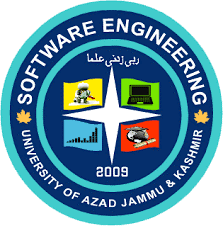
****

**The University of Azad Jammu and Kashmir**

**Department of Software Engineering**

**PROJECT REPORT**

**Course Instructor:** Engr. Ahmed Khawaja **Semester:** Fall-2024

**Submission Date:** Feb 12, 2024 **Session:** 2022-2026

**Course Name:** Machine Learning **Code:** SE-3102

**Submitted By:**

**Farah Qayoom** (2022-SE-28)

**Khushbakht Razzaq**(2022-SE-38)

**Rejja Abbasi**(2022-SE-04)

Table of Contents

[**Historical Background:** 3](#_Toc195307554)

[**Competitions and Benchmarks:** 3](#_Toc195307555)

[**Significance in Machine Learning:** 3](#_Toc195307556)

[**Conclusion:** 4](#_Toc195307557)

[**Step-by-Step Explanation of Kaggle CIBMTR Competition** 4](#_Toc195307558)

[**Step 1: Setup Kaggle Account** 4](#_Toc195307559)

[**Step 2: Join the Competition** 4](#_Toc195307560)

[**Step 3: Download the Dataset** 4](#_Toc195307561)

[**Step 4: Data preprocessing , Training** 5](#_Toc195307562)

[**Step 5: Testing** 10](#_Toc195307563)

[**Model Inference and Submission Pipeline Explanation** 10](#_Toc195307564)

[**Key Points** 14](#_Toc195307565)

[**Model Inference and Submission Code – Detailed Explanation** 14](#_Toc195307566)

[**Explanation Of Used Models:** 19](#_Toc195307567)

[**1.** **cat\_model.pkl** 19](#_Toc195307568)

[**2.** **lgb\_model.pkl** 19](#_Toc195307569)

[**3.** **xgb\_model.pkl** 19](#_Toc195307570)

[**4.** **meta\_model.pkl** 19](#_Toc195307571)

[**5.** **ensemble\_models.pkl** 19](#_Toc195307572)

[**Final Scores Result:** 20](#_Toc195307573)

​

**Overview about competition**

The **MNIST (Modified National Institute of Standards and Technology**) dataset is a renowned collection of handwritten digits widely utilized in the field of machine learning, particularly for image classification tasks. It comprises **70,000** grayscale images of handwritten digits from **0** to **9**, each sized at **28×28** pixels. Specifically, the dataset is divided into **60,000** training images and **10,000** testing images.

# **Historical Background:**

The origins of the **MNIST** dataset trace back to the late 1980s and early 1990s when the U.S. National Institute of Standards and Technology (NIST) sought to facilitate the development of optical character recognition (OCR) systems. Initially, NIST released datasets such as Special Database 3 (SD-3) and Special Database 7 (SD-7), containing segmented handwritten characters. However, these datasets exhibited variations in writing styles and quality. To create a more standardized dataset, researchers Yann LeCun, Corinna Cortes, and Christopher J.C. Burges combined samples from SD-3 and SD-7, resulting in the MNIST dataset. This new dataset underwent preprocessing steps, including size normalization and centering of digit images, to ensure uniformity. ​

# **Competitions and Benchmarks:**

The **MNIST** dataset has been pivotal in benchmarking image classification algorithms. Notably, in 1992, NIST and the U.S. Census Bureau organized a competition to evaluate the performance of various OCR systems using datasets like SD-3 and SD-7. This competition attracted participation from multiple organizations and highlighted the challenges of handwritten digit recognition. The insights gained from this event contributed to the development of more robust machine learning models and underscored the importance of standardized datasets like MNIST for algorithm evaluation. ​

# **Significance in Machine Learning:**

The **MNIST** dataset has become a foundational benchmark for assessing image classification algorithms. Its simplicity and well-structured format make it an ideal starting point for individuals new to machine learning. Researchers and practitioners have employed MNIST to test various models, including neural networks, support vector machines, and convolutional neural networks, facilitating the comparison of methodologies and tracking advancements in the field. ​

# **Conclusion:**

Incorporating details about the MNIST dataset's history, its role in competitions, and its significance in machine learning will enrich your project report, providing a comprehensive overview of its impact and applications in the field.

# **Step-by-Step Explanation of Kaggle CIBMTR Competition**

## **Step 1: Setup Kaggle Account**

1. Sign up for **Kaggle** .
2. Verifying **email, identity (persona),** and **phone number** (required for full access).

## **Step 2: Join the Competition**

1. Then we Go to the **CIBMTR** Competition page.
2. Click **"Join Competition"** and we accept the terms.

## **Step 3: Download the Dataset**

* 1. Download the training dataset .

