DEPARTMENT OF COMPUTER SCIENCE
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI – 600036

Student Response Prediction using Advanced Feature Extraction and Optimization Techniques

A Thesis

Submitted by

GAURAV KANWAT

For the award of the degree

Of

MASTERS OF TECHNOLOGY

Oct 2024

THESIS CERTIFICATE

This is to undertake that the Thesis titled **STUDENT RESPONSE PREDICTION USING ADVANCED FEATURE EXTRACTION AND OPTIMIZATION TECHNIQUES**, submitted by me to the Indian Institute of Technology Madras, for the award of **Masters of Technology**, is a bona fide record of the research work done by me under the supervision of **Dr. Chandrashekar Lakshminarayanan**. The contents of this Thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

In order to effectively convey the idea presented in this Thesis, the following work of other authors or sources was reprinted in the Thesis with their permission:

Chennai 600036 **Gaurav Kanwat**

Date: Oct 2024

Dr. Chandrashekar Lakshminarayanan
Research advisor
Professor
Department of Computer Science
IIT Madras

ABSTRACT

In the field of educational data mining, predicting student performance and responses plays a critical role in improving personalized learning and adaptive teaching strategies. This project focuses on the NeurIPS 2020 Education Challenge, which involves predicting students' correctness on tasks and their responses to questions, given a rich and multi-layered dataset containing student interactions, question attributes, and metadata.

Our approach utilizes advanced feature extraction techniques, including Singular Value Decomposition (SVD) for dimensionality reduction and multilabel stratified K-Fold cross-validation for robust model validation. We explored GPU-accelerated libraries like CuPy and cuDF for optimization but relied primarily on PyTorch for model training. A variety of machine learning models were developed, evaluated, and optimized using hyperparameter tuning through Weights & Biases (W&B). Special attention was given to feature engineering by employing encoding techniques such as target and label encoding, along with entropy-based selection methods to capture meaningful patterns from the data.

The final models were evaluated using accuracy. Our optimized pipeline achieved competitive results, demonstrating the impact of carefully designed features and a structured approach to cross-validation. Challenges related to data size, computational limits, and imbalanced classes were addressed systematically. This study provides insights into the practical strategies needed to tackle large-scale educational data and emphasizes the importance of feature extraction and model tuning in achieving superior prediction performance.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND PROBLEM STATEMENT

1.1 OVERVIEW OF EDUCATIONAL DATA MINING (EDM)

In recent years, educational institutions and online platforms have started collecting large amounts of data from students-everything from quiz responses to time spent on questions. Analyzing this data is at the heart of Educational Data Mining (EDM), which aims to improve learning by uncovering patterns and insights hidden in these interactions. Predicting how students perform on tasks or respond to questions is one of the most meaningful challenges in this space. With accurate models, we can build personalized learning experiences, offer timely feedback, and help students achieve better outcomes.

The NeurIPS 2020 Education Challenge [12] offers an exciting opportunity to dive into this problem. It provides a multi-layered dataset filled with student interactions, question details, and subject hierarchies. By using this data, the goal is to predict whether students will answer questions correctly and to explore the nature of their responses across different types of questions. This challenge is a perfect playground to apply machine learning techniques while tackling real-world problems in education.

1.2 PROBLEM DEFINITION AND MOTIVATION

The challenge posed by NeurIPS consists of two tasks:

- **Task 1:** Predict if a student will answer a specific question correctly.
- **Task 2:** Predict the nature of the student's response to different types of questions (for example, how confident or accurate the response is).

These tasks require us to dive deep into various factors: How do students perform across different subjects? Do some types of questions tend to confuse students more

than others? How can we make sense of the relationships between subjects and topics? Finding answers to these questions will help us design models that offer more than just predictions—they can provide meaningful insights that educators can act on.

The motivation behind this project is not just about achieving high scores in the competition; it's also about learning how to handle large datasets, extract meaningful patterns, and build scalable models. Educational platforms are growing rapidly, and the insights from this project could be applied to real-world systems that benefit both students and teachers.

1.3 OBJECTIVE OF THE STUDY

This project is guided by a set of practical objectives that align with the tasks outlined in the NeurIPS challenge. The goal is to design a complete and optimized workflow that is robust, repeatable, and capable of delivering accurate insights from the data. The following key objectives were pursued:

1. **Explore and Clean the Dataset:** Before building models, it is essential to fully understand the structure of the data, handle missing values, and ensure consistency. This involves identifying and removing duplicates, filling missing metadata using parent-child relationships, and transforming raw timestamps into meaningful features. Intermediate data checkpoints are saved to avoid reprocessing and speed up execution.

