#### A Project Report on

**“DEVELOPMENT OF AMES: AUTOMATED ML EXPERT SYSTEM”**

**Submitted in partial fulfillment of the requirement for Degree in Bachelor of Engineering (Information Technology)**

**By**

Keertana Kappuram (501830)

Prithviraj Sanjay Patil (501850)

Russel Rumao (501858)

#### Guided by:

Prof. Poonam Bari



**Department of Information Technology**

**Fr. Conceicao Rodrigues Institute of Technology**

Sector 9A, Vashi, Navi Mumbai – 400703

**University of Mumbai 2021-2022**

i

**CERTIFICATE**

This is to certify that the project entitled

**“DEVELOPMENT OF AMES: AUTOMATED ML EXPERT SYSTEM**

##### Submitted By

|  |  |
| --- | --- |
| Keertana Kappuram | 501830 |
| Prithviraj Sanjay Patil | 501850 |
| Russel Rumao | 501858 |

In partial fulfillment of degree of **B.E**. in **Information Technology** for term work of the project is approved.

##### External Examiner Internal Examiner

**Internal Guide Head of the Department**

**Principal**

**Date: - College Seal**

ii

**DECLARATION**

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the institute and can evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

-----------------------------------------

Keertana Kappuram (501830)

-----------------------------------------

(Prithviraj Sanjay Patil (501850)

-----------------------------------------

Russel Rumao (501858)

Date: / /

## ABSTRACT

##### There has been an exponential rise in the quantity of data in the last few decades and as a consequence of this, the need for Machine Learning related applications has increased in every domain. In recent years, Machine Learning has been used in many fields to achieve significant breakthroughs. These fields include financial services, transportation, healthcare, e-commerce, retail, etc. wherein Machine Learning has been used for innovation, transformation and optimization to get highly promising results. In today’s world, Machine Learning is not used only for research and development applications but also in the enterprise domain. However, traditional Machine Learning methods are dependent on humans and that is not a feasible option for businesses having limited resources and those which cannot invest in a highly qualified data science team. And even in the case of ML Engineers who are in high demand across various industries, improving the efficiency of the task of Machine Learning has become a challenge. This calls for the creation of an application that can automate the end-to-end process of applying machine learning solutions to real-world problems.

##### “AMES: Automated ML Expert System” will make Machine Learning available in a true sense, even to people with no major expertise in this field. Such a system will increase productivity by automating repetitive tasks, help to avoid errors that might creep in due to manual processes and democratize Machine Learning by making the power of ML accessible to everybody. Tasks such as hyperparameter optimisation, feature engineering, data preprocessing, visualisation and even model selection, if automated, will prove to be of great benefit to ML Engineers and novice users alike. This project aims to automate all these tasks and make the process of building a model simple, quick and efficient.

##### Keywords: Machine Learning, AutoML, Hyperparameter Optimisation, Model Selection

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Topic** | **Page No** |
| **1** | **Introduction** |  |
|  | 1.1 Background |  |
|  | 1.2 Motivation |  |
|  | 1.3 Objectives |  |
|  | 1.4 Customer and Users |  |
|  | 1.5 Problem Statement |  |
|  | 1.6 Project Description |  |
|  | 1.7 Scope |  |
|  | 1.8 Key milestones |  |
|  | 1.9 Known Risks and Issues |  |
|  | 1.10 Advantages |  |
| **2** | **Literature Survey and Analysis** |  |
|  | 2.1 Related published works |  |
|  | 2.2 Existing systems |  |
|  | 2.3 Analysis  2.4 Requirement Analysis |  |
|  | 2.4 Requirement Specification |  |
| **3** | **System Design** |  |
|  | 3.1 Architecture diagram |  |
|  | 3.2 Use Case diagram |  |
|  | 3.3 Data Flow diagram |  |
|  | 3.4 Flow Chart |  |
| 4 | **Implementation Details**   * 1. Minimum Hardware and Software Requirements      1. Hardware Requirements      2. Software Requirements   2. Solution Approach |  |
| **5** | **Software Testing/Test Cases**   * 1. Timeline Chart for VIII Semester   2. Test Cases |  |
| **6** | **Experimental Results**  6.1 GUI  6.2 Result Analysis |  |
| **7** | **Conclusion and Future Scope**  7.1 Conclusion  7.2 Future Scope |  |
|  | **References** |  |
|  | **Appendix A: Code Sample** |  |
|  | **Appendix B: Certificates and Paper Published** |  |
|  | **Acknowledgement** |  |

**LIST OF FIGURES**

v

|  |  |  |
| --- | --- | --- |
| **Fig. No** | **Figure Name** | **Page No** |
| **2.1**  2.1.1  2.1.2 | **Related published works** |  |
| **2.2** | **Existing Systems** |  |
| 2.2.1 |  |  |
| 2.2.2 |  |  |
| 2.2.3 |  |  |
| **2.4.1** | **Hardware Requirements** |  |
| 2.4.1.1 |  |  |
| 2.4.1.2 |  |  |
| 2.4.1.3 |  |  |
| **3** | **System Design** |  |
| 3.1 | Architecture Diagram |  |
| 3.2 | Block Diagram |  |
| 3.3 | Flow Chart |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **4.2**  4.2.1  4.2.2 | **Solution Approach** |  |
| **5**  5.1 | **Software Testing/ Test Cases**  Time Line Chart for Semester 8 |  |
| **6.1** | **Graphical User Interface** |  |
| 6.1.1 |  |  |
| 6.1.2 |  |  |
| 6.1.3 |  |  |
| 6.1.4 |  |  |
| 6.1.5 |  |  |
| 6.1.6 |  |  |
| 6.1.7 |  |  |
| 6.1.8 |  |  |
|  | Appendix B: Paper Published and Certificates |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Table Name** | **Page No.** |
| **4** | **Implementation details** |  |
| 4.1 | Minimum Hardware/Software Requirements |  |
| **5**  5.1  5.2 | **Software Testing/ Test Cases** Time Line Chart for Semester 8 Test Cases |  |
| **6** | **Experimental Results** |  |
| 6.2 |  |  |

ix

x

# CHAPTER 1 INTRODUCTION

## INTRODUCTION

#### Background

Automated Machine Learning provides methods and processes to make Machine Learning available for non-Machine Learning experts, to improve efficiency of Machine Learning and to accelerate research on Machine Learning. Machine learning has achieved considerable successes in recent years and an ever-growing number of disciplines rely on it. However, this success crucially relies on human machine learning experts to perform Preprocessing and cleaning, Selection and Construction of appropriate features, Selection of a model family, Optimisation of hyper parameters, and Critically analysing the results obtained. As the complexity of these tasks is often beyond non-ML-experts, the rapid growth of machine learning applications has created a demand for off-the-shelf machine learning methods that can be used easily and without expert knowledge.

#### Motivation

Model creation and Hyperparameter Tuning have time and again proven to be the most challenging and time consuming tasks in the whole Machine Learning workflow. When it comes to these operations it is not possible to trust the defaults because the parameters suitable for one model may not yield significant results in the other. There are too many moving pieces when it comes to model creation - the number of hyper parameters, the error of overfitting or underfitting, and the mistakes related to choosing the wrong metrics for evaluating a model and it’s parameters. Considering all these issues, it would be an innovative approach to use Machine Learning to solve the problems in Machine Learning. This will not only save the time of Machine Learning experts but also allow novices to enter into the field and integrate Machine Learning into the domain of their choice.

#### Objectives

The main objective of the system is to eliminate human intervention at every step of the Machine Learning Model Creation process.

1. To help not only Machine Learning experts, but also novices of this domain to access highly accurate machine learning models.
2. To create a web application which will be used by the users to easily upload datasets and create models which can be saved by them and used in the future.
3. To generate an optimum model, using the ‘AutoML’ Library and using a Custom Algorithm Selection Process based on the input data for a given Machine Learning task.

#### Users

The Users of AMES are:

##### Machine Learning Novices

* + To get ready-to-use machine learning models that can be implemented without expert knowledge
  + To enter the Machine Learning field and integrate Machine Learning into the domain of their choice.

##### Machine Learning Experts

* + To deal with repetitive and time-consuming ML tasks

##### Enterprises

* + To quickly adopt machine learning solutions which lets the data scientists of the company pay attention to more complicated issues which lead to improving the overall plan

#### Problem Statement

The performance of a given model depends on both the fundamental quality of the algorithm and the details of its tuning. There is a need to provide automated Hyperparameter optimization and algorithm configuration in order to automate the tedious, time-consuming and error-prone process of constructing a model. In order to provide a high performing model for a given dataset and simultaneously reduce the time invested by a Machine Learning engineer, an efficient and easy to use system is required to be created. The aim of this system should be to eliminate human intervention at every stem of the Model Creation process.

#### Project Description

**AMES: Automated ML Expert System** aims to provide Machine Learning enthusiasts with a website which lets users upload a dataset and gives them an ML model with the highest accuracy as the output which can be saved and reused.

The **website** consists of three main tabs which are the “**Data Preprocessing**”, “**Build Model**” and “**Complete Model**” tabs. In the “Data Preprcoessing” tab, the user can upload a dataset and get a preprocessed dataset as an output which can be used by them for any ML application that they want to. The “Build Model” tab let’s the user choose which parts of the preprocessed dataset do they want to work on and lets the user choose whether they want to perform “Binary Classification”, “Multiclass Classification” or “Regression”. Once the user chooses one of these options, our website automatically build a model which gives the highest accuracy using multiple algorithms for a given dataset. The “Complete Model” tab can be used to download the final model. The machine learning model built on the website is also mailed to the user so that they can reuse it as they please.

#### Scope

1. The system assumes that the input dataset has undergone the preprocessing stage and hence will provide optimum results in terms of model selection.
2. The system is capable of generating an optimum model based on the input data for a given Machine Learning task.
3. The input data is fed to the AutoML library and a Custom algorithm selection process and the results only between these two modules are compared and the best performing model is provided as an output.
4. The output model is stored in the database and hence made available for future use.

#### Key Milestones

* + 1. Creating the website
    2. Connecting the website with the database
    3. Letting the user choose which Machine Learning problem they are dealing with and manipulate the dataset uploaded
    4. Developing the custom algorithm selection process
    5. Developing the model building process which chooses the algorithm with the highest accuracy
    6. Connecting the backend with the front end
    7. Saving the final ML model in the database and sending emails containing the model right after it is created

#### Limitations

1. The Training Process will be time consuming since it also involves Validation and Scoring.
2. The system will require a lot of computational resources during deployment since extensive and costly operations are needed to be carried out at the backend.
3. The system will only be able handle input data upto a given upper limit of input dataset size. beyond that the processing will become too intensive for deployment via a server.
4. Ensuring that the results of the Custom Algorithm selection process are at par with the AutoML library will be a primary challenge, if it is not overcome the results will always be skewed in the direction of the AutoML library.
5. Deploying the application on a mobile platform will not be possible since the processing and User Interface are more suited to Desktop usage

#### Advantages

* + 1. Saves Time:

The time take by a user to go through multiple processes like dealing with datasets, choosing algorithms to use, comparing their accuracies, building a model, saving it for further use, etc. can be saved by the proposed system as it does everything on one website without wasting the user’s time.

* + 1. Can Be Used By Anyone:

AMES can be used by Machine Learning Novices, Machine Learning Experts and Enterprises. Traditional Machine Learning methods are dependent on humans and that are not a feasible option for businesses having limited resources and those which cannot invest in a highly qualified data science team. And even in the case of ML Engineers who are in high demand across various industries, improving the efficiency of the task of Machine Learning has become a challenge which can be solved using automated machine learning practices.

* + 1. Reduce Human Intervention

Machine Learning Models rely on human Machine Learning experts to perform Pre-processing and cleaning, Selection and Construction of appropriate features, Selection of a model family, Optimisation of hyperparameters, and Critically analysing the results obtained. Human Intervention can be reduced by automating such processes.

* + 1. Reduce Manual Errors

Working on tedious Machine Learning processes can result in occasional manual errors which should be avoided to get optimal results, which can be done by using our system.

