Optimum College, branch and course Predictor

Kshitij Shrivastava[1]

Abstract— **The Joint Entrance Examination (JEE) is one of the most competitive and prestigious examinations in India, serving as a gateway for undergraduate admissions into premier engineering institutions such as the Indian Institutes of Technology (IITs) and National Institutes of Technology (NITs). The dataset analyzed in this project is sourced from https://cutoffs.iitr.ac.in/https://cutoffs.iitr.ac.in/ and encompasses detailed admission records spanning multiple years.**

The dataset includes features such as institution type (IIT/NIT), program names, reservation categories, gender-based reservation pools, admission round numbers, and rank cutoffs for various courses. This data offers a comprehensive view of the admission trends and eligibility criteria, facilitating the development of predictive models to recommend suitable colleges and courses based on a student's rank, category, and preferences.

The project aims to leverage this dataset to build a machine learning-based college and branch predictor tailored for JEE aspirants. By utilizing advanced algorithms, such as CatBoost, the predictor models the relationships between input features (e.g., rank, category, and program) and admission outcomes. This tool can assist students in making informed decisions about their college and course choices, thereby enhancing the admission planning process.

The insights and predictions generated through this project highlight patterns in admissions, the impact of reservation policies, and the competitiveness of various institutions and programs over the years. This research underscores the potential of data-driven approaches to simplify complex decision-making processes in education.

.

I. Introduction

The Optimum College, Course & Branch Predictor aims to revolutionize the decision-making process for IIT-JEE aspirants by leveraging the power of machine learning and deep learning techniques. With the increasing competition and complexity of admission trends, students often face challenges in identifying the best-fit colleges, courses, and branches based on their rank, category, and quota. This project addresses these challenges by providing a data-driven approach to predict key information, including the institute, program name, program duration, and institute type.

Existing college prediction tools often suffer from accessibility issues, requiring subscriptions or payments, and provide limited insights confined to basic recommendations. This project bridges these gaps by offering a free, inclusive, and insightful solution. The model is trained using historical admission data from prestigious institutions like IITs and NITs, incorporating features such as opening and closing ranks, quotas, and categories. By analyzing past trends and using algorithms like CatBoost, the system ensures accurate and personalized recommendations. [1]

Additionally, the project explores advanced deep learning techniques, such as LSTMs[2], to capture year-over-year trends, offering a holistic view of admission dynamics. The integration of machine learning and sequential modeling enhances the predictive capabilities, enabling students to make well-informed choices about their academic futures. This initiative underscores the importance of accessibility, accuracy, and innovation, making it a valuable resource for a diverse range of student.[3]

II. Material and Method

A. Problem Description/ Specific goals

With the rising competition and increasing complexity of admission trends, students often struggle to identify colleges, courses, and branches that align with their ranks, categories, and quotas. This project addresses these challenges by providing a data-driven, accessible, and inclusive solution that predicts key information, including the institute, program name, program duration, and institute type. Existing prediction tools often fall short due to their limited scope, accessibility barriers, or subscription costs, offering only basic recommendations without meaningful insights. The goal of this project is to overcome these limitations by delivering a free, robust, and insightful model that leverages historical admission data from prestigious institutions like IITs and NITs. The model incorporates critical features such as opening and closing ranks, quotas, and categories to analyze past trends and provide accurate predictions.

B. DataSet Description

The dataset utilized in this project is a comprehensive compilation of historical admission data sourced primarily from the official cut-off archives of premier engineering institutes like IITs and NITs. This dataset captures various critical features that influence the admission process, ensuring a detailed and robust foundation for predictive modeling. Below is a detailed breakdown of the dataset's components:

1) Source and Structure : The dataset is sourced from publicly available resource, such as the official website **https://cutoffs.iitr.ac.in**.It is structured in a tabular format, ensuring compatibility with machine learning frameworks. The structured nature of the data facilitates efficient preprocessing and feature extraction, which are critical for developing accurate predictive models .

[1]Smart Mobility, Indian Institute of Technology Hyderabad, India.