2. **Design, Extract, and Optimize Features:** Raw data is just the starting point. To improve the predictive power of our models, we need to add meaningful features. This includes applying **SVD [5]** for dimensionality reduction on question metadata and calculating group-level statistics (like average student performance) to capture trends in behavior. Hierarchical encoding is also performed to reflect dependencies between questions and subjects.

3. **Build and Evaluate Predictive Models with Cross-Validation:** A range of models—including **CatBoost [9], XGBoost [2], LightGBM**, **LightGBM-2 [7]** and **MLP [6]**—are built to address the challenge. Since the dataset is large and contains imbalanced data, **multilabel stratified K-Fold cross-validation [11]** is used to ensure best generalization across different splits.

4. **Implement Ensemble Models for Improved Performance:** To increase accuracy and generalization, ensemble techniques are employed. This involves combining predictions from individual models, such as LightGBM, CatBoost, XGBoost, and MLP to create more reliable forecasts. Different configurations are also explored to enhance diversity in the ensemble (such as LightGBM-2).

5. **Optimize Performance Using Modern Tools:** The models need to be efficiently trained, So we used PyTorch for model development and track progress through Weights & Biases (W&B) . W&B sweeps help us tune hyperparameters like learning rates and batch sizes for each model, ensuring peak performance.

1.4 SCOPE AND LIMITATIONS

The scope of this project is focused primarily on two aspects: effective feature engineering and the development of scalable predictive models. The dataset offers a rich playground with information about student interactions, question metadata, and subject hierarchies, which makes it a perfect test case for advanced machine learning techniques. However, the project also brings several challenges that need to be addressed:

1. **Handling Large-Scale Data:** With millions of records spread across several files, the sheer size of the data can be overwhelming. We carefully managed the memory utilization and used tools like GPU acceleration to speed things up where possible.

2. **Class Imbalance and Data Sparsity:** Some questions are attempted by only a small group of students, and the distribution of correct and incorrect responses is not always even. This imbalance can skew the predictions, so we'll need to use

cross-validation techniques that account for this.

3. **Extracting Meaning from Hierarchical Data:** Questions in the dataset are organized in a parent-child structure that reflects subject hierarchies. Understanding and encoding these relationships into features is crucial but non-trivial. A thoughtful approach will be needed to capture the dependencies between subjects and questions.

4. **Resource Constraints:** While we utilized Kaggle's GPU resources for training, we still encountered various restrictions which includes limited GPU and CPU resources, need to be mindful of execution time and storage limits. Writing efficient code and selecting the right optimization strategies played a key role in keeping the project on track.

Despite these challenges, we aimed to create a workflow that is not only functional but also insightful. The lessons learned from tackling these challenges will provide valuable experience in handling large datasets and building robust machine learning pipelines.

CHAPTER 2

METHODOLOGY AND IMPLEMENTATION

2.1 DATASET DESCRIPTION

The dataset provided for the NeurIPS 2020 challenge [12] consists of multiple interconnected files, each contributing valuable information about student responses, question metadata, and subjects.

Key Components:

- **Train Data (train_task_1_2.csv):**
  Contains records of student attempts on questions, with details like student IDs, question IDs, timestamps, correctness, and responses. This is the core dataset used for training our models.

- **Test Data (test_public_answers_task_1.csv):**
  Structured similarly to the train data but without the target variables (correctness and response type). This data will be used to evaluate the model's generalization performance.

- **Metadata Files:**

  - **answer_metadata_task_1_2.csv:** Details about question attempts, including submission timestamps (including date), confidence, and attempt counts.

  - **question_metadata_task_1_2.csv:** Provides information about question difficulty, associated subjects, and hierarchical relationships (parent-child).

  - **student_metadata_task_1_2.csv:** Contains demographic information about students, such as date of birth, gender and pupil premium.

2.2 DATA PREPROCESSING AND CLEANING

The quality of the dataset directly influences model performance, so careful preprocessing is essential.

1. **Handling Missing Data:**

    - Missing student or question-level metadata is imputed using averages or filled by propagating values from parent-child subject hierarchies. In some cases, non-critical missing data points are dropped to ensure smoother training.

2. **Checkpoints and Intermediate Data Saves:**
   Given the size of the data, intermediate checkpoints are saved after key steps in preprocessing.

    - Datasets like df_train and df_test are saved in parquet format to avoid repeated processing. This strategy also allows us to resume from checkpoints in case of interruptions during model training or when switching between environments (e.g., Local Machine, Kaggle notebooks).

3. **Data Cleaning and Transformation:**

    - Duplicate and inconsistent records are removed to maintain data integrity.

    - Timestamps are transformed into meaningful time-based features, such as the time between attempts or time spent on individual questions.

4. **Encoding Categorical Features:**

    - Label Encoding is used for categorical columns like subjects.

