Chapter 4

Existing Methodology

4.1 Data Preparation

We make use of this dataset totaling 2,111 samples, collected from 11 different features, in this experiment. The dataset was collected in detail with our larger study on performance and learning behaviors among learners-in-programming.

Our study had 30 participants. All of them were learners who were at the very beginning stage of their programming journey. Thus, they were from various backgrounds and experiences but possessed a common characteristic: that they had just started learning about programming.

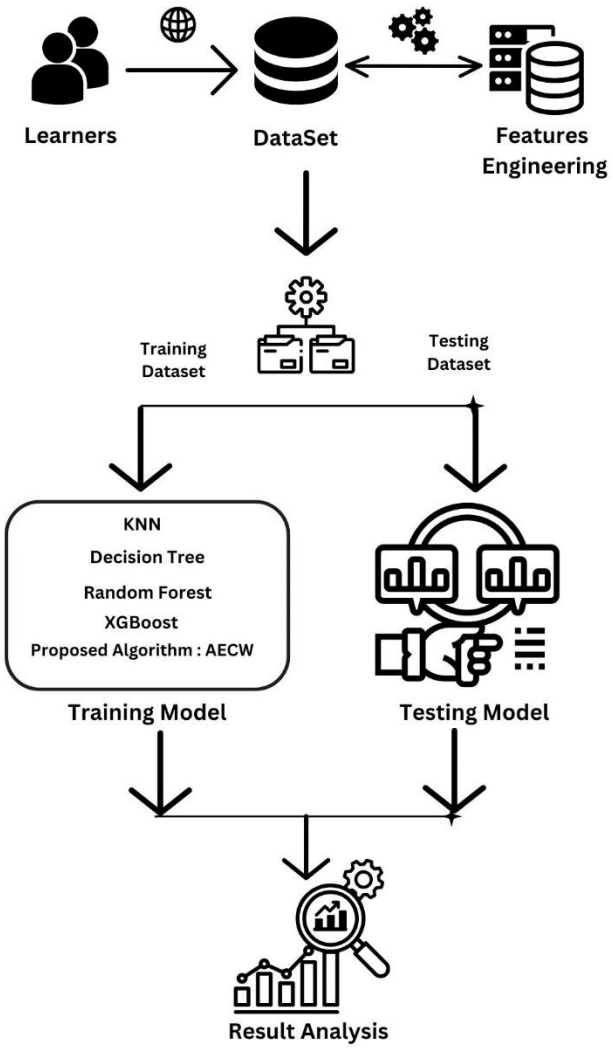

Figure 4.1: Workflow of study

To ensure uniform and relevant data collected, each learner was provided with a customized interface designed specifically to support their programming. This way, the participants could engage with the challenges in programming while systematically recording the interactions, performance metrics, and progress. The results of such interactions were carefully compiled and processed to create the dataset, encompassing all aspects of how learners engaged in programming - whether they completed tasks with specified rates, committed errors with observed frequencies, followed certain coding patterns, or spent time working on these activities. It has all these features aggregated for use in analyzing any difficulty novice learners face during programming and devising ideas on how to improve their learning situation.

This carefully curated dataset is the basis of our research that will allow us to explore and evaluate the effectiveness of various machine learning techniques in predicting and improving the performance of beginner learners.

4.1.1 Data Preprocessing

In this stage of the experiment, we filtered the data set to be more relevant to the objectives of the research. We removed one attribute that we considered to be out of scope of our analysis. We considered the functionality after much thought and noticed that it had no apparent effect or relationship with the aspects that concerned the programming performance that we aimed to study.

To make the dataset more relevant and of greater depth for analysis, we have included four more attributes from the existing ones. The four new ones were computed systematically to give a very fine grain view of the programming behaviors and how well learners performed. The following list reflects these:

Correctness Score (fc(S): This attribute determines the correctness of a program as its ability to satisfy the specified functional requirements. It measures the extent to which the learner's solution meets the requirements of the problem; therefore, it is an objective measure of program correctness.

Error Score of a Program (fe(S)): This is the measure of errors in the learner's program. This includes syntax errors, logical errors, and runtime exceptions. It acts as a diagnostic indicator of the difficulties the learner faces in writing error free code.

Performance Score (p(S)): It combines several metrics such as efficiency, completeness, and adherence to best coding practices. The overall performance of the learner is measured in terms of how well he or she performs on a combination of correctness, error rates, and other performance-related measures. This characteristic is important to apply predictive models and for overall analysis of learner progress.