2) Features : The dataset comprises several features that play a pivotal role in determining admission outcomes. These features can be categorized as follows:

1) **Basic Admission Parameters**:
   - **Opening Rank**: The rank of the first candidate admitted to a program under a specific quota and category.
   - **Closing Rank**: The rank of the last candidate admitted to a program under a specific quota and category.
2) **Categorical Features**:
   - **Category**: Represents the reservation category of the student (e.g., GEN, OBC-NCL, SC, ST, GEN-EWS, etc.).
   - **Quota**: Denotes the quota type under which the admission is made (e.g., All India (AI), Home State (HS), Other State (OS)).
3) **Institution Details**:
   - **Institute Short Name**: A short identifier for the institute (e.g., IITD for IIT Delhi).
   - **Institute Type**: Categorization of the institute, such as IIT or NIT.
4) **Program Details**:
   - **Program Name**: The full name of the academic program (e.g., Mechanical Engineering, Computer Science, Civil Engineering).
   - **Program Duration**: Specifies the duration of the program (e.g., 4 Years, 5 Years).
5) **Additional Contextual Information**:
   - **Year**: The admission year, allowing for trend analysis and the identification of temporal patterns.
   - **Gender Pool**: Gender-specific pools, such as Gender-Neutral or Female-only.

3) Data Challenges: The dataset posed several challenges:

1) **Data Quality Issues**:
   - Missing Values: Some entries lacked specific information, such as closing ranks for under-subscribed programs.
   - Inconsistent Formats: Variations in how categories or quotas were recorded necessitated standardization.
2) **Categorical Encoding**: Many features, such as categories and quotas, required encoding into numerical formats for compatibility with machine learning models.
3) **Imbalanced Data**: Certain categories or quotas had significantly fewer data points, potentially biasing the model.

4) Preprocessing and Transformation : To address these challenges, the following preprocessing steps were performed:

- **Handling Missing Values**: Missing data was imputed or excluded based on the relevance of the feature.
- **Encoding Categorical Features**: Features like category and quota were label-encoded, ensuring numerical representation without losing interpretability.
- **Feature Scaling**: Numerical features, such as opening and closing ranks, were standardized to improve model performance.
- **Data Splitting**: The dataset was divided into training and testing subsets to evaluate model performance effectively.

III. Analytics Results

1) Number of Programs per Institute: (Fig: 1)

- **Overview**:
  This bar graph illustrates the total number of academic programs offered by various institutes, including both IITs and NITs.
- **Key Observations**:
  - **IITs**: Among IITs, **IIT Kharagpur** stands out with the highest number of programs, surpassing 30. Other prominent IITs like **IIT Bombay**, **IIT Delhi**, and **IIT Madras** also offer a substantial variety of programs (20–25 programs).
  - **NITs**: For NITs, **NIT Rourkela** leads in program offerings, followed by **NIT Warangal** and **NIT Trichy**, which also offer a wide array of programs. Some newer NITs, such as **NIT Arunachal Pradesh** and **NIT Mizoram**, offer fewer programs, reflecting their smaller scale or more recent establishment.
- **Insights**: The diversity in programs indicates how certain institutes, particularly older IITs and NITs, have established a broader curriculum to cater to varied academic interests. On the other hand, newer institutions are still expanding their program portfolio.

2) Yearly Trends in Closing Ranks by Caste: (Fig: 2)

- **Overview**: This line graph tracks the trends in closing ranks over the years (2016–2021) for different seat categories, such as General (GEN), OBC-NCL, SC, ST, EWS, and their respective PWD subcategories.
- **Key Observations**:
  - **General Category (GEN)**: The closing ranks for the General category exhibit a significant spike around 2019, followed by a downward trend. This spike suggests a possible policy or seat allocation change during that year.
  - **OBC-NCL & SC**: These categories also show a similar pattern, with OBC-NCL consistently having lower closing ranks than SC, indicating higher competition among candidates.
  - **PWD Categories**: The closing ranks for PWD candidates (e.g., GEN-PWD, SC-PWD) remain relatively higher, reflecting a more lenient rank cutoff for accessibility purposes.
  - **GEN-EWS Introduction**: The **EWS** category, introduced in recent years, shows a closing rank

[H]

