

AI-Based Student Campus Placement Prediction System A Web Application for Predicting Student Placements Using Machine Learning Algorithms

Snehal Nandkishor Dhulgande¹, Ankita Tirupati Bansode², Sandhyarani Navnath Bansude³, Jyoti Dnyaneshwar Gadadare⁴, Mrs. Tanuja S Dhage⁵

^{1,2,3,4}*Dattakala Group of Institutions, Faculty of Engineering Swami-Chincholi, Daund, Pune*

⁵*Department of Information Technology, Dattakala Group of Institutions,
Faculty of Engineering Swami-Chincholi, Daund, Pune*

Abstract—Campus placement is an important stage in the transition from higher education to employment. Students often want to understand how their academic and experience-related profiles may relate to placement outcomes, while institutions may seek ways to use historical records to support career guidance. This paper presents a Campus Placement Predictor web application that uses machine-learning classification to estimate a placement outcome from selected student attributes. The project presentation identifies age, gender, academic stream, number of internships, CGPA, hostel status, and history of backlogs as input features, with “placed or not” as the target label. The proposed implementation uses Python and machine-learning libraries such as NumPy, Pandas, Matplotlib, and Scikit-learn, with Flask connecting the prediction model to a frontend developed using HTML, CSS, and JavaScript. The candidate classifiers are Logistic Regression, Support Vector Classifier, K-Nearest Neighbours, Decision Tree, Random Forest, and Gradient Boosting. The intended process is to inspect and prepare the data, train candidate models, compare their performance, select an appropriate model, and integrate it into the web application. The supplied presentation does not provide the dataset size, data source, train/test split, measured metrics, or final best-performing classifier. Therefore, this paper explains the system and evaluation plan without inventing numerical results. The application should be understood as an advisory tool for exploring patterns in data, not as a guarantee of an individual student’s employment outcome.

I. INTRODUCTION

Campus placement activities connect students completing their studies with organizations recruiting entry-level employees. In a competitive employment

environment, students may be interested in understanding which parts of their profiles are represented in historical placement records. Educational institutions may also wish to review placement trends to plan training, internships, technical workshops, and career-support activities.

Machine learning offers methods for finding patterns in labelled data and using those patterns to classify new examples. In a placement prediction task, a model is trained on records where student-related attributes and a known placement outcome are available. After training, the model can receive a new set of input attributes and estimate the likely class. The quality of this estimate depends on the quality and representativeness of the dataset, the way the data are prepared, the selected algorithm, and the evaluation process.

The project described in the supplied presentation is a web-based Campus Placement Predictor. It includes a landing page, a form page for collecting student information, and an output page that displays the prediction in simple language. HTML, CSS, and JavaScript are identified for the frontend, while Python and Flask are identified for the backend. Several classifiers are considered so their performance can be compared rather than relying on a single algorithm without evaluation.

This paper expands the project description into a structured research-paper format. It distinguishes the proposed design and intended methodology from measured experimental findings that were not supplied in the presentation.

II. PROBLEM STATEMENT

Students may find it difficult to interpret how a combination of academic and experience-related information corresponds to historical placement outcomes. Institutions may have placement records but lack a simple interface through which students can explore a model-generated estimate. The project addresses the problem of designing a web application that accepts selected student attributes and uses a trained classification model to predict a placement category.

The system does not directly secure a job or determine a student's actual future. Recruitment outcomes depend on additional factors such as employer-specific requirements, interview performance, communication, portfolios, market conditions, vacancies, and selection policies. The prediction is therefore an estimate based on the patterns learned from the available data.

III. MOTIVATION

The project is motivated by the possibility of making machine-learning concepts accessible through a practical web application. A prediction interface can demonstrate how data are collected, processed, passed to a trained model, and returned as a result. Comparing multiple algorithms also provides an opportunity to study the strengths and limitations of different classification approaches.

