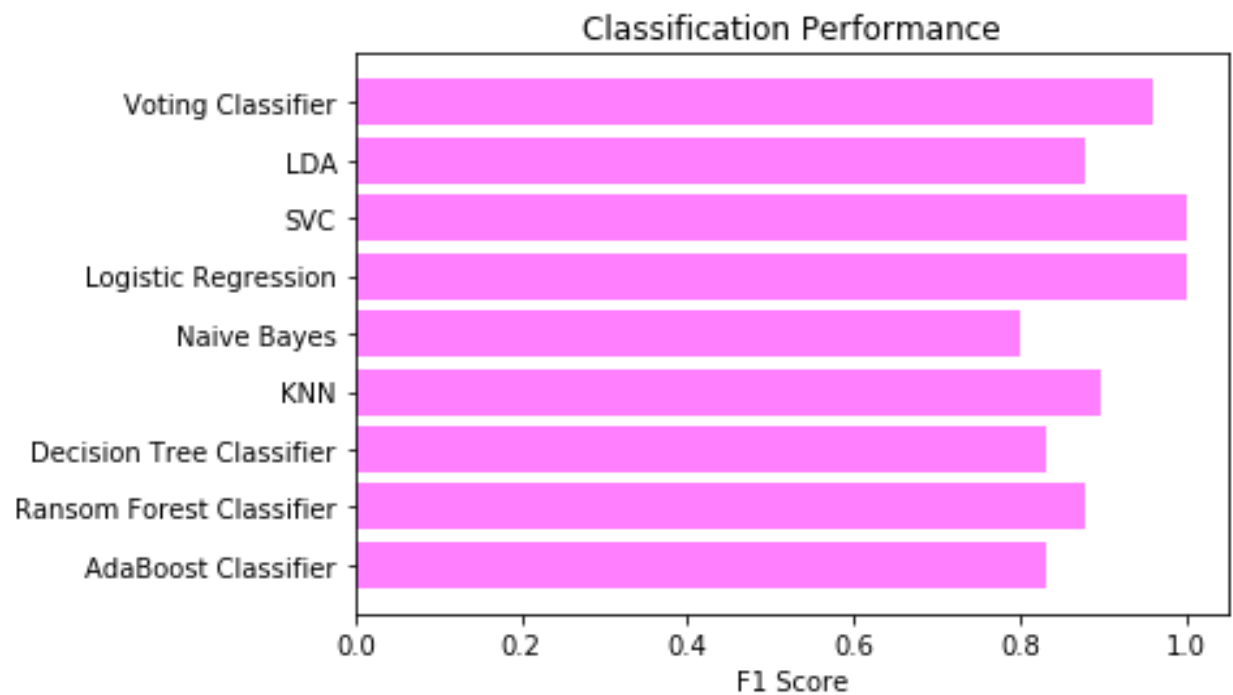


Fig 9.4 F1 Score Comparison Graphs - Quantile Transformer

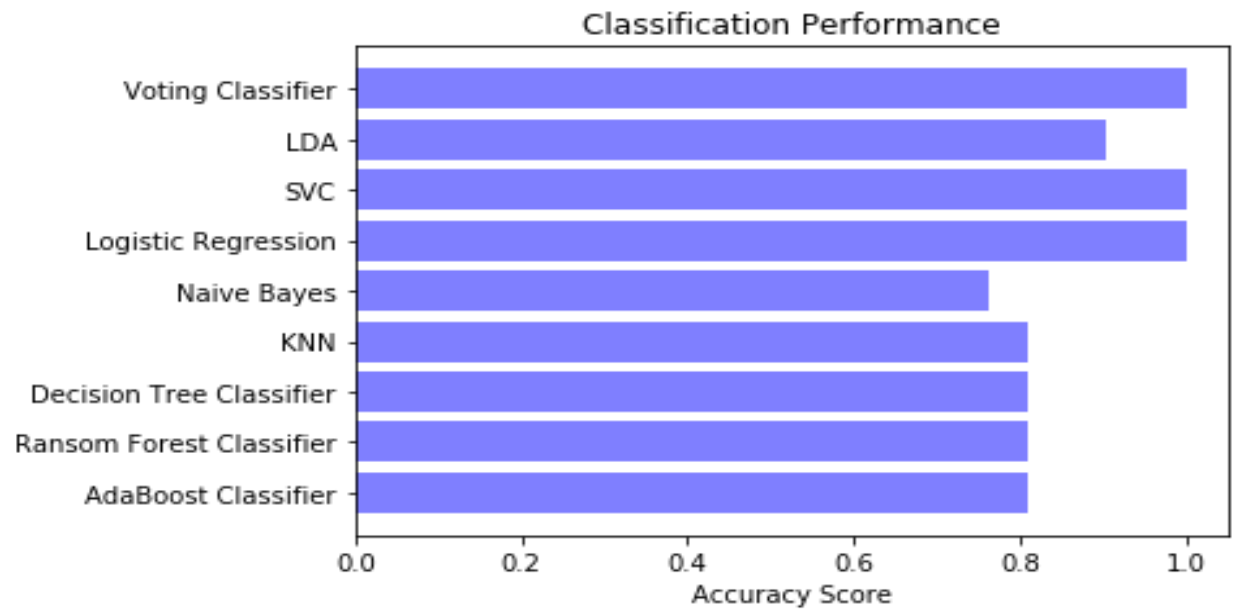


Fig 9.5 Accuracy Comparison Graphs - Power Transformer

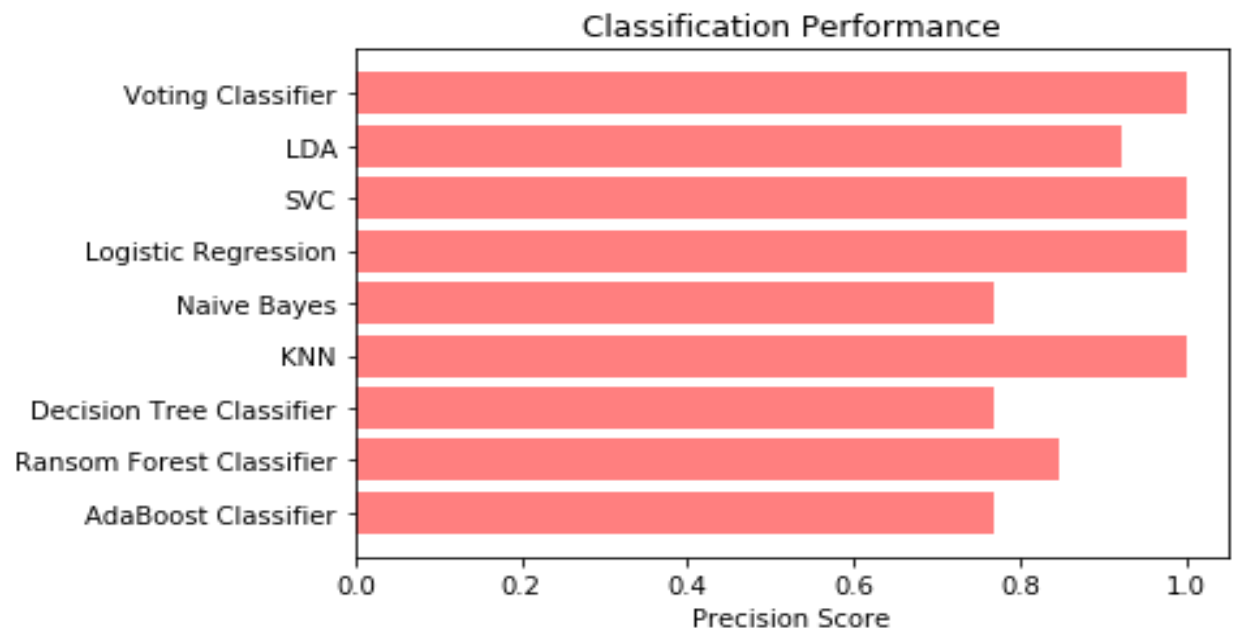


Fig 9.6 Precision Comparison Graphs - Power Transformer