    - Target Encoding is applied for high-cardinality variables (e.g., student IDs) to retain predictive power.

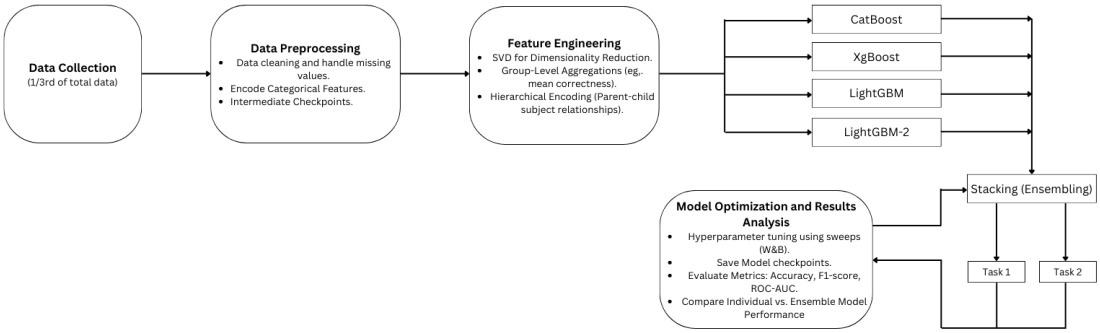


Figure 2.1: Workflow for Model Development, Feature Engineering, and Ensembling Strategy. This figure outlines the sequential steps taken.

2.3 FEATURE ENGINEERING

Feature engineering was crucial in extracting meaningful patterns from the data.

1. **Dimensionality Reduction using SVD:**

    - SVD [5] is used to reduce the dimensionality of question metadata while retaining important patterns. This helps the models learn effectively without being bogged down by irrelevant data.

2. **Group-level Aggregations:**

    - Features such as average correctness per student and mean question difficulty per subject are created. These aggregated statistics provide context that individual data points alone cannot capture.

3. **Hierarchical Encoding:**

    - Parent-child relationships are encoded to capture dependencies, with SubjectId_with_parents representing multi-level subject hierarchies.

2.4 MODEL DEVELOPMENT AND CROSS-VALIDATION STRATEGY

Building robust models requires thoughtful design and evaluation techniques.

1. **Baseline Models:**

    - **Logistic Regression [3]** and **Random Forests [1]** provide a simple but effective benchmark for the advanced models that follow.

2. **Advanced Models:**

    - **CatBoost** (Categorical Boosting) [9] works without one hot encoding, therefore used as a strong baseline to compare with other boosting algorithms, especially given its ability to handle imbalanced data and categorical metadata effectively.

    - **XgBoost** (Extreme Gradient Boosting) [2] was included because of its speed and regularization capabilities, particularly when training models on large datasets. It provided reliable results, serving as both a benchmark and part of ensemble models.

    - **LigthGBM** (Light Gradient Boosting Machine) [7] was employed as one of the primary models due to its ability to handle sparse and high-dimensional data. We used it extensively for both single-model predictions and ensemble strategies.

    - **LightGBM-2** was trained with a different boosting strategy (such as GOSS or DART) to complement the default GBDT-based LightGBM. This version refers to another instance of LightGBM with different configurations. This provided diversity in the ensemble and improved the overall performance.

    - **MLP** [6] is a neural network model that can capture non-linear patterns through fully connected layers. It explores non-linear feature interactions, especially with feature-engineered inputs like SVD-reduced data.

- **MLP MultiClass** handles multi-class classification, where the task involves predicting one of several possible outcomes. MLP MultiClass was used to model student responses with multiple categories, leveraging softmax activation and cross-entropy loss for training.

3. **Ensemble Strategies:** Combining the strengths of multiple models resulted in improved generalization and prediction accuracy.

    - **Ensembling Techniques [4]** like stacking and averaging were used to combine predictions from LightGBM, XGBoost, CatBoost, and MLP models. The ensemble models better than individual algorithms with an accuracy of 0.7094 considering 1/3rd of the data was used.

    - **LightGBM Variants in Ensembles:** Different configurations of LightGBM, including GOSS and DART, were combined to improve robustness. These variants brought complementary strengths, boosting the ensemble's overall accuracy.

4. **Saving Model Checkpoints:**

    - During training, checkpoints are saved after each epoch to avoid loss of progress.

    - This also allows us to resume training without starting from scratch in case of interruptions.

2.5 OPTIMIZATION TECHNIQUES AND TOOLS USED

To make the models more efficient and accurate, we applied multiple levels of optimization.

1. **Optimizer Selection:**

    - **Stochastic Gradient Descent (SGD)** [10] with momentum speeds up convergence by reducing oscillations in the learning process.

    - **Adam** and **Nadam optimizers** [8] are employed to handle sparse gradients effectively, ensuring smoother training on large datasets.