For students, such a tool may encourage reflection on their profiles and prompt discussions about skill development, internships, and academic preparation. For educational institutions, aggregated and responsibly handled results may support planning of placement-related training. These potential benefits depend on careful evaluation and should not be interpreted as proof that any single feature causes placement success.

IV. AIM AND OBJECTIVES

The aim of the project is to develop a web-based student placement prediction system using machine-learning classification and to compare candidate models before selecting one for application integration.

- To understand the student attributes included in the available dataset.

- To inspect, clean, and prepare the data for supervised learning.
- To explore patterns and distributions through exploratory data analysis.
- To train and compare Logistic Regression, Support Vector Classifier, K-Nearest Neighbours, Decision Tree, Random Forest, and Gradient Boosting classifiers.
- To evaluate candidate models using suitable classification metrics rather than relying only on accuracy.
- To develop a user-friendly frontend using HTML, CSS, and JavaScript.
- To display the model output in clear language and explain that the result is an estimate.

V. SCOPE OF THE PROJECT

The scope described by the presentation includes a web interface, student-information form, Python/Flask backend, dataset-based prediction, and comparison of multiple machine-learning classifiers. The listed input variables are age, gender, stream, internships, CGPA, hostel status, and history of backlogs. The target is placement status.

The supplied presentation does not specify the dataset provider, number of records, exact data collection procedure, application hosting environment, or measured model results. This paper does not assume those details. They should be documented from the actual implementation before final academic submission.

VI. LITERATURE SURVEY

The project presentation lists four related publications concerning placement prediction, placement analysis, and comparison of machine-learning models. The titles suggest that the project belongs to a wider area of research that applies classification methods to student placement data. The presentation does not include the authors, detailed abstracts, experimental settings, datasets, or reported findings of these publications; accordingly, the following summaries are limited to the bibliographic information shown in the source presentation.

6.1. Placement Prediction and Analysis Using Machine Learning

The presentation lists this work in the International Journal of Engineering Research & Technology (IJERT), Special Issue, 2022, ISSN 2278-0181. Based on its title, the work concerns analysis and prediction of placement outcomes using machine learning. The precise algorithms, dataset, and findings are not provided in the supplied material and should be checked from the original paper before drawing detailed comparisons.

6.2. Students Placement Prediction System

The presentation lists this work in the International Journal for Research in Applied Science & Engineering Technology (IJRASET), November 2022, ISSN 2321-9653. Its title indicates a student placement prediction system. The presentation does not state its system architecture, input variables, evaluation measures, or model performance.

6.3. Placement Prediction Using Various Machine Learning Models and Their Efficiency Comparison

The presentation lists this work in the International Journal of Innovative Science and Research Technology (IJISRT), dated 20 May 2020, ISSN 2456-2165. The title indicates a comparison of multiple machine-learning models for placement prediction. Such a comparison is relevant to the present project's plan to evaluate several classifiers using consistent test data and metrics. Specific reported results cannot be inferred from the title alone.

6.4. Prediction of Student Placement Using Machine Learning Algorithm

The presentation lists this work in the International Journal of Creative Research Thoughts (IJCRT), dated 5 May 2020, ISSN 2320-2882. Its title concerns predicting student placement with a machine-learning algorithm. The supplied presentation does not provide sufficient information to identify the algorithm or verify the reported results.

6.5. Research Gap and Project Relevance

The presentation's literature list and proposed classifier comparison indicate an interest in applying data-driven classification to placement outcomes. The project's practical emphasis is to connect the prediction model to a simple web application so a user can submit information and receive a result. A stronger

final literature review should retrieve and read the original papers, record their authors and full publication details, compare their datasets and methods, and identify a specific gap supported by evidence.

VII. PROPOSED SYSTEM

The proposed system accepts student information through a web form and returns a predicted placement category. The application is described as having three visible stages: a landing page, a form page, and an output page. The model is trained separately using labelled historical data; the web application then loads the trained model and uses it to make predictions for submitted inputs.