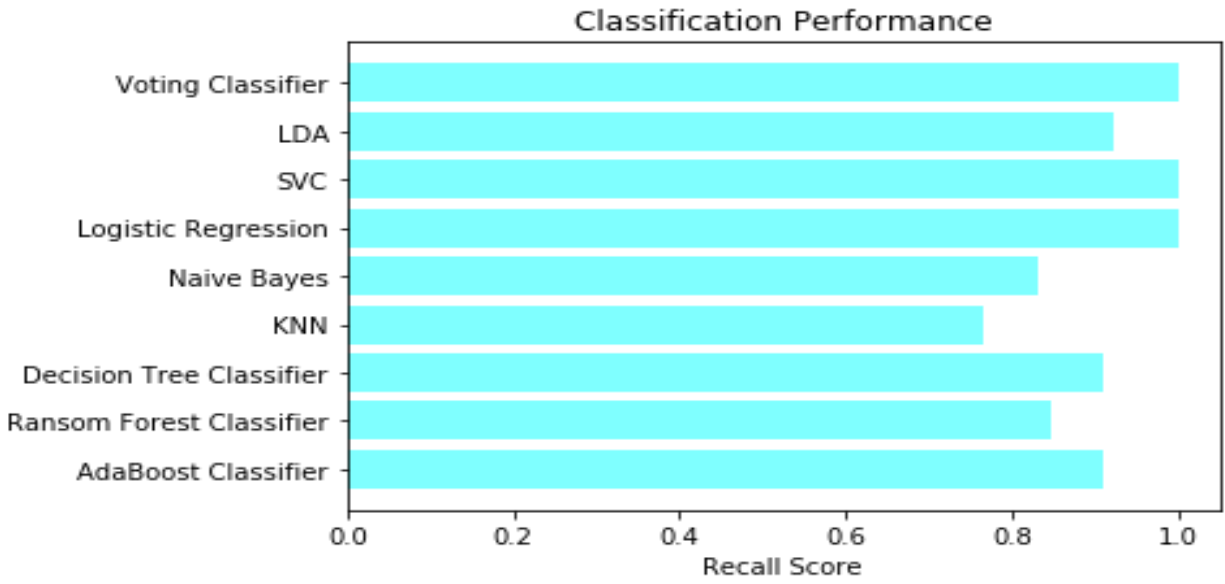


Fig 9.7 Recall Comparison Graphs - Power Transformer

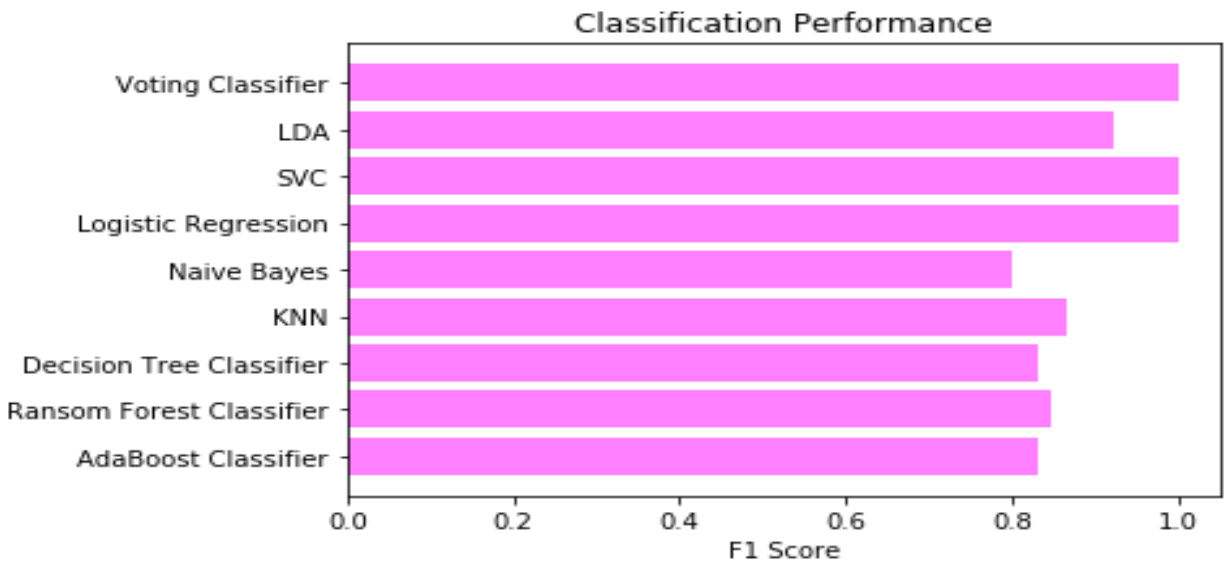


Fig 9.8 F1 Score Comparison Graphs - Power Transformer

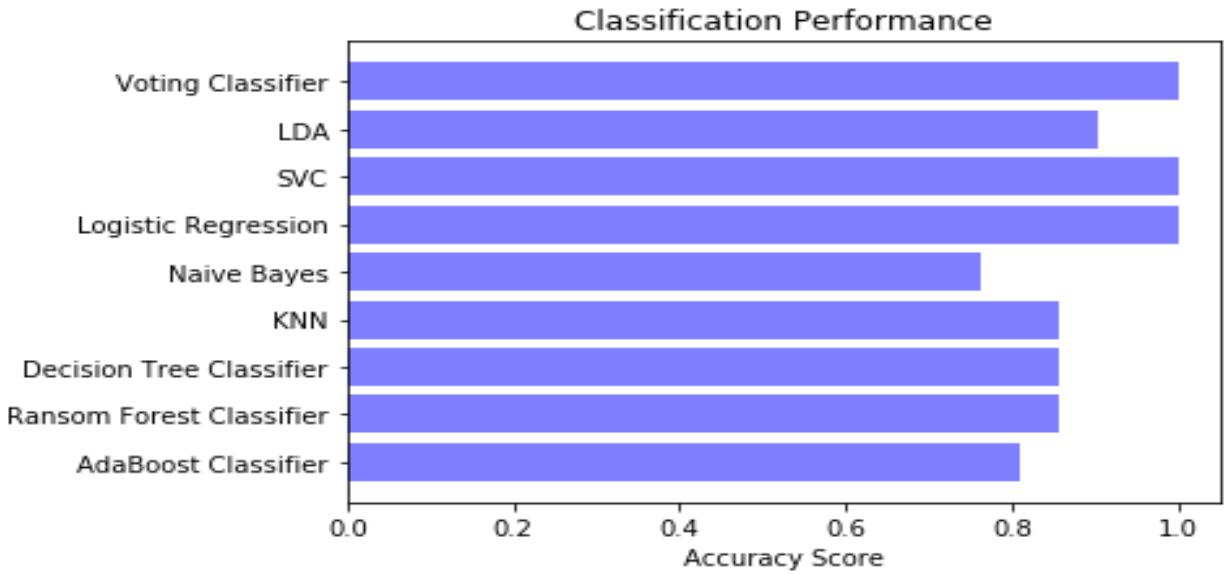


Fig 9.9 Accuracy Comparison Graphs - MAXABSSCALER

