WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

College Placement Analysis: An Interactive Dashboard for Placement Statistics and Readiness Prediction

Dr. T. Esther, Krithik K, Mohamed Ishak J, Muthusukesh M, Praveen D

Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore-641062, Tamil Nadu, India.
Email ID: krithikk25it@srishakthi.ac.in

ABSTRACT

College placement records are often maintained in spreadsheets, making it challenging to manage student information, review placement indicators, and provide individual guidance efficiently. This paper presents the College Placement Analysis System, a Python-based web application developed using Streamlit, Pandas, and SQLite. The system provides a dashboard summarising key indicators such as total students, placement rate, average package, average CGPA, and department count. Its student record management features support searching, filtering, editing, and deleting records, while a student registration form enables new records to be added with input validation and duplicate student-ID protection. The system includes a rule-based placement-readiness predictor that calculates an indicative score from student academic performance, skill scores, internship status, and backlogs. A career guidance module recommends potential job roles and identifies areas where a student may need further improvement. Student data can be imported from CSV or Excel files and exported as CSV, while SQLite provides persistent storage for student records. The system was tested with a sample dataset of 500 student records and through functional checks of its main operations. The readiness score is a weighted estimate rather than a machine-learning prediction, and the sample dataset is intended to demonstrate system functionality rather than represent actual placement outcomes. The proposed system offers a centralised and practical way to manage placement information and support students with accessible analysis and career guidance.

Keywords: College Placement Analysis, Streamlit, Python, SQLite, Student Record Management, Placement Readiness, Career Guidance, CSV and Excel Data Handling.

INTRODUCTION

Campus placement is one of the most visible outcomes of an engineering college. Students, parents, faculty, and management all look at the number of students placed, the companies that visited, and the salaries offered. The placement cell therefore collects a large amount of data every year, including the academic record of each student, skill and aptitude scores, internship and certification details, and the final placement result. In many colleges this information is stored in spreadsheets that are prepared for record keeping rather than for analysis.

Data analysis libraries and dashboard frameworks now make it possible to build analysis tools with modest effort. Python libraries such as Pandas [1] handle tabular data, Plotly produces interactive charts, and Streamlit turns a Python script into a web application. Combining these tools can give the placement cell an interactive view of its data without requiring any separate reporting software.

Existing challenges in manual placement analysis

When placement data is analysed by hand, every question needs a fresh set of filters, counts, and formulas. Finding the placement rate for one department, comparing the CGPA of placed and unplaced students, or listing the students with active backlogs each takes repeated manual work and is easy to get wrong. Data quality problems such as missing values, duplicate rows, or misspelt column names are often noticed only after a wrong figure has been reported. Reports are rebuilt from scratch each time, and different people may calculate the same indicator in different ways.

General-purpose business intelligence tools can solve some of these problems, but they need installation, licensing, and training, and they do not know the meaning of fields such as backlogs, aptitude score, or placement status. A small purpose-built application that understands the placement dataset can be simpler for a college to adopt.

Role of data analytics and machine learning

Descriptive analytics summarises what has already happened, for example how many students were placed in each department. Interactive dashboards make such summaries easy to explore because the user can change a filter and immediately see the effect on every chart. Machine learning goes one step

further by learning patterns from past records. A classifier such as a Random Forest [3] can estimate whether a student with a given profile is likely to be placed. Educational data mining has been widely studied for this kind of task [4]. Predictions of this type are probabilistic and depend heavily on the quality of the training data, so they should be used to guide preparation and counselling, not to make decisions about individual students.

Problem statement and objectives

Placement data kept in spreadsheets is time-consuming to analyse, prone to calculation and data-quality errors, and difficult to present clearly. There is a need for a simple application that validates the data, calculates the standard placement indicators automatically, and presents them in a form that faculty and management can explore.

The main objective of this work is to develop a web-based College Placement Analysis System that analyses student placement data and demonstrates placement prediction. The specific objectives are:

1. To load placement datasets from CSV and Excel files and verify that the required columns are present.
2. To calculate placement indicators: total students, students placed, placement rate, and average and highest salary. iii. To provide filters for department, CGPA range, and placement status that apply to all views.
3. To visualise placement status, department-wise strength and placement rate, CGPA distribution, and salary distribution.
4. To provide a searchable student record table with CSV download.
5. To generate a placement summary and a department summary report that can be downloaded.
6. To provide a placement readiness estimate in the interface and to develop a Random Forest model for placement prediction. viii. To evaluate the application through functional testing on valid and invalid datasets.

