[1. INTRODUCTION 2](#_Toc1538832224)

[1.1 Background 3](#_Toc569883111)

[1.2 Problem Statement 3](#_Toc1847154652)

[2.DATA DESCRIPTIVE ANALYSIS 4](#_Toc170649694)

[2.1 Target descriptive analysis 4](#_Toc2014602087)

[2.2 Medical claim descriptive analysis 8](#_Toc722235016)

[2.3 Pharmacy claim descriptive analysis 9](#_Toc1740528154)

[2.4 Training Data descriptive analysis 11](#_Toc2064816046)

[3. EVALUATION METRICS (Helen) 14](#_Toc182746831)

[3.1 Confusion matrix 15](#_Toc1561055716)

[3.1.1 Precision 15](#_Toc125950246)

[3.1.2 Recall 15](#_Toc599171061)

[3.1.3 F1 Score 16](#_Toc234646502)

[3.2 Micro-averaged Metrics 16](#_Toc488857219)

[3.3 Macro-averaged Metrics 17](#_Toc2064045451)

[3.4 Weighted-averaged Metrics 17](#_Toc1811076731)

[4 Machine Learning Modeling and Results 17](#_Toc1951193626)

[4.1 Machine Learning Models 18](#_Toc7290056)

[4.1.1 Long Short-Term Memory (LSTM) Model (Chandrika, Helen) 18](#_Toc92436300)

[4.1.2 Random Forest Algorithms 18](#_Toc362168358)

[4.1.3 Extreme Random Tree Algorithms 19](#_Toc661664151)

[4.1.4 Boosting Tree Algorithms 19](#_Toc1435399253)

[4.1.5 Neural Network Algorithms 21](#_Toc764585966)

[4.2 Modeling Results 21](#_Toc1009770314)

[4.2.1 LSTM 21](#_Toc702228538)

[4.2.2 Random Forest Algorithms 22](#_Toc287584563)

[4.2.3 Extreme Random Tree Algorithms 23](#_Toc1653343907)

[4.2.4 Boosting Tree Algorithms 24](#_Toc1658665195)

[4.2.5 Neural Network Algorithms 27](#_Toc1448227144)

[4.3 AUC Score 28](#_Toc1693487059)

[4.4 Feature Importance 29](#_Toc327450098)

[3. EVALUATION METRICS (Helen) 1](#_Toc578035581)

[3.1 Confusion matrix 1](#_Toc2074494257)

[3.2 Precision 1](#_Toc1659802751)

[3.3 Recall 1](#_Toc1093570332)

[3.4 F1 1](#_Toc1648185618)

[4 Machine Learning Modeling 1](#_Toc44425384)

[4.1 Long Short-Term Memory (LSTM) Model (Chandrika, Helen) 1](#_Toc428120811)

[4.11.1 LSTM Introduction (Chandrika) 1](#_Toc1234995589)

[4.1.2 Model parameters (Helen) 1](#_Toc1584005779)

[4.1.3 Model results (Helen) 1](#_Toc201923113)

[4.2 LightGBM Model 1](#_Toc698745443)

[4.3 XGBoost 1](#_Toc1623307834)