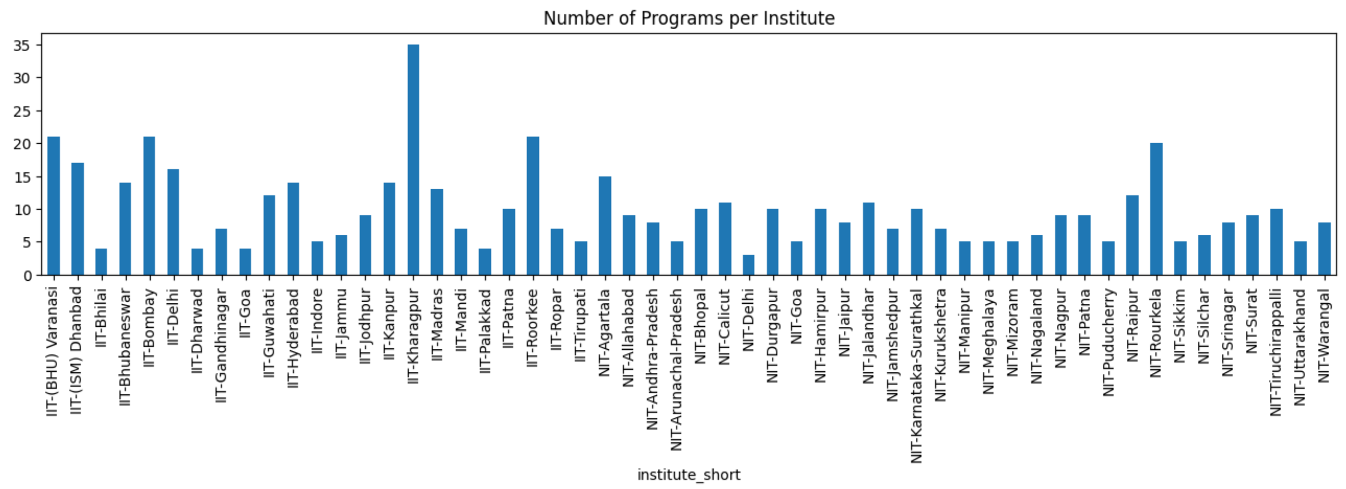

Fig. 1: Scatter plot of number of programs per Institute

trend aligning between the General and OBC-NCL categories.

- **Insights**: These trends demonstrate how policies such as reservation and the introduction of new seat categories (e.g., GEN-EWS) significantly impact closing ranks. The spike in 2019 may relate to increased seat intake or changes in exam difficulty.

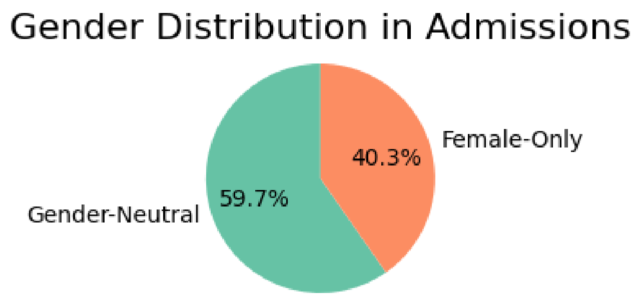

Fig. 2: Gender Distribution in Admission

3) Distribution of Seat Types: (Fig: 3)

- **Overview**:
  This bar graph shows the frequency distribution of different seat types/categories in the dataset. Categories include General (GEN), OBC-NCL, SC, ST, GEN-EWS, and their PWD subcategories.
- **Key Observations**:
  - **Dominant Categories**: The **GEN** and **OBC-NCL** categories have the highest counts, closely followed by SC and ST. This reflects the demographic proportions of applicants and available seats.
  - **GEN-EWS**: The GEN-EWS category, although relatively new, has gained a notable share of seats.
  - **PWD Categories**: PWD subcategories (GEN-PWD, SC-PWD, etc.) have significantly lower counts compared to non-PWD categories, reflecting the reserved percentage allocated to them.

- **Insights**: This graph highlights the impact of reservation policies, with a balanced distribution for major reserved categories (SC/ST/OBC-NCL). It also indicates the relatively smaller share of PWD seats due to their limited proportion in overall allocations.