MATERIALS AND METHODS

Development approach

The project follows a structured development approach consisting of problem identification, requirement analysis, design, implementation, testing, and documentation. During problem identification, the difficulties of analysing placement records in spreadsheets were studied. Requirement analysis identified the major functions needed: file upload, validation, indicators, filters, charts, record search, reports, and prediction.

The dashboard, the data-validation function, and the prediction module were developed as separate parts of one Python project. The source code is maintained in a Git repository, and the dashboard work was carried out on a separate feature branch that was merged into the main branch through a pull request. Testing was used to check the behaviour of the data handling and prediction functions and to identify the areas that still need improvement.

System architecture

The system is organised into three layers: the presentation layer, the data-processing layer, and the analytics modules. The presentation layer is a Streamlit application (app.py) with seven pages reached from a sidebar. The data-processing layer uses Pandas to read the uploaded file, check the columns, apply the filters, and calculate the indicators, and Plotly Express to draw the charts. The analytics modules are the data-validation function (validation.py) and the Random Forest prediction module (prediction.py).

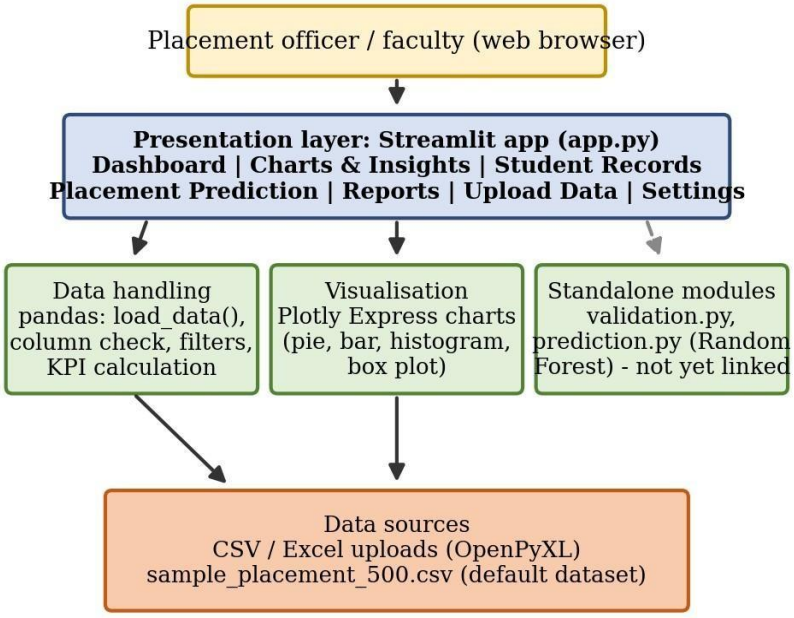


Fig. 1. System Architecture of the College Placement Analysis System.

The whole application runs as a single Python process, and the active dataset is held in the Streamlit session state. There is no separate backend server or database in the current version. In Fig. 1 the dashed arrow shows that the two analytics modules exist in the project but are not yet called from the dashboard; this is discussed in Section 3.5.

Technology stack and development environment

The application is written in Python. The interface is built with Streamlit and styled with custom CSS to give a dark theme. Pandas is used for data handling, Plotly Express [6] for interactive charts, and OpenPyXL [7] for reading Excel files. The machine-learning module uses scikit-learn [2]. The packages required to run the project are listed in a requirements file, and the dashboard is started with a single Streamlit command [5].

Table 1. Technology stack of the proposed system.

Layer	**Technology**
Language	Python
Interface	Streamlit with custom CSS dark theme
Data handling	Pandas
Layer	**Technology**
Visualisation	Plotly Express
File support	CSV (Pandas) and Excel .xlsx/.xls (OpenPyXL)
Machine learning	scikit-learn (Random Forest, pipelines, imputation, one-hot encoding)

Dataset and required fields

The application expects a table with thirteen columns, grouped in Table 2. Column names are trimmed of extra spaces on loading, and a file that lacks any of the required columns is rejected with a message that lists the missing names. The project includes a synthetic sample file of 500 students (sample_placement_500.csv, also provided as an Excel file), which is loaded by default. It covers seven departments (AI&DS, Civil, CSE, ECE, EEE, IT, and Mechanical), with CGPA values from 5.01 to 9.99, and 263 of the 500 students have completed an internship. Two further files, invalid_test.csv and invalid_test.xlsx, contain unrelated columns and are used to test the rejection of wrong files.

Table 2. Fields of the placement dataset.