7.1. Frontend

The frontend is identified in the presentation as HTML, CSS, and JavaScript. HTML defines page elements such as headings, labels, input controls, and buttons. CSS controls layout, spacing, typography, and visual appearance. JavaScript can support client-side form interactions and input checks. The landing page introduces the application, the form page collects the selected attributes, and the output page displays the prediction.

7.2. Backend

The backend is identified as Python with Flask. Flask can receive a submitted form request, validate and convert values into the expected format, call the trained machine-learning pipeline, and return a rendered result page. The backend should reject invalid values and handle missing or unexpected inputs safely. The same preprocessing rules used during training must be applied during prediction.

7.3. Machine-Learning Component

The machine-learning component is responsible for learning relationships between the input features and the known placement label. The presentation identifies Scikit-learn as a relevant library and lists six classifiers for consideration. The final model should be selected based on measured validation performance and the practical purpose of the application.

7.4. System Workflow

- The user opens the landing page.
- The user enters the requested attributes on the form page.

- The application checks that the submitted values are present and valid.
- Flask converts the values into the format expected by the trained model.
- The model returns a predicted class, and, if supported and calibrated appropriately, a probability estimate.
- The output page displays the result in understandable language.
- The application avoids presenting the estimate as a guaranteed placement outcome.

VIII. SYSTEM ARCHITECTURE AND DESIGN

The system architecture consists of three main layers: the presentation layer, the application layer, and the prediction layer. The presentation layer provides the landing page, student information form, and result

page. The application layer uses Flask to receive user requests, validate input data, and manage communication with the machine-learning model. The prediction layer performs the required preprocessing and applies the trained classifier to predict the placement category.

The model-training workflow operates separately from the live prediction process. It includes dataset inspection, data cleaning, train/test splitting, preprocessing, model training, validation, performance comparison, and final evaluation. The selected preprocessing and classification pipeline is then saved for use by the Flask application.

The overall architecture provides a structured connection between the user interface and the machine-learning component, allowing student information to be processed and the predicted placement category to be displayed on the result page.