Final Decision: This characteristic is a synthesized outcome of the above metrics. Here new task will be assigned based on performance score. It will help in classifies the learner's performance into predetermined decision classes. This characteristic is important to apply predictive models and for overall analysis of learner progress.

These calculated characteristics greatly enrich the data set, making it a richer and more informative data set for predictive analysis. Through these supplementary metrics, we hope to gain further insights into how the learning patterns of novice learners can be analyzed and better understood.

4.1.2 Data Splicing

To determine the performances of the machine learning models and their capability of generalization, the given dataset was divided into training and testing sets. A split of 70-30% was used for this division. This would imply that 70% of the data was utilized for training models and 30% for the performance evaluation on data unseen by the model. This approach is one of the standard and well-established methods applied in machine learning research where models are trained on a reasonable amount of data and a significant portion for assessing the ability to generalize to unseen scenarios.

Of the 2,111 records in the dataset, 1,477 records were assigned to the training set, which the models were trained on, to learn patterns and relationships about the data. The training records formed a basis for the learning process of the model, so it could update its parameters and increase its predictive powers.

The testing set consisted of the remaining 634 records that also happened to be an independent validation set. These were not exposed to the model during the training phase and hence formed a fair assessment of the performance of the model. Testing on this subset enabled us to test many important performance metrics such as accuracy, precision, recall, and F1-score, as well as determine the efficiency to which models could predict outcome for new, unseen data points.

We tried to overcome the overfitting phenomenon with this data splitting. In general, overfitting occurs in a model that performs outstandingly well on the training set but fails to generalize or perform well on the unseen data. The division of the data into the training set and the testing set gave a severe test in which to check the working of the models and their viability while ensuring that the results were stable and significant.

4.2 Existing Machine Learning Algorithm

Modi et al. (2024) explained how regression models in machine learning can be used to predict the outcome based on complex data in the health domain, such as asthma control test scores. This approach involves using robust predictive models to analyze key variables that affect test outcomes, which depicts the flexibility of machine learning across different domains. This study underlines the relevance of selecting appropriate machine learning algorithms and features for reliable prediction. A principle that applies to educational contexts, it follows that in predicting the performance of novice learners, techniques such as regression-based approaches or other machine learning algorithms can provide insight into the learning trajectory of individual learners and help to target interventions and improve their performance. Applying analogous methodologies, educational researchers can achieve comparable success in predicting and improving programming skill development [29].

We used and exhaustively analyzed four popular machine learning models: KNN, Decision Tree, Random Forest, and XGBoost, to predict the final decision from the preprocessed dataset. Based on their characteristics and strengths, we chose one for each, which ensured an all-around evaluation of the capabilities of the models in predicting as well as suitability in handling the complexities associated with our dataset. We have used weighted average method for calculation of following indicators.

4.2.1 KNN Algorithm

One of the simplest yet powerful algorithms in classification is the KNN model. Being inherently proximity-based as a predictor made it a good candidate to help with understanding the importance of feature distributions and data density. Throughout the experiment, much attention had to be paid to tuning the number of neighbors, k, since this number did much to influence the ability of the model to generalize. Smaller values of k made the model sensitive to local patterns, hence, it might have captured subtleties in the data, while for larger values, it smoothed the predictions and prevented overfitting. The KNN algorithm provided insights into the natural structure of the dataset and the importance of its features, serving as a useful benchmark with which to compare the performance of more complex models.

4.2.2 Decision Tree Algorithm

Another very important component to our assessment was the interpreted, easy-to-implement Decision Tree model. At every node of a decision tree in its hierarchy, the samples get split based on one of the feature values, according to which the prediction or further decision is made regarding which target class the sample needs to be classified into; and by experimentation, we opted to tune critical hyperparameters such as the maximum depth, minimum samples at every split, and another common splitting criterion, Gini impurity or entropy. These adjustments allowed our control of the tree's properties, tuning between accuracy and generalizations. The Decision Tree provided an extremely good performance though with interpretable predictions; additionally, it was prone to regularize overfitting judiciously.