* + 1. Increase Productivity

Productivity can be increased by automating the Machine Learning model creation process which result in reduction of repetitive processes, users paying attention to more complex issues which require human intervention and increased motivation due to decrease of tedious tasks.

8

# CHAPTER 2

**LITERATURE SURVEY AND ANALYSIS**

## LITERATURE SURVEY AND ANALYSIS

The “DEVELOPMENT OF AMES : Auto ML expert system” aims to take in a preprocessed dataset, train and test it in a manner such that it yields the optimum result in terms of testing scores. In doing so this system will eliminate the need to go through the cumbersome process of adjusting the hyperparameters retraining models. The algorithm selection process is the heart of this whole system. In order to arrive at the best approach it was important to thoroughly examine the existing systems and algorithm selection method. Knowledge regarding concepts like Hyperparameter Optimisation[1], Meta Learning [2] is imperative to understand this system. Hence, several papers and existing systems have been analysed to get a better understanding of the system.

#### Related Work

In **OBOE** [3] as demonstrated by Chengrun Yang, Yuji Akimoto, et. al., a collaborative filtering method for time-constrained model selection and hyperparameter tuning has been suggested by the authors. OpenML and UCI datasets with 150 to 1000 data points and no missing value were used for training purposes. Instead of meta features, latent meta features were used and the optimization procedure ensured that these latent meta-features best predicted the cross-validated errors, among all bilinear models. The Oboe system is primarily divided into Offline and Online stages. The Offline stagecomprises data preprocessing, error matrix generation and computation of low dimensional algorithm features, where training datasets were used as input. Data preprocessing , time constrained model selection, and ensemble are the parts of the online stage where the test dataset and low dimensional algorithm features are used as input.

**Auto-WEKA** [4] by Chris Thornton, Frank Hutter, et. al., puts forth a ‘auto classifier and optimizer’ which have a good performance on popular datasets of UCI and variants of MNIST and CIFAR. Algorithms with optimal generalized performance are selected for the model and then hyperparameters are optimized for better accuracy. Sequential Model-Based Optimization (SMBO) which comes under Bayseian optimization is used for this purpose as it works explicitly on categorical and continuous hyperparameters. Auto-Weka uses Sequential Model-based Algorithm Configuration (SMAC) and Tree-structured Parzen Estimator (TPE) are the two SMBO algorithms. Auto-Weka has two top level boolean parameters which leads to a very wide tree that allows the creation of a single hyperparameter optimization problem with four hierarchical layers of a total of 786 parameters. SMAC and TPE find models with much better cross-validation performance than by grid search over WEKA’s 29 base classifiers.

**ATM** [5] by Thomas Swearingen et. al., presents ‘Auto Tuned Models’ where users can upload a dataset and choose a subset of modelling methods and the results are provided in the form of ready-to-predict models, confusion matrices, cross-validation results, and training timing. ATM provides two model/optimisation selection/search methods; these are (1) A hybrid Bayesian and multi-armed bandit optimisation system and (2) A model recommender system that works by exploiting the modeling techniques’ previous performances on a variety of datasets. ATM is a distributed asynchronous system that can generate several classifiers simultaneously. ATM’s hierarchical algorithm allows search at the method level and tunes hyperparameters. When A user uploads their dataset it gets stored and registered to the ModelHub database. The ATM workers query ModelHub for classifiers to run and report the results back to the ModelHub. The results of the models are obtained via streaming in a load-balanced fashion to ensure speed. The system follows an Open Source approach and abstractions are provided so that AutoML experts can modify and propose new methods for model selection. The system also consists of a database repository of several hundred thousand classifiers trained on hundreds of datasets.

In **Auptimizer** [6] by Jiaji Liu, Samarth Tripathi, et. al., the authors review the challenges faced while using HPO (Hyperparameter Optimisation) and show a general Hyperparameter Optimization tool called Auptimizer to speed up model tuning and bookkeeping. The system design simplifies creating, controlling and tracking of typical machine learning projects. It also allows for the integration of new HPO algorithms. By using a few lines of code to an existing project the user can switch between different HPO algorithms and computing resources without rewriting their entire training scripts.It has two key components i.e Resource Manager and Proposer and Tracking and Visualization as supplementary components. (1)The Proposer controls how the Auptimizer interacts with HPO algorithms for recommending new hyperparameter values. (2) The resource Manager connects computing resources to model training automatically thus allowing codes to run on resources based on their availability. It also has a basic tool to visualize the results from history. Experiment tracking where the history can be tracked in a user-specified database is also present.

#### Existing Systems

#### Auto-WEKA [2]

#### In the paper ‘The WEKA Data Mining Software: An Update’ [7] by Mark Hall, Eibe Frank,

#### et. al., The authors review the WEKA workbench, history of project, user interface, new features

#### in the recent update, etc. Auto-WEKA is a tool that performs algorithm selection and

#### hyperparameter optimization over classification and regression algorithms which are

#### implemented in WEKA to give a model with the best performance. In ‘User Guide for

#### Auto-WEKA version 2.6’ [8] by Lars Kotthoff, Chris Thornton, et. al., The authors have

#### discussed Auto-WEKA’s GUI, classifiers, parameters, methodology, etc.

#### H2O’s AutoML[9]

#### H2O’s AutoML is a tool that automates the machine learning workflow which includes automating

#### the process of training candidate models and finding the “best” model without any

#### prior knowledge. It is based on a combination of fast random search and stacked ensembles which helps it achieve results equivalent to and often better than other methods which work on complex model tuning techniques. The result is a leaderboard with a ranked list of all models which can be easily exported into the user’s application environment. The ranking is based on multiple model performance measures like mean squared error, accuracy, F1 score, Gini coefficient, mean absolute error, etc. The goal is to save time, reduce manual code-writing time, improve performance, increase reproducibility and scale training data sets into clusters

#### Analysis

After looking at the existing systems available for AutoML tools, it is understood that that these systems mainly deal with choosing the best algorithm but do no give the user a way to automatically build a complete Machine Learning model which can be used by Machine Learning novices to Machine Learning experts.

Possible Improvements In The Existing Systems:

* + - 1. Developing one system that automates every part of the Machine Learning model building process
      2. Increase in accuracy for better experience
      3. Easy User interface
      4. Letting users save models so that they can reuse them

**2.4 Requirement Analysis**

**2.4.1 Functional Requirements**

Based on the features and needs of the system the Functional requirements can be described as follows -

1. Input dataset - The user should be able to upload a preprocessed dataset which gets stored on the server.
2. Specify Machine Learning Task - The user must be able to specify the task which needs to be performed so that a model is built appropriately.
3. Output Model - The user should be provided the ability to download the output model generated by the website and hence make it available for future use.
4. Storage - The created model should be stored in the User account’s Database.
5. Visualisation - The training process using the validation dataset should be shown to the user in the form of graphs and other visualisation methods.
6. Registration - The web application will allow only registered and authenticated users to use its features.

**2.4.2 Non Functional Requirements**

The model selection system has the following non functional requirements -

1. Speed - Given that the process of selecting a model and running the input data through AutoML is a time consuming task the overall speed of the website needs to be ensured. The web application must be fast in loading the data and processing it into a model.
2. Usability - The web application should be user-friendly. It should provide an easy to use User Interface that is suitable for Machine Learning Professionals and Beginners alike.
3. Security - Validation of the registered Sign In credentials should be carried out thoroughly.
4. Maintainability - The website should be capable of incorporating new features requirements as they are created.
5. Scalability- System should be able to handle requests from a large number of users.

#### 2.5 Requirement Specification

**2.5.1 Hardware Requirements**

**For Server**

1. 16 GB RAM
2. 1 TB ROM
3. Intel i7/i9 (10 th Gen) Core Processor
4. Any GPU
5. Windows 8/10 or Linux 18/20

**For User**

1. 8 GB RAM
2. 500 GB ROM
3. Intel i5 Core Processor

**2.5.2 Software Requirements**

1. **Python:** Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.
2. **EvalML Library:** EvalML is an AutoML library that builds, optimizes, and evaluates machine learning pipelines using domain-specific objective functions.
3. **Numpy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
4. **Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
5. **Pickle:** Pickle is the standard way of serializing objects in Python. You can use the pickle operation to serialize your machine learning algorithms and save the serialized format to a file. Later you can load this file to deserialize your model and use it to make new predictions.
6. **Flask:** Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.
7. **Firebase:** Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.
8. **React.js:** React. js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps.

17

# CHAPTER 3 SYSTEM DESIGN

## SYSTEM DESIGN

#### Architectural Block Diagram

The architecture of this system, as shown in Figure 3.1, requires a preprocessed dataset as an input from the user. This preprocessed dataset will be acquired from the “Automated Data Cleaning” module. The dataset will undergo Data Cleaning - which involves the Handling of Missing values, Outlier detection and removal and Normalisation of data. After this process is complete the data will be converted to the required format for the further process of Model Selection. Once the dataset is uploaded it will be split into a Training, Testing and Validation dataset. Using these 3 sub-datasets Algorithm selection will be carried out. Once the most efficient algorithm is selected the model will be trained on the Training dataset. Upon completion of the Training process Validation and Testing will be carried out to conclude the performance of the selected model. Then all the parameters of the model along with it’s scores will be stored as a Pickle file into the database for future use. The resultant model will also be returned to the user via email, who can now use this model for further tasks.

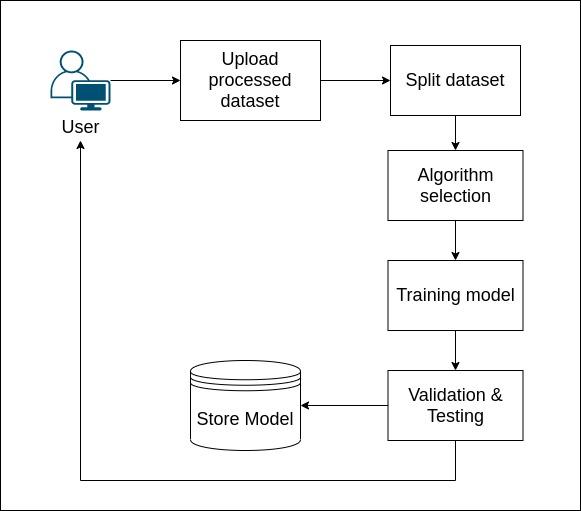
****

Figure 3.1 - Architecture Diagram

**3.2 Block Diagram**

The Block diagram for AMES includes the various modules and data objects present in the system, and how the system decides to select the best possible model using both, the custom algorithm selection process and the AutoML library. As depicted in Fig. 3.2, when the preprocessed dataset arrives as an input, it will be fed to the algorithm selection module. This module will split the dataset into a training, testing and validation dataset. A training dataset is defined as a dataset of examples used during the learning process and is used to fit the parameters. This dataset will be fed to the combined module which consists of Algorithm Selection and Hyperparameter Tuning. This module also takes in the validation dataset to score the model. This whole process of training and validation can be visualized by the user to better understand the model generation process. The output from this module will be given to the Evaluation module which will make use of the testing dataset and provide the final selected output module. However, in order to get the best results, the preprocessed dataset will also be fed to the AutoML Library. This library will provide its own output model and both the output models will be compared with their accuracies. The best performing model will be returned as the final output model.

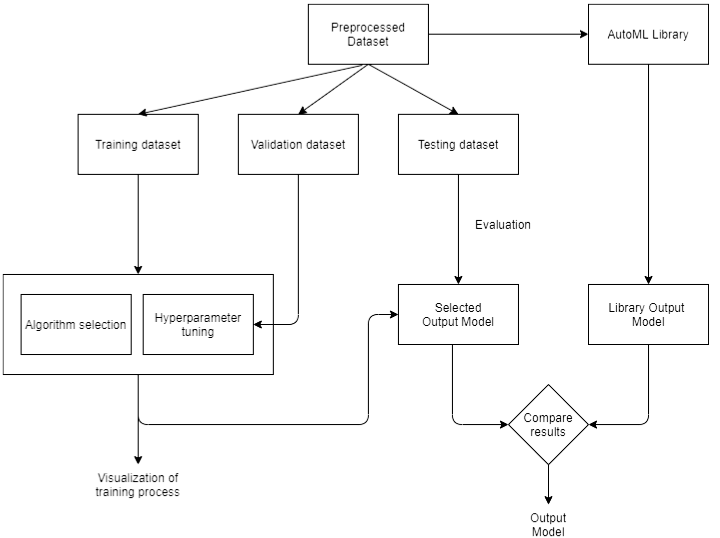


Figure 3.2 - Block Diagram

**3.3 Flow Chart**

In Figure 3.3, the flow of data through the system and the various processes that are carried out using it.