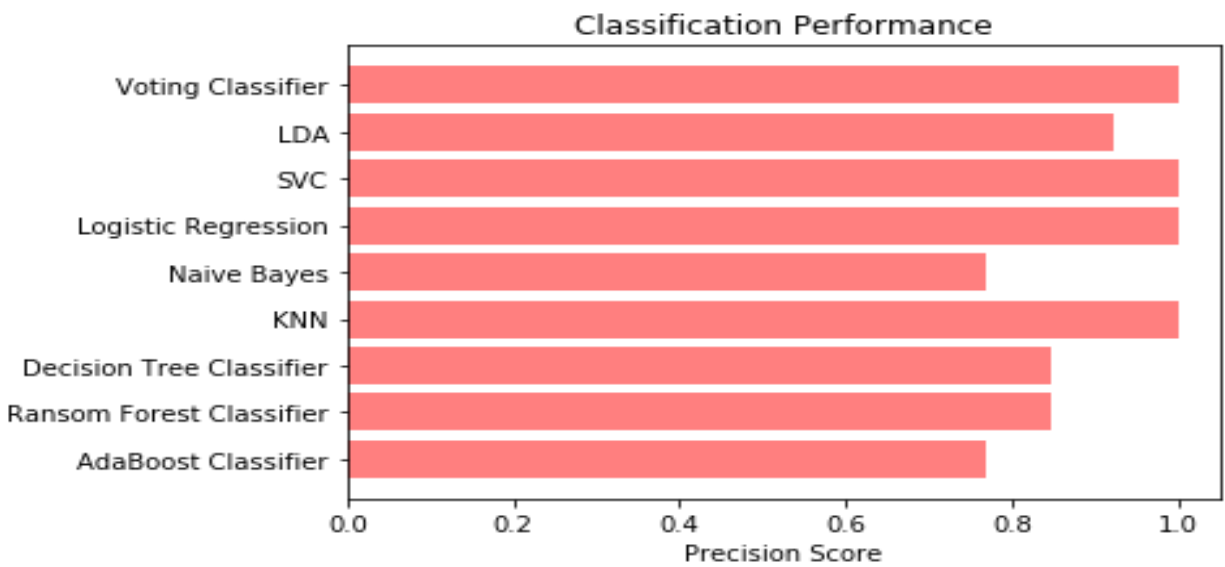


Fig 9.10 Precision Comparison Graphs - MAXABSSCALER

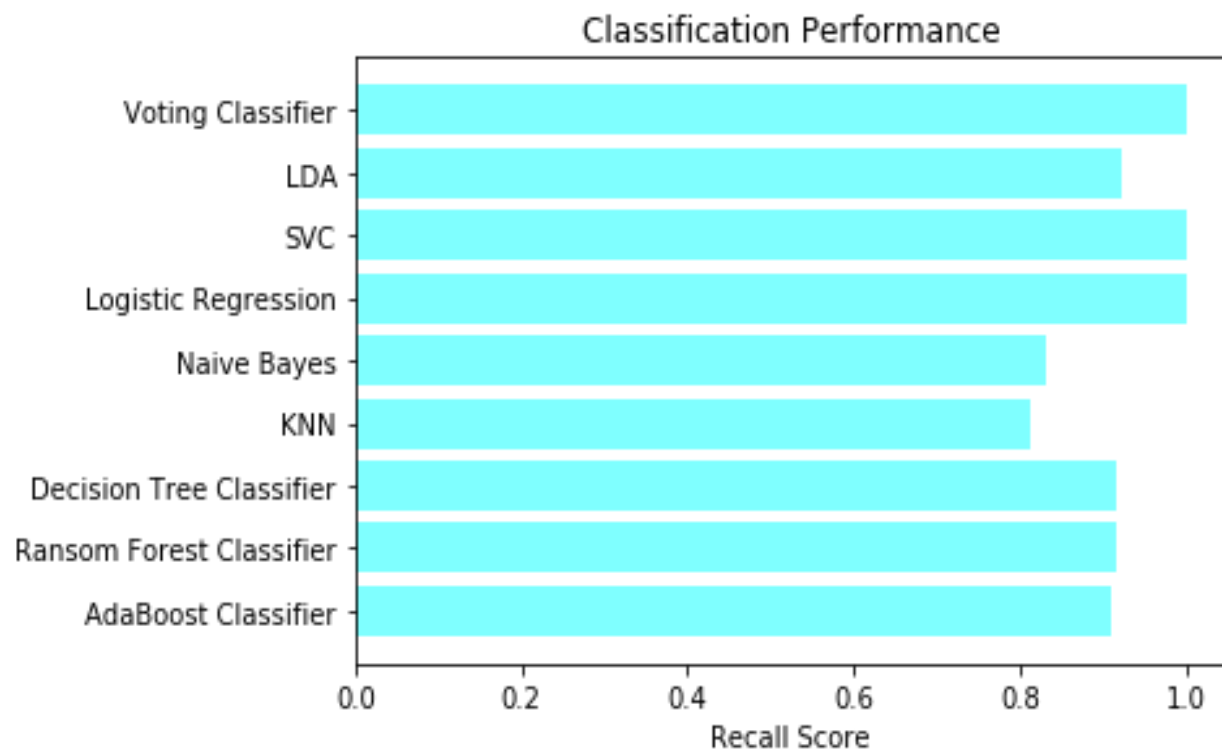


Fig 9.11 Recall Comparison Graphs - MAXABSSCALER

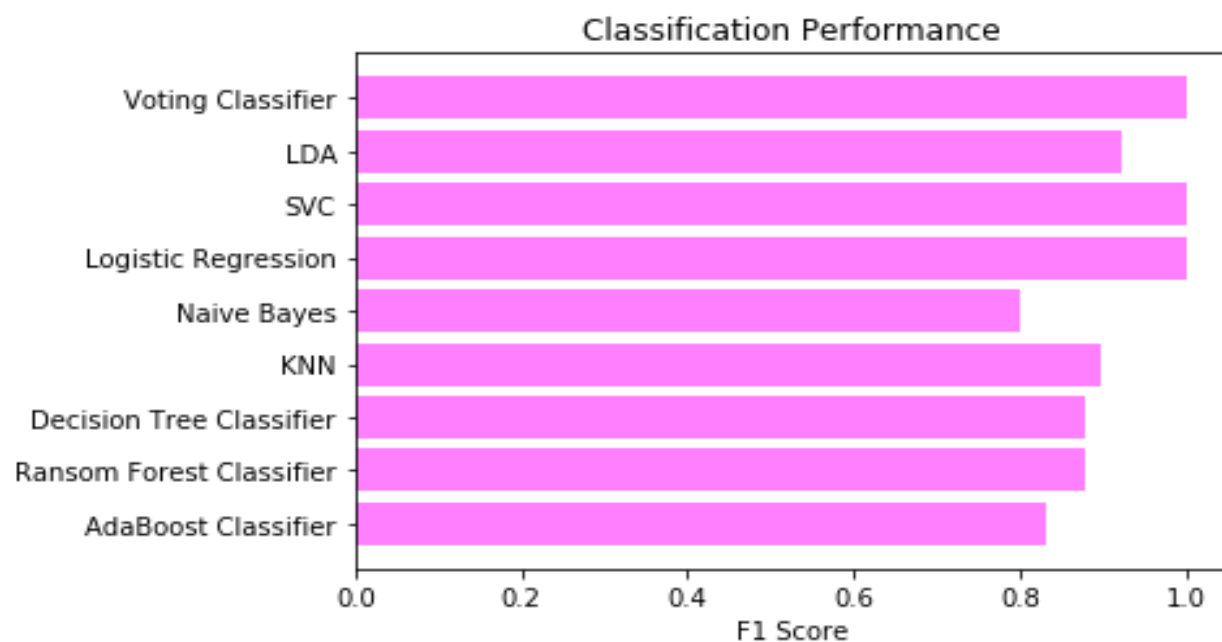


Fig 9.12 F1 Score Comparison Graphs - MAXABSSCALER