4.2.3 Random Forest Algorithm

Random Forest, an extension to the Decision Tree for the ensemble-based approach, had been used to leverage from both bagging and random feature selection. It achieved strong predictions by training many decision trees on random subsets of the data and then aggregating them, overcoming overfitting. These include the optimization of the number of trees, maximum depth, and the number of features to consider for splits during the tuning of parameters. Because of the complexity of the interactions it captured, this model didn't lose stability on a wide range of feature interaction distributions. Random Forest is reliable and versatile because it is ensemble in nature, hence the consistency in performance on different distributions of data.

4.2.4 XGBoost Algorithm

XGBoost was an advanced boosting algorithm that was developed to reach the best accuracy and efficiency. It has developed exceptional proficiency in recognizing subtle patterns in data through iterative construction of weak learners and optimization of residual errors. Such sophisticated features as regularization, weighted sampling, and efficient parallel computation were given a serious boost in terms of predictive performance. Many hyperparameter tunings are given by XGBoost, including learning rate, depth, or number of estimators, or regularization terms given and played the most important roles in optimized performance. Considering that it has the possibility to iteratively reduce its bias and variance until attaining a low error rate possible, XGBoost became one of the best performances in our experiments.

4.3 Tools and Library Used

For performing all machine learning algorithm Google Colab platform is used in this study. So no additional configuration or set need to create in local system. Dataset is kept in csv format which can be easily uploaded and used with colab [63]. Python 3.0 is used by google colab for executing python code.

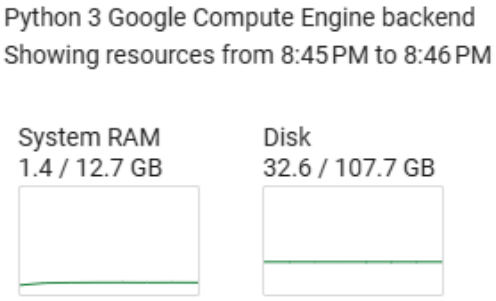

Figure 4.2: Colab Environment Setup

Following are the introductions of library used in implementations of machine learning algorithms in python.

Pandas: The pandas library is a staple in Python when it comes to manipulating and analyzing data. Its data structures are primarily of two types: DataFrame and Series. They allow organizing, filtering, and transformation of data. With its functions for grouping, merging, and reshaping, pandas simplify complex workflows on data. It allows reading and writing of files from multiple file formats such as CSV, Excel, and SQL databases. Its built-in methods for missing values, statistical analysis, and custom functions make the preprocessing process easier. Data science has widely adopted pandas in combination with other Python libraries like NumPy and Matplotlib, allowing efficient exploration and visualization of data.

matplotlib.pyplot: Matplotlib.pyplot is a general-purpose library for creating static, animated, and interactive visualizations in Python. It has plot types such as line, bar, scatter, and histogram to represent data properly. It provides control of figure properties such as axes, colors, labels, and legends, which makes customization possible according to needs. Matplotlib seamlessly integrates with other libraries such as pandas and NumPy. Users can easily create visualizations directly from structured data. It is widely used for exploratory data analysis, where visual insights are necessary to understand patterns and trends. Although it has a steep learning curve for more advanced features, Matplotlib remains a favorite for flexibility and comprehensive documentation.

Numpy: NumPy is an acronym for Numerical Python, it is a fundamental library used in Python for numerical computations. It introduces the concept of ndarray object that efficiently allows the storage and manipulation of big multi-dimensional arrays and matrices. NumPy offers all kinds of mathematical operations that make it a must tool in scientific computing, including linear algebra, Fourier transforms, and statistical functions. It also provides tools for generating random numbers essential for simulations and probabilistic modeling. NumPy is optimized for performance, using C and Fortran libraries to make computations fast. Seamless integration with libraries like pandas, Matplotlib, and Scikit-learn ensures that NumPy will play a central role in Python's data science ecosystem.

sklearn.ensemble.RandomForestClassifier: The RandomForestClassifier from sklearn.ensemble may be a powerful machine learning algorithm that is based on the group learning method. It creates many choice trees during training and ranks the direction with the most votes for classification tasks. This classifier has many positive aspects such as transposing the high-dimensional data rather than high-dimensional datasets to the lower memory of the computer, robustness to outliers and noise, and of course, it is basically a noise reduction technique. Random Forest is highly praised due to its recovery of information from lost data, its support of both the categorical and numerical features, as well as its capability to estimate feature importance. Its parallelizable nature makes it computationally efficient for large datasets. It also contains features through which you can set the importance of each feature accurately, which in turn enables feature selection and model interpretability. Even in a noisy environment, the algorithms clustering nature guarantees accurate predictions and thus becomes a popular solution for various real applications.