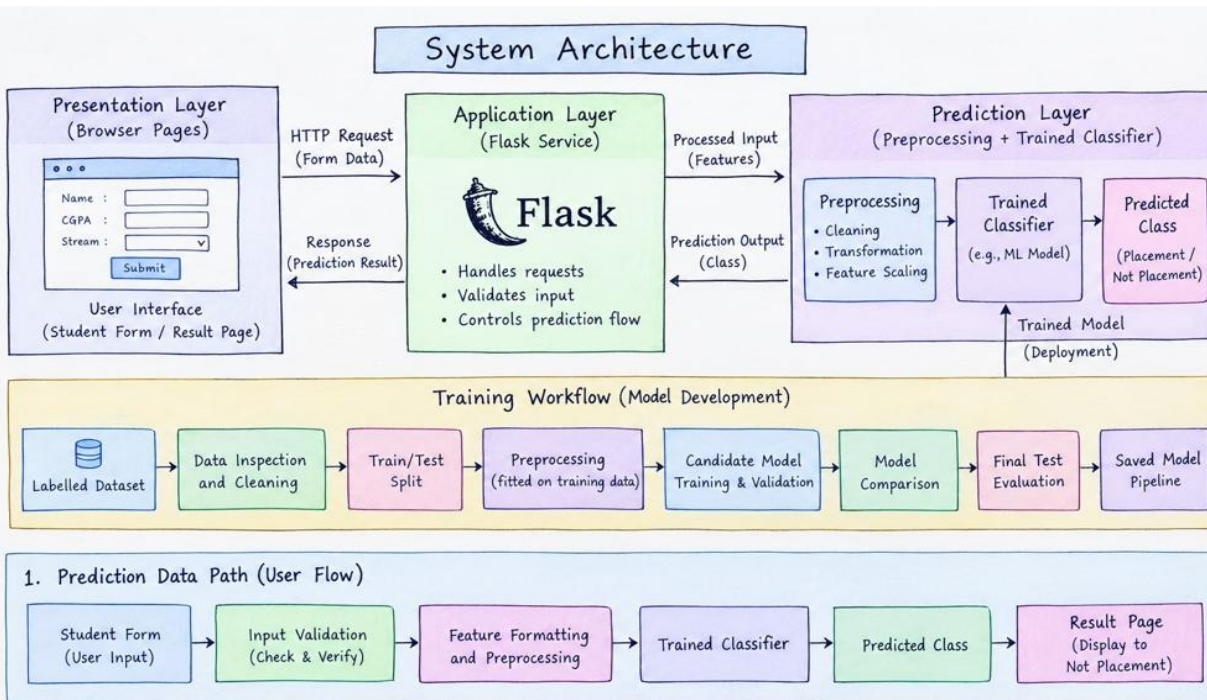


Figure 1: System Architecture of the Campus Placement Prediction System

IX. DATASET DESCRIPTION

The presentation identifies seven potential predictor attributes and one outcome field. The dataset is described as student-related information, but its source, size, collection period, licensing, and privacy

arrangements are not stated. Those details should be recorded from the dataset actually used.

No.	Column	Meaning stated in the presentation
1	Age	Age of the student.
2	Gender	Gender of the student.

3	Stream	Student's academic stream.
4	Internships	Number of internships completed by the student.
5	CGPA	Student's result / CGPA.
6	Hostel	Whether the student stays in a hostel.
7	History of Backlogs	Whether the student had backlogs previously.
8	Placed or not	Whether the student was placed; target label.

In supervised classification, the target column represents the known outcome that the model is trained to predict. It must not be accidentally included among the input features, because doing so would leak the answer into the model. Data rows should be checked for duplicates, missing values, inconsistent spelling of categories, impossible values, and differences in units or formats. The target labels should be checked to understand how many examples belong to each class.

Categorical features such as stream, gender, and hostel status generally need encoding into a numerical representation accepted by the chosen model. Numerical variables such as age, internship count, and CGPA should be checked for valid ranges and appropriate data types. Scaling can be particularly important for distance- or margin-based methods. Preprocessing should be implemented in a reproducible pipeline so training and live predictions use identical transformations.

X. METHODOLOGY

10.1. Data Collection and Understanding

The first step is to identify the actual dataset used by the project and document where it came from, what each column means, how the placement label was recorded, and whether the data can be used for this purpose. The presentation does not provide these details. A final paper should state the record count, time period, inclusion criteria, and any permission or anonymization process.

10.2. Data Cleaning and Preprocessing

The dataset should be inspected for missing entries, duplicate rows, inconsistent categories, incorrect data types, and implausible values. The treatment of missing data should be documented. Categorical fields may be encoded, and numerical fields may be scaled

when appropriate. Preprocessing steps should be fitted only on the training subset and applied to validation and test subsets without refitting, reducing the risk of data leakage.

10.3. Exploratory Data Analysis

Exploratory data analysis (EDA) helps summarize the dataset before model training. It may include counts of placed and not-placed students, distributions of CGPA and internship counts, and comparisons of feature distributions across the target classes. NumPy and Pandas can support data manipulation and summary calculations, while Matplotlib can be used to create charts. Associations found in EDA should be treated as descriptive patterns, not evidence that a feature causes placement.

10.4. Training and Validation

The labelled dataset should be separated into training and test subsets before fitting the final model. If the dataset is small or imbalanced, a stratified split can help preserve class proportions. Cross-validation on the training data can provide a more stable basis for comparing models and tuning parameters. The held-out test set should be used for final evaluation only after model selection is complete.

10.5. Model Comparison

All candidate classifiers should be evaluated on the same split or equivalent cross-validation folds, using consistent preprocessing and a clearly stated scoring procedure. This makes the comparison more meaningful. The model-selection criterion should reflect the use case; for example, recall may matter when missing a student who could benefit from support is a concern, while precision may matter for different kinds of follow-up. The choice should be explained rather than determined by a single metric without context.