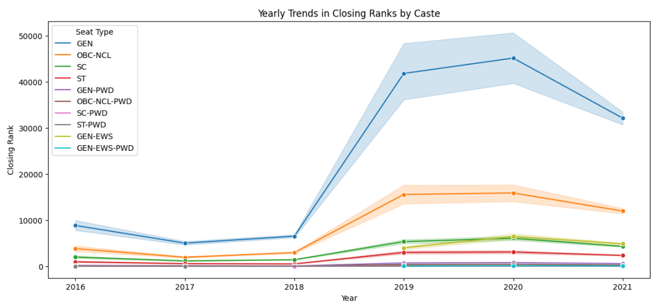

Fig. 3: Yearly trend of closing rank by category

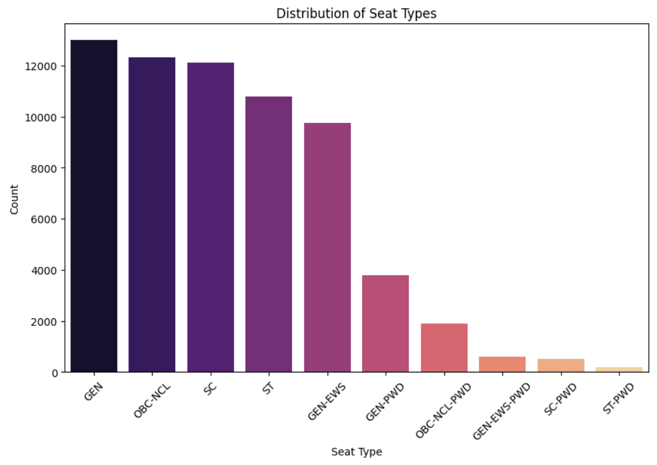

Fig. 4: Distribution of seat Type

IV. Methodology / Approach

Here I provides a comprehensive theoretical overview of how to implement **CatBoost**, **Artificial Neural Networks**

(ANN), **LSTM** and **Random Forest** models from data preparation to evaluation. Each model's methodology is explained with a focus on key processes, model-specific nuances, and best practices.

A. *Methodology for CatBoost*

CatBoost is a gradient boosting algorithm specifically designed to handle categorical data effectively. It minimizes the need for extensive preprocessing by incorporating categorical feature handling directly within its training process.[4]

1) Step 1: Data Preprocessing:

- **Handling Missing Data**: CatBoost can handle missing values natively, but preprocessing missing data (e.g., imputation or creating a ”missing” category) can sometimes yield better performance.[5]
- **Label Encoding for Categorical Data**:
  CatBoost can process categorical data directly by specifying which features are categorical. However, in cases where the data format does not allow specifying categorical columns, **Label Encoding** can be used as an alternative:
  - **Label Encoding** involves assigning unique integer labels to each category in a feature. For example, ”Male” = 0, ”Female” = 1.
  - CatBoost converts these labels internally into advanced encodings (like target-based encoding) during training to retain statistical information about the categories.
  - **Why Use Label Encoding?**: While CatBoost works best with raw categorical data, label encoding simplifies preprocessing and ensures compatibility when working with pipelines or frameworks that might not natively support categorical column specifications.
- **Data Splitting**: Split the dataset into training, validation, and test sets. The test set remains untouched until the final evaluation stage.

2) Step 2: Model Training:

- **Hyperparameter Tuning**: Key hyperparameters include:
  - Learning Rate: Controls the step size during optimization.
  - Depth: The maximum depth of trees, controlling the model's complexity.
  - Iterations: The number of boosting iterations (trees).
  - L2 Regularization: Helps reduce overfitting by penalizing large weights.
- **Categorical Handling During Training**: CatBoost's **ordered boosting** ensures that predictions are calculated in a statistically robust manner, avoiding target leakage for categorical features.
- **Overfitting Prevention**: Use early stopping based on validation loss to prevent overtraining. Additional regularization parameters like `random_strength` can help reduce overfitting further.

3) Step 3: Model Evaluation:

- Evaluate the model using metrics appropriate to the task:
  - **Classification**: Accuracy, F1-score, precision, recall, and AUC-ROC curve.
  - **Regression**: Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared.
- **Feature Importance Analysis**: CatBoost provides a feature importance mechanism to understand which features most influence predictions.