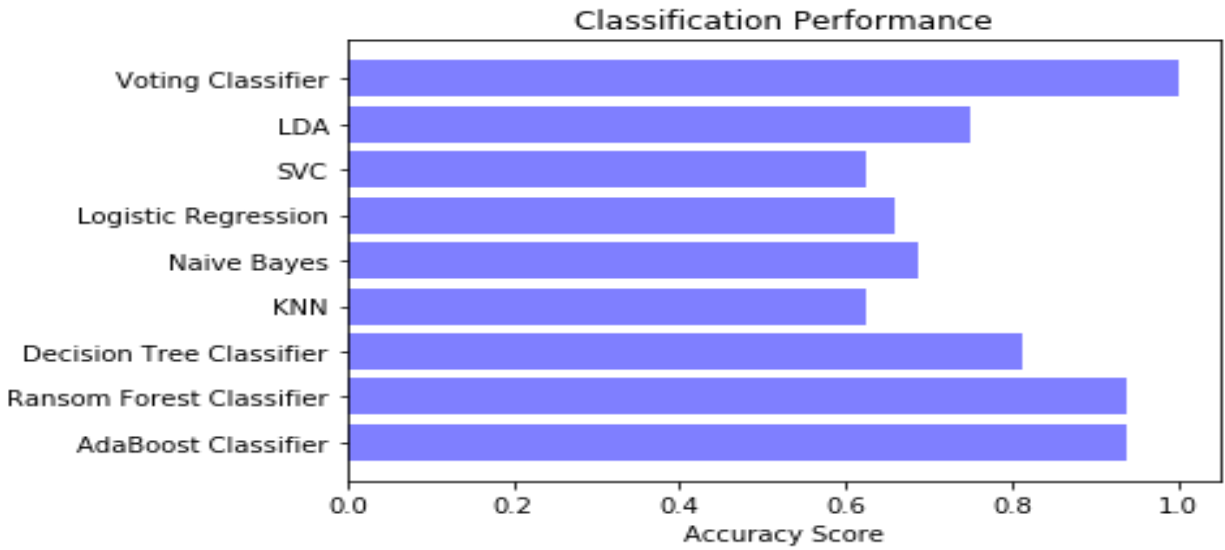


Fig 9.13 Accuracy Comparison Graphs - Normalizer

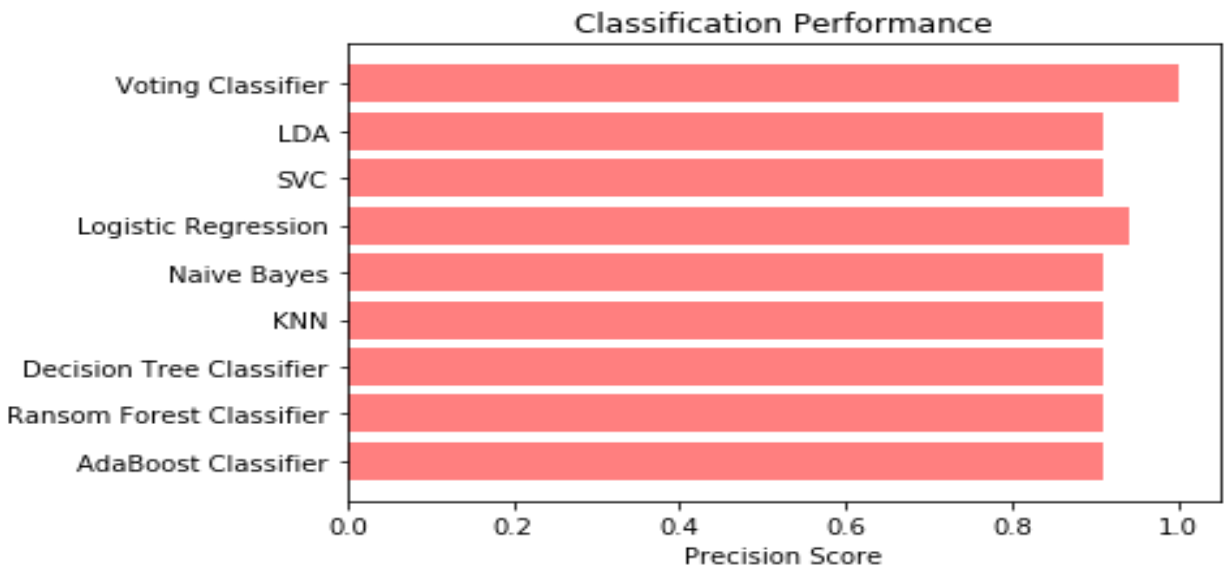


Fig 9.14 Precision Comparison Graphs - Normalizer

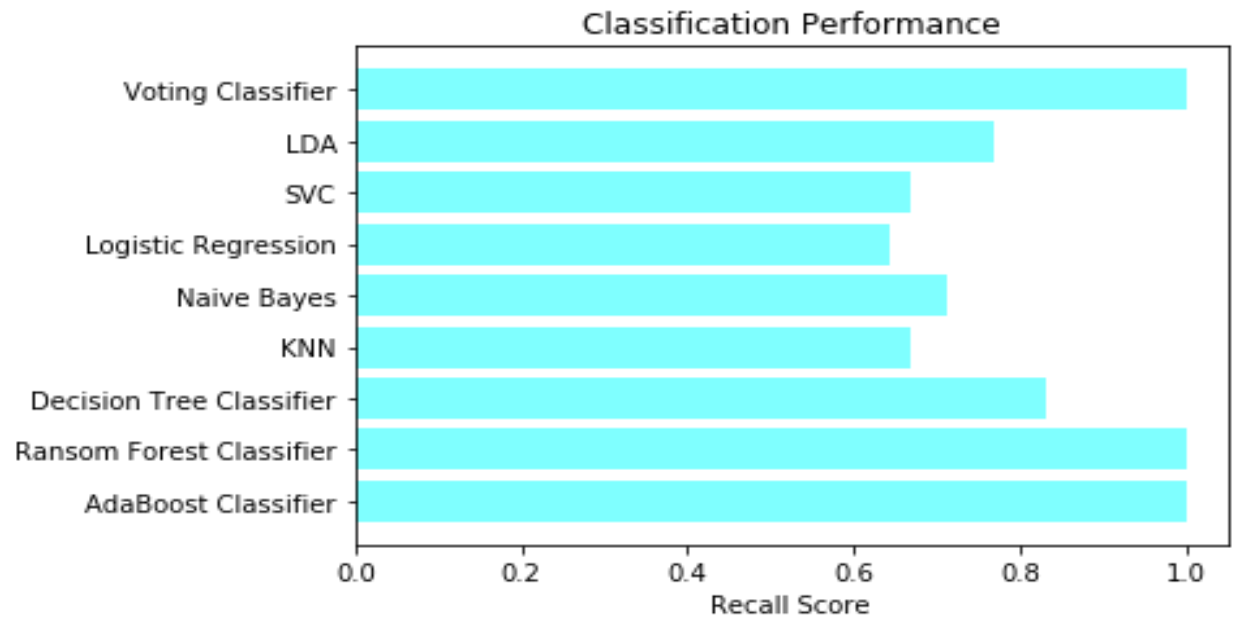


Fig 9.15 Recall Comparison Graphs - Normalizer

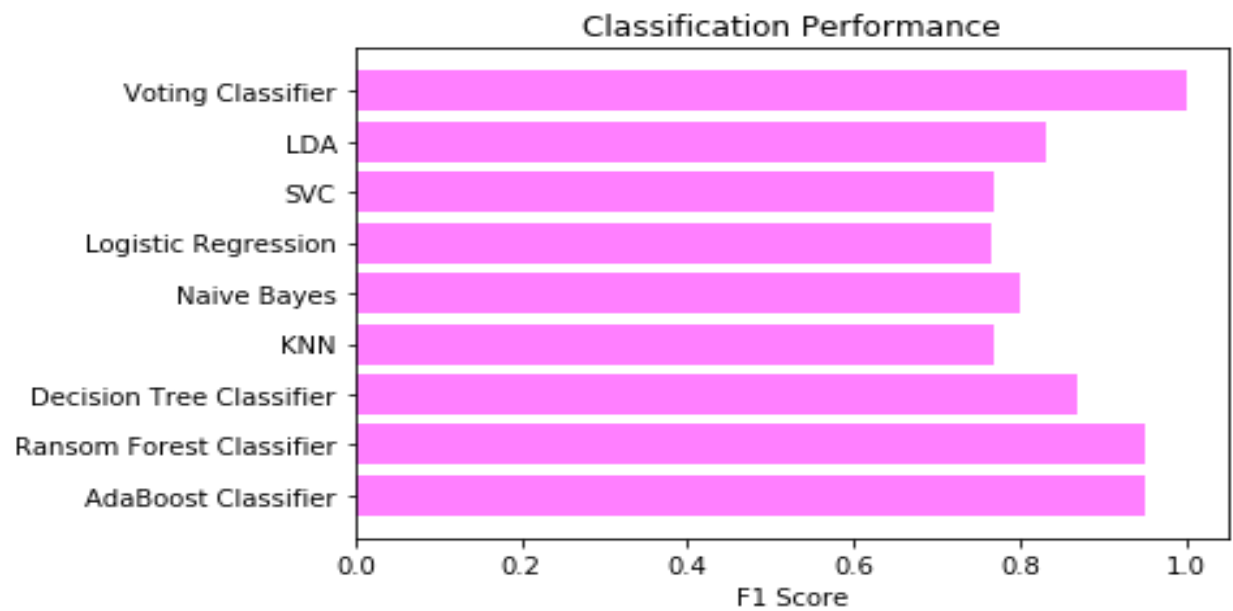