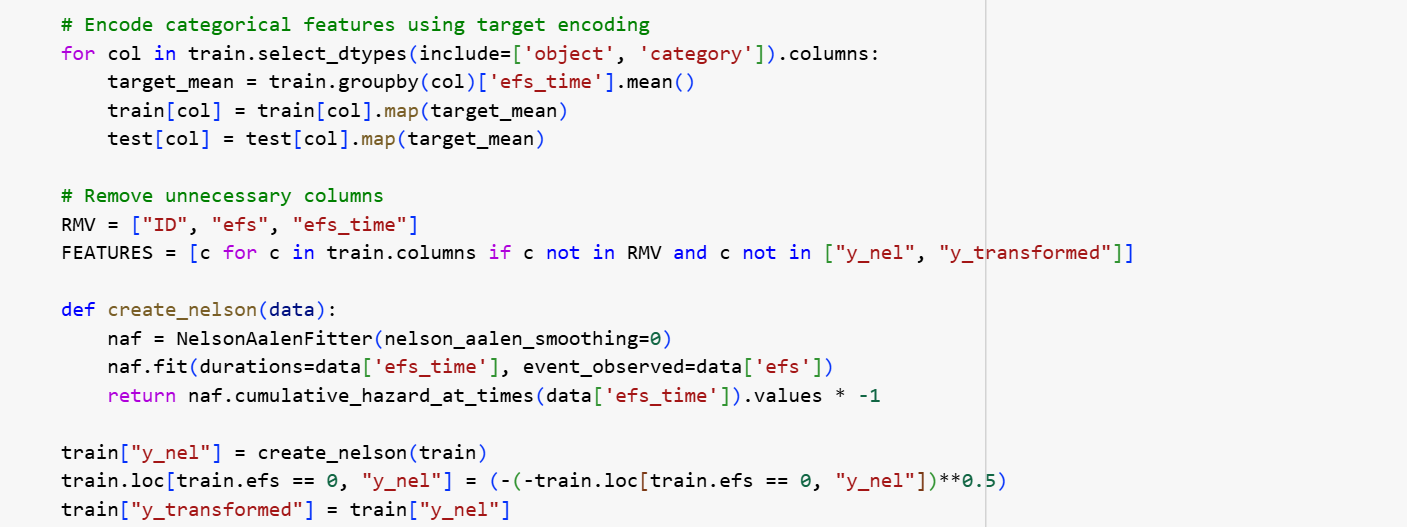


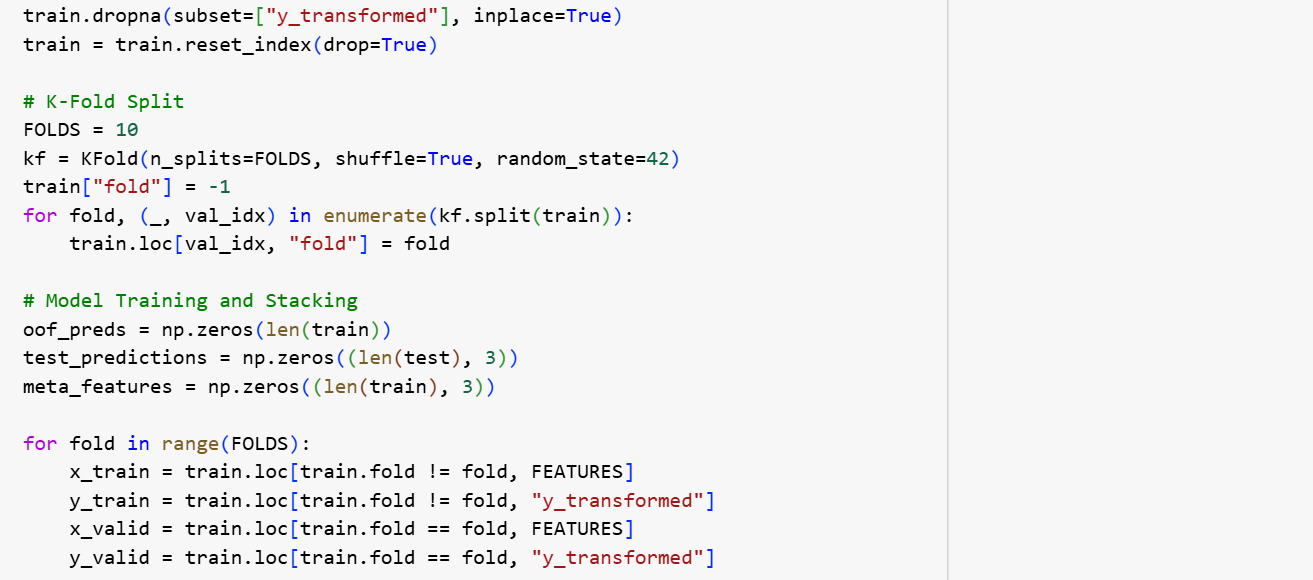
Figure 1

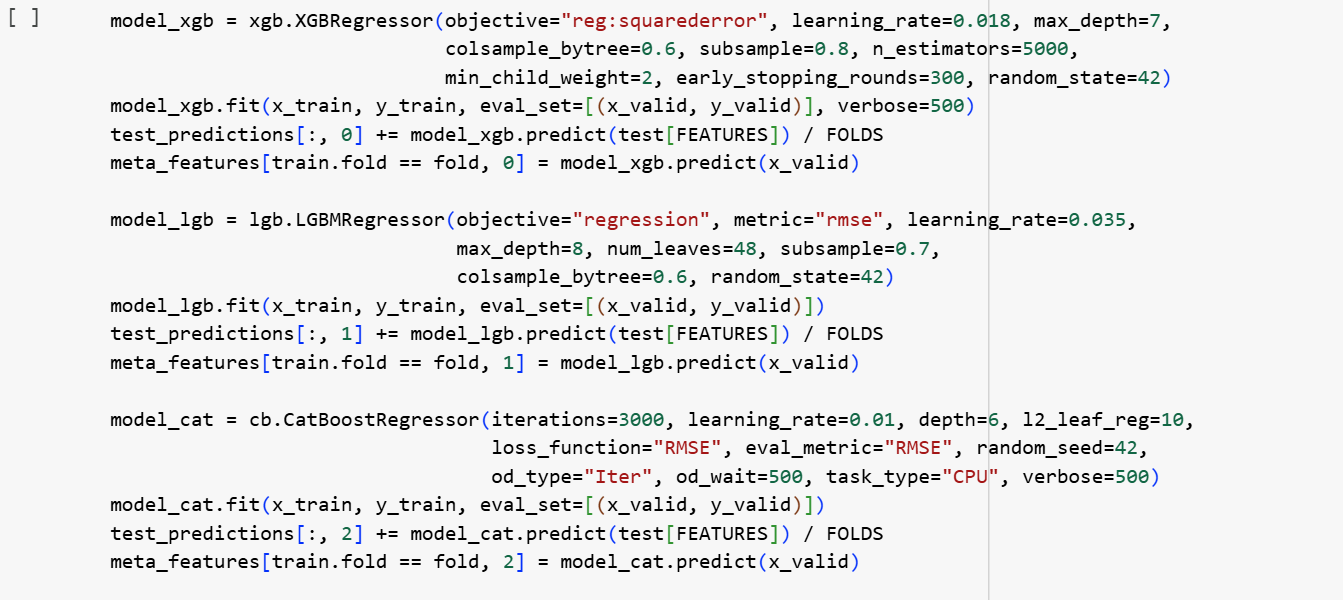
## **Step 4: Data preprocessing , Training**