**2023 Humana Mays Healthcare Analytics Case Competition Problem Prompt**

# 1. INTRODUCTION

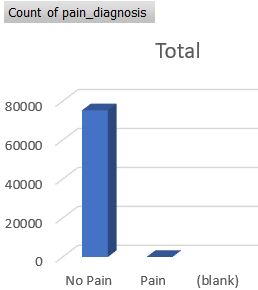
## 1.1 Background

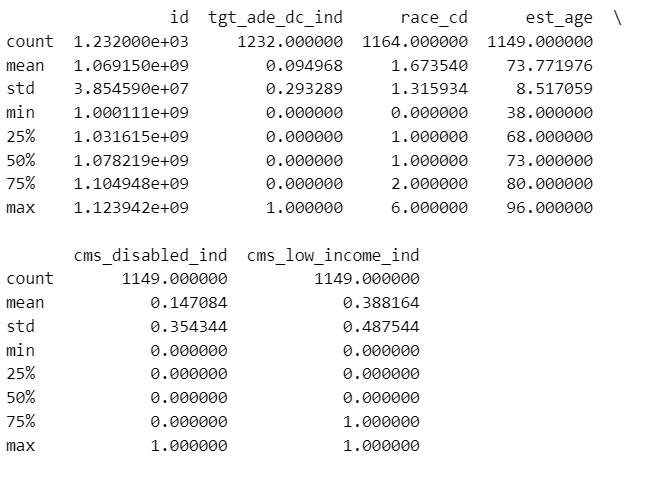
Oncology has witnessed noteworthy advancements in treatments, yet cancer remains a prominent cause of mortality in the US, claiming around 600,000 lives annually. A key medication, Osimertinib, an oral tyrosine kinase inhibitor, is employed for early-stage non-small cell lung cancer patients with a specific EGFR mutation. While its efficacy is commendable, doubling survival rates and reducing recurrence by 80% when adhered to, its associated side effects, including nausea, fatigue, and high blood glucose, present challenges. Roughly 25% of Humana members discontinue Osimertinib within six months due to these side effects. Leveraging data analytics can aid in identifying at-risk members, promoting medication adherence, and enhancing the quality of life for oncology patients.

## 1.2 Problem Statement

The goal is to utilize data and analytics to identify members at risk of discontinuing Osimertinib due to adverse drug events (ADEs), promote medication adherence, and consequently improve patient outcomes.2.DATA DESCRIPTIVE ANALYSIS

## 2.1 Target descriptive analysis





A bar graph with blue lines

Description automatically generated

From The above visualization we can observe the time-series analysis of the Count of therapies ending over time.

A graph of a number of people

Description automatically generated

The data describes a distribution of individuals across different racial or ethnic categories. Each category has a unique code (race\_cd) that corresponds to a specific racial or ethnic description (race\_cd\_desc). 0 - unknown:

This code is likely reserved for instances where an individual's racial or ethnic background cannot be determined or is not specified.

1 - white: Represents individuals who identify as White.

2 - black: Represents individuals who identify as Black or African American.

3 - other: This category might include individuals who do not fit into any of the specific categories provided, or it could be a collection of several smaller racial or ethnic groups not separately listed.

4 - Asian: Represents individuals who identify as Asian. This might encompass a wide range of ethnicities from the Asian continent, such as Chinese, Japanese, Indian, etc.

5 - Hispanic: Represents individuals who identify as Hispanic or Latino. This might include people from a Spanish-speaking background or from Latin American countries.

6 - n Amer native: Represents individuals who identify as Native American or Indigenous to North America.

A graph with blue rectangular bars

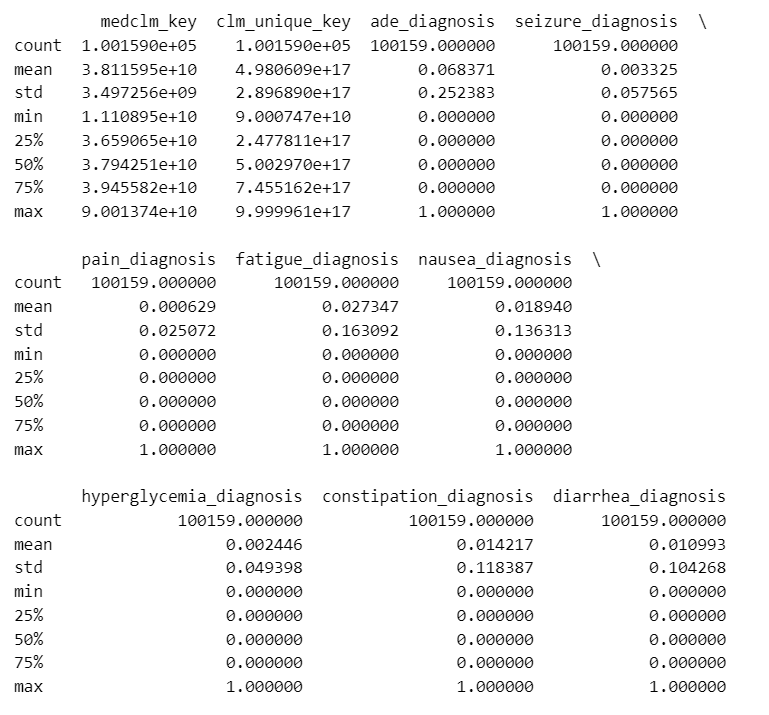
Description automatically generated

The bar chart depicts the distribution of count of gender distribution with respect to each category which is male and female, and it clearly shows the high distribution is in females rather than males.