Fig 9.16 F1 Score Comparison Graphs - Normalizer

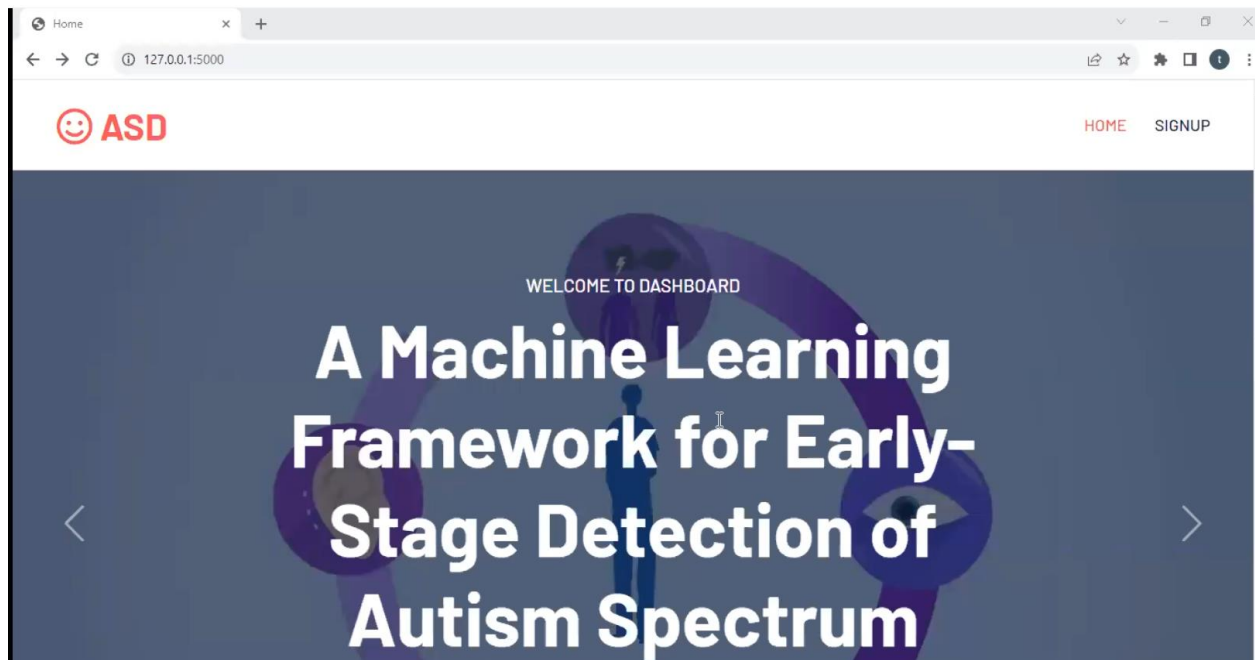


Fig 9.17 Home Page

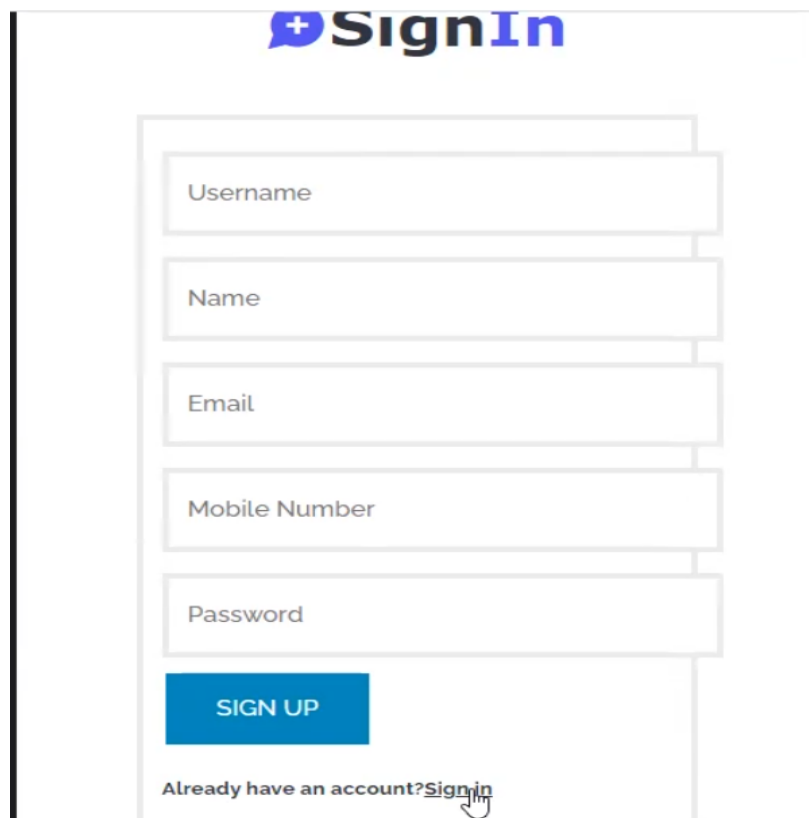
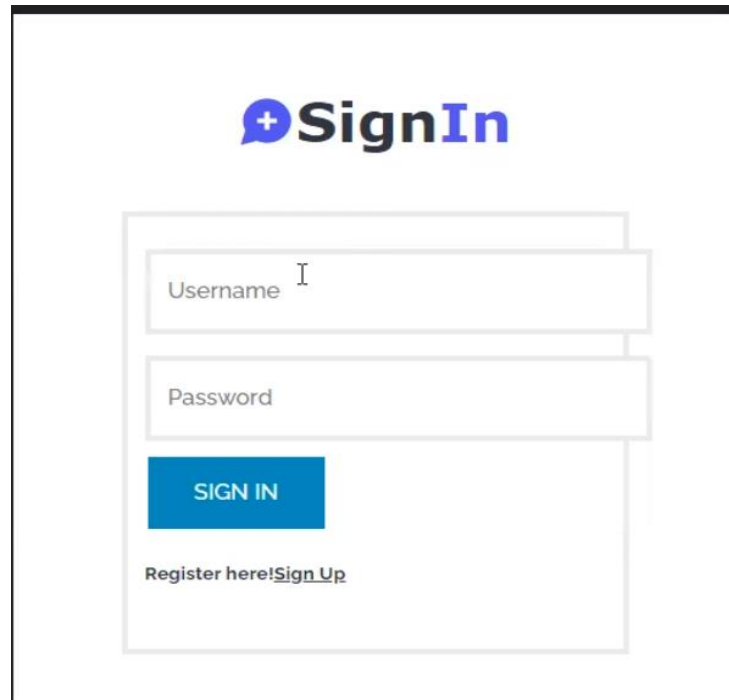
A screenshot of a web page titled 'Sign In' with a blue plus icon. The page contains a registration form with five input fields: 'Username', 'Name', 'Email', 'Mobile Number', and 'Password'. Below these fields is a blue button labeled 'SIGN UP'. At the bottom, there is a link that says 'Already have an account? Sign in', with a mouse cursor hovering over the 'Sign in' text.