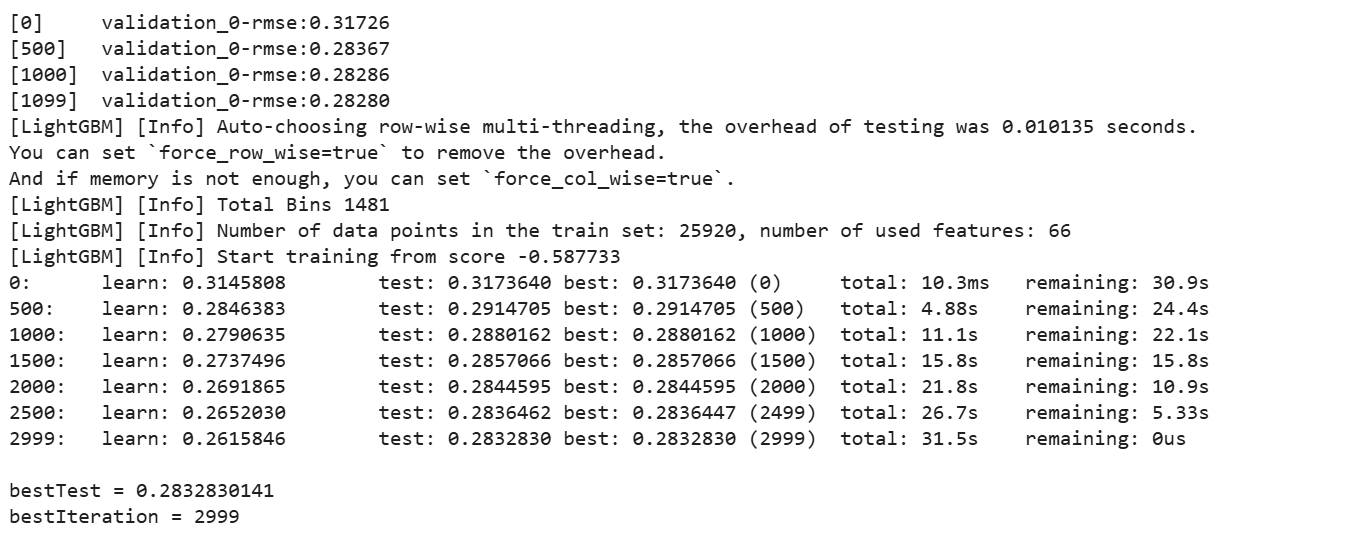


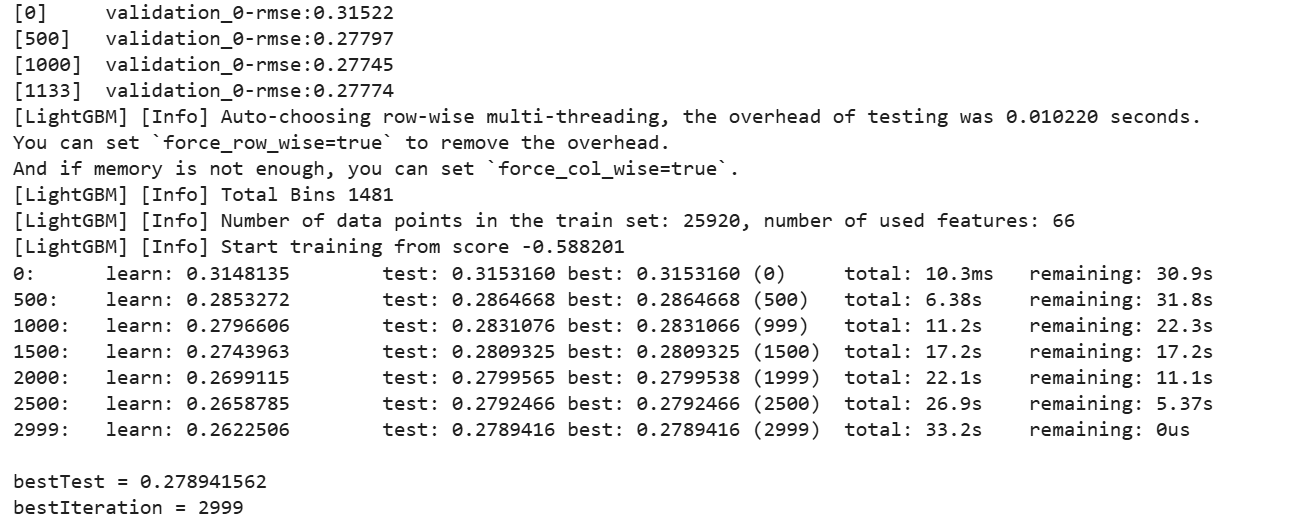


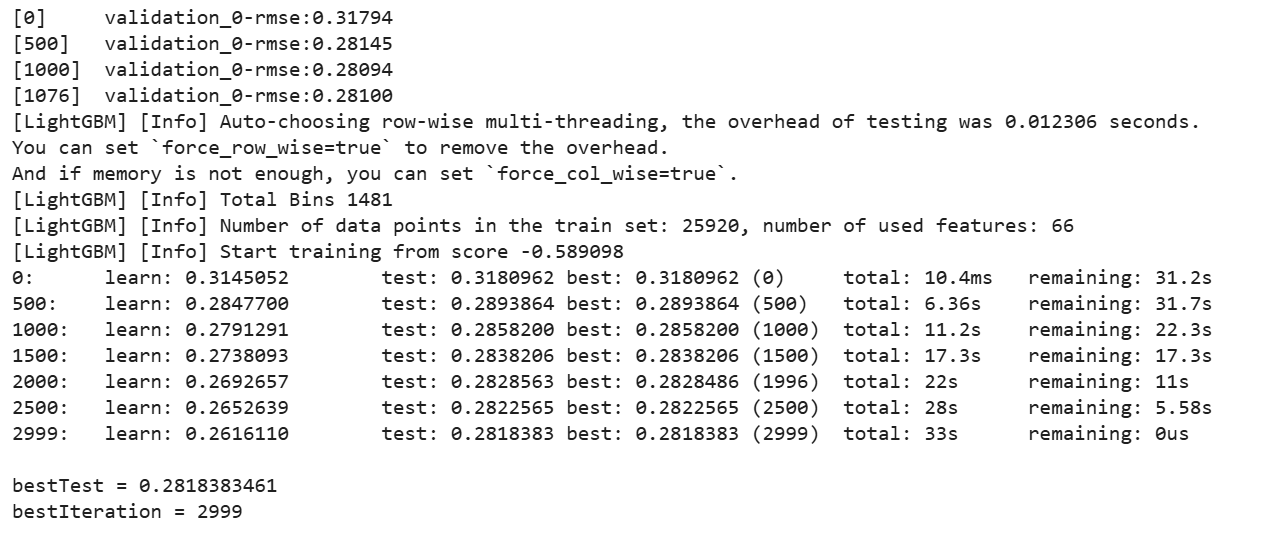


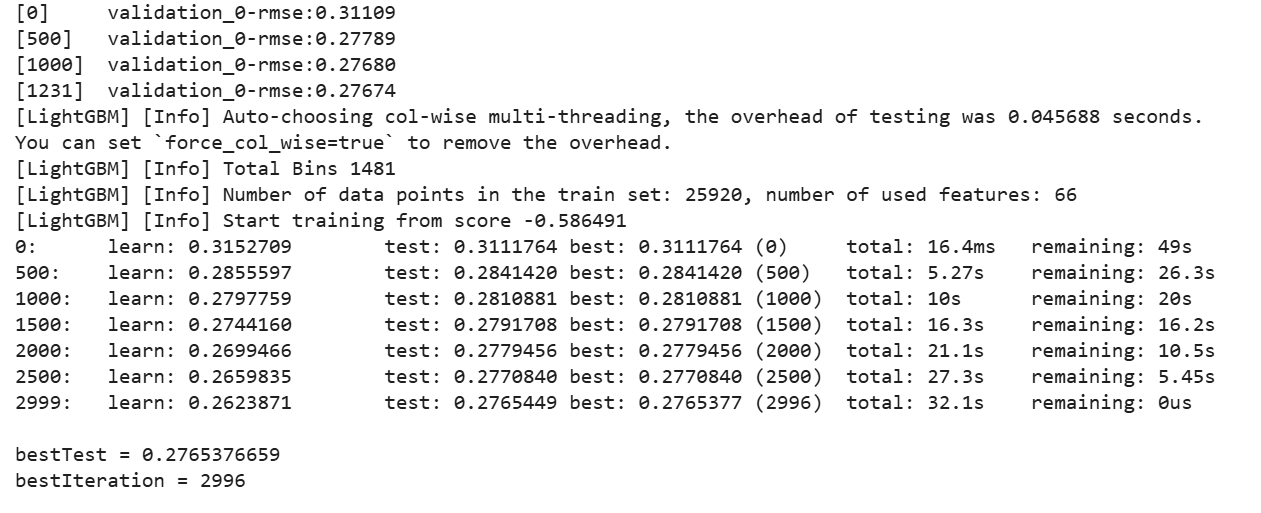