2. **Hyperparameter Tuning with W&B Sweeps:**

    - Using Weights & Biases (W&B), we performed hyperparameter sweeps to explore various learning rates, batch sizes, and optimizer configurations. This process helps identify the best parameters for each model.

3. **GPU Acceleration:**

    - Although **CuPy and cuDF** were explored initially, we found that **PyTorch** provided better integration with our models.

4. **Data Management Optimization:**

    - Large intermediate datasets were saved in parquet format to minimize memory usage and speed up I/O operations.

    - Preprocessed data and trained models were stored across multiple checkpoints, allowing us to pick up from any stage without reprocessing everything.

CHAPTER 3

RESULTS AND DISCUSSION

3.1 EVALUATION METRICS

Given the nature of our tasks, accuracy was chosen as the sole evaluation metric to measure the models' performance. Accuracy reflects the proportion of correct predictions over the total predictions made:

1. **Accuracy:** Measures the percentage of correct predictions. However, it alone is insufficient for imbalanced datasets.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Since accuracy aligns with the goal of the task—predicting the correctness of student responses—it provided a clear and interpretable measure of how well the models performed. While other metrics like F1 Score or ROC-AUC could offer nuanced insights for imbalanced datasets, accuracy remained sufficient in this context, as our models were trained with stratified cross-validation to address class imbalances.

3.2 MODEL PERFORMANCE ON VALIDATION & TEST DATA

1. **Baseline Models:**

    - Logistic Regression and Random Forests served as benchmarks, achieving moderate accuracy but struggling with imbalanced data.

2. **Advanced Models:** We achieved better results by using advanced models capable of capturing complex patterns in the data.

    - **CatBoost:** CatBoost handled categorical features efficiently and performed

well with minimal preprocessing, providing competitive results.

- **XGBoost:** XGBoost's speed and built-in regularization made it effective, especially on large datasets, contributing significantly to the ensemble.

- **LightGBM:** LightGBM demonstrated excellent performance on high-dimensional data, becoming one of the key models in the ensemble strategy.

- **LightGBM-2:** A second instance of LightGBM, configured with GOSS/DART, improved the ensemble's diversity and robustness.

3. **MLP (Multi-Layer Perceptron):** MLP captured non-linear interactions in the data, offering improved predictive power.

4. **MLP MultiClass:** This model effectively handled multi-class prediction tasks, helping capture the complexity of student responses with multiple possible categories.

5. **Ensemble Models:** The use of ensemble techniques (e.g., stacking or averaging) improved generalization and boosted overall accuracy by leveraging the strengths of multiple models.

    - The ensemble of CatBoost, XGBoost, and LightGBM models outperformed individual models, reducing the impact of overfitting and handling imbalanced data more effectively.

6. **Impact of Feature Engineering:** Models with **SVD-based features** showed a noticeable improvement in convergence and accuracy. The **hierarchical encoding** of subjects and questions added depth to the feature set, improving prediction performance further.

Addressing the complexities of missing values, hierarchical subject relationships, and class imbalances required a combination of careful preprocessing, advanced feature

Algorithm	Validation Accuracy	Test Accuracy
CatBoost	0.7431	0.7093
LightGBM	0.7422	0.7104
LightGBM-2	0.7421	0.7097
XgBoost	0.7426	0.7103
MLP	0.6408	0.6410
MLP-MultiClass	0.6410	0.6408
Ensemble (Wgtd. [1]Voting)	0.7425	0.7111

Table 3.1: Comparison of Validation and Test Accuracy for Various Models.

engineering, and diverse modeling approaches. The use of ensemble techniques and hyperparameter tuning further ensured that the models were capable of delivering meaningful predictions across varied educational scenarios.

3.3 CHALLENGES IN THE DATASET AND TASKS

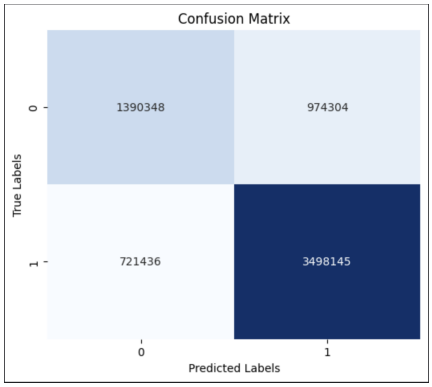
The dataset and tasks posed several challenges that required strategic solutions, efficient resource management, and thoughtful design:

1. **Handling Large-Scale Data with Resource Constraints:** Given the size of the dataset (millions of records) and limited GPU resources, processing the entire dataset at once was infeasible. To overcome this, we processed 1/3rd of the data at a time, focusing on subsets for each model. Intermediate models and outputs were saved to disk, and unnecessary variables were deleted from memory to free up resources. This allowed us to run multiple models sequentially without exhausting system memory.