Fig 9.18 Registration Page



The image shows a login page with a white background. At the top center is a logo consisting of a blue circle with a white plus sign, followed by the text "SignIn" in a bold, blue, sans-serif font. Below the logo is a light gray rectangular box containing the login form. Inside this box, there are two white input fields: the top one is labeled "Username" and the bottom one is labeled "Password". Below these fields is a blue rectangular button with the text "SIGN IN" in white, uppercase letters. At the bottom of the gray box, there is a link that says "Register here!Sign Up" in a small, gray font.

Fig 9.19 Login Page

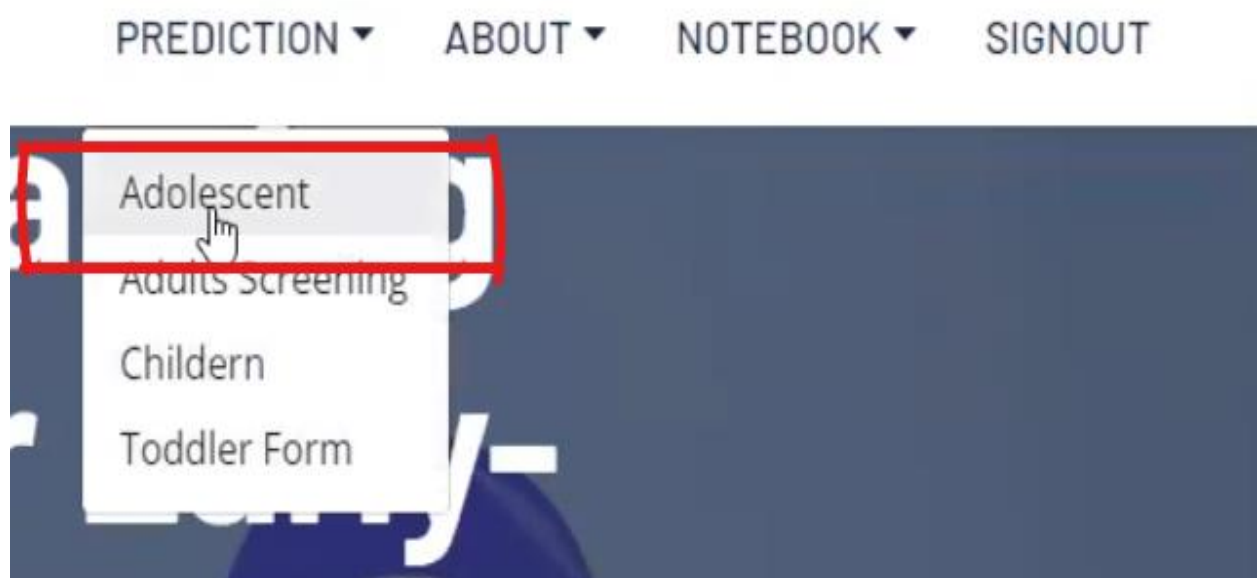


Fig 9.20 Click on Adolescent

A1: <input type="text" value="No"/>	
A2: <input type="text" value="No"/>	
A3: <input type="text" value="No"/>	Ethnicity <input type="text" value="Hispanic"/>
A4: <input type="text" value="No"/>	Jaundice <input type="text" value="Yes"/>
A5: <input type="text" value="Yes"/>	Austim <input type="text" value="Yes"/>
A6: <input type="text" value="Yes"/>	Country of Resdeince <input type="text" value="Austria"/>
A7: <input type="text" value="Yes"/>	Used App Before <input type="text" value="No"/>
A8: <input type="text" value="Yes"/>	Age Desc <input type="text" value="12-16 Years"/>
A9: <input type="text" value="Yes"/>	Relation <input type="text" value="Parent"/>
A10: <input type="text" value="Yes"/>	<input type="button" value="Predict"/>
Gender <input type="text" value="Male"/>	

Fig 9.21 Upload Input Data

Result: You have no ASD based on the input provide!

Fig 9.22 Output Screen

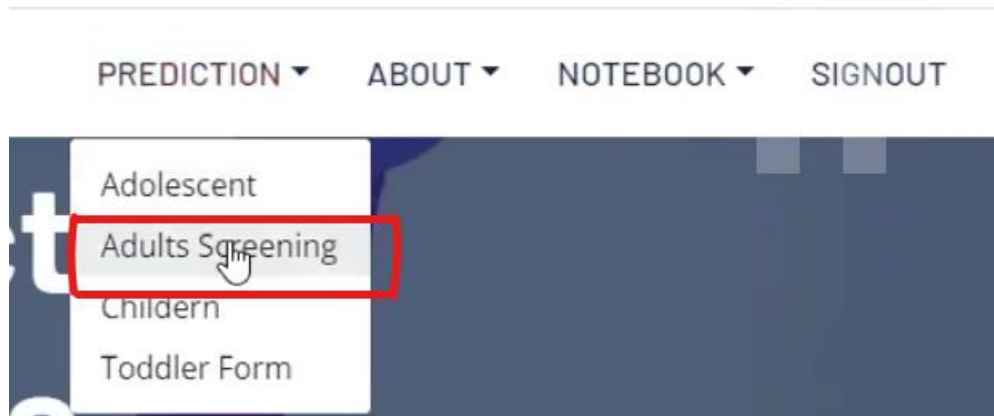


Fig 9.23 Click on Adults Screening

A1:	Yes	Ethnicity	White-European
A2:	Yes	Jaundice	No
A3:	Yes	Austim	No
A4:	Yes	Country of Resdeince	United States
A5:	No	Used App Before	No
A6:	No	Result:	25
A7:	Yes	Age Desc	18 and more
A8:	No	Relation	Self
A9:	No	Predict	

Fig 9.24 Upload Input Data

Result: You have ASD, based on the input provide!