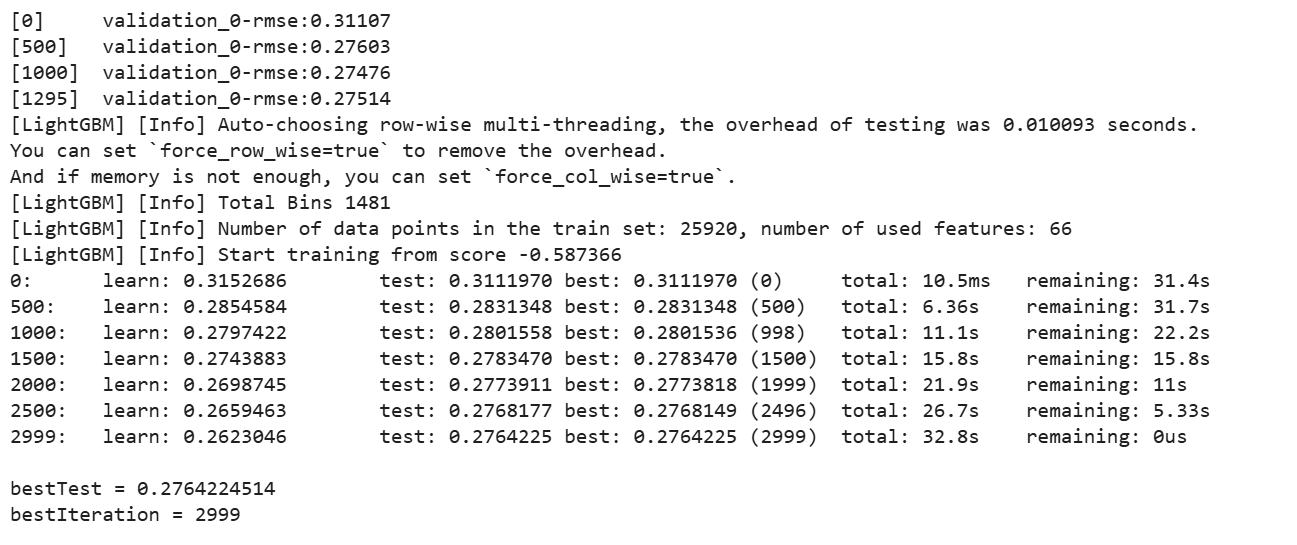
**OUTPUT**

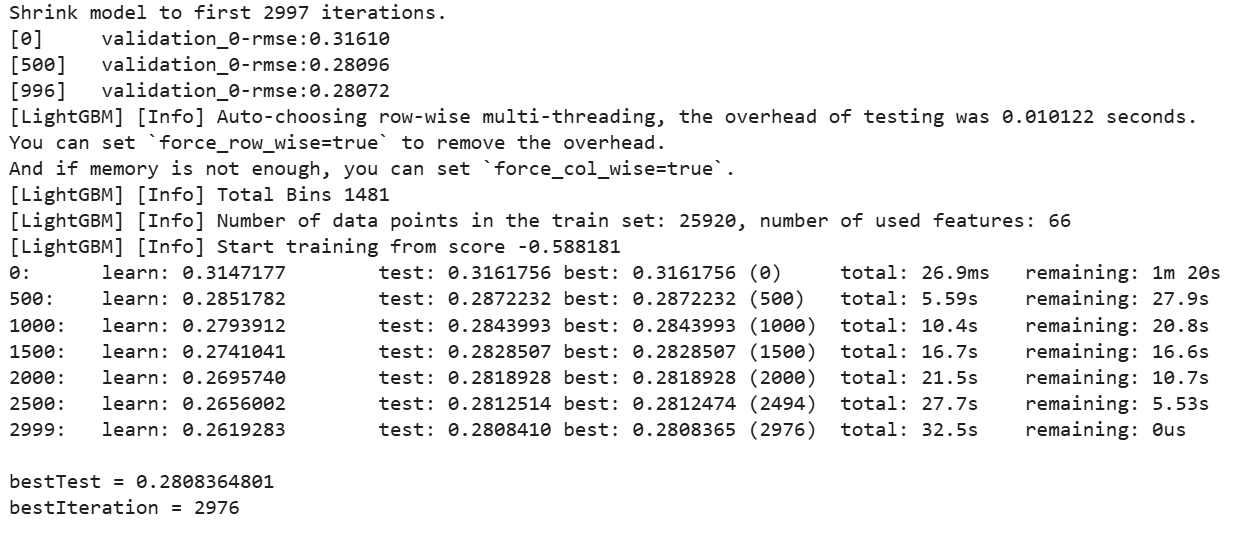


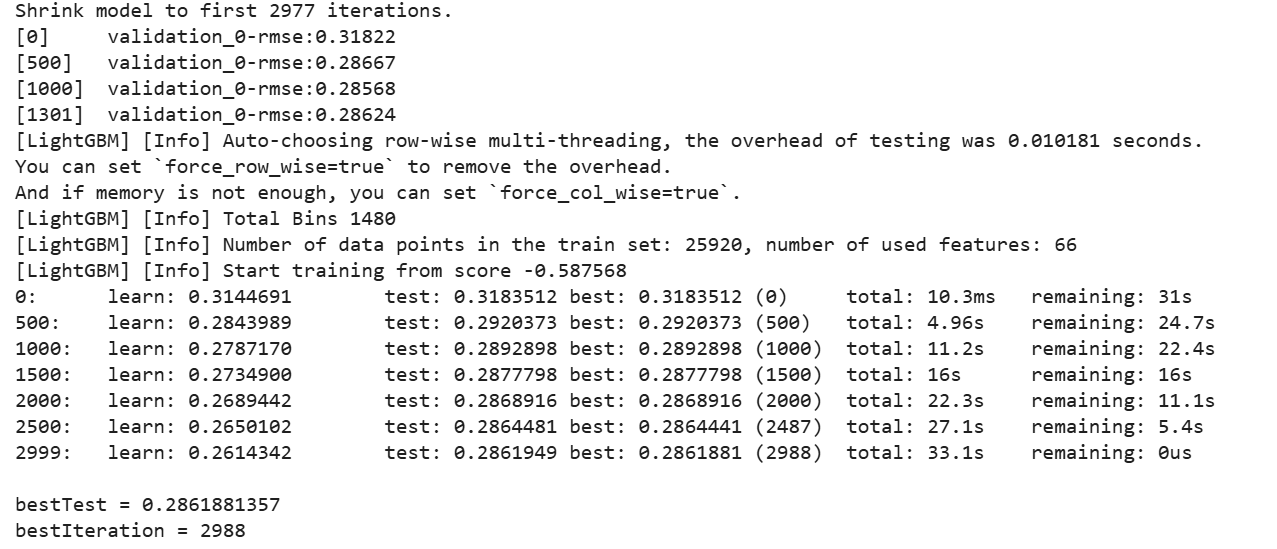