A graph of age distribution

Description automatically generatedThe bar chart spans the age range from 40 to 90 years. In such charts, each bar represents an Estimated age, and the height of each bar corresponds to the count of patients within that age. A higher bar in the 70-75 age range indicates that a significant number of patients fall within this age group compared to other age groups on the chart.

## 2.2 Medical claim descriptive analysis



We have data on 100,159 medical claims, each with unique IDs.

Most claims don’t have the mentioned diagnoses, but when they do:

The most common is the “ade\_diagnosis” (6.84% of claims), followed by “fatigue\_diagnosis” (2.73%).

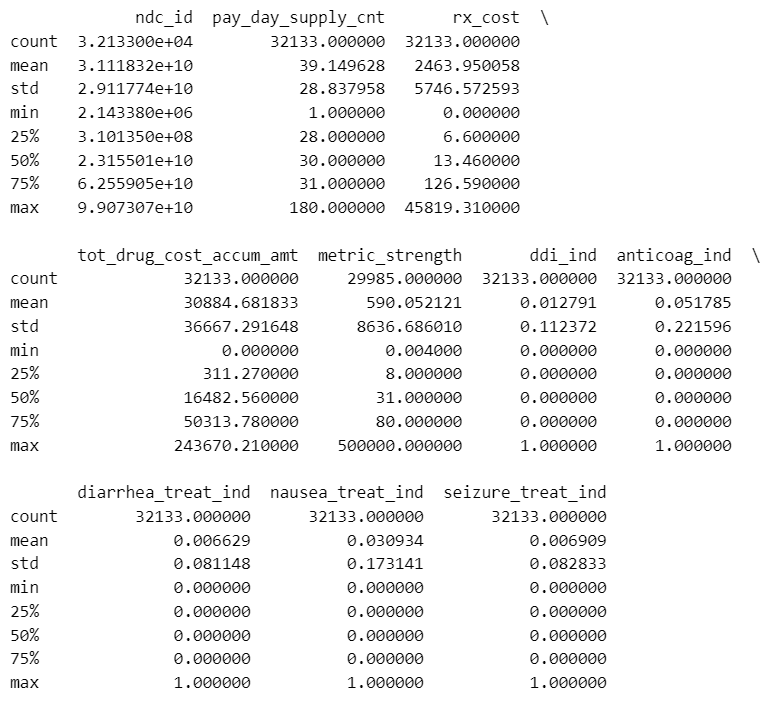
Diagnoses like seizures, pain, and hyperglycemia is rare in this dataset.

This dataset essentially gives us an idea about the distribution of various diagnoses across 100,159 medical claims.

Table and figures

## 2.3 Pharmacy claim descriptive analysis

Table and figures



figures

Mean: The average prescription cost is approximately $2463.95.

Std: There's a high variability in prescription costs, with a standard deviation of about $5746.57.

Min-Max: The costs range from $0 to $45819.31. Given the huge difference between the mean and median (50% quartile) which is only $13.46, it indicates that there are some high-cost outliers affecting the average.

The dataset has 32,133 observations for most variables.

Prescription costs (rx\_cost) and metric strength (metric\_strength) have high variability.

Most binary indicators (like ddi\_ind, anticoag\_ind, etc.) have a low percentage of "yes" (or 1) responses, implying that many observations in these categories are "no" (or 0).

Target Data: Contains therapy start and end dates, target identifiers, and protected attributes like sex, race, and age.

target\_train with 1232 records and target\_holdout with 420 records.

Medical Claims Data: Contains information about all medical claims for a member for 90 days (about 3 months) before starting Osimertinib therapy and until the end of therapy.

medclms\_train with 100,159 records and medclms\_holdout with 23,232 records.

Pharmacy Claims Data: Provides details about all pharmacy claims for an individual during the 90 days (about 3 months) before starting Osimertinib therapy and until the end of therapy.

rxclms\_train with 32,133 records and rxclms\_holdout with 6,670 records.