Group	Fields	Description
Identification	Student_ID	Unique number of the student
Academic	Department, CGPA, Backlogs	Branch, cumulative grade point average, and number of pending arrears
Skill scores	Aptitude_Score, Communication_Score, Technical_Score	Scores recorded for each student (entered on a 0-100 scale in the prediction form)
Profile	Internship, Certifications, Primary_Skill	Internship completed (Yes/No), number of certifications, and main technical skill
Outcome	Placement_Status, Company, Salary_LPA	Placed or Not Placed, recruiting company, and salary in lakhs per annum

Application pages

The sidebar of the application provides seven pages, summarised in Table 3. The dashboard filters in the sidebar (department, CGPA range, and placement status) are applied before any page is drawn, so the indicators, charts, tables, and reports always describe the same group of students.

Table 3. Pages of the College Placement Analysis System.

Page	Functions
Dashboard	Cards for total students, students placed, placement rate, and average salary; placement status chart; department-wise student count; first ten records
Charts & Insights	CGPA histogram, CGPA versus placement box plot, department placement rate, and salary distribution of placed students
Student Records	Free-text search over all fields, record count, and CSV download of the displayed records
Placement Prediction	Input form for CGPA, backlogs, three scores, and internship; indicative readiness score with a feedback message
Reports	Summary table of indicators; department summary (students, placed, average CGPA, average salary, placement rate); CSV download
Upload Data	Upload of CSV or Excel files, validation, preview, option to use the file, and restoring the default dataset

Page	Functions
Settings	Dataset information: rows, columns, missing cells, duplicate rows, column names, and missing values per column

Data loading and validation workflow

When the application starts, the default sample dataset is loaded into the session state. A user can upload a new CSV or Excel file on the Upload Data page. The loader checks the file type, reads the file, trims the column names, checks the thirteen required columns, and rejects an empty file. Any problem is returned as a message instead of stopping the application. A valid file is shown as a ten-row preview and replaces the active dataset only after the user confirms. The workflow is shown in Fig. 2.

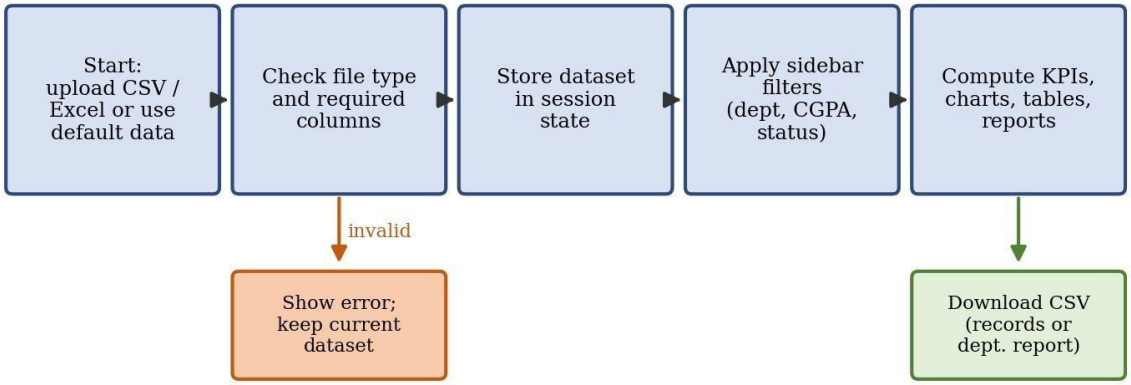


Fig. 2. Workflow of Data Loading, Filtering, and Reporting.

A separate validation function (validation.py) is available for data-quality checks. It reports an error for a missing or empty dataset and gives warnings for blank column names, missing values, and duplicate rows. The Settings page shows the number of missing cells and duplicate rows for the active dataset.

Analysis and prediction methods

Placement status is converted to a placed or not-placed flag by comparing the text, ignoring case, with the values placed, yes, true, 1, and selected. The placement rate is the number of placed students divided by the total number of students after filtering, and salary figures are calculated only for placed students whose salary is a valid number.

The Placement Prediction page calculates an indicative readiness score from the form inputs. The score is 30% of CGPA/10, plus 20% of the aptitude score, 15% of the communication score, and 25% of the technical score (each divided by 100), plus 10 points for a completed internship. If the student has backlogs, 5 points per backlog are deducted up to a maximum of 25, and the result is limited to the range 0 to 100. A score of 75 or more is reported as strong readiness, 50 to 74 as moderate, and below 50 as needing more preparation. These weights were chosen by the developers and were not learned from data.