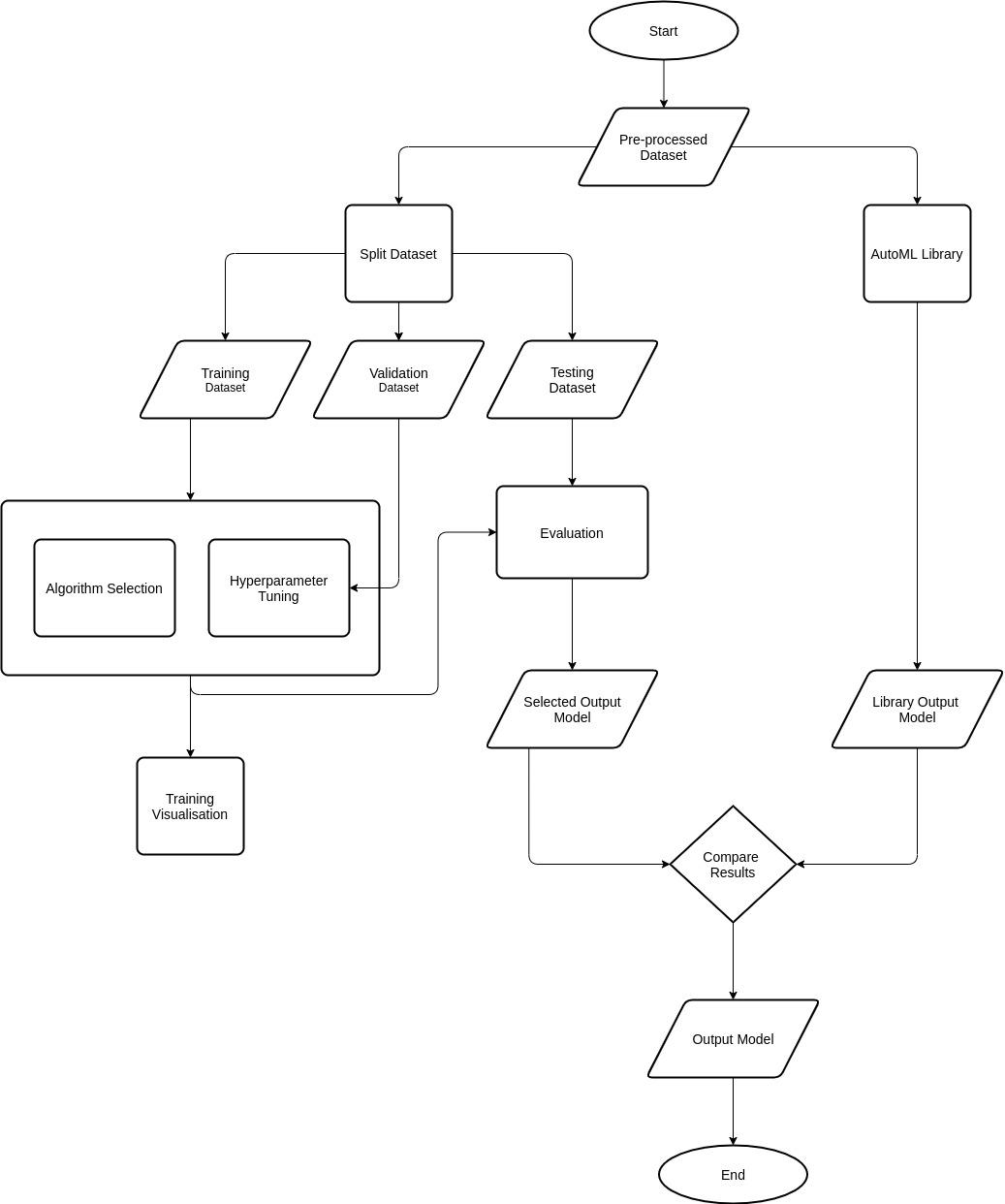


Figure 3.3 - Flow Chart

The processes in the system start with the essential input required for the system, that is the preprocessed dataset. This data is now provided as an input to two other processes. The first process is splitting of the dataset into training, validation and testing sections for the custom algorithm selection process and the second process where the preprocessed dataset goes to is the AutoML library model creation. The training and validation datasets then go through hyperparameter tuning and algorithm selection whereas the training dataset goes through evaluation which happens after algorithm selection. Representation of all algorithms’ performances for a given dataset happens in the “Training visualization” section of the data flow. Once the evaluation for the custom process is done, the most accurate algorithm is chosen to create the first model. Another model is created using the AutoML library. Both of the models are compared to find the model with higher accuracy which will be the final model. This model can then be downloaded and used by Machine Learning enthusiasts as per their requirements.

24

# CHAPTER 4

**IMPLEMENTATION DETAILS**

## IMPLEMENTATION DETAILS

#### Hardware/Software Requirement

Table 4.1 Hardware/Software Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Site** | **Requirements** | | |
| 1 | Developer | **Sr. No** | **Hardware** | **Software** |
| 1 | 16 GB RAM | Python 3.7 + |
| 2 | 1 TB ROM | EvalML |
| 3 | Intel i7/i9 (10th Gen) Core Processor | Keras |
| 4 | Any GPU | Tensorflow |
| 5 | Windows 8/10 or Linux 18/20 | Numpy |
| 6 |  | Pandas |
| 7 |  | Pickle |
| 8 |  | Flask |
| 9 |  | Firebase |
| 10 |  | React.js |
| 2 | User | 11 | 8 GB RAM | Access to website |
| 12 | 500 GB ROM  Intel i5 Core Processor | Email account to download model |

#### Solution Approach

* + 1. **Dataset Classification Module - Custom Model**

The custom algorithm selection process consists of comparison between the python library, sklearn’s classification algorithms applied on the user’s uploaded dataset. The classification algorithms considered by the model are Support Vector Classification, Decision Tree, Random Forest and Logistic Regression. The algorithms are compared to get the one algorithm which gives the highest accuracy for a given dataset. This accuracy is then compared with EvalML’s AutoML library’s accuracy for the same dataset and the most accurate algorithm is chosen as the final model which can be saved by the user who can then reuse it.

Figure 4.1 shows the results of the custom algorithm selection process on the dataset “Bank Note Authentication”[12] which is used to predict whether the banknotes are genuine or fraudulent using 4 different parameters namely, skewness, curtosis, entropy and variance. Here, Support Vector Classification, Logistic Regression, Decision Tree and Random Forest algorithms give an accuracy of 100%, 98.15%, 98.91% and 99.34% respectively. The algorithm with the highest accuracy is Support Vector Classification which is why, it will be compared with the AutoML library, out of which the option with the highest accuracy will be chosen as the final algorithm.

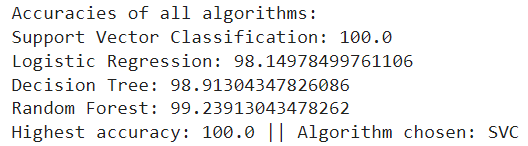


Figure 4.1 Custom Model Results- Classification

* + 1. **Dataset Regression Module - Custom Model**

The custom algorithm selection process consists of comparison between the python library, sklearn’s classification algorithms applied on the user’s uploaded dataset. The regression algorithms considered by the model are Support Vector Regression, Decision Tree, Random Forest and Linear Regression. The algorithms are compared to get the one algorithm which gives the highest accuracy for a given dataset. This accuracy is then compared with EvalML’s AutoML library’s accuracy for the same dataset and the most accurate algorithm is chosen as the final model which can be saved by the user who can then reuse it.

Figure 4.2 shows the results of the custom algorithm selection process on the dataset “Auto mpg”[14] which has information about mileage per gallon performance of various cars. Here, Support Vector Regression, Linear Regression, Decision Tree and Random Forest algorithms give an accuracy of 81.45%, 88.51%, 69.89% and 75.91% respectively. The algorithm with the highest accuracy is Random Forest which is why, it will be compared with the AutoML library, out of which the option with the highest accuracy will be chosen as the final algorithm.

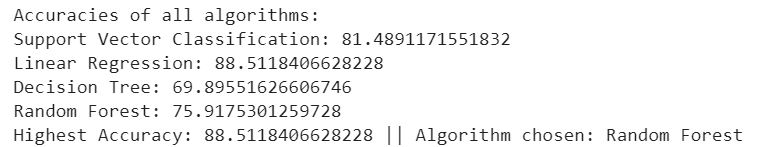


Figure 4.2 Custom Model Results- Regression

**4.2.3 EvalML Model:**

This module is responsible for performing the task of Classification in Machine Learning using text data. Unlike the previous module, it uses a library named EvalML. EvalML is an AutoML library which builds, optimizes, and evaluates machine learning pipelines using domain-specific objective functions. Therefore Classification on text input will be performed using EvalML and a Custom Model and the best result will be provided to the user as an output model. In order to demonstrate the usage of EvalML for Binary Classification, the Breast Cancer dataset is used. In this dataset the real values are computed for each cell nucleus and it has features like radius (mean of distances from the center to points on the perimeter), texture (standard deviation of gray-scale values), perimeter, area, smoothness (local variation in radius lengths), compactness (perimeter^2 / area – 1.0), concavity (severity of concave portions of the contour), concave points (number of concave portions of the contour), symmetry and fractal dimension. The dataset has a class distribution of 357 records with the target class as ‘*Benign’* and 212 records as *‘Malignant’.* The approach for building this module involves loading in the required dataset and splitting it into X\_train, X\_test, Y\_train and Y\_test values using the functions provided. During the splitting process the problem type is specified as ‘binary’. The various available problem types can be listed and then AutoMLSearch is used to find the best algorithm for the provided dataset. Next the automl rankings function is used to find out the best performing model and the pipeline for this model is fetched. The problem can also be optimised for a specific objective. The model can now be saved as a pickle file and loaded in again for future use.

**Working**

The demonstration of Text Dataset Classification using EvalML is carried out using the Breast Cancer dataset as shown in Figure 4.3. This figure shows the first five rows of the dataset and it’s various attributes. This output is fetched using the head() method.

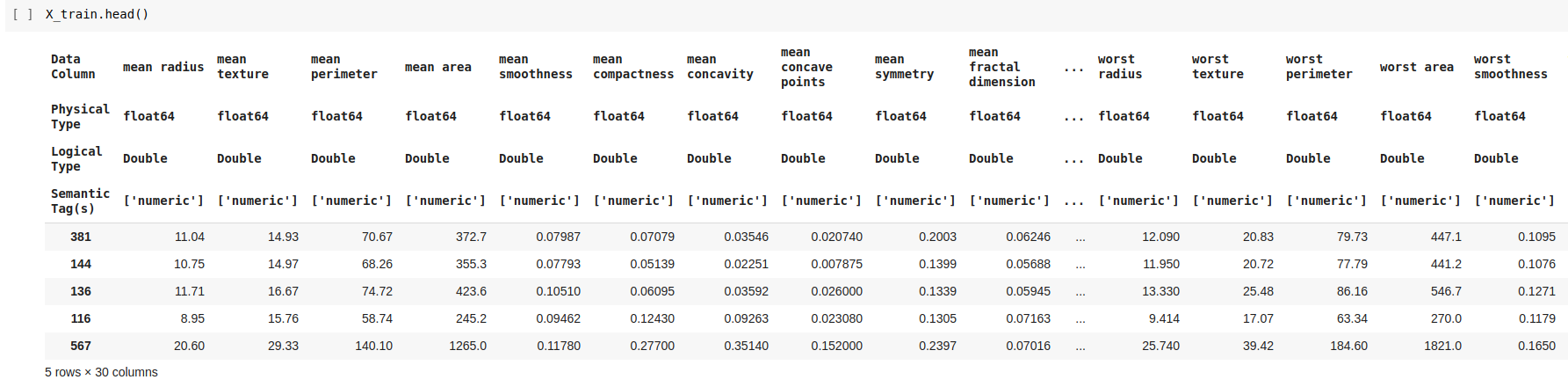
****

Figure 4.3 - Breast Cancer Dataset

EvalML provides many problem types on which a given dataset can be trained on. These are shown using the all\_problem\_types attribute of the ProblemTypes class. The problem types available are binary classification, multi class classification, regression, time series regression, time series binary, and time series multiclass. These problem types are shown in Figure 4.4.