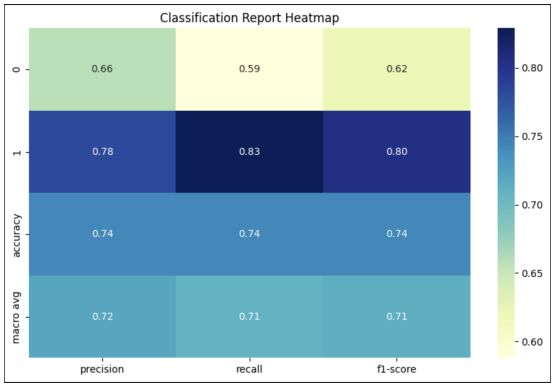
2. **Model-by-Model Execution:** Instead of running all models simultaneously, we ran one model at a time, saved its output, and then cleared memory to make space for the next run. This strategy minimized memory usage and avoided GPU session interruptions.

3. **High Dimensionality:** With a large number of interdependent features (e.g.,

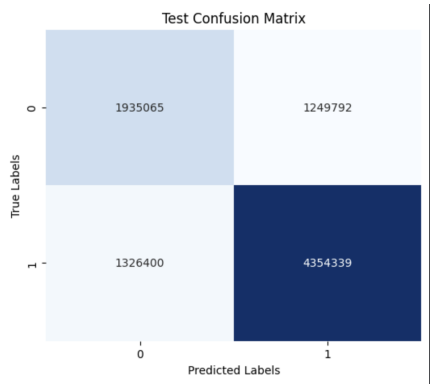
[1]*Wgtd - Weighted



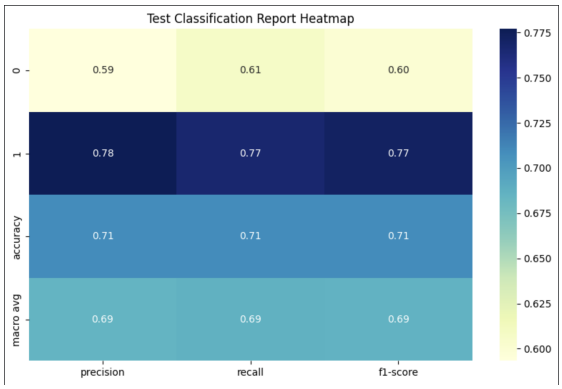
(a) Confusion Matrix (Validation Data).



(b) Classification Report (Validation Data).



(c) Confusion Matrix (Test Data).



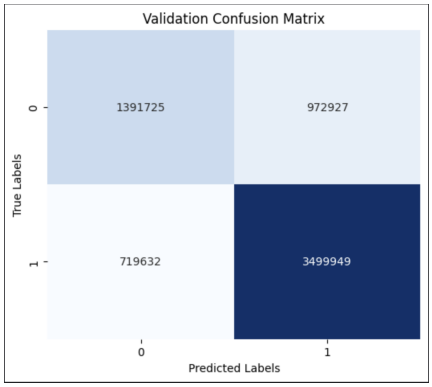
(d) Classification Report (Test Data).

Figure 3.1: Performance Metrics for Stacking Ensemble without individual algorithms.

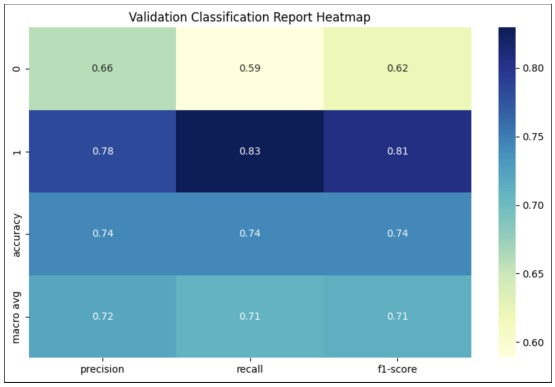
hierarchical relationships between questions and subjects), dimensionality reduction techniques such as SVD were essential to reduce complexity and speed up model training without losing predictive power.

4. **Missing Data:** Certain fields, such as student demographics, had missing values. These were imputed using relevant statistics (e.g., mean or mode), and non-critical missing fields were dropped to avoid introducing noise.