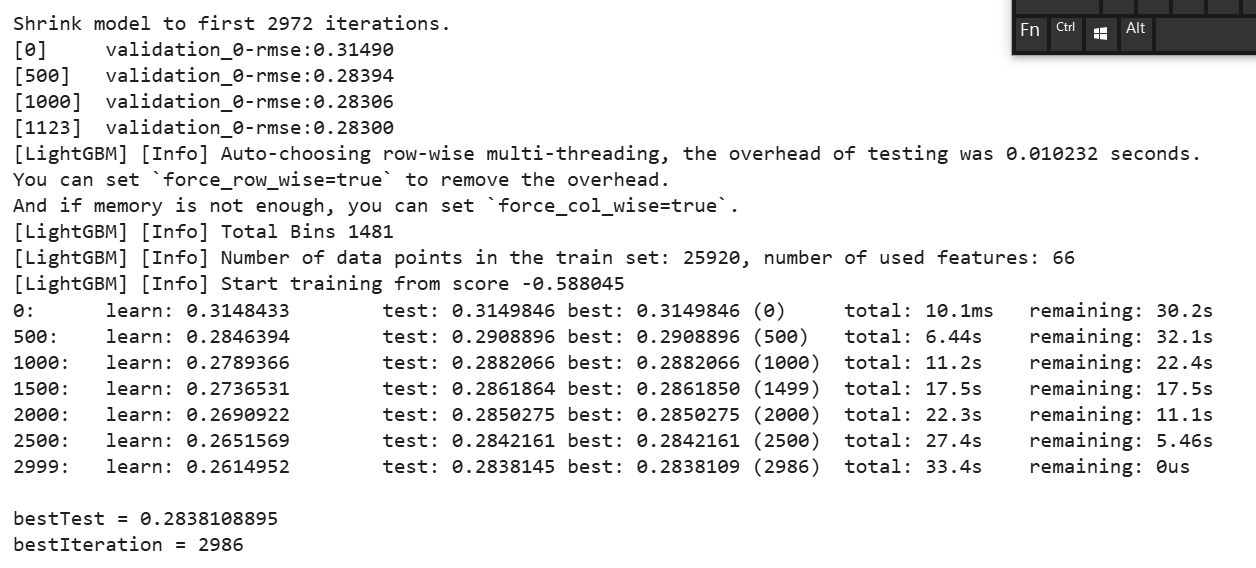


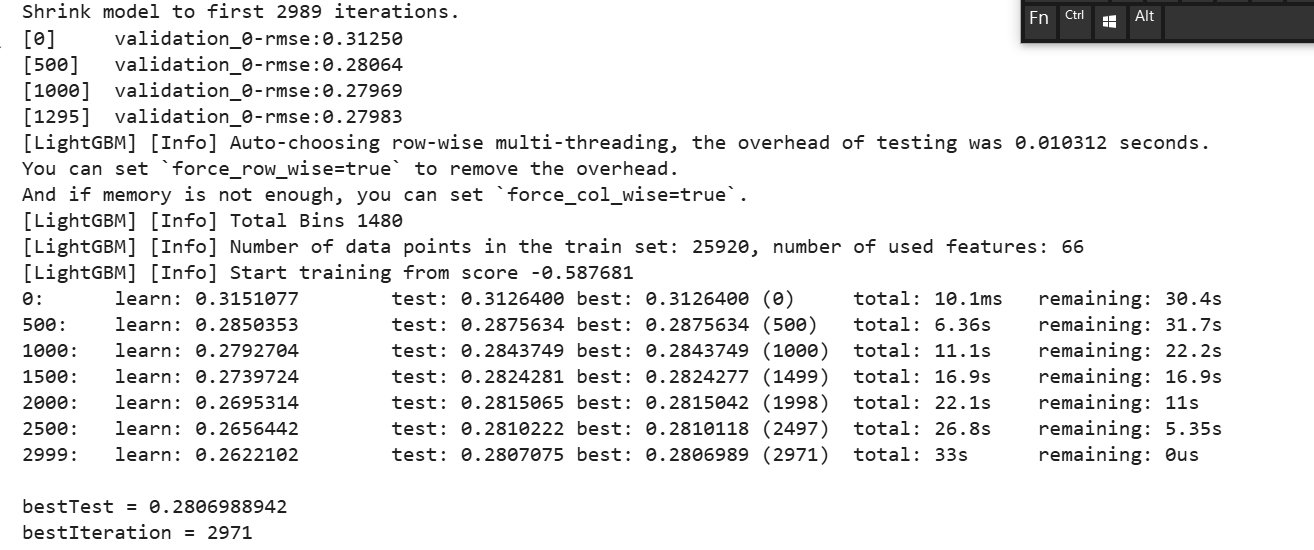


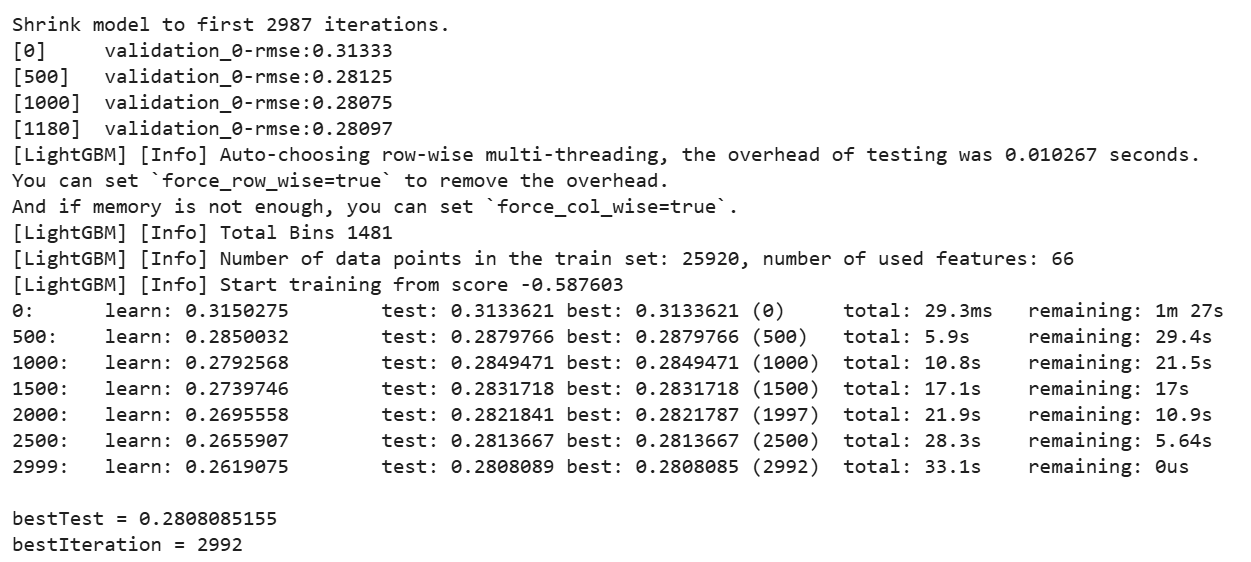


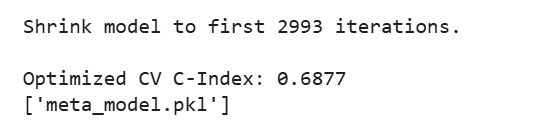