Data Dictionary: Describes each data column available in the claims datasets.

Race Code Descriptions: Gives definitions for the race codes in the target files.

Outcome Metrics:

The predictive model's outcomes should be evaluated using the target column tgt\_ade\_dc\_ind

## 2.4 Training Data descriptive analysis

2.4.1 Data connection

**Data Connection**

* We have 3 training data sources, **target\_train.csv**, **medclms\_train.csv**, **rxclms\_train.csv**. We have the unique ids **therapy\_id** in all of them. Using these and other columns to join the three.

We first start with the target train data, where we find users by the Id column, for grouping all the data we use the column **therapy\_id**, also checking if one id only one has **therapy\_id** using group by and count operations.

* In the **medclms\_train** dataset, we use the columns. **therapy\_id , visit\_date , clm\_unique\_key**

Duplicates are dropped based on a subset of columns (**therapy\_id**, **visit\_date**, **clm\_unique\_key**).

* The third dataset is for **rxclms\_train** dataset: which contains the prescription data, here we focus on columns **therapy\_id** and **service\_date** and we aggregate these by sum of their multiple rows, to get a unique entry for the same **therapy\_id**.

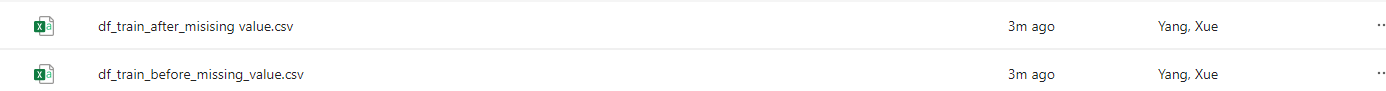
**We then go ahead and merge the datasets, below are the steps for connecting the datasets.**

1. Merging medclms\_train with aggregated rxclms\_train\_ on columns therapy\_id and visit\_date.
   * This is done using pd.merge in python
   * Result: **df\_medclms\_rx\_train** DataFrame.
2. Cleaning **df\_medclms\_rx\_train**:
   * Dropping specified columns such as 'medclm\_key' and ‘clm\_unique\_key'
   * Filling NaN values with 0.0.

1. We then sort the dataset **df\_medclms\_rx\_train** by therapy\_id and visit\_date.

2.4.3 Descriptive analysis

Using these two file (vish)



1. How many unique therapy ID, the race, sex, age, and disable, low\_income (figures)

unique therapy ID (around 500) table and figures

1. A table
2. Other disg# and treat: just use all data (2000 records) table and figures

A screenshot of a computer

Description automatically generated

A graph with blue bars

Description automatically generated

The treatment location for the claims is divided into six categories: Other, Outpatient, ER, Inpatient, Telephone, and Observation. The bar chart shows that the "Other" category has the most occurrences, while Telephone and Observation have the fewest.

A graph of blue and orange lines

Description automatically generated

A screenshot of a computer screen

Description automatically generated

The correlation analysis shows the relationship between the target variable and the other variables. Hence it shows that there is a strong relationship between the variables and the target variable.

Table -Vishwesh

Data Connection:

Target Train Dataset Handling:

Grouped the target\_train dataset by the "id" column.

Counted the "therapy\_id" occurrences for each individual ID.

Filtered the dataset to only include IDs with more than one associated "therapy\_id".

Medical Claims Data Cleaning:

Duplicates were removed from the medclms\_train dataset.

Further, deduplication was conducted in the medclms\_train DataFrame based on the columns: 'therapy\_id', 'visit\_date', and 'clm\_unique\_key'.

Pharmacy Claims Data Exploration and Grouping:

Delved into various columns of rxclms\_train, specifically examining columns like 'gpi\_drug\_group\_desc' and 'gpi\_drug\_class\_desc' and

A subset of rxclms\_train was grouped by both 'therapy\_id' and 'service\_date'.

During this grouping, specific columns were aggregated using summation. This effectively combined indicators for each therapy occurring on a particular service date.

Data Frame Handling with Unique Therapy Identifiers:

Computed the count of unique 'therapy\_id' present within the DataFrame df1.