B. *Methodology for Artificial Neural Networks (ANN)*

ANNs are versatile and capable of capturing complex, non-linear relationships between input features and target variables.[6]

1) Step 1: Data Preprocessing:

- **Normalization and Scaling**: Normalize or standardize numerical features. This ensures consistent performance by preventing large feature values from dominating others.
- **Encoding Categorical Features**:
  - Use **one-hot encoding** for categorical features in ANN models because ANNs cannot handle categorical data directly.
- **Handling Imbalanced Data**: For classification tasks, address class imbalances using oversampling (e.g., SMOTE) or class weighting.

2) Step 2: Architecture Design:

- **Input Layer**: Number of neurons equals the number of input features.
- **Hidden Layers**:
  - Decide the number of hidden layers and neurons per layer based on the problem's complexity.
  - Apply activation functions like **ReLU** (Rectified Linear Unit) to introduce non-linearity.
- **Output Layer**:
  - Binary Classification: A single neuron with a sigmoid activation function.
  - Multi-class Classification: Softmax activation with as many neurons as the number of classes.
  - Regression: A single neuron with no activation (linear output).

3) Step 3: Training Process:

- **Loss Functions**:
  - Binary Classification: Binary Cross-Entropy Loss.
  - Multi-class Classification: Categorical Cross-Entropy Loss.
  - Regression: Mean Squared Error (MSE).
- **Optimization Algorithm**: Use optimizers like Adam or RMSprop, which adaptively adjust the learning rate for efficient convergence.
- **Batch Size and Epochs**:
  - Choose an appropriate batch size (e.g., 32, 64) based on memory constraints and dataset size.

    - Train the model for several epochs, but use **early stopping** to terminate training once validation loss stops improving.
- **Regularization**:
    - Dropout: Randomly drop a fraction of neurons during training to prevent overfitting.
    - L2 Regularization: Adds a penalty term to the loss function based on the magnitude of weights.

4) Step 4: Model Evaluation:

- Evaluate the model using test data and calculate relevant metrics:
    - **Classification**: Accuracy, confusion matrix, F1-score, and AUC-ROC.
    - **Regression**: MSE, MAE, and R-squared.
- Plot learning curves to assess the training process and identify overfitting or underfitting.

C. Methodology for Random Forest

Random Forest is an ensemble learning method that makes multiple decision trees using bagging (Bootstrap Aggregating) to improve accuracy and reduce overfitting. [7] It is particularly effective for predicting categorical and numerical outcomes like institute and program allocation based on ranks.

1) Step 1: Data Preprocessing:

- **Handling Missing Data**: Although Random Forest can handle missing values to some extent, imputing missing data (e.g., using mean/mode imputation) ensures greater stability and accuracy.
- **Filtering Data**: Focus the dataset on relevant features such as `year`, `round_no`, `quota`, `pool`, `category`, `opening_rank`, and `closing_rank`.
- **Label Encoding for Categorical Features**: Random Forest requires numerical inputs, so categorical variables are converted into numeric values:
    - **Label Encoding:** Assigns a unique integer to each category (e.g., `"IIT" = 0`, `"NIT" = 1` for `institute_type`).
    - **Why Use Label Encoding?** Decision trees split data based on feature values. Label encoding enables Random Forest to use numerical thresholds for creating effective splits.

2) Step 2: Model Training:

- **Tree Construction**: Random Forest constructs multiple decision trees by:
    - Training each tree on a random subset of the data (bagging).
    - Splitting features randomly at each node to ensure diversity among trees.
- **Hyperparameter Tuning**: Optimize key parameters for improved model performance:
    - **Number of Trees**: Increasing the number of trees provides better stability but adds computational cost.
    - **Maximum Depth**: Restricts the depth of trees to prevent overfitting.
    - **Minimum Samples per Split**: Ensures splits occur only when sufficient data points are available.
    - **Random State**: Set a random state for reproducibility.[8]

3) Step 3: Model Evaluation:

- Evaluate the trained Random Forest model on the test dataset using regression metrics:
    - **Mean Absolute Error (MAE):** Measures the average absolute error between actual and predicted ranks.
    - **Mean Squared Error (MSE):** Provides the average squared difference between actual and predicted values.
    - **R-squared (R^2):** Evaluates how well the model explains variance in the data.
- **Feature Importance**: Analyze which features contribute most to the model's predictions using Random Forest's feature importance scores.