The Random Forest module (prediction.py) trains a classifier from the dataset. It accepts several alternative column names, converts the placement status into two classes, fills missing values (most frequent value for categories, median for numbers), and one-hot encodes department, internship, and primary skill. The model uses 150 trees with balanced class weights. Twenty percent of the records are held out for testing, and the final model is then trained again on all valid records. The function that makes a prediction returns a label and the estimated probability of placement.

Testing and evaluation

Testing covered the data loading and validation rules, the calculation of the indicators, and the training of the prediction model. These were run as scripts on the sample and invalid files supplied with the project. The behaviour of the Streamlit interface itself, such as filters, search, and downloads, requires an interactive session and is marked as pending in Table 6. The folder reserved for automated tests currently contains only a description file, so no automated test suite has been written yet.

RESULTS AND DISCUSSION

Application functionality

The application brings the main placement analysis functions into one interface: validated file upload, filters, indicator cards, charts, record search, department summaries, downloadable CSV outputs, and a readiness estimate. Its main benefits compared with analysis in a spreadsheet are summarised in Table 4.

Table 4. Comparison between manual spreadsheet analysis and the proposed system.

Manual spreadsheet analysis	Proposed system
Counts and percentages are calculated with handwritten formulas	Indicators are calculated automatically for the selected students
Department comparison needs separate pivot tables	Department summary and placement rate chart are generated automatically
Records are sorted and filtered manually for each question	Sidebar filters update all pages together
Wrong files or missing columns are noticed late	Required columns are checked on upload; missing cells and duplicates are shown
Reports are rebuilt each time	Department report and student records download as CSV
Readiness is judged informally	Weighted readiness score gives a consistent indication

Analysis of the sample dataset

The default sample dataset contains 500 students, of whom 288 (57.6%) are marked as placed and 212 as not placed. Among the placed students the average salary is 10.97 LPA, the highest is 20.08 LPA, and the lowest is 2.42 LPA. The average CGPA of placed students (7.76) is higher than that of students not placed (7.08), and the same holds for the average aptitude score (69.01 against 66.26) and technical score (65.30 against 63.22). Table 5 and Fig. 3 give the department-wise results.

Table 5. Department-wise placement results for the sample dataset (500 records).

Department	Students	Placed	Placement rate (%)	Average CGPA
Mechanical	59	39	66.1	7.67
EEE	70	45	64.3	7.45
IT	70	42	60.0	7.33
AI&DS	97	55	56.7	7.44
CSE	64	35	54.7	7.42
ECE	74	39	52.7	7.53
Civil	66	33	50.0	7.49
Overall	500	288	57.6	7.47

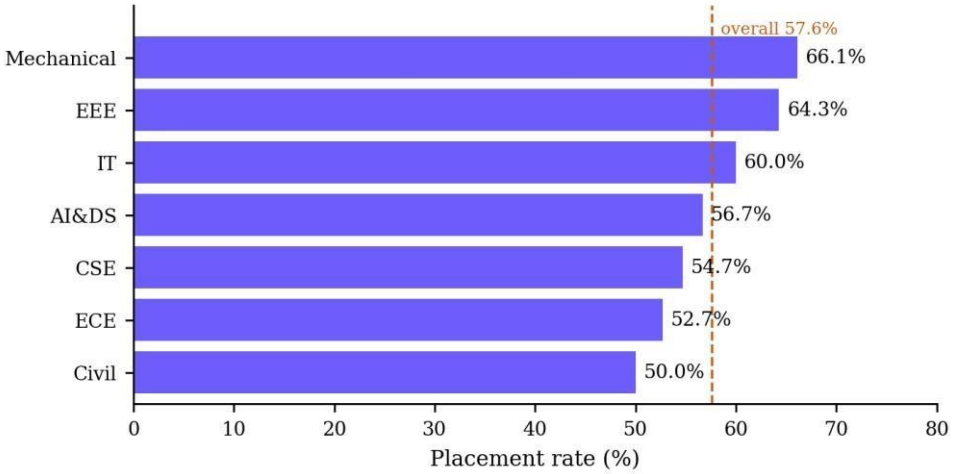


Fig. 3. Placement rate by department for the sample dataset.

Mechanical (66.1%) and EEE (64.3%) show the highest placement rates in the sample, and Civil (50.0%) the lowest. Because the sample data is synthetic, these differences only show how the dashboard presents department-level results and must not be interpreted as real trends in any college.