A new DataFrame, named target\_train\_rv, was initialized to accommodate reordered rows based on 'therapy\_id'. Iterated over the unique 'therapy\_id' values in df1, and for each unique 'therapy\_id', the corresponding rows from target\_train were extracted and subsequently appended to target\_train\_rv. This step reordered the rows in target\_train\_rv based on the sequence of 'therapy\_id' in df1.

Overall, the procedures outlined above represent systematic data preprocessing steps to ensure clean, structured, and ordered data, setting the stage for further analytics or modeling.

# **3. EVALUATION METRICS** (Helen)

This project is a binary classification problem. Therefore, precision, recall, and F1 scores were considered to evaluate the performance of the model.

## 3.1 Confusion matrix

A confusion matrix is a table that is normally used as a tool for computing the performance of a classification model. The key function of this table is to present a comparison between “Predicted Labels” from the model and “Actual Labels” from the ground truth. Figure 1 shows the example of the classification outcome of data instances from two groups, “Positive” and “Negative.” The classification can be divided into four status categories.

* True Positive (TP): The number of instances that a model predicts correctly such that the “Actual Labels” is positive and “Predicted Labels” is positive as well.
* True Negative (TN): The number of instances that a model predicts correctly such that the “Actual Labels” is negative and “Predicted Labels” is negative as well.
* False Positive (FP): The number of instances that a model predicts incorrectly such that the “Actual Labels” is negative but “Predicted Labels” is positive.
* False Negative (FN): The number of instances that a model predicts incorrectly such that the “Actual Labels” is positive but the “Predicted Labels” is negative.

Table

Description automatically generated

Figure 1. Example Confusion Matrix

### 3.1.1 Precision

Precision is defined as the ratio of correctly predicted positive observations to the total predicted positive observations. The precision score equals 1 for a classifier with zero false positives.

(1)

### 3.1.2 Recall

The recall is the ratio of correctly predicted positive observations to all observations as shown in. Under an ideal case, both precision and recall are equal to 1 with zero counts of false positives and false negatives.

(2)

### 3.1.3 F1 Score

F1, a weighted average of precision and recall values, takes both precision and recall into account in. The F1 score is high only when both precision and recall are high, and the F1 score becomes 1 only when both precision and recall equal to 1.

(3)

Add something about AUC （chandrika）

AUC represents the "Area Under the Curve" of an ROC plot. This plot shows a model's ability to distinguish between two classes. The ROC curve is a plot of the True Positive Rate (sensitivity) against the False Positive Rate (1-specificity) for various threshold values. The AUC provides a scalar value of how well a model can distinguish between the two classes. In essence, it quantifies the overall ability of the model to discriminate between positive and negative cases, regardless of the specific threshold chosen.

AUC values range from 0 to 1:

AUC = 0.5: The model's predictions are no better than random guessing. The ROC curve would be a diagonal line.

AUC > 0.5: The model has some discriminative capability. The greater the AUC, the better the model is at distinguishing between positives and negatives.

AUC = 1: The model has perfect discriminative capability. It can separate the two classes perfectly.

In practice, an AUC closer to 1 indicates a better-performing model. AUC values range from 0 to 1: 0.5 means the model is as good as random guessing, while 1 means perfect classification. A higher AUC indicates a better model

## 3.2 Micro-averaged Metrics

With the confusion matrices, the classification models’ predictive performance can be measured in terms of accuracy, precision, recall, and F1-score. Accuracy is the ratio of correct results to total cases. Precision is the ratio of correct results over all positive results. The recall is the part of all true results returned. F1-score combines the precision and recall scores of a model. As most of such measurements are designed for binary classification problems, they should be transformed for multi-class classification using the weighted average conditions (micro-averaged and macro-averaged) in the scikit-learn Python library package. While micro-averaging represents the weighted average based on the frequency of samples from each class, macro-averaging represents the unweighted mean of precision, recall, and accuracy metrics (Yang, 1999). In general, micro-averaged F-measure is heavily influenced by larger classes, while macro-averaged F-measure gives equal weight to each class regardless of size. Since smaller classes tend to be harder to classify, the macro-averaged F-measure is often supposed to emphasize the smaller classes, while the micro-averaged F-measure emphasizes the large classes.