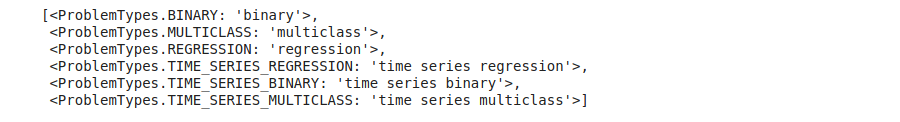


Figure 4.4 - Problem Types available in EvalML

After viewing the available problem types, the AutoMLSearch class’s search() method is used to find all the suitable models for a given task while considering the input dataset. The result of this search can be viewed using the rankings() method. The various pipelines and their ratings are shown in figure 4.5.

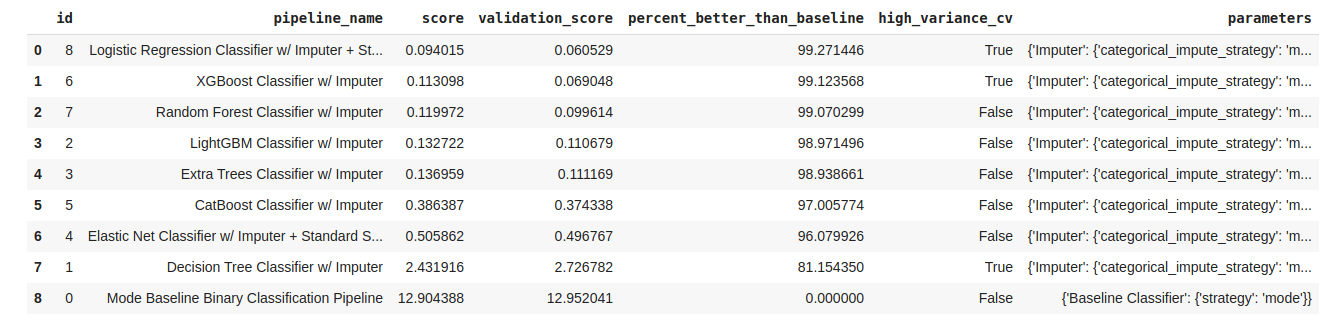
****

Table 4.5 - Various Pipelines and their Rankings

As visible clearly in Table 5.1 the Logistic Regression Classifier performs the best for this dataset. This algorithm is followed by the XGBoost and Random Forest algorithms. After viewing these rankings the best\_pipeline() method is used to get the best model which in this case will be Logistic Regression Classifier. The detailed description of the selected pipeline can be given by the describle\_pipeline() method. This detailed pipeline is displayed in Figure 4.6.

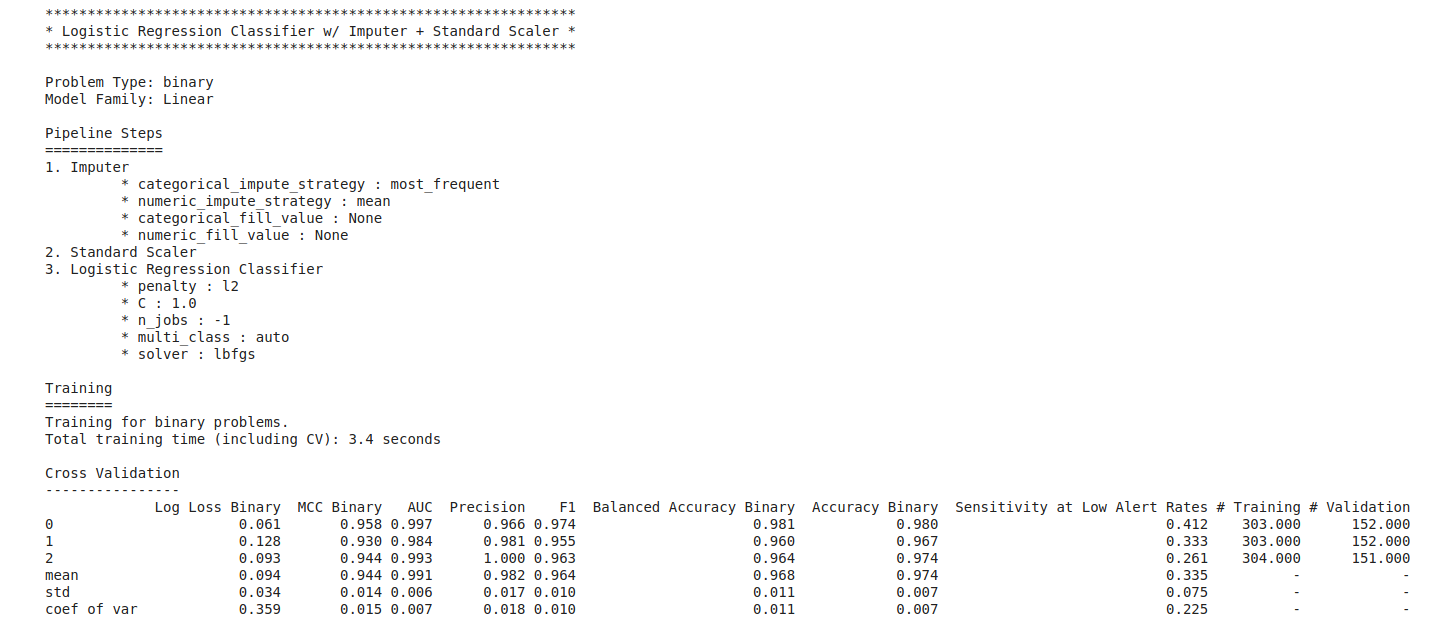


Figure 4.6 - Description of the best Pipeline

This whole optimisation process can also be carried out for a specific objective. For example an optimisation can be carried out by considering accuracy and recall. The final best pipeline is stored using the save() method which takes in a path to the pickle file for the model. This file can be loaded again using the load function and the predict function can be used on this loaded model and the results for predictions can be obtained as shown in Table 4.2.

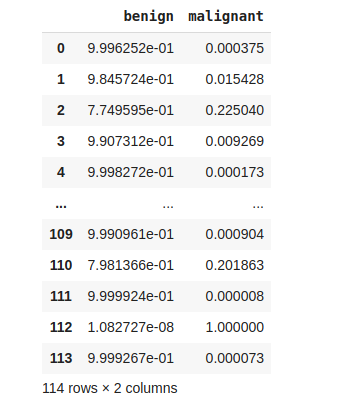


Table 4.2 - Predictions from the saved model

**4.2.4 Selecting the most accurate model**

The algorithms processed by the custom algorithm selection process and the AutoML library are compared to get the most accurate algorithm which is used to get the final results by the model.

Figure 4.8 shows the results of comparison between all the algorithms for the dataset “Bank Note Authentication” wherein Support Vector Classification gives the highest accuracy.