Prediction components

With the default form values (CGPA 7.0, no backlogs, all three scores 70, and an internship completed) the readiness score is 73.0%, which falls in the moderate band. The score responds in the expected direction: each backlog lowers it, and higher scores or a completed internship raise it. It is a heuristic and not a learned model.

When the Random Forest module was run on the 500-record sample, using a stratified 80:20 split, it correctly classified 65% of the 100 hold-out students. The sample contains 57.6% placed students, so always predicting placed would already be right for roughly that share of cases, and the Random Forest is only modestly above this level. This is expected for synthetic data in which the outcome has a weak relation to the input features, and the result should be taken only as proof that the training and evaluation pipeline runs correctly. A single split on 100 records also gives a rough estimate; cross-validation on real data would be needed for a reliable one.

Functional evaluation

The test scenarios are listed in Table 6. Tests marked Verified were run as scripts on the project files and gave the stated outcome. Tests marked Pending need the Streamlit interface to be used interactively and have not yet been recorded.

Table 6. Functional test scenarios and status.

Test ID	Module	Test scenario	Expected behaviour	Status
T1	Data loading	Load sample_placement_500.csv	500 rows and all 13 required columns found	Verified
T2	Data loading	Load sample_placement_500.xlsx	Same 500 rows and 13 columns as the CSV file	Verified
T3	Upload check	Files with columns Student_ID, Name, Marks (invalid_test.csv/.xlsx)	Required columns reported missing; file rejected	Verified
T4	Upload check	Short file sample_placement.csv (5 columns)	Required columns reported missing; file rejected	Verified

Test ID	Module	Test scenario	Expected behaviour	Status
T5	Validation	Dataset with missing values and duplicate rows; empty dataset	Warnings for missing and duplicate data; error for empty data	Verified
T6	Indicators	Calculate placement rate and average salary for the sample	57.6% placement rate; 10.97 LPA average salary	Verified
T7	Prediction	Train the Random Forest on the sample dataset	Model trains and reports hold-out accuracy (65%)	Verified
T8	Readiness score	Default form values	Score of 73.0% in the moderate band	Verified
T9	Filters	Select a department, CGPA range, and status	All pages update to the selected students	Pending
T10	Records	Search text and download CSV	Matching rows shown and exported	Pending

The pending tests should be completed and any failures corrected and repeated. Automated tests for the loader, the validation function, and the indicator calculations would allow these checks to be run every time the code changes.

Discussion and limitations

The system provides a working framework for validating placement data, calculating indicators, and exploring them through filters and charts. Its modular structure keeps the data checks, the charts, and the prediction code separate, which makes each part easier to improve. Several limitations should be stated clearly.

First, the dataset supplied with the project is synthetic, so none of the statistics in this paper describe real students. Second, the Placement Prediction page uses the weighted score described in Section 2.7 and does not call the Random Forest model; the validation function is likewise not called by the dashboard, which performs its own column check. The Random Forest and validation modules are therefore developed but not yet integrated. Third, the project has no login, user roles, or database: uploaded data stays only in the session, and anyone who can open the application can see all student records. Real placement data contains personal information, so access control and data protection rules would be needed before real use. Fourth, only files that contain all thirteen required columns are accepted, and company-wise hiring analysis is not yet part of the interface. Fifth, the model evaluation uses a single hold-out split and no automated tests exist yet.

CONCLUSION

The College Placement Analysis System shows how a placement dataset can be turned into an interactive analysis using Python, Streamlit, Pandas, and Plotly. The application validates uploaded CSV and Excel files, calculates the main placement indicators, applies filters across all views, presents charts and department summaries, supports record search, and exports results as CSV files. A weighted readiness score and a Random Forest module demonstrate how placement prediction can be added to the analysis.

The results reported here come from a synthetic dataset and therefore demonstrate the working of the system, not real placement performance. The usefulness of the predictions in particular depends on the quality and size of the real data used for training.

Future work includes connecting the Random Forest and validation modules to the interface, testing the model on verified real data with cross-validation and feature-importance reporting, adding companywise hiring and salary analysis, introducing login and role-based access with a database, writing automated tests, and deploying the application for use by the placement cell.

ACKNOWLEDGMENTS

The authors would like to thank Sri Shakthi Institute of Engineering and Technology for providing the academic environment and facilities required for this project. The authors also express their sincere gratitude to the Head of the Department of Information Technology, the project guide Mrs. T. Esther, and the faculty members for their guidance, encouragement, and support throughout the development of the project.