sklearn.tree.DecisionTreeClassifier: The DecisionTreeClassifier, which is constructed from the sklearn.tree library, is a simple but very performance wise machine learning algorithm commonly used for the choice of a class or classification tasks. It is built up of nodes which are accompanied by dividing mathematical formulae to classify the new examples on the basis of prevailing

nodes. It uses our raw attributes to separate our classes best and it also use information gain to Informative the process of learning too and of course, it uses Gini entropy which is a measure which shows how probable it is that a choice with a given attribute will be correct if we have a decent sample as the child f a particular parent. Such a system can be easily explained to the lay man because it displays graphs to show the outcomes of decisions, which means it is a preferred option for expert-aided decision making. The case of dealing with factors in the form of objects instead of numbers does not pose an impediment to decision trees. Additionally, the data preprocessing phase is not so extensive when we are operating with decision trees. However, if we try to fit a decision tree with many levels, we will face the problem of overfitting and it will not be able to be used for only the data that contains a high noise level. In such cases it becomes inevitable to assign the hyperparameters properly such as max_depth or min_samples_split. On the other hand, the use of the DecisionTreeClassifier as a basic building block for the more advanced methods like RandomcfForest or Gradient Boosting must be also mentioned.

Seaborn: Seaborn is a Python information visualization library built on best of Matplotlib, offering a straightforward and stylishly satisfying way to form measurable illustrations. It rearranges complex plots like heatmaps, combine plots, and violin plots with fair some lines of code. Seaborn exceeds expectations at visualizing information disseminations and connections, with built-in bolster for categorical and numerical information. It can be integrated with pandas DataFrames, which allows coordinate plotting of data without broad preprocessing. Seaborn has a default set of topics and color palettes that enhance the visual offer of charts, making it suitable for introductions and reports. It is broadly used in exploratory data analysis to identify patterns, relationships, and trends in data.

Xgboost: XGBoost can be a high-performance machine learning library for slope boosting. It is distinguished by its speed and competency. Xgboost is capable of supporting classification, relapse, and positioning errands. It makes utilization of advanced strategies, including regularization, weighted quantile portrayal, and parallelized tree building, making it more intense than ordinary calculations for

slope boosting. The presentation is highly customizable concerning hyperparameters for fine-tuning demonstrate execution. It handles lost values intellectuals and gives built-in back for positioning highlight significance. It's broadly utilized in competitive machine learning challenges, where it ensures tall precision and adaptability when managing with expansive datasets.

sklearn.metrics.confusion_matrix: The confusion_matrix work from sklearn.metrics may be a effective apparatus for assessing classification models. It gives a outline of forecast comes about by comparing the real and anticipated names in a lattice arrange. The network comprises four key values: Genuine Positives (TP), Untrue Positives (FP), Genuine Negatives (TN), and Wrong Negatives (FN). These values frame the premise for calculating other execution measurements like accuracy, review, and F1 score. Disarray frameworks are especially valuable for recognizing demonstrate inclinations and understanding classification mistakes, such as when untrue positives are more basic than untrue negatives. Its visual representation helps in surveying show execution and optimizing limits for way better classification results.

sklearn.metrics (Precision, Recall, F1-Score, Accuracy): Precisely, recall, f1-score, and accuracy-score functions in sklearn.metrics offer a more detailed evaluation of the classification model. Precision refers to the proportion of true positives out of the positive predictions made; this reflects the importance of the predictions. Recall computes the capability of the model to classify all actual positive instances. The F1-score is a harmonic mean between precision and recall that measures the relative weight of false positives versus false negatives. The most basic metric among these calculates the proportion of correct predictions out of total predictions. These metrics are imperative to understanding the performance of models and their various contexts, especially the imbalanced dataset.

sklearn.model_selection.StratifiedShuffleSplit: It is one of the techniques for cross-validation offered in sklearn.model_selection and has the quality of stratified splitting; thus, this ensures class label distributions remain invariant to the way that both train and test sets are developed. They find especial usage for unevenly

represented data. As these classes remain balanced because it stratifies them. The shuffling process will result in a randomized split, avoiding overfitting to the specific data patterns. It is a critical tool in ensuring fair and unbiased validation in machine learning workflows.