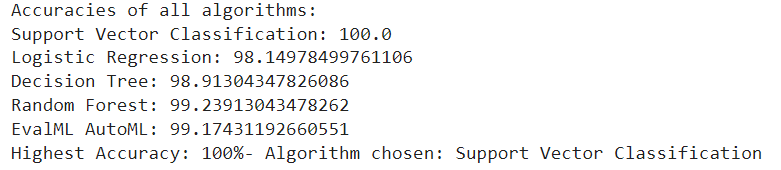


Figure 4.8 Comparison Between Custom Model Algorithms

30

# CHAPTER 5

**SOFTWARE TESTING/ TEST CASES**

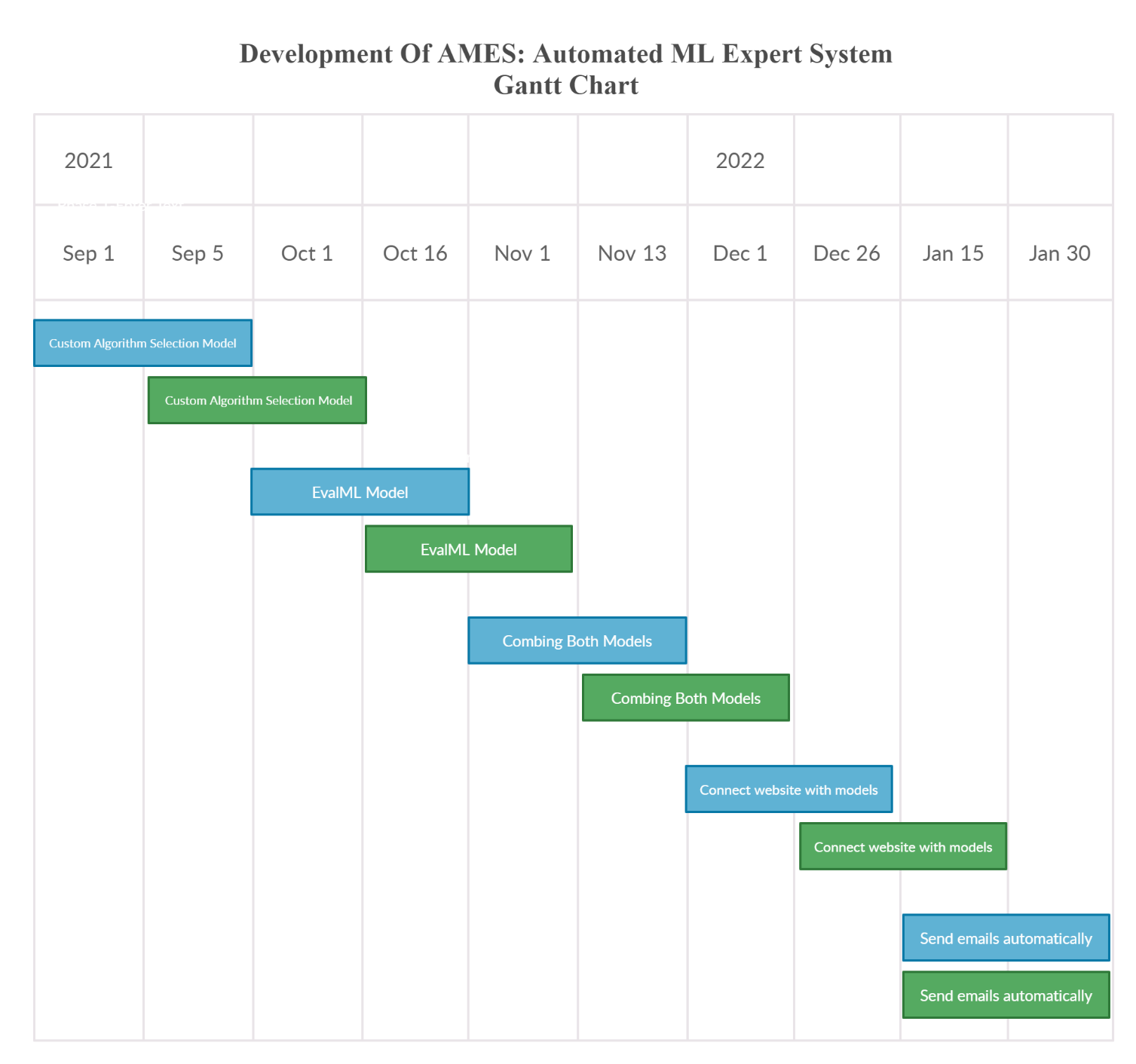
## SOFTWARE TESTING/ TEST CASES

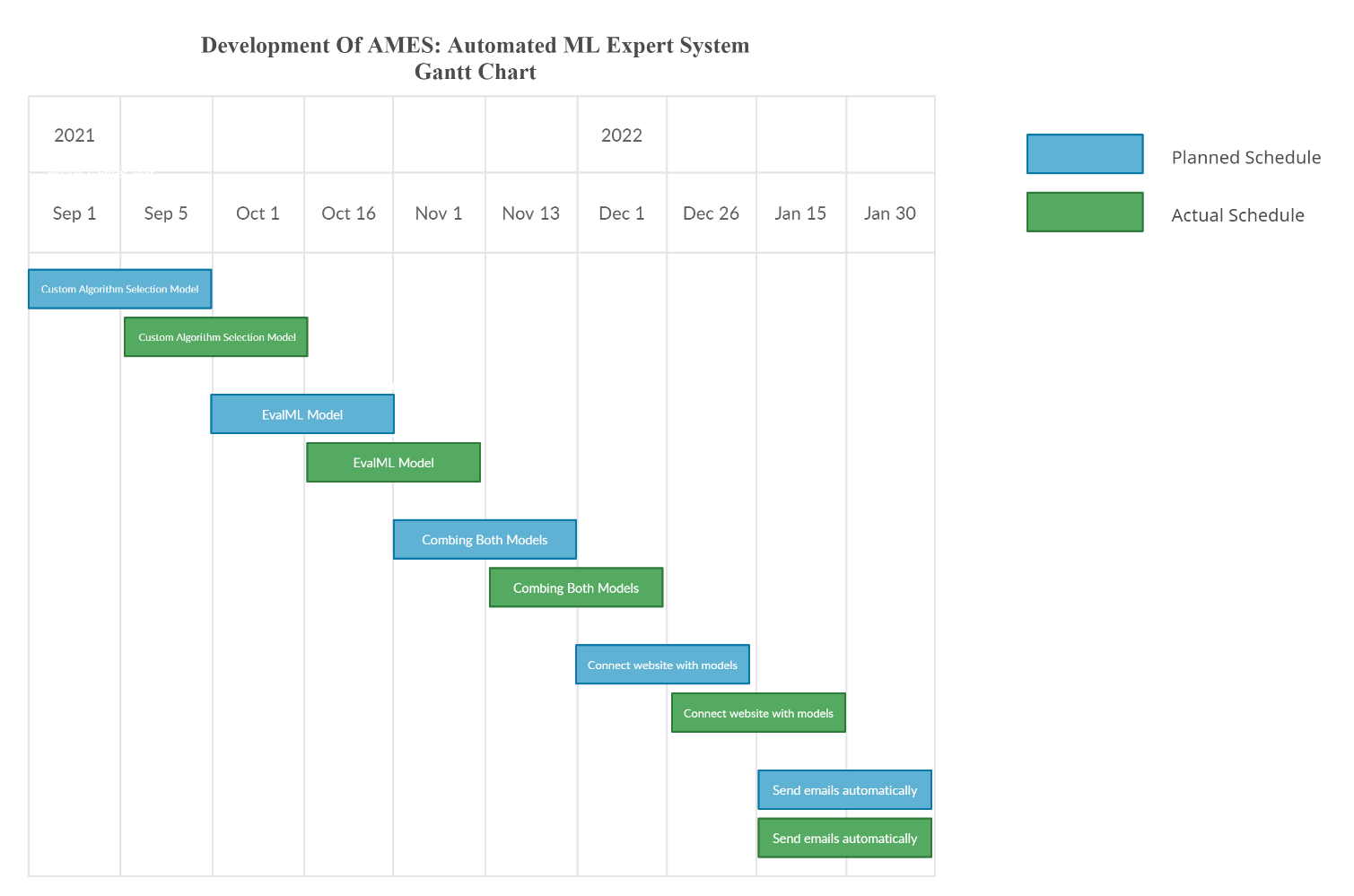
#### Time Line Chart for Semester 8

Table 5.1 Time Line Chart for Semester 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Scheduled Start Date** | **Scheduled End Date** | **Actual Start Date** | **Actual End Date** |
| Custom Algorithm Selection Model | 01/09/2021 | 01/10/2021 | 05/09/2021 | 16/10/2021 |
| EvalML Model | 01/10/2021 | 01/11/2021 | 16/10/2021 | 13/11/2021 |
| Combing Both Models | 01/11/2021 | 01/12/2021 | 13/11/2021 | 26/12/2021 |
| Connecting website with models | 01/12/2021 | 15/01/2022 | 26/12/2021 | 15/01/2022 |
| Sending emails automatically | 15/01/2022 | 30/01/2022 | 15/01/2022 | 30/01/2022 |

Fig 5.1 Time Line Chart for Semester 8





#### Test Cases

Table 5.2 Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case**  **ID** | **Test Case Execution Steps** | **Expected Result** | **Tester** | **Test Result** |
| TC1 | Front End | Fully working front end | Prithviraj | Pass |
| TC2 | Upload dataset | Dataset should be uploaded successfuly | Prithviraj | Pass |
| TC3 | User requirements satisfied | User should be able to choose ML problem, required output column along with required rows and columns of the dataset | Prithviraj | Pass |
| TC4 | Data validation on the website | Error should be shown when email format is wrong or no columns are chosen for the input | Prithviraj | Pass |
| TC5 | Categorical Data | Categorical data should be dealt with during regression | Keertana | Pass |
| TC6 | Custom Model | Custom Algorithm Selection Process should choose algorithm with the highest accuracy | Keertana | Pass |
| TC7 | EvalML Model | EvalML library's should choose it's algorithm with the highest accuracy | Keertana | Pass |
| TC8 | Integration of custom model and AutoML model | Algorithm with highest accuracy amongst the custom and EvalML models should be chosen for the final model | Keertana | Pass |
| TC9 | Final Output | Final model's output should be displayed on the web application | Prithviraj  Keertana | Pass |
| TC10 | Email | Email consisting the model should be sent to the user when model is created | Prithviraj | Pass |

33

# CHAPTER 6

**EXPERIMENTAL RESULTS**

## EXPERIMENTAL RESULTS

#### Graphical User Interface

The Graphical User Interface comprises of a web application which provides the user the ability to upload a preprocessed dataset. In the Home page of the website as shown in Figure 4.10, the user is shown the various tasks the website can perform. This includes the Data Preprocessing functionality, the functionality to build a model based on input dataset and a section to go through the complete Machine Learning process. The GUI also provides a side navigation bar which will help the user switch between different sections easily.

Figure 6.1 Shows the home page of the website.

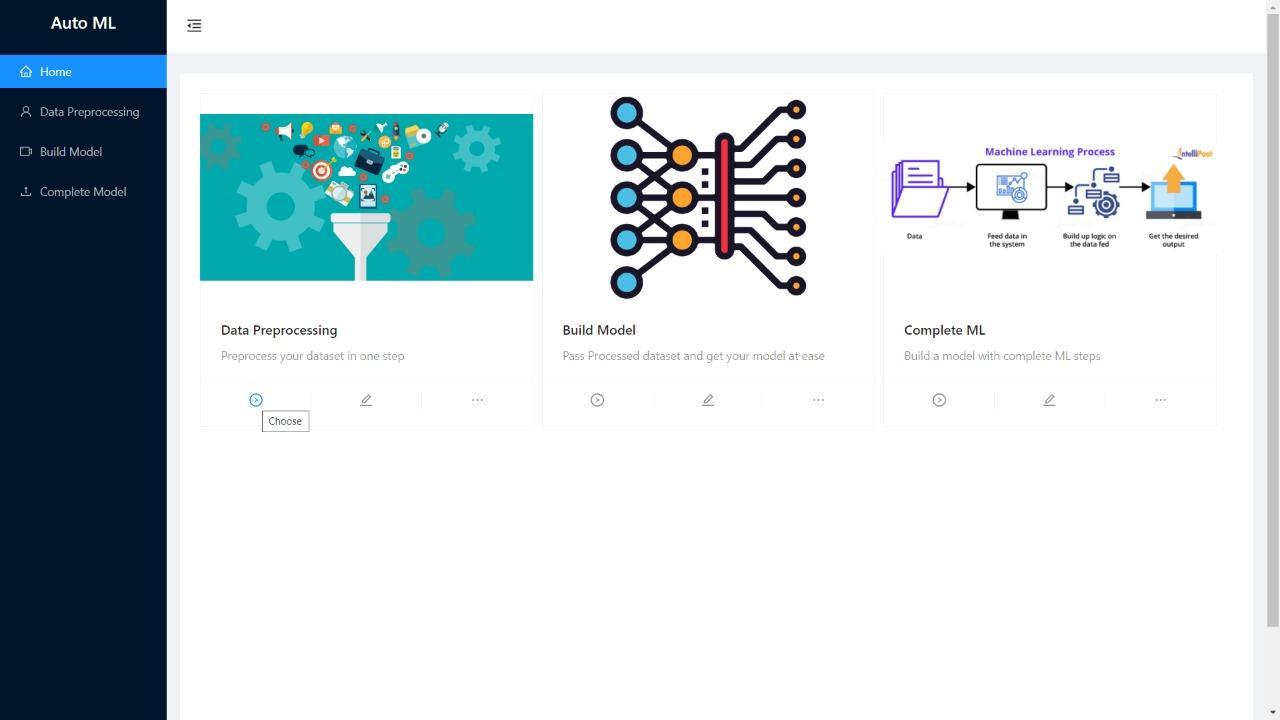


Figure 6.2 shows the “Build Model” tab of the web application wherein the user can upload a preprocessed dataset.