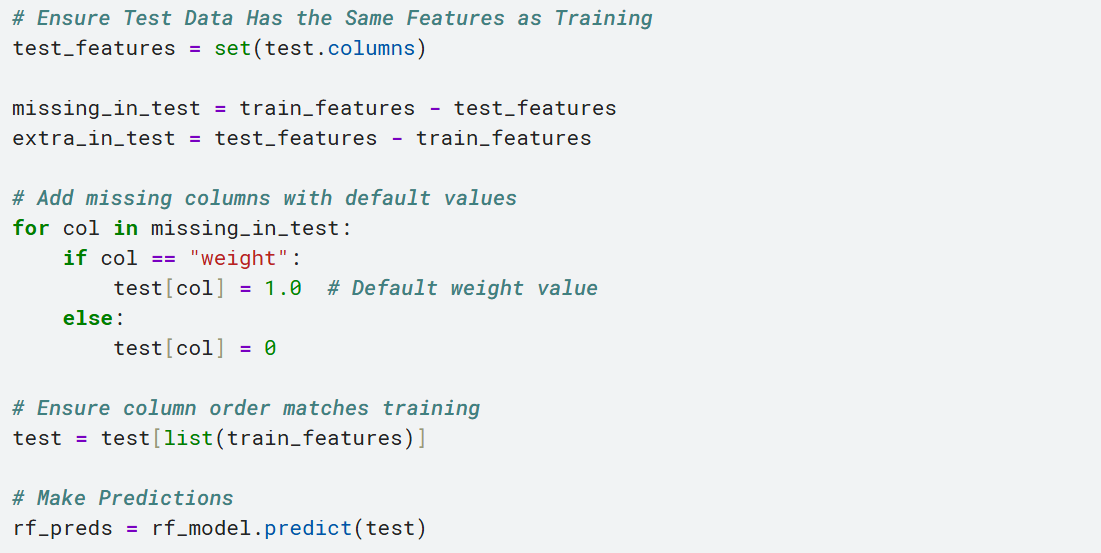
## **Step 5: Testing**

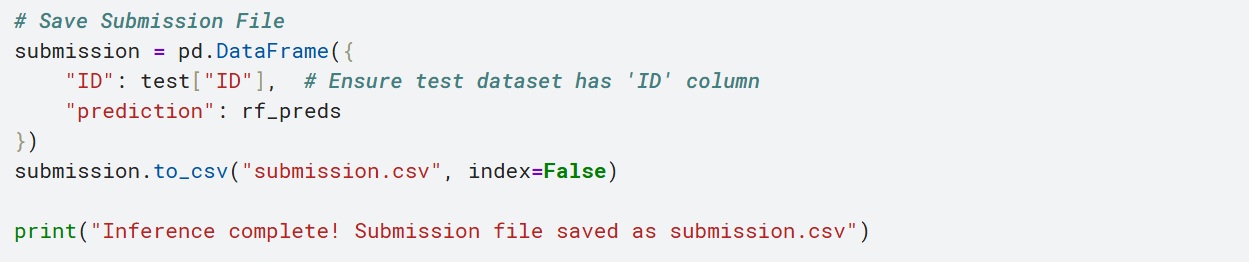
## **Model Inference and Submission Pipeline Explanation**