REFERENCES

1. Khodaskar, M., Gajare, A., Bochare, C., & Patil, S. (2025). AI-driven placement prediction and recommendation system for college students. International Research Journal of Engineering and Technology, 12(10).
2. Kumar, A. D. P., Kuchhadia, V., Charan, G., Sreeja, G., & Pavan, N. (2025). Predicting student employability using machine learning: A comparative study of classification algorithms. In Proceedings of the International Conference on Computer Science and Communication Engineering (ICCSCE 2025). https://www.atlantis-press.com/proceedings/iccsce-25/126017440 Atlantis Press.
3. Pathak, A., Matcha, M., Gopisetti, M., & Joshi, S. (2025). A machine learning framework for predicting student placement outcomes. Ingénierie des Systèmes d'Information, 30(7), 1715–1721.https://iieta.org/node/21236
4. Rao, K. E., Terlapu, P. R. V., Salakapuri, R., Ramakrishna, G., Pemula, R., & Rao, D. P. (2025). Student placement performance analysis and prediction systems at higher education institutions in South India region: A machine learning approach with ranked features. U.Porto Journal of Engineering, 11(2), 183–219. https://ijooes.fe.up.pt/index.php/upjeng/article/view/2183-6493_0011-002_002916
5. Batool, S., Rashid, J., Nisar, M. W., Kim, J., Kwon, H.-Y., & Hussain, A. (2023). Educational data mining to predict students' academic performance: A survey study. Education and Information Technologies, https://doi.org/10.1007/s10639-022-11152-y 28(1), 905–971.
6. Bin Roslan, M. H., & Chen, C. J. (2022). Educational data mining for student performance prediction: A systematic literature review (2015–2021). International Journal of Emerging Technologies in Learning, https://doi.org/10.3991/ijet.v17i05.27685 17(5), 147–179.
7. Yağcı, M. (2022). Educational data mining: Prediction of students' academic performance using machine learning algorithms. Smart Learning Environments, 9, Article 11. https://doi.org/10.1186/s40561-022-00192-z
8. Khan, A., & Ghosh, S. K. (2021). Student performance analysis and prediction in classroom learning: A review of educational data mining studies. Education and Information Technologies, 26(1), 205–240. https://doi.org/10.1007/s10639-020-10230-3
9. Maurya, L. S., Hussain, M. S., & Singh, S. (2021). Developing classifiers through machine learning algorithms for student placement prediction based on academic performance. Applied Artificial https://doi.org/10.1080/08839514.2021.1901032 Intelligence, 35(6), 403–420.
10. Harihar, V. K., & Bhalke, D. G. (2020). Student placement prediction system using machine learning. SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology, 12(Special Issue 2). https://doi.org/10.18090/samriddhi.v12is2.17
11. Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. WIREs Data Mining and Knowledge Discovery, 10(3), Article e1355. https://doi.org/10.1002/widm.1355
12. Srinivas, C. K., Yadav, N. S., Pushkar, A. S., Somashekar, R., & Sundeep, K. R. (2020). Students placement prediction using machine learning. International Journal for Research in Applied Science and Engineering Technology, 8(5). https://doi.org/10.22214/ijraset.2020.5466
13. Francis, B. K., & Babu, S. S. (2019). Predicting academic performance of students using a hybrid data mining approach. Journal of Medical Systems, 43(6), Article 162. https://doi.org/10.1007/s10916-019-1295-4
14. Khandale, S., & Bhoite, S. (2019). Campus placement analyzer: Using supervised machine learning algorithms. International Journal of Computer Applications Technology and Research, 8(9), 358–362.
15. Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x
16. Thangavel, S. K., Bharathi, P. D., & Sankar, A. (2017). Student placement analyzer: A recommendation system using machine learning. In 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS) (pp. 1–5). IEEE. https://doi.org/10.1109/ICACCS.2017.8014632
17. Shahiri, A. M., Husain, W., & Rashid, N. A. (2015). A review on predicting student's performance using data mining techniques. Procedia Computer Science, 72, 414–422. https://doi.org/10.1016/j.procs.2015.12.157
18. Romero, C., & Ventura, S. (2010). Educational data mining: A review of the state of the art. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 40(6), 601–618. https://doi.org/10.1109/TSMCC.2010.2053532
19. Baker, R. S., & Yacef, K. (2009). The state of educational data mining in 2009: A review and future visions. Journal of Educational Data Mining, 1(1), 3–17. https://doi.org/10.5281/zenodo.3554657

20. Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861– 874. https://doi.org/10.1016/j.patrec.2005.10.010