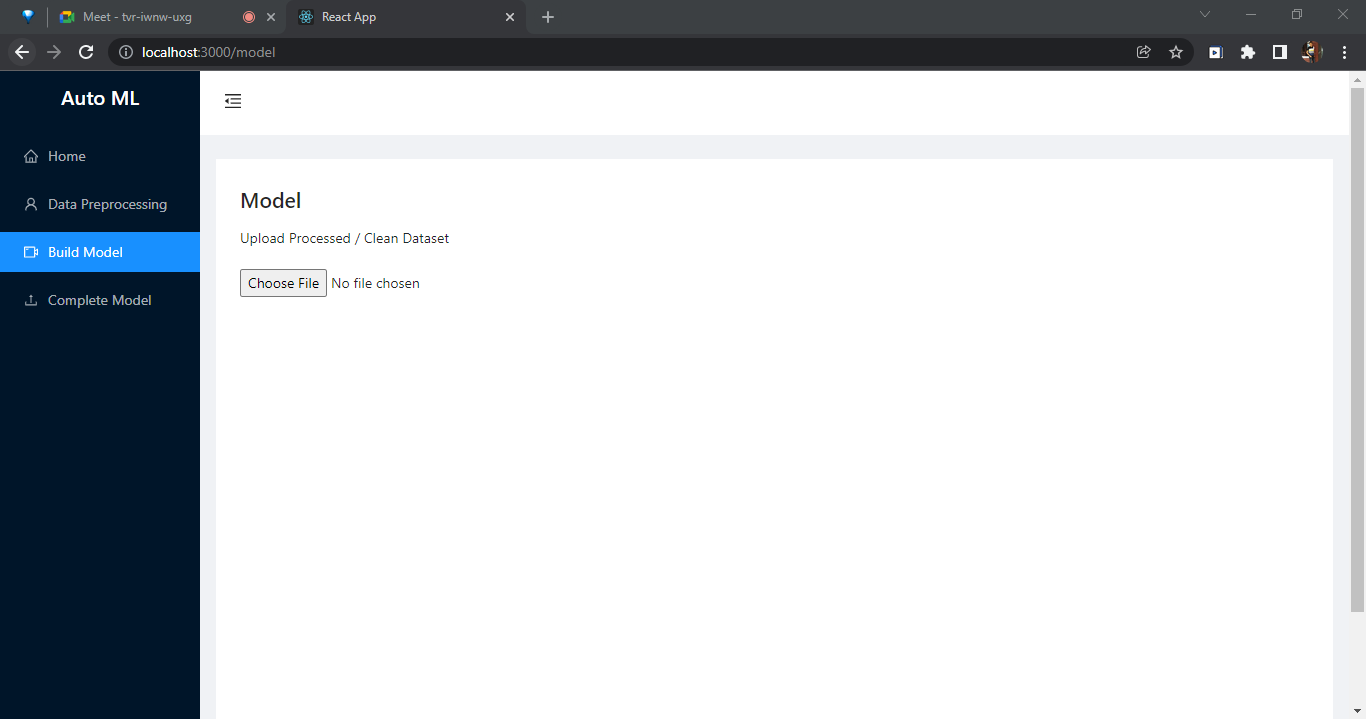


Figure 6.3 shows a user uploading a preprocessed dataset

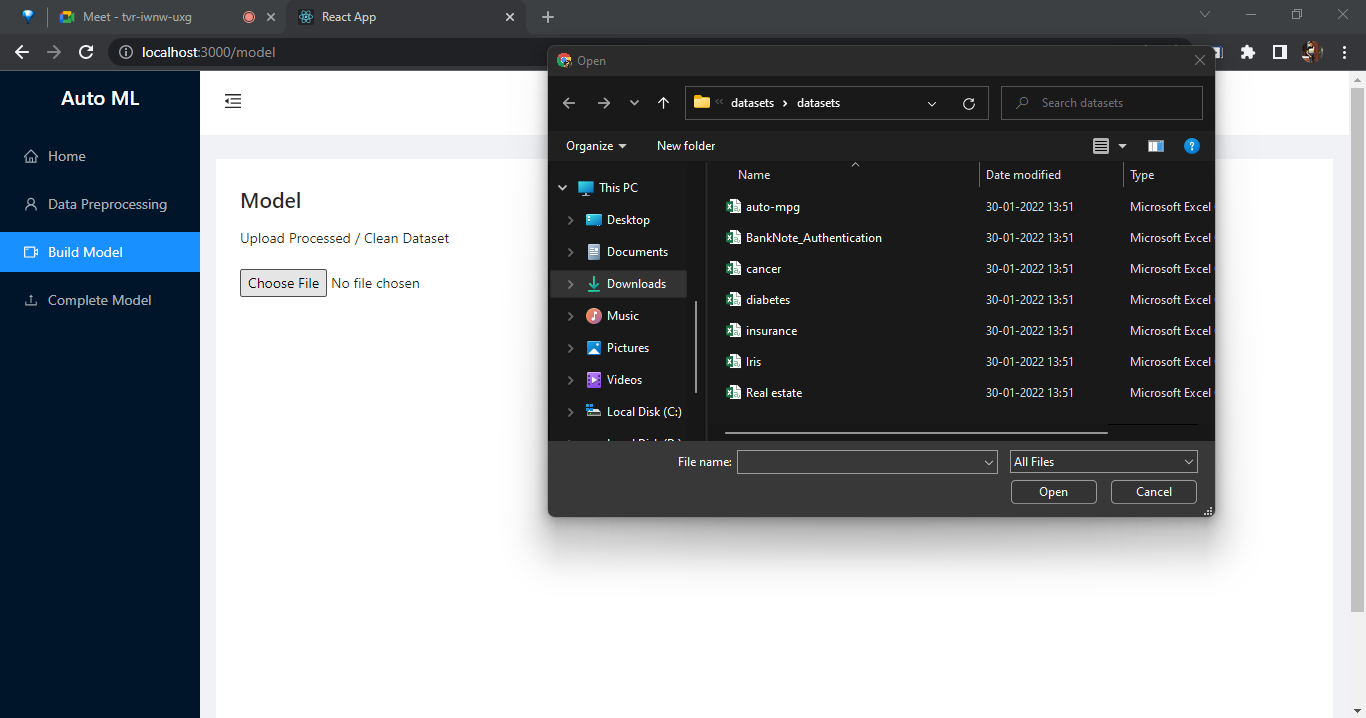


Figure 6.4 shows how the website lets the user choose which Machine Learning problem (Binary Classification, Multiclass Classification or Regression) they want to work on. After this, they can make changes in the dataset as they please and finally, they get to choose which feature of the dataset they want to be the required output of the model.

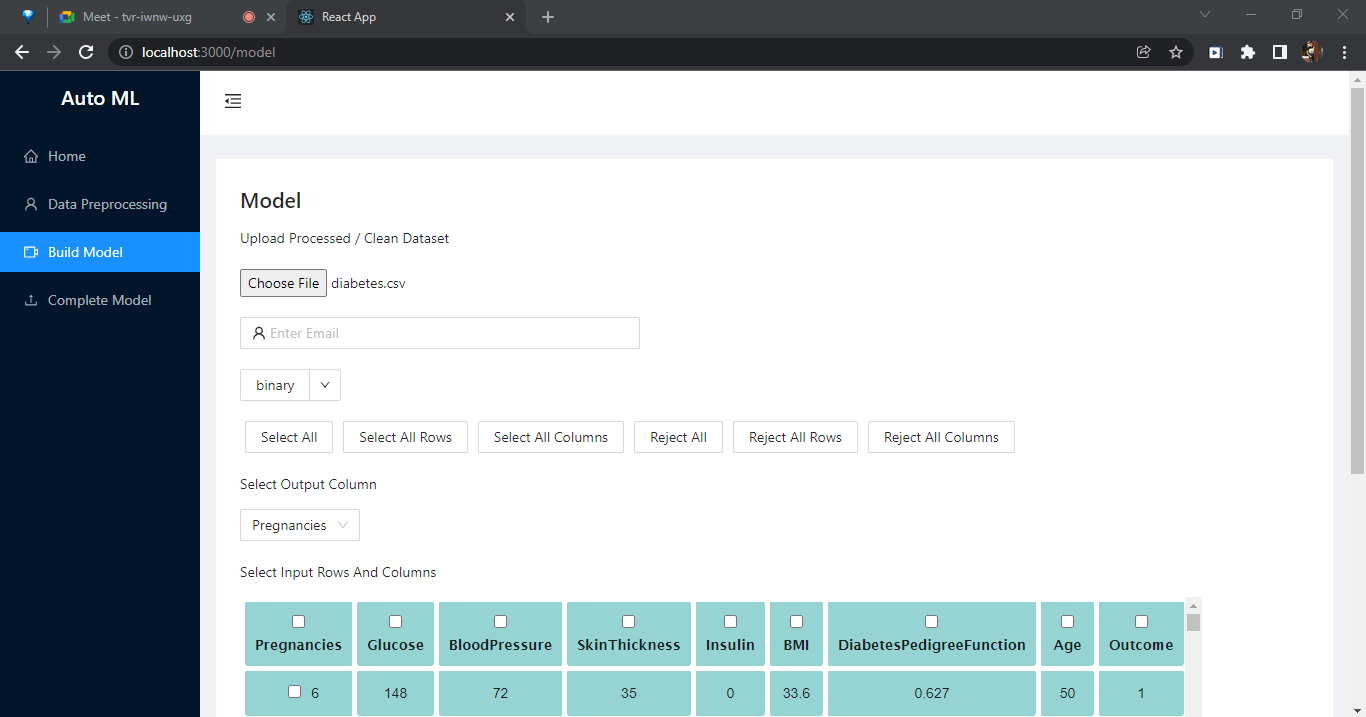


Figure 6.5 shows that the user has chosen Binary Classification and the output column as “Outcome” after providing the website with their email address where they want the final model to be sent.

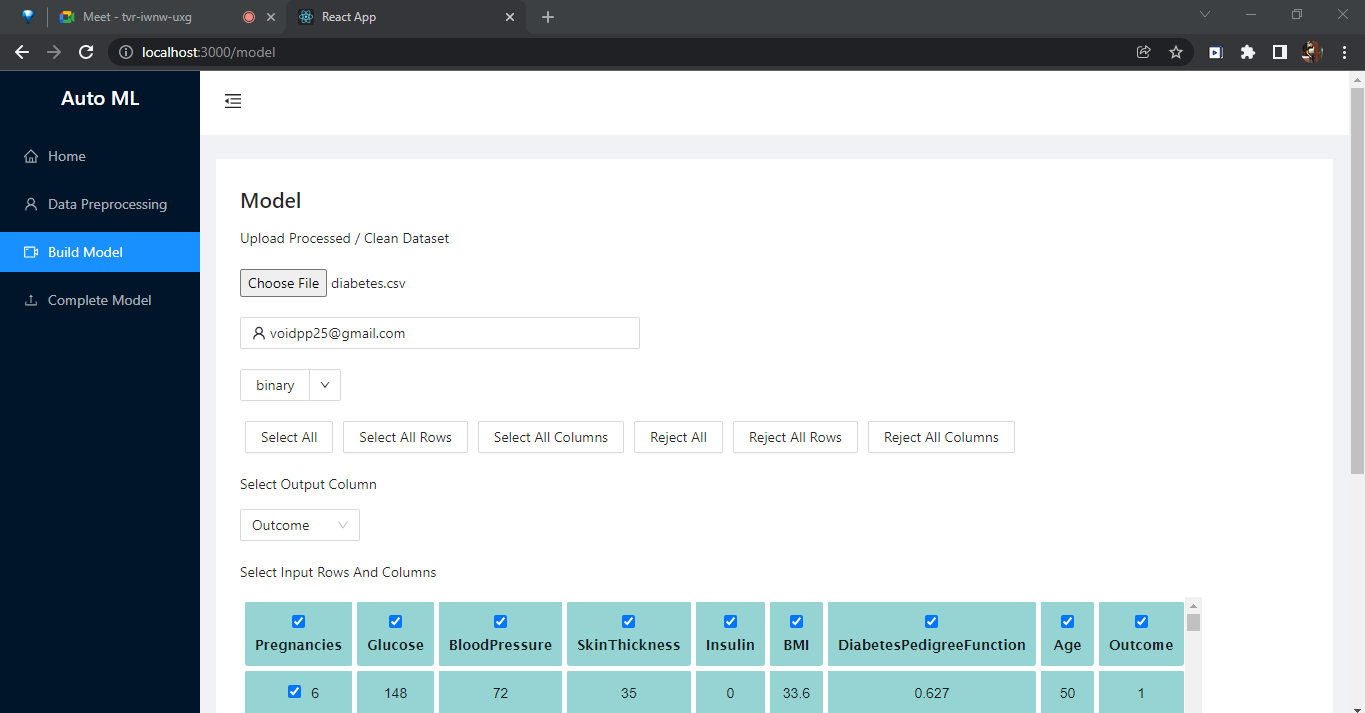


Figure 6.6 shows the validation done for the website wherein the user needs to input a valid email and has to choose input rows and columns to continue working.

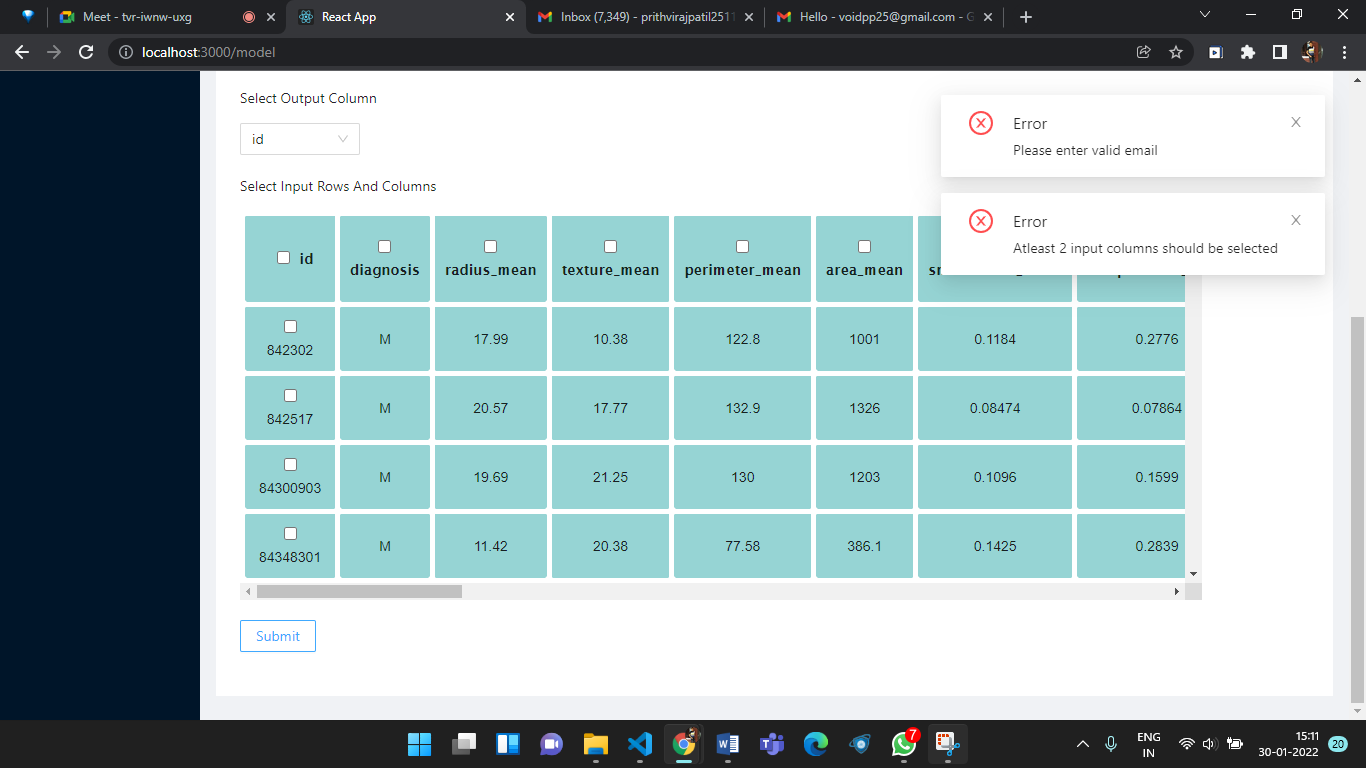


Figure 6.7 shows that the user has chosen all the columns except for “Outcome” as the input features. The user can also choose certain columns and rows to work with if they want.