10.6. Deployment

After choosing a model, the complete preprocessing-and-model pipeline can be saved and loaded by Flask. The web application should verify form values, return useful messages for invalid submissions, and avoid exposing private student information in logs or pages.

10.7. System Workflow Diagram

The system workflow illustrates the major stages

involved in developing and operating the Campus Placement Prediction System. During model development, the labelled dataset is inspected and cleaned before being divided into training and testing subsets. The candidate classifiers are trained and compared using appropriate evaluation metrics, and the selected model is prepared for deployment.

During application execution, the student enters the required information through the web form. The Flask backend validates the submitted values and passes them through the same preprocessing pipeline used during model training. The trained classifier generates a predicted placement category, which is returned to the user through the result page.

The workflow diagram presents the relationship between model development, model evaluation, deployment, and real-time prediction. It provides a clear representation of the complete process from dataset preparation to the display of the prediction result.

XI. MACHINE-LEARNING CLASSIFIERS

11.1. Logistic Regression

Logistic Regression is a supervised classification method that estimates the probability of a class using a logistic function. It is often used as a baseline because it is relatively straightforward to train and interpret. Its performance depends on how well the relationship between the features and the target is represented by the model and on suitable preprocessing. For this project, it provides a reference point against which more complex models can be compared.

11.2. Support Vector Classifier (SVC)

A Support Vector Classifier aims to find a decision boundary that separates classes with a large margin. Depending on its kernel, it can represent linear or more complex boundaries. Feature scaling and choices such as the regularization parameter and kernel settings can affect performance. Model parameters should be selected using training-only validation rather than the final test set.

11.3. K-Nearest Neighbours (KNN)

K-Nearest Neighbours classifies a new example by examining the labels of nearby examples in the training data. The number of neighbours, distance measure, and feature scaling can strongly influence the

result. KNN can be intuitive, but prediction can become slower as the training dataset grows, and irrelevant or differently scaled features may distort distances.

11.4. Decision Tree Classifier

A Decision Tree learns a sequence of feature-based splits that divide the training records into groups. Trees can represent non-linear relationships and their decision rules may be easier to explain than those of some other models. However, a deep tree can memorize training examples and perform poorly on unseen records. Depth, minimum leaf size, and other complexity controls should be evaluated.

11.5. Random Forest Classifier

Random Forest combines predictions from multiple decision trees trained with randomized samples and feature selection. This ensemble approach often reduces the instability of an individual tree. It can model complex patterns but may be less directly interpretable than a small tree. Its performance should be measured on held-out data rather than assumed from the algorithm name.

11.6. Gradient Boosting Classifier

Gradient Boosting builds an ensemble sequentially, where later learners focus on reducing errors made by the current ensemble. It can provide strong classification performance on structured tabular data, but its behaviour depends on settings such as the number of estimators, learning rate, and tree complexity. Tuning must be performed without using the held-out test outcomes.

The supplied presentation lists all six algorithms but does not provide verified model scores or identify the best-performing classifier. It would therefore be inaccurate to claim that one of them is the winner without running and documenting the experiments.

XII. PERFORMANCE EVALUATION

Performance measurement is necessary to determine how well a model predicts outcomes for examples it has not seen during training. The presentation contains a section titled “Performance Measurement and Model Selection,” but the extracted content does not include numerical results. The following measures are recommended for reporting the experiment.

12.1. Accuracy

Accuracy is the number of correct predictions divided by the total number of predictions. It gives a broad summary, but it can be misleading if placed and not-placed records are distributed unevenly.

12.2. Precision

Precision measures how many examples predicted as positive are actually positive. It is useful when false positive predictions have meaningful consequences.

12.3. Recall

Recall measures how many actual positive examples are correctly identified. It is useful when failing to identify a positive example is a particular concern.

12.4. F1-Score

The F1-score is the harmonic mean of precision and recall. It combines these two aspects into one measure, though the choice of averaging method should be stated for a multi-class problem.

12.5. Confusion Matrix

A confusion matrix shows the counts of correct and incorrect predictions for each class. It helps reveal

whether a model tends to misclassify one category more often than another.