This Python script is used for generating predictions using a pre-trained machine learning model as part of the **CIBMTR - Equity in post-HCT Survival Predictions** competition hosted on Kaggle. The aim is to predict transplant survival rates for patients who underwent allogeneic Hematopoietic Cell Transplantation (HCT). Below is a breakdown of the steps performed by the code:

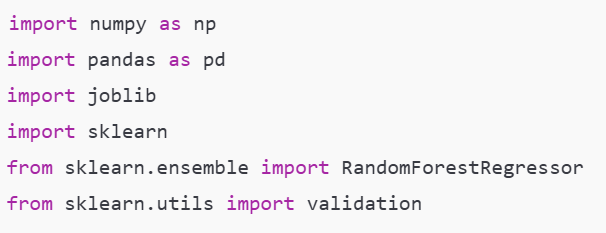








1. **Importing Required Libraries:**

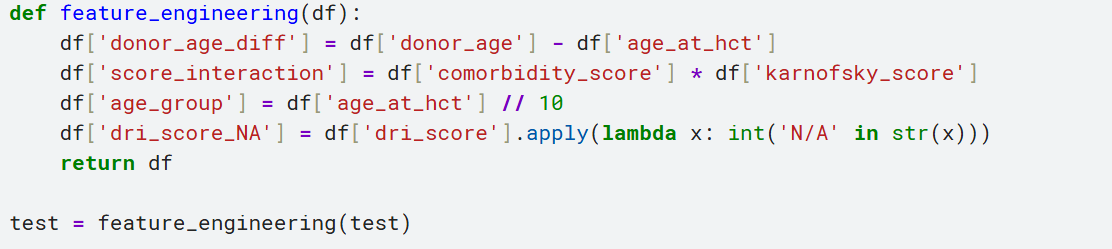


1. **NumPy & Pandas**: For data manipulation.
2. **joblib**: For loading the serialized model and other preprocessed data.
3. **sklearn**: To check model compatibility and use machine learning utilities.
4. **RandomForestRegressor**: Represents the model type (even though it’s not used directly for training here).
5. **validation**: To ensure the loaded model is fitted before prediction.
6. **Loading and Preprocessing the Test Data**



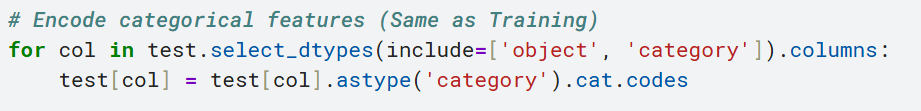
* **Loads the test data from the specified Kaggle dataset directory.**

#### **Feature Engineering Function**



* **Adds new features to enhance model performance:**
  1. **donor\_age\_diff:** Difference between donor and patient age.
  2. **score\_interaction:** Product of comorbidity and Karnofsky scores.
  3. **age\_group:** Binned age.
  4. **dri\_score\_NA:** Flag for missing DRI scores.

1. **Encoding Categorical Variables**



Converts object or category type columns into numeric codes so that they are compatible with the model.

1. **Scikit-learn Version Check**



Prints the version to ensure compatibility with the model.

1. **Load Pre-trained Model**



* Loads a trained RandomForestRegressor model from a .pkl file.
* Uses check\_is\_fitted() to confirm the model is ready for inference.

1. **Aligning Test Features with Training**



* Loads the original training feature set to ensure the test data has the same columns and order.
* Any missing features are filled with zeros (or 1.0 for "weight").
* Extra columns not present in training are ignored.

1. **Make Predictions**



* Runs the test data through the model to generate predictions.

1. **Save Submission File**



**Creates a CSV file for submission containing:**

* **ID:** The identifier from the test set.
* **Prediction:** The model’s predicted survival score.