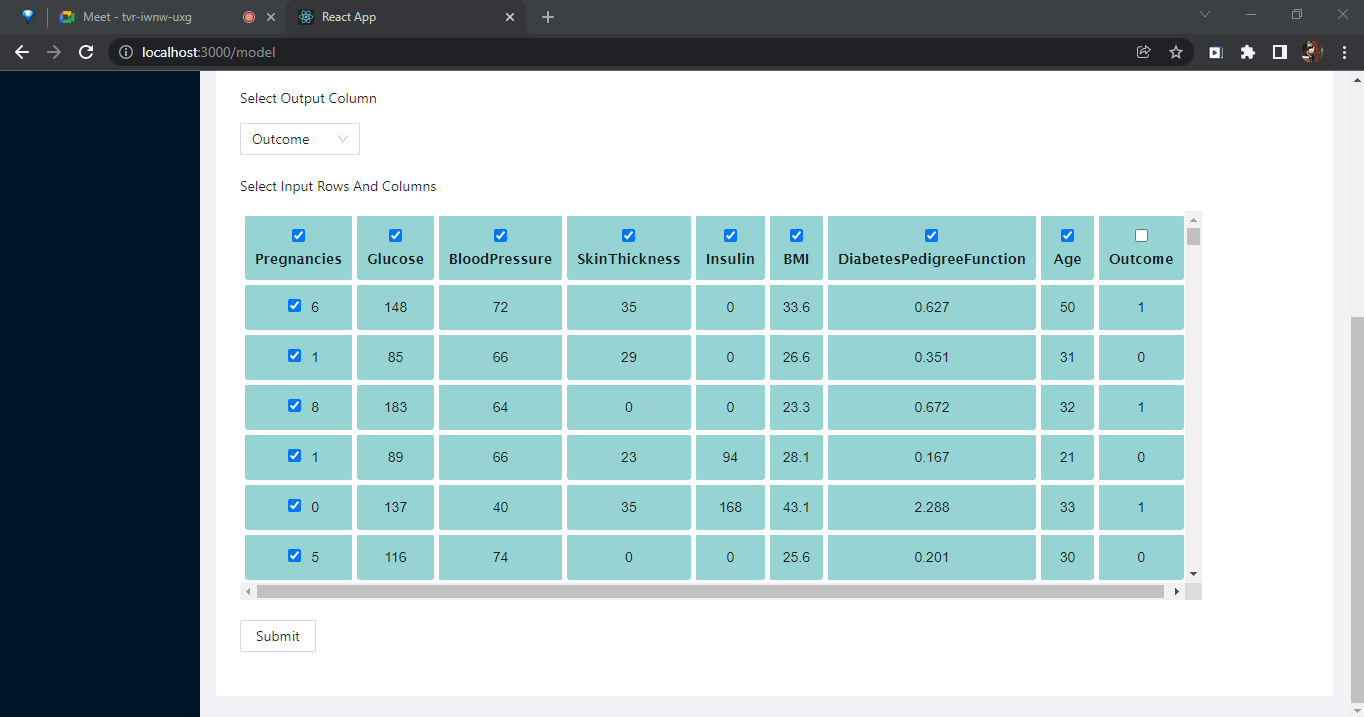


Figure 6.8 shows the output of the model wherein the details of the most accurate algorithm for the given dataset and Machine Learning problem are displayed.

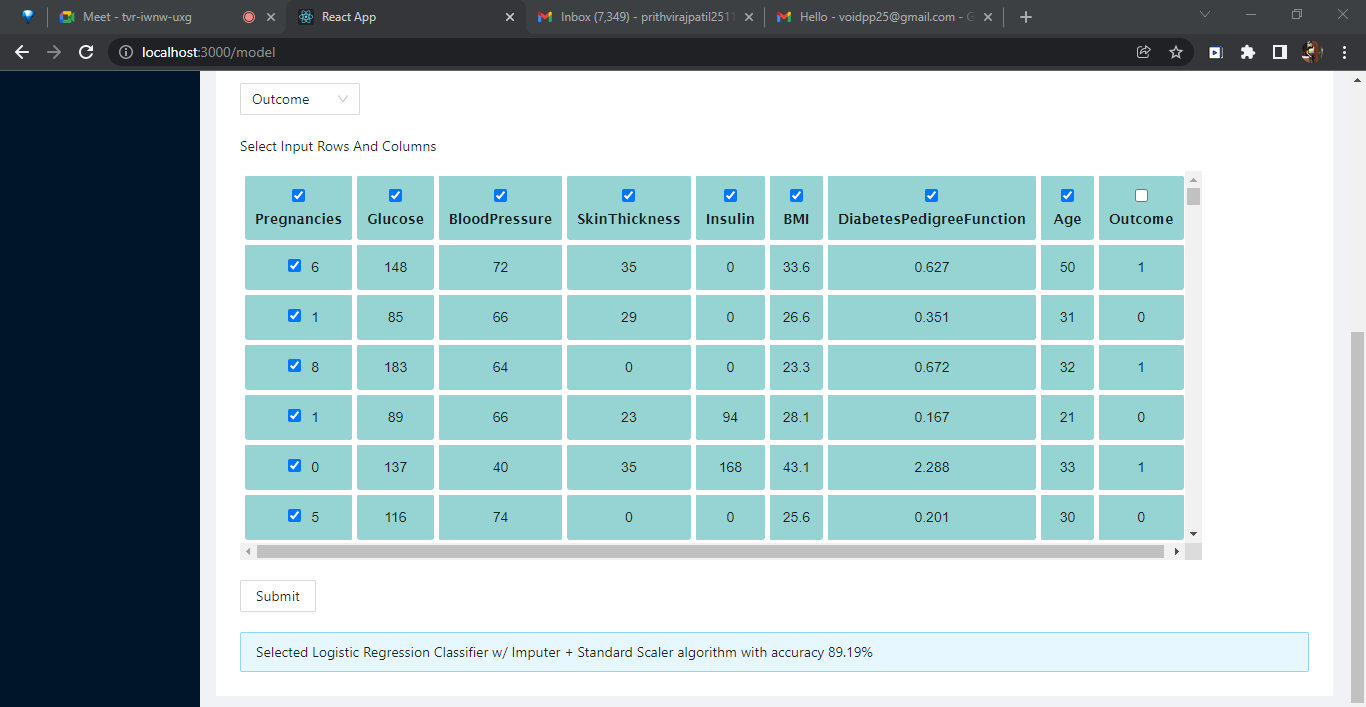
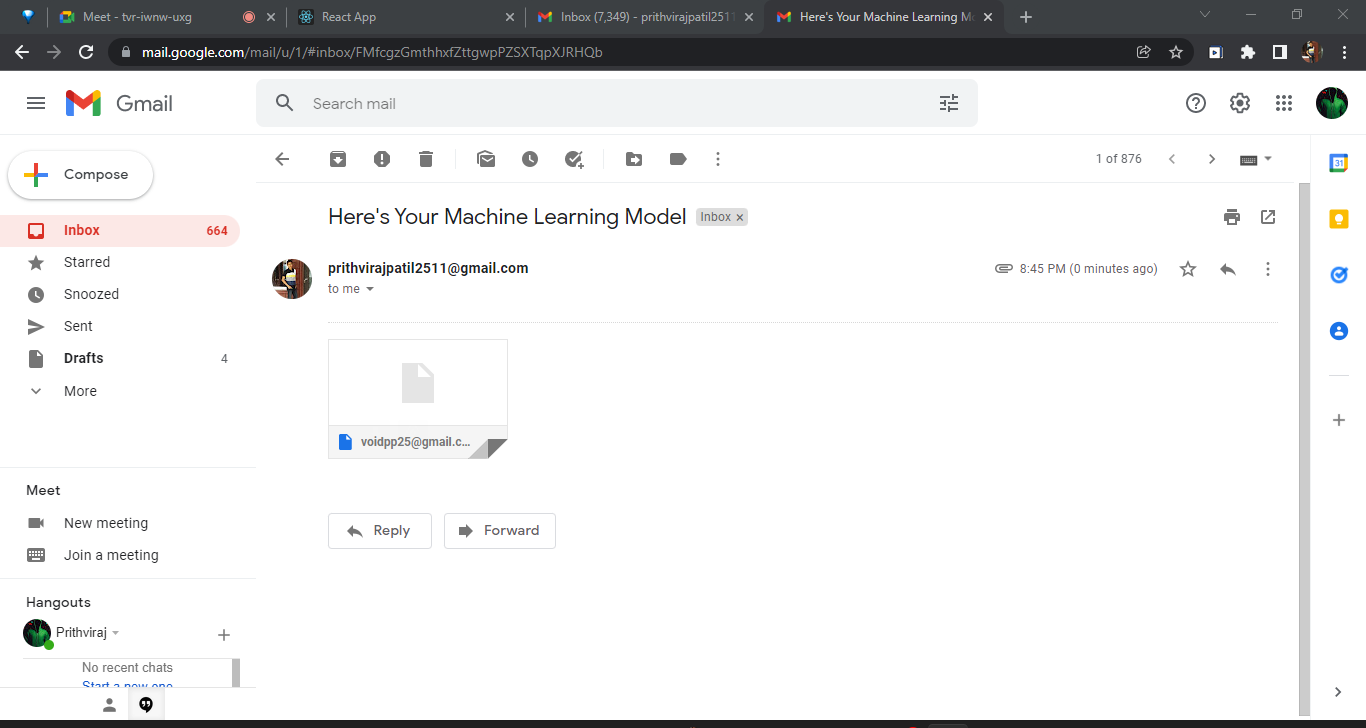


Figure 6.9 shows the email that the user receives after successful model creation done by the website.



#### Result Analysis

On getting the input dataset, the proposed system’s custom algorithm selection process chooses the algorithm with the highest accuracy after training, validation and testing and this algorithm is used to get the final results. The Iris plant dataset from the UCI Repository of Machine Learning Databases [10] contains 150 instances from three classes: Iris-virginica, Iris-versicolor and Iris-setosa. Table 6.1 shows the results of the custom algorithm selection process while performing multiclass classification on the Iris dataset. The algorithm chosen by the custom process of the proposed system for this classification instance is SVC (Support Vector Classifier) as it has the highest accuracy of 95.83% among other algorithms like Logistic Regression, Decision Tree and Random Forest for the given dataset. Fig. 4.8 shows the graphical representation of the results of the custom process.