Fig 9.25 Output Screen

Similarly we can choose other Datasets do the prediction based on the parameters

CONCLUSION

10. CONCLUSION

In this work, we proposed a machine-learning framework for ASD detection in people of different ages (Toddlers, Children, Adolescents, and Adults). We show that predictive models based on ML techniques are useful tools for this task. After completing the initial data processing, those ASD datasets were scaled using four different types of feature scaling (QT, PT, normalizer, MAS) techniques, classified using eight different ML classifiers (AB, RF, DT, KNN, GNB, LR, SVM, LDA). We then analyzed each feature scaled dataset's classification performance and identified the best-performing FS and classification approaches. We considered different statistical evaluation measures such as accuracy, ROC, F1-Score, precision, recall, Mathews correlation coefficient (MCC), kappa score, and Log loss to justify the experimental findings. Consequently, our proposed prediction models based on ML techniques can be utilized as an alternative or even a helpful tool for physicians to accurately identify ASD cases for people of different ages. Additionally, the feature importance values were calculated to identify the most prominent features for ASD prediction by employing four different FSTs (IGAE, GRAE, RFAE, and CAE). Therefore, the experimental analysis of this research will allow healthcare practitioners to take into account the most important features while screening ASD cases. In the future, we intend to collect more data related to ASD and construct a more generalized prediction model for people of any age to improve ASD detection and other neuro-developmental disorders.

BIBLIOGRAPHY

11. REFERENCES

- [1] 1] M. Bala, M. H. Ali, M. S. Satu, K. F. Hasan, and M. A. Moni, “Efficient machine learning models for early stage detection of autism spectrum disorder,” *Algorithms*, vol. 15, no. 5, p. 166, May 2022.
- [2] D. Pietrucci, A. Teofani, M. Milanesi, B. Fosso, L. Putignani, F. Messina, G. Pesole, A. Desideri, and G. Chillemi, “Machine learning data analysis highlights the role of parasutterella and alloprevotella in autism spectrum disorders,” *Biomedicines*, vol. 10, no. 8, p. 2028, Aug. 2022.
- [3] R. Sreedasyam, A. Rao, N. Sachidanandan, N. Sampath, and S. K. Vasudevan, “Aarya—A kinesthetic companion for children with autism spectrum disorder,” *J. Intell. Fuzzy Syst.*, vol. 32, no. 4, pp. 2971–2976, Mar. 2017.
- [4] J. Amudha and H. Nandakumar, “A fuzzy based eye gaze point estimation approach to study the task behavior in autism spectrum disorder,” *J. Intell. Fuzzy Syst.*, vol. 35, no. 2, pp. 1459–1469, Aug. 2018.
- [5] H. Chahkandi Nejad, O. Khayat, and J. Razjouyan, “Software development of an intelligent spirography test system for neurological disorder detection and quantification,” *J. Intell. Fuzzy Syst.*, vol. 28, no. 5, pp. 2149–2157, Jun. 2015.
- [6] F. Z. Subah, K. Deb, P. K. Dhar, and T. Koshiba, “A deep learning approach to predict autism spectrum disorder using multisite resting-state fMRI,” *Appl. Sci.*, vol. 11, no. 8, p. 3636, Apr. 2021.
- [7] K.-F. Kollias, C. K. Syriopoulou-Delli, P. Sarigiannidis, and G. F. Fragulis, “The contribution of machine learning and eye-tracking technology in autism spectrum disorder research: A systematic review,” *Electronics*, vol. 10, no. 23, p. 2982, Nov. 2021.

- [8] I. A. Ahmed, E. M. Senan, T. H. Rassem, M. A. H. Ali, H. S. A. Shatnawi, S. M. Alwazer, and M. Alshahrani, “Eye tracking-based diagnosis and early detection of autism spectrum disorder using machine learning and deep learning techniques,” *Electronics*, vol. 11, no. 4, p. 530, Feb. 2022.
- [9] P. Sukumaran and K. Govardhanan, “Towards voice based prediction and analysis of emotions in ASD children,” *J. Intell. Fuzzy Syst.*, vol. 41, no. 5, pp. 5317–5326, 2021.
- [10] S. P. Abirami, G. Kousalya, and R. Karthick, “Identification and exploration of facial expression in children with ASD in a contact less environment,” *J. Intell. Fuzzy Syst.*, vol. 36, no. 3, pp. 2033–2042, Mar. 2019.
- [11] M. D. Hossain, M. A. Kabir, A. Anwar, and M. Z. Islam, “Detecting autism spectrum disorder using machine learning techniques,” *Health Inf. Sci. Syst.*, vol. 9, no. 1, pp. 1–13, Dec. 2021.
- [12] C. Allison, B. Auyeung, and S. Baron-Cohen, “Toward brief ‘red flags’ for autism screening: The short autism spectrum quotient and the short quantitative checklist in 1,000 cases and 3,000 controls,” *J. Amer. Acad. Child Adolescent Psychiatry*, vol. 51, no. 2, pp. 202–212, 2012.
- [13] F. Thabtah, F. Kamalov, and K. Rajab, “A new computational intelligence approach to detect autistic features for autism screening,” *Int. J. Med. Inform.*, vol. 117, pp. 112–124, Sep. 2018.
- [14] M. M. Ali, B. K. Paul, K. Ahmed, F. M. Bui, J. M. W. Quinn, and M. A. Moni, “Heart disease prediction using supervised machine learning algorithms: Performance analysis and comparison,” *Comput. Biol. Med.*, vol. 136, Sep. 2021, Art. no. 104672.

- [15] E. Dritsas and M. Trigka, “Stroke risk prediction with machine learning techniques,” *Sensors*, vol. 22, no. 13, p. 4670, Jun. 2022.
- [16] V. Chang, J. Bailey, Q. A. Xu, and Z. Sun, “Pima Indians diabetes mellitus classification based on machine learning (ML) algorithms,” *Neural Comput. Appl.*, early access, pp. 1–17, Mar. 2022.
- [17] F. Thabtah, “Machine learning in autistic spectrum disorder behavioral research: A review and ways forward,” *Inform. Health Social Care*, vol. 44, no. 3, pp. 278–297, 2018.
- [18] K. S. Omar, P. Mondal, N. S. Khan, M. R. K. Rizvi, and M. N. Islam, “A machine learning approach to predict autism spectrum disorder,” in *Proc. Int. Conf. Electr., Comput. Commun. Eng. (ECCE)*, Feb. 2019, pp. 1–6.
- [19] H. Abbas, F. Garberson, E. Glover, and D. P. Wall, “Machine learning approach for early detection of autism by combining questionnaire and home video screening,” *J. Amer. Med. Informat. Assoc.*, vol. 25, no. 8, pp. 1000–1007, 2018.
- [20] K. L. Goh, S. Morris, S. Rosalie, C. Foster, T. Falkmer, and T. Tan, “Typically developed adults and adults with autism spectrum disorder classification using centre of pressure measurements,” in *Proc. IEEE Int. Conf. Ac.*