The final paper should provide a results table containing the actual metrics for all six classifiers, the test-set size, the class definitions, and the selected model. No figures are filled in here because the supplied presentation does not provide experimental values.

12.6. Graphical Representation of Model Performance

Graphical representations can help compare the performance of the candidate machine-learning classifiers. A model comparison chart can present the accuracy, precision, recall, and F1-score of Logistic Regression, Support Vector Classifier, K-Nearest Neighbours, Decision Tree, Random Forest, and Gradient Boosting.

A confusion matrix can provide additional information about the correctly and incorrectly classified placement outcomes. These visualizations help identify differences between classifiers and support the selection of an appropriate model.

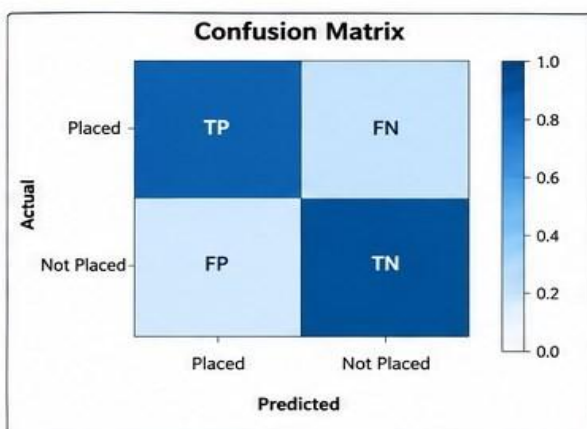


Figure 4: Confusion Matrix of the Selected Classifier

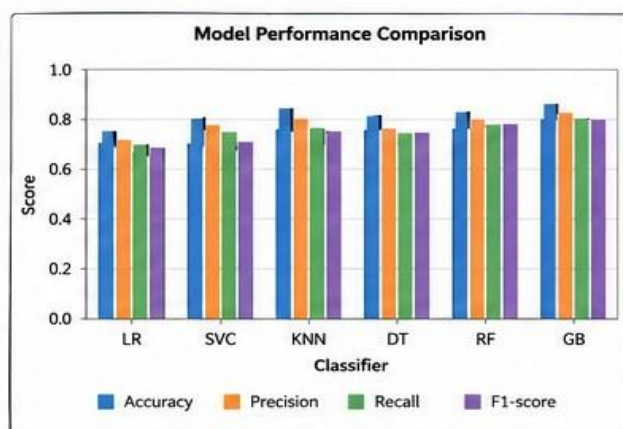


Figure 5: Performance Comparison of Machine-Learning Classifiers

The comparison chart and confusion matrix should be generated from the actual experimental results obtained using the test dataset. The figures should include clearly labelled axes, appropriate metric definitions, and a consistent presentation style.

XIII. LIBRARIES AND TECHNOLOGIES

- Python: primary programming language identified for backend and machine-learning tasks.
- Flask: web framework used to connect the web

pages with the prediction logic.

- Pandas: useful for loading, inspecting, and manipulating tabular datasets.
- NumPy: supports numerical operations and array-based processing.

- Matplotlib: supports charts for exploratory analysis and performance visualization.
- Scikit-learn: provides classification algorithms, preprocessing tools, and evaluation metrics.
- HTML: defines the structure of the web pages and form elements.
- CSS: controls the visual style and layout of the frontend.
- JavaScript: can provide client-side interactions and basic form validation.

The presentation identifies these technologies but does not provide exact version numbers, a dependency file, or deployment configuration. For reproducibility, the final project documentation should include library versions and the commands or steps required to run the application.

XIV. EXPECTED OUTPUT AND APPLICATION USE

The presentation describes an output page that gives the user an overall answer in simple words. The exact wording and decision threshold should match the implemented model. If the classifier returns only a class label, the application should not present it as a probability. If a probability is shown, its interpretation and calibration should be checked and explained.

A useful output page can state that the result is generated from the submitted attributes and historical data, and that real placement depends on many additional factors. The page should not shame, label, or discourage a student based on the prediction.