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| SVC | 95.83 |
| Logistic Regression | 95.00 |
| Decision Tree | 94.99 |
| Random Forest | 93.33 |

Table 6.1: Custom Algorithm Selecting Process Results

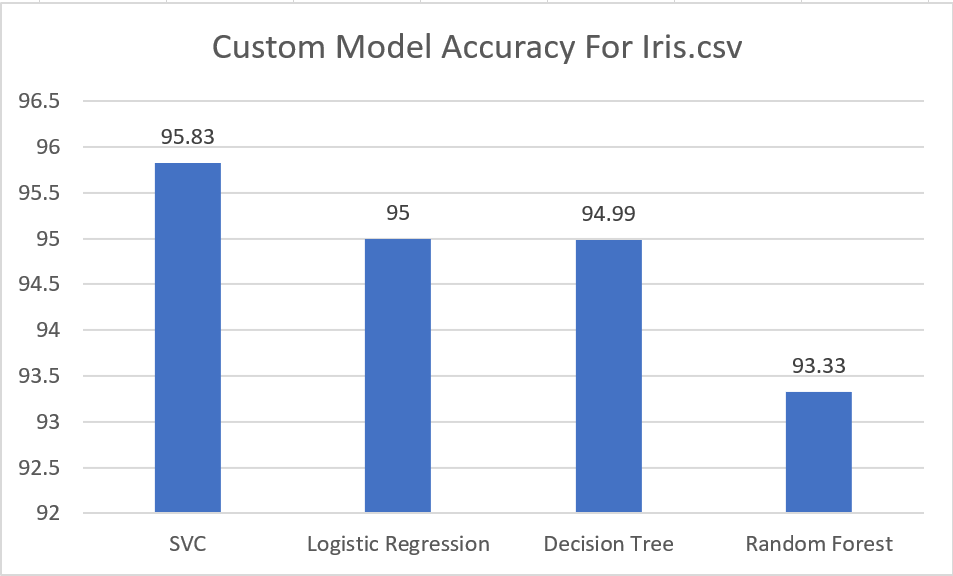


Fig 6.10 Custom Algorithm Selecting Process Results

The proposed system chooses the model with a higher accuracy amongst the models created using the custom algorithm selection process and the AutoML library. Table 6.2 shows the results of both the processes (custom algorithm selection process and the AutoML library) applied on four datasets which are the Iris plant dataset, Bank Note Authentication and Auto-mpg from the UCI Repository of Machine Learning Databases [10] along with the Insurance dataset [11], provided by Kaggle. The comparative results of both the processes are graphically depicted in Fig. 6.11.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Custom Process | EvalML AutoML | Model Chosen |
| Bank Note Authentication | 100 | 61.3 | Custom Model |
| Iris | 95.83 | 100 | EvalML AutoML |
| Auto-mpg | 84.61 | 93.73 | EvalML AutoML |
| Insurance | 91.74 | 82.27 | Custom Model |

Table 6.2: Custom Algorithm Selection Process VS. AutoML Library.

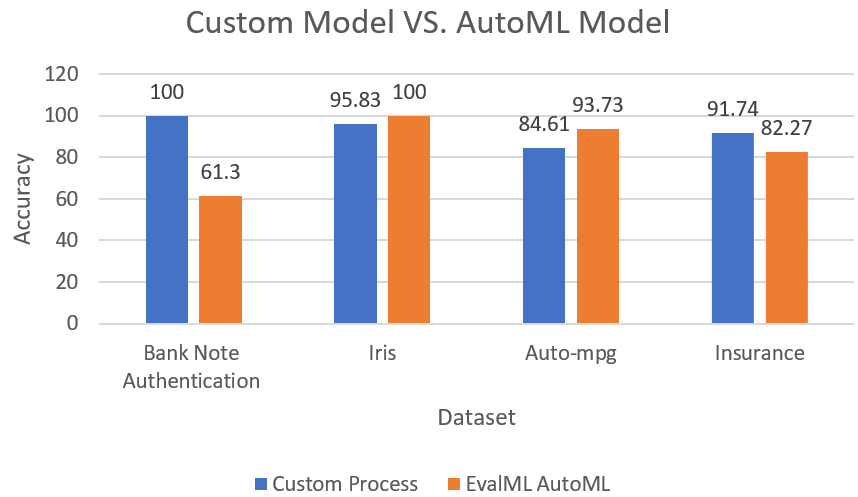


Fig 6.11 Custom Algorithm Selection Process VS. AutoML Library

40

# CHAPTER 7

**CONCLUSION AND FUTURE SCOPE**

## CONCLUSION

‘AMES: AUTOMATED ML EXPERT SYSTEM’ has provided an efficient and much needed solution to combat the many challenges in Machine Learning. The method consists of two major steps - generation of a model using the ‘AutoML’ Library and using a Custom Algorithm Selection Process to find another model with high performance. The results from these two steps are compared and a final output model is generated and saved for future use. It has successfully eliminated the need for human intervention in the model building process and hence reduced the time required by ML users, novices and professionals alike, in the building and deployment of their application. Since the system is designed in the form of a web application, it is easily accessible and it will be of great benefit to Machine Learning Engineers.

## FUTURE SCOPE

1. Support for more Machine Learning tasks - In the modules developed so far support is provided for the Classification task in ML. More tasks like Prediction, Clustering and Pattern mining can also be implemented to make the web application cater to diver se tasks.
2. More options for Algorithms - Each task should be carried with a wide variety of algorithms. This will ensure the optimum result.
3. Time Optimization - The computation time for running several algorithms together needs to be reduced.
4. Language - Currently the system supports building the model in Python Language only. This can be expanded to other languages which also provide ML functionalities like R, Java and C++.
5. Options for deploying the Saved Model - The functionality to easily deploy a saved model can also be provided to the user.

## REFERENCES

[1] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, Si-Hao Deng, Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization, Journal of Electronic Science and Technology, Volume 17, Issue 1, 2019, Pages 26-40, ISSN 1674-862X.

[2] Hospedales, Timothy & Antoniou, Antreas & Micaelli, Paul & Storkey, Amos. (2020). Meta-Learning in Neural Networks: A Survey.

[3] Yang, Chengrun & Akimoto, Yuji & Kim, Dae & Udell, Madeleine. (2019). OBOE: Collaborative Filtering for AutoML Model Selection. 1173-1183. 10.1145/3292500.3330909.

[4] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '13). Association for Computing Machinery, New York, NY, USA, 847–855.

[5] T. Swearingen, W. Drevo, B. Cyphers, A. Cuesta-Infante, A. Ross and K. Veeramachaneni, "ATM: A distributed, collaborative, scalable system for automated machine learning," 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 151-162, doi: 10.1109/BigData.2017.8257923.

[6] J. Liu, S. Tripathi, U. Kurup and M. Shah, "Auptimizer - an Extensible, Open-Source Framework for Hyperparameter Tuning," 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 339-348, doi: 10.1109/BigData47090.2019.9006330.

[7] Witten, Ian & Hall, Mark & Frank, Eibe & Holmes, Geoffrey & Pfahringer, Bernhard & Reutemann, Peter. (2009). The WEKA data mining software: An update. SIGKDD Explorations. 11. 10-18. 10.1145/1656274.1656278.

[8] Lars Kotthoff, Chris Thornton, Frank Hutter. User Guide for Auto-WEKA version 2.6 .

[9] LeDell, Erin. “H2O AutoML: Scalable Automatic Machine Learning.”, 2020

[10] C. Blake, E. Keogh and C.J. Merz, "UCI Repository of Machine Learning Databases", 1998

[11] Kaggle Medical Cost Personal Datasets, [online] Available: http://www.kaggle.com

[12] Dua, D. and Graff, C., “UCI Machine Learning Repository”, 2019

## APPENDIX A: CODE SAMPLE

##### Sample Code:

def getClassificationCustomModel(X,Y):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.33, random\_state=1)

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

classifier1 = SVC(kernel = 'rbf', random\_state = 0)

classifier1.fit(X\_train, y\_train)

classifier2 = LogisticRegression(random\_state = 0)

classifier2.fit(X\_train, y\_train)

classifier3 = DecisionTreeClassifier(criterion='entropy', random\_state=0)

classifier3.fit(X\_train, y\_train)

classifier4 = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 0)

classifier4.fit(X\_train, y\_train)

accuracies1 = cross\_val\_score(estimator = classifier1, X = X\_train, y = y\_train, cv = 10)

Mean1=accuracies1.mean()

accuracies2 = cross\_val\_score(estimator = classifier2, X = X\_train, y = y\_train, cv = 10)

Mean2=accuracies2.mean()

accuracies3 = cross\_val\_score(estimator = classifier3, X = X\_train, y = y\_train, cv = 10)

Mean3=accuracies3.mean()

accuracies4 = cross\_val\_score(estimator = classifier4, X = X\_train, y = y\_train, cv = 10)

Mean4=accuracies4.mean()

Final=max(Mean1,Mean2,Mean3,Mean4)

if(Final==Mean1):

y\_pred = classifier1.predict(X\_test)

algo="Support Vector Classification"

elif(Final==Mean2):

algo="Logistic Regression"

y\_pred = classifier2.predict(X\_test)

elif(Final==Mean3):

algo="Decision Tree"

y\_pred = classifier3.predict(X\_test)

else:

algo="Random Forest"

y\_pred = classifier4.predict(X\_test)

return(algo,Final)

def getRegressionCustomModel(X,Y):

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.33, random\_state=1)

    sc = StandardScaler()

    X\_train = sc.fit\_transform(X\_train)

    X\_test = sc.transform(X\_test)

    regressor1 = LinearRegression()

    regressor1.fit(np.array(X\_train), y\_train)

    regressor2 = RandomForestRegressor(random\_state = 0)

    regressor2.fit(np.array(X\_train), y\_train)

    regressor3 = SVR(kernel='rbf')

    regressor3.fit(np.array(X\_train), y\_train)

    regressor4 = DecisionTreeRegressor(random\_state = 0)

    regressor4.fit(np.array(X\_train), y\_train)

    accuracies1 = cross\_val\_score(estimator = regressor1, X = X\_train, y = y\_train, cv = 10)

    Mean1=accuracies1.mean()

    accuracies2 = cross\_val\_score(estimator = regressor2, X = X\_train, y = y\_train, cv = 10)

    Mean2=accuracies2.mean()

    accuracies3 = cross\_val\_score(estimator = regressor3, X = X\_train, y = y\_train, cv = 10)

    Mean3=accuracies3.mean()

    accuracies4 = cross\_val\_score(estimator = regressor4, X = X\_train, y = y\_train, cv = 10)

    Mean4=accuracies4.mean()

    Final=max(Mean1,Mean2,Mean3,Mean4)

    if(Final==Mean1):

        y\_pred = regressor1.predict(np.array(X\_test))

        algo="LinearRegression"

    elif(Final==Mean2):

        algo="Random Forest Regression"

        y\_pred1 = regressor2.predict(np.array(X\_test))

    elif(Final==Mean3):

        algo="Support Vector Regression"

        y\_pred1 = regressor3.predict(np.array(X\_test))

    else:

        algo="Decision Tree Regression"

        y\_pred1 = regressor4.predict(np.array(X\_test))

    return(algo,Final)

## ACKNOWLEDGEMENT

We would like to express our appreciation to all those who provided us the opportunity to complete this project. We thank Dr. S.M Khot, Principal of Fr. C. Rodrigues Institute of Technology and Prof. Dhanashree Hadsul, H.O.D. of IT department. We are extremely grateful to Prof. Dhanashree Hadsul, Prof. Lakshmi Gadhikar and Prof. Kalpana Wani project coordinators of IT department for extending their invaluable support to us. We would also like to extend our heartfelt gratitude to our project guide Mrs. Poonam Bari, who took keen interest in our work and guided us all along, till the completion of our project by providing all the necessary information for developing a good system. We are thankful for getting constant encouragement, support and guidance from all the teaching staff of the FCRIT Information Technology Department. Finally, we would like to extend our warm thanks to our family members and well-wishers for their continued support.

Yours sincerely,

Prithviraj Patil

Russel Rumao

Keertana Kappuram

x

**64**