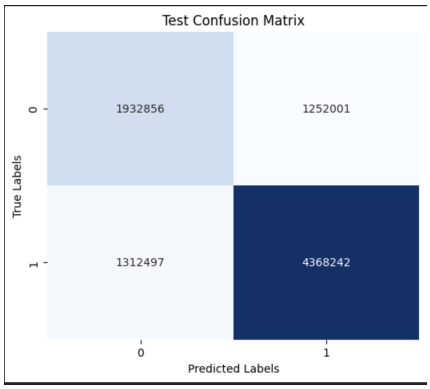
5. **Hierarchical Relationships:** The questions were organized in a hierarchical structure (subject, parent, child), and encoding these dependencies was essential but computationally intensive. We optimized feature engineering to preserve key relationships without overloading the models, ensuring the **hierarchical encoding** enriched the feature set meaningfully.



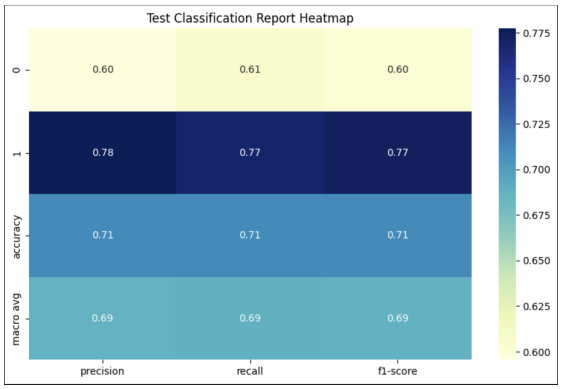
(a) Confusion Matrix (Validation Data).



(b) Classification Report (Validation Data).



(c) Confusion Matrix (Test Data).



(d) Classification Report (Test Data).

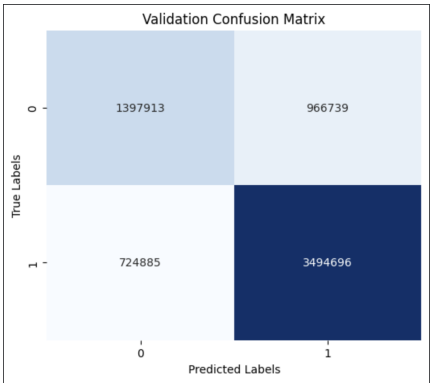
Figure 3.2: Performance Metrics for Stacking Ensemble using CatBoost algorithm.

6. **Limited Training Resources:** With limited resources, we focused on writing optimized code to minimize execution time. GPU sessions were carefully managed with checkpoints to prevent loss of progress. W&B sweeps allowed us to efficiently tune hyperparameters without wasting resources on redundant runs.

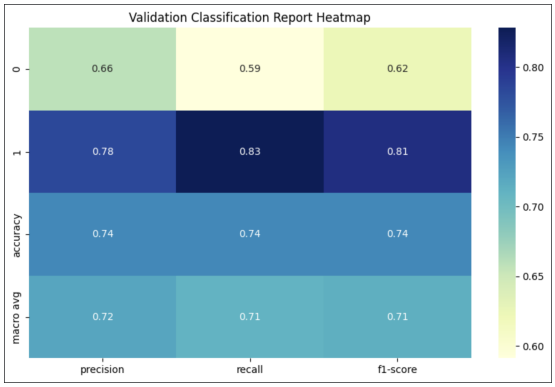
3.4 KEY LEARNINGS AND OBSERVATIONS

This project provided valuable insights into handling large-scale data, resource management, and model optimization. Below are the key learnings and observations that emerged from the process:

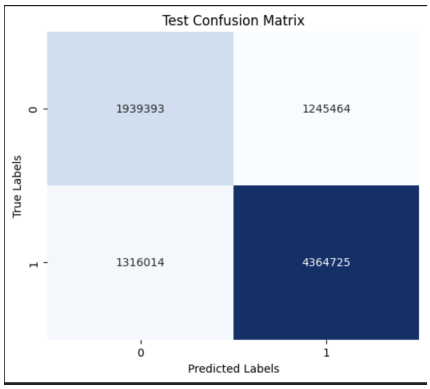
1. **The Importance of Incremental Data Processing** Breaking the dataset into smaller, manageable chunks allowed us to work within limited memory constraints. This incremental approach ensured that GPU resources were used efficiently which



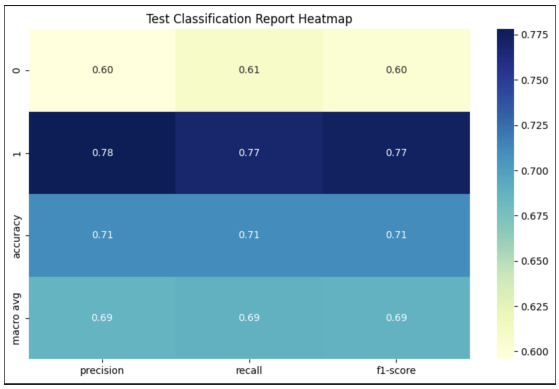
(a) Confusion Matrix (Validation Data).