4) Step 4: Prediction and Visualization:

- **Prediction**: Use the trained model to predict target variables such as `institute_short` and `program_name` based on ranks.
- **Visualization**: Compare predicted and actual values:
    - Plot the predicted `opening_rank` and `closing_rank` against the actual values for different years and rounds.
    - Highlight the trends in allocation over time to assess prediction accuracy.

D. Methodology for LSTM

The step-by-step methodology for implementing the LSTM model [9] is as follows:

1) **LSTM Model Design:**
    - Use the Keras Sequential API to define the LSTM model architecture:
        - **Input Layer:** Specify the input shape as $(n\_steps, \text{number\_of\_features})$.
        - **LSTM Layer:** Add an LSTM layer with a specified number of units (e.g., 50) and an activation function (e.g., ReLU).
        - **Dense Layer:** Add a fully connected layer to output the predictions (e.g., `Dense(2)` for predicting opening and closing ranks).
2) **Model Compilation:**
    - Compile the model by specifying:
        - **Optimizer:** Use `adam` for efficient learning.
        - **Loss Function:** Use `mean_squared_error (MSE)` as the loss metric for regression tasks.
3) **Model Training:**
    - Train the model using the training data (X_train, y_train).

- Validate the model on the testing data (X_test, y_test) during training to monitor performance.
- Configure training parameters such as:
  - **Number of epochs:** e.g., 50.
  - **Batch size:** e.g., 16.

4) **Prediction:**
   - Use the trained model to predict future ranks.
   - Take the last sequence from the dataset and iteratively predict the next values for the desired future years (e.g., 2022, 2023, 2024).
5) **Inverse Transformation:**
   - Convert the scaled predictions back to the original scale using the inverse transformation of the `MinMaxScaler`.
6) **Visualization:**
   - Plot the actual and predicted values to visually compare performance.
   - Highlight trends over time for both opening and closing ranks.
7) **Evaluation:**
   - Assess the model's performance using metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE).
   - Analyze the model's ability to capture patterns and trends in the data.

V. Key Observations - Model Results

The predicted opening and closing ranks shown in Table 1 are generated using an LSTM model for the dataset filtered with the following conditions: the institute is **IIT-Bombay**, the program is **Electrical Engineering**, the quota is **AI (All India)**, the pool is **Gender-Neutral**, the program duration is **4 Years**, and the category is **General (GEN)** Fig: 5. These predictions provide insights into the rank trends for this specific program and candidate pool over the years 2022 to 2025.

Year	Opening Rank	Closing Rank
2022	99.88	435.74
2023	101.39	440.23
2024	103.54	446.45
2025	106.51	454.82

TABLE I: Predicted Opening and Closing Ranks (2022-2025)

The approach used for prediction is highly flexible, enabling users to customize the filtering criteria according to their specific requirements. For instance, a candidate can adjust parameters such as the institute, program, quota, gender pool, program duration, or category to predict opening and closing ranks for their personalized preferences. This customization ensures that the model can generate tailored predictions for any unique scenario or set of conditions.

The original data for opening and closing ranks for the program **Electrical Engineering (IIT-Bombay, AI Quota, Gender-Neutral, 4 Years, General Category)** has been taken from **College Pravesh** for the years 2022, 2023, and 2024. The recorded ranks are as follows:

Model Name	Programme	Degree	Institute	Mean Accuracy
CatBoost	0.31	0.84	0.81	0.58
ANN	0.20	0.81	0.19	0.40
Random Forest	0.76	0.94	0.75	0.81

TABLE II: Comparison of Models

In **2022**, the opening rank was **103**, and the closing rank was **369**. In **2023**, the opening rank was **142**, and the closing rank was **463**. In **2024**, the opening rank was **15**, and the closing rank was **464**.

The LSTM model predictions for these years are quite close to the original recorded ranks, which indicates the effectiveness and reliability of the model in capturing rank trends accurately.