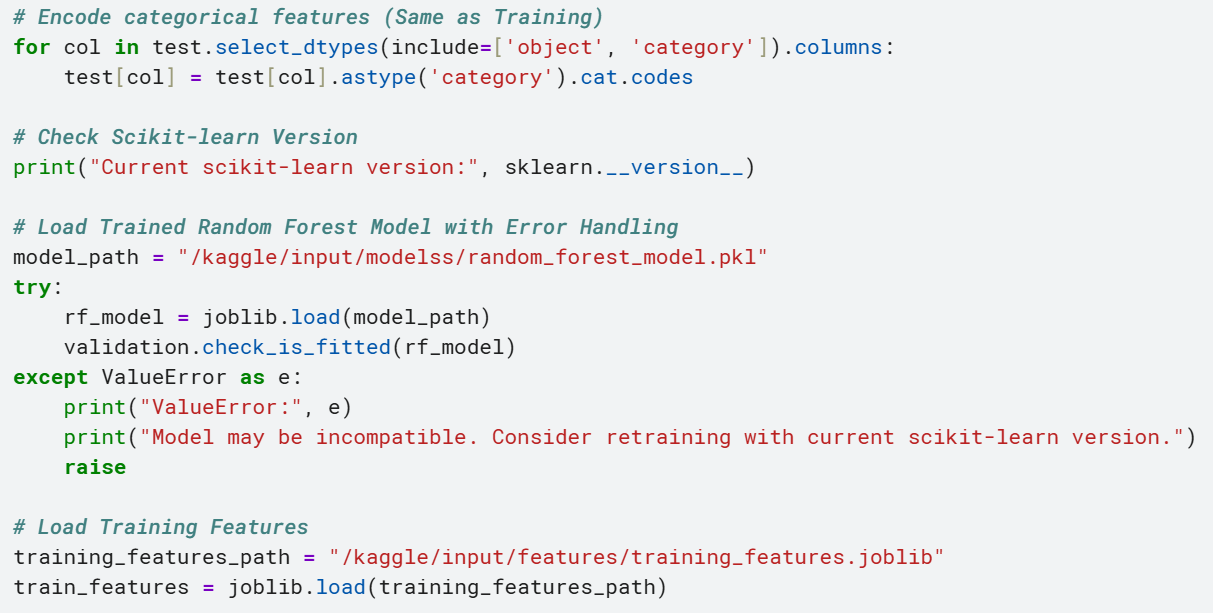
**Key Points**

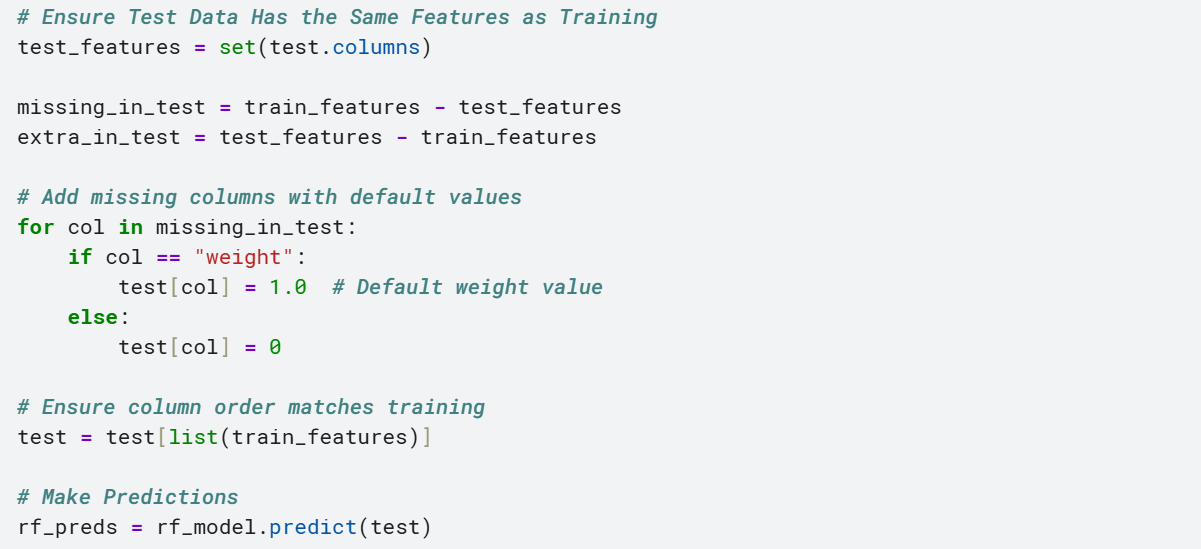
* **Reproducibility**: The same feature engineering and encoding steps are applied to test data as were used during training.
* **Error Handling**: Ensures that model compatibility issues are caught and reported.
* **Modularity**: The feature engineering function can be reused for training or further model development.

# **Model Inference and Submission Code – Detailed Explanation**

This script is used to **generate predictions** on test data using a **trained Random Forest model** for predicting post-HCT survival outcomes. Below is a breakdown of each step in the code:









**1**. **Library Imports**



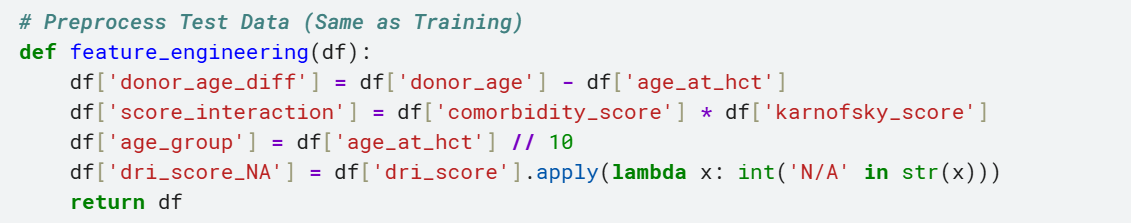
1. **NumPy** and **Pandas**: Handle numerical operations and tabular data.
2. **joblib**: Load pre-trained machine learning models and serialized objects.
3. **sklearn**: Used for the model and checking if it is fitted properly.

**2**. **Load the Test Dataset**



Loads the test dataset from Kaggle input directory.

**3**. **Feature Engineering**



This function creates **new features** to improve model accuracy:

1. **donor\_age\_diff**: Age difference between donor and patient.
2. **score\_interaction:** Interaction term between comorbidity score and Karnofsky score.
3. **age\_group:** Categorizes age into decades.
4. **dri\_score\_NA:** Binary flag to indicate if dri\_score is missing or set to 'N/A'.

This engineered data ensures the model receives the same type of input it was trained on.  
**4.** **Categorical Encoding**



Converts categorical (text-based) features into numerical codes so the model can interpret them.

**5.** **Check scikit-learn Version**



* Ensures compatibility between the training environment and current environment.

**6.** **Load the Trained Model**



1. Loads the previously trained **Random Forest Regressor** model using joblib.
2. validation.check\_is\_fitted(rf\_model) ensures the model is ready for inference.
3. If there's an issue, a ValueError is raised and explained.

7. **Load Training Features**



Loads the list of features used during training. This ensures **feature alignment** with the test data.

* 1. **Align Test Data with Training Features**



1. Ensures all features present during training are also in the test set.
2. **Missing features** are added with default values:

* "weight" is filled with 1.0.
* All others are filled with 0.

1. Columns are reordered to match the model’s training format.
   1. **Make Predictions**



Uses the model to make survival predictions on the preprocessed test data.

* 1. **Save Submission File**



1. **Creates a CSV file in the correct submission format.**
2. **Columns: ID (from test set), prediction (model output).**

**11.** **Final Confirmation**

****

Prints a message confirming the process has completed.

# **Explanation Of Used Models:**

## **cat\_model.pkl**

A model trained using CatBoost, a gradient boosting algorithm designed to handle categorical features efficiently. Works well for tabular data.

## **lgb\_model.pkl**

A model trained using LightGBM, another gradient boosting framework optimized for speed and performance. It’s highly effective on structured datasets.

## **xgb\_model.pkl**

A model trained using XGBoost, a powerful gradient boosting algorithm known for accuracy and scalability.

## **meta\_model.pkl**

This is likely the meta-learner in a stacking ensemble, which combines predictions from cat\_model, lgb\_model, and xgb\_model to make a final prediction.

## **ensemble\_models.pkl**

This file probably contains the entire stacked ensemble structure—base models (CatBoost, LightGBM, XGBoost) and the meta-model. It might also include any preprocessing pipelines.

**Which Model is Best for MNIST?**

**MNIST** is an image classification dataset. Deep learning models (like CNNs) are usually better suited for such tasks.

For the **MNIST** dataset (flattened), **ensemble\_models.pkl** is likely the best because:

1. It combines the strengths of multiple models.
2. Reduces overfitting.
3. Usually yields better generalization on unseen data.

**Final Scores Result:**