(b) Classification Report (Validation Data).



(c) Confusion Matrix (Test Data).



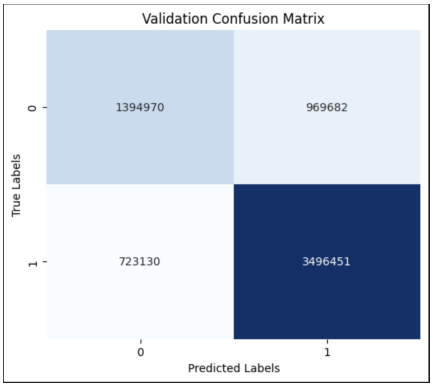
(d) Classification Report (Test Data).

Figure 3.3: Performance Metrics for Stacking Ensemble using XGBoost algorithm.

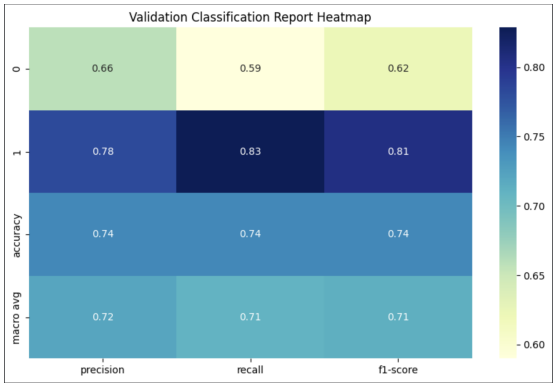
may result in compromising the quality of the models.

2. **Model Diversity and Ensemble Effectiveness** Using multiple algorithms significantly improved prediction performance. The ensemble strategy introduced diversity in predictions and reduced the impact of overfitting, delivering better generalization across validation sets.

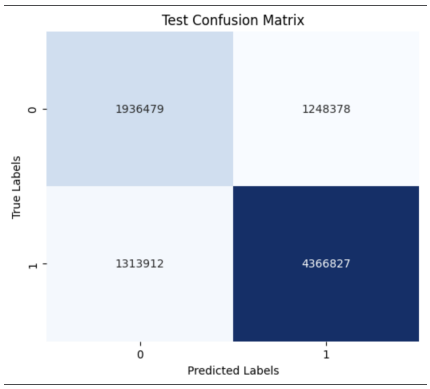
3. **Impact of Feature Engineering and Hierarchical Encoding** Feature engineering techniques, such as SVD-based dimensionality reduction and hierarchical encoding of subject dependencies, played a critical role in improving model performance. Encoding hierarchical relationships (e.g., using SubjectId_with_parents) preserved important dependencies without overwhelming the model with unnecessary complexity.



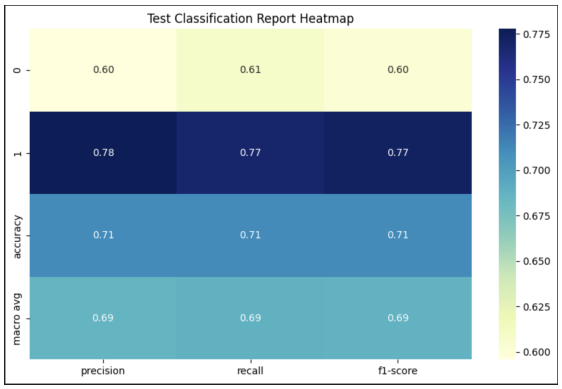
(a) Confusion Matrix (Validation Data).



(b) Classification Report (Validation Data).



(c) Confusion Matrix (Test Data).



(d) Classification Report (Test Data).

Figure 3.4: Performance Metrics for Stacking Ensemble using LightGBM algorithm.

4. **Balancing Class Distribution with Cross-Validation** The use of multilabel stratified K-Fold cross-validation ensured that class imbalances (e.g., correctness vs. incorrect responses) did not bias the models. This approach improved the reliability of model evaluation and ensured better generalization.

5. **Managing Resource Constraints with Model-by-Model Execution** Running one model at a time and saving outputs incrementally allowed us to optimize memory usage and avoid session interruptions. This method ensured that we could explore a range of models and configurations despite limited computational resources.

By employing incremental data processing, ensembling techniques, and careful tuning, we achieved reliable performance under constrained conditions. These insights provide a valuable framework for tackling similar large-scale machine learning tasks in future projects.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 SUMMARY OF FINDINGS

This project addressed the NeurIPS 2020 challenge by combining advanced machine learning models, feature engineering, and ensemble strategies. Given the dataset's complexity, careful preprocessing and feature extraction were essential to capture meaningful patterns. SVD-based dimensionality reduction managed high-dimensional metadata while retaining key information, and hierarchical encoding reflected subject, parent, and child relationships, enriching the feature space.