XV. ETHICAL CONSIDERATIONS AND LIMITATIONS

XStudent data can be personal and should be handled with care. Only necessary fields should be collected, access should be restricted, and the application should avoid exposing individual records. If the dataset contains demographic features such as gender, the project team should consider whether their use is justified, measure possible differences in model performance across groups where appropriate, and document the risks. Historical placement data may reflect prior inequalities or hiring practices, which a model can reproduce.

The model cannot account for every factor that affects

recruitment. A dataset collected from one institution, stream, or period may not generalize to other contexts. If the underlying labour market or hiring process changes, performance may decline. The system should not be used as the sole basis for hiring, eligibility, academic evaluation, or access to placement support. The principal project-specific limitation is that the supplied presentation does not report the dataset source or size, preprocessing decisions, train/test procedure, parameter settings, or measured results. These are essential for assessing scientific validity and reproducibility.

XVI. FUTURE SCOPE

- Document the dataset source, sample count, collection period, data dictionary, and permissions.
- Evaluate the models using stratified cross-validation and a held-out test set.
- Report accuracy, precision, recall, F1-score, confusion matrices, and class balance.
- Add interpretable explanations that describe model behaviour without implying that correlations are causal.
- Conduct fairness checks and review whether sensitive attributes should be included in the model.
- Add input validation, accessibility improvements, security protections, and clearer user guidance.
- Monitor performance over time and retrain only with appropriately governed, representative data.
- Explore additional relevant features only when they are available lawfully, ethically, and consistently.

XVII. CONCLUSION

This paper presents an expanded description of the Campus Placement Predictor web application using machine learning. The proposed system combines a form-based frontend built with HTML, CSS, and JavaScript, a Python/Flask backend, and a classification pipeline that predicts placement status from selected student attributes. The project presentation identifies Logistic Regression, Support Vector Classifier, K-Nearest Neighbours, Decision

Tree, Random Forest, and Gradient Boosting as candidate algorithms.

at: <https://ijcrt.org/papers/IJCRT2005188.pdf>

A reliable conclusion about the best model can be made only after evaluating the algorithms on a documented dataset with reproducible preprocessing and appropriate metrics. Since the supplied presentation does not include measured scores, this paper does not claim a particular accuracy or winning classifier. The project can serve as a practical demonstration and potential career-guidance aid when its limitations, privacy requirements, and fairness concerns are addressed. The prediction should remain advisory and should never be treated as a guarantee of employment.

REFERENCES

- [1] Naresh Patel, K. M., Goutham, N. M., Inzamam, K. A., Kandi, S. V., and Sharan, V. R., "Placement Prediction and Analysis Using Machine Learning," *International Journal of Engineering Research & Technology (IJERT)*, ICEI 2022, vol. 10, no. 11, 2022. ISSN: 2278-0181. Available at: <https://www.ijert.org/placement-prediction-and-analysis-using-machine-learning>
- [2] Khamkar, P., Lagad, R., Shinde, P., Londhe, S., and Bhosale, S. S., "Students Placement Prediction System," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 10, no. XI, pp. 784–786, November 2022. ISSN: 2321-9653. DOI: <https://doi.org/10.22214/ijraset.2022.47448>
- [3] Jose, I. T., Raju, D., Aniyankunju, J. A., James, J., and Vadakkel, M. T., "Placement Prediction Using Various Machine Learning Models and Their Efficiency Comparison," *International Journal of Innovative Science and Research Technology (IJISRT)*, vol. 5, no. 5, May 2020. ISSN: 2456-2165. Available at: <https://www.ijisrt.com/placement-prediction-using-various-machine-learning-models-and-their-efficiency-comparison>
- [4] Gautami, G., Hayat, M., Banerjee, P., and Kumar, B., "Prediction of Student Placement Using Machine Learning Algorithm," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 8, no. 5, pp. 1379–1384, May 2020. ISSN: 2320-2882. Available