The table below compares the original data with the predicted data from the LSTM model:

Year	Original Opening Rank	Original Closing Rank
2022	103	430
2023	142	463
2024	15	464

TABLE III: Original Data for Opening and Closing Ranks (2022-2024)

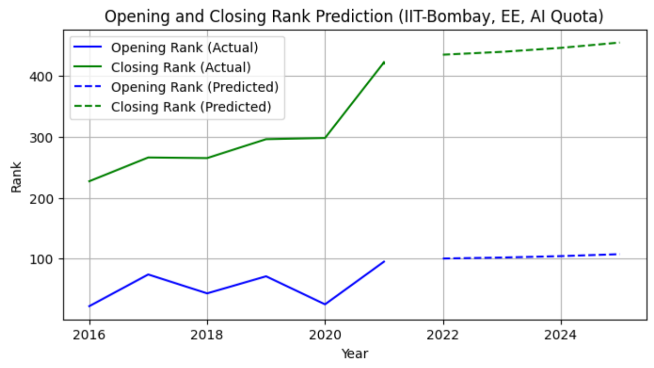

Fig. 5: LSTM Predicted Result

Fig. 6: Yearly trend of closing rank by category

The table II compares the performance of three machine learning models—**CatBoost**, **ANN**, and **Random Forest**—evaluated based on their mean accuracy across key features such as *Programme Name*, *Degree Short*, and *Institute Name*. Among these models, **Random Forest** achieved the highest accuracy of **81%**, demonstrating its robustness and effectiveness for this task.

In this project, **Random Forest** proved to be a highly reliable model for predicting college, its branch, and course duration, achieving results closely aligned with the original data. Additionally, an **LSTM** model was successfully implemented to forecast rank trends, with its predictions closely approximating the recorded data. This highlights the complementary role of **Random Forest** and **LSTM** in providing accurate and insightful predictions for college and program rank trends.

VI. Conclusion

Throughout this project, I gained a deeper understanding of the mathematics and practical applications behind several machine learning and deep learning algorithms. In Gradient Boosting and XGBoost, I explored gradient descent, where models iteratively minimize loss functions such as mean squared error and log loss by combining weak learners like decision trees. CatBoost stood out for its innovative use of target-based encoding, leveraging Bayesian principles to efficiently transform categorical data, and its ordered boosting algorithm.

I also learned about the workings of Random Forest, which uses decision tree splitting and bagging to create a robust, interpretable model capable of processing categorical data through label encoding. Hyperparameter tuning in these models helped me understand how parameters like learning rate, tree depth, and feature selection impact model performance, complexity, and overfitting.

Deep learning algorithms like LSTMs, focusing on their unique ability to retain temporal information through the mathematical roles of gates (input, forget, and output) and cell states. Understanding activation functions such as sigmoid and tanh further enhanced my grasp of how LSTMs address challenges like the vanishing gradient problem.

The practical implementation of these algorithms—CatBoost, Random Forest, ANNs, and LSTMs—underscored the importance of combining theoretical knowledge with real-world applications.

References

[1] Y. G. S. C. Xiao Chen, Yi Peng, "A competition model for prediction of admission scores of colleges and universities in chinese college entrance examination," *PLOS ONE*, 2024.

[2] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2017.

[3] N. R. S. . M. P. . R. S. . E. Y. . J. S. . Aga Maulana 1, Teuku Rizky Noviandy 1 and R. I. 7, "Optimizing university admissions: A machine learning perspective," *Journal of Educational Management and Learning*, vol. 1, no. 1, 2023.

[4] M. Imani and H. R. Arabnia, "Hyperparameter optimization and combined data sampling techniques in machine learning for customer churn prediction," *Journal of Computational Science*, vol. 54, p. 101452, 2023.

[5] M. I. Al-Sarem, M. A. Alfarisy, and A. Alsaedi, "Feature selection and classification using catboost method for improving the performance of predicting parkinson's disease," *Computers, Materials Continua*, vol. 64, no. 3, pp. 1545–1565, 2020.

[6] J. Welbl, "Casting random forests as artificial neural networks (and profiting from it)," *Neural Networks*, vol. 60, pp. 91–96, 2014.

[7] C. . Paelinckx, "Evaluation of random forest and adaboost tree-based ensemble classification," 12 2008.

[8] L. Breiman, "Random forests," *Machine Learning, 45(1), 5-32*, 2001.

[9] X. Ma, G. Zhang, and X. Wang, "Stock prediction based on random forest and lstm neural network," *Journal of Forecasting*, vol. 38, no. 5, pp. 455–470, 2019.