A key takeaway was the importance of model diversity. We experimented with models like CatBoost, XGBoost, LightGBM (and its variants), and MLPs, each contributing uniquely. LightGBM-2 with alternate boosting strategies (e.g., GOSS, DART) further diversified the ensemble, enhancing generalization. Ensemble methods proved crucial for accuracy and reliability, as aggregating predictions reduced overfitting and mitigated class imbalance, allowing for better performance on both common and rare responses.

Despite resource constraints, incremental data processing and checkpoints ensured smooth model execution. W&B hyperparameter sweeps helped efficiently optimize parameters, while GPU acceleration with PyTorch enabled faster training.

In conclusion, this project demonstrated the value of model diversity, ensembling, and optimized resource management. It provides a robust framework for handling large-scale machine learning challenges by emphasizing feature engineering, ensemble strategies, and efficient workflows.

4.2 LIMITATIONS OF THE STUDY

While the project achieved promising results, several limitations were encountered:

1. **Resource Constraints:** Due to limited GPU resources, we processed only 1/3rd of the dataset at a time. This restricted the scale of our experiments and limited some ensemble strategies.

2. **Sequential Model Execution:** Running one model at a time and clearing memory between runs, although necessary, slowed down the experimentation process.

3. **Handling Class Imbalances:** Despite using stratified cross-validation, some models still struggled with imbalanced data, indicating the need for more advanced techniques such as SMOTE or weighted loss functions.

4. **Feature Engineering Scope:** Although hierarchical encoding and aggregation improved performance, further exploration of interaction effects (e.g., pairwise feature combinations) could have yielded better results.

4.3 FUTURE SCOPE

This project opens several avenues for further exploration and improvements:

1. **Expand the Scale of Experiments:** With access to more powerful computational resources, larger portions of the dataset can be processed at once, enabling deeper analysis and more extensive hyperparameter tuning.

2. **Complete Data Utilization for Better Accuracy:** Future iterations will focus on processing the complete dataset to improve model accuracy and extract more robust insights across tasks.

3. **Explore Advanced Ensembling Techniques:** Future work could include experimenting with stacking, weighted averaging, and meta-learning ensembles to further boost performance.

4. **Incorporate Advanced Imbalance Handling:** Applying techniques like SMOTE, adaptive resampling, or class-weighted loss functions could improve model performance on imbalanced data.

5. **Deep Learning Approaches:** The use of more complex neural networks, such as transformers or graph-based models, could enhance performance, particularly for capturing relationships within hierarchical data.

6. **Extend to Additional Tasks:** In addition to improving on the current task, future work will focus on Task 2 and Task 3 to complete the full challenge. Using deep learning algorithms across these tasks will be prioritized to improve predictive accuracy.

4.4 FINAL TAKEAWAYS

This project effectively addressed the NeurIPS 2020 challenge by utilizing advanced machine learning models, feature engineering, and ensemble techniques. Careful preprocessing, SVD-based dimensionality reduction, and hierarchical encoding of subject relationships allowed us to uncover meaningful patterns in the data.

We employed multiple models—CatBoost, XGBoost, LightGBM (and its variants), and MLPs—to enhance prediction performance. Ensemble methods boosted accuracy further by reducing overfitting and improving generalization across diverse data subsets. Despite limited computational resources, we maintained smooth execution through checkpoints, intermediate saves, and sequential model training.

BIBLIOGRAPHY

[1] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[2] Tianqi Chen and Carlos Guestrin. "XGBoost: A scalable tree boosting system". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 785–794.

[3] J.S. Cramer. *The Origins of Logistic Regression*. Tech. rep. 02-119/4. Tinbergen Institute, 2002.

[4] M. A. Ganaie et al. "Ensemble deep learning: A review". In: *arXiv preprint arXiv:2104.02395* (2021).

[5] Gene H Golub and William Kahan. "Calculating the singular values and pseudo-inverse of a matrix". In: *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* 2.2 (1965), pp. 205–224.

[6] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[7] Guolin Ke et al. "LightGBM: A highly efficient gradient boosting decision tree". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[8] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[9] Liudmila Prokhorenkova et al. "CatBoost: unbiased boosting with categorical features". In: *arXiv preprint arXiv:1706.09516* (2018).

[10] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[11] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. "On the stratification of multi-label data". In: *Machine Learning and Knowledge Discovery in Databases* (2011), pp. 145–158.

[12] Zichao Wang et al. "Diagnostic Questions: The NeurIPS 2020 Education Challenge". In: *NeurIPS 2020 Competition and Demonstration Track*. Vol. 133. PMLR, 2021, pp. 191–205. URL: https://proceedings.mlr.press/v133/wang21a.html.