Since the ADE data are extremely unbalanced, the evaluation metrics of the ML models are based on micro-averaged, macro-averaged, and weighted-average accuracy, precision, and recall. For HRGC crashes, most majority HRGCs have zero crashes and are within the large class, while those with crashes are within the smaller classes. In this case study, there are 2 classes (0 and 1). The micro-averaged evaluation matrix is calculated by considering the total TP, total FP, and total FN of the model globally, but does not consider each individual crash class. The total TP, total FP, and total FN are calculated as:

Eq. (1)

The micro-averaged measures are then computed as shown in Eq. (21).

Eq. (21)

## 3.3 Macro-averaged Metrics

The macro-averaged evaluation matrix considers each class individually and then takes the unweighted mean of the measures. First, the precision, recall, and F1-score for each crash class should be calculated:

Eq. (22)

Subsequently, the macro precision, macro recall, and macro F1-score are calculated.

Eq. (23)

## 3.4 Weighted-averaged Metrics

Because micro- and macro-average matrices focus differently on larger or smaller classes, it is desirable to weighted-average the two evaluation matrices. Assuming is the weight for each crash class calculated by the number of samples in that crash class divided by the total samples of all classes:

Eq. (24)

# 4 Machine Learning Modeling and Results

## 4.1 Machine Learning Models

### 4.1.1 Long Short-Term Memory (LSTM) Model (Chandrika, Helen)

**Longitudinal Data Overview:**

Longitudinal data, otherwise known as panel or time series data, holds a unique value in the world of data analysis due to its ability to track a specific sample across multiple time intervals. This contrasts with cross-sectional data which captures data at a singular moment in time.

**Key Attributes of Longitudinal Data:**

Multiple observations of the same sample over a timeline. Sequential collection enables the examination of causality and directional relationships among variables. The capability to analyze both intra-unit (within the same individual or unit) and inter-unit variations (across different units or individuals). Vital for observing developmental shifts, intervention impacts, and the progression of different phenomena. Analyzing longitudinal data can be complex due to potential autocorrelation and other related challenges.

**Understanding of LSTM Networks:** LSTM networks can be viewed as an advancement of recurrent neural networks (RNNs). The inception of LSTM networks was primarily to tackle the issues posed by RNNs. RNNs face challenges in retaining information over extended periods. This becomes problematic when predicting current outputs requires referencing older stored data. RNNs lack the nuanced control needed to decide which contextual elements to retain and which past elements to discard. RNNs also suffer from "exploding and vanishing gradient" issues during network training via backtracking.

4.1.2 Model parameters (Helen)

A screenshot of a computer program

Description automatically generated

### 4.1.2 Random Forest Algorithms

The random forest is a supervised ML algorithm that grows and combines multiple decision trees to create a “forest.” The set of decision trees is randomly selected as a subset of the training set, and then the votes from different decision trees are collected to decide the final prediction. Since decision trees in a random forest are relatively uncorrelated models, the random forest always outperforms the individual constituent model with diversity and stability (Géron, 2019).

Node splitting, which divides a node into multiple sub-nodes to create relatively pure nodes, is a key concept in the random forest algorithm. Node impurity represents how well the trees split the data. The ways of splitting a node can be broadly divided into two categories based on the type of target variable (continuous and categorical). For categorical variables, the Shannon entropy and the Gini index are two common impurity functions used for calculating the purity of a node. The lower the value of entropy or the Gini impurity, the higher the purity (or homogeneity) of the node. Random forests using the entropy and Gini impurity as the splitting criteria for classification problems are denoted as “RandomForestEntr” and “RandomForestGini,” respectively. When the target variable is continuous, i.e., regression problems, reduction in variance is a widely used method for node splitting. Random forests using the mean squared error (MSE) as the splitting criterion for regression problems are denoted as “RandomForestMSE.”

### 4.1.3 Extreme Random Tree Algorithms

Extremely randomized trees (Extra trees in short) are an ensemble supervised ML method. This method is similar to random forests in that they both construct multiple decision trees to use for the task at hand, whether classification or regression. The Extra trees algorithm works by creating a large number of unpruned decision trees from the training dataset. Predictions are made by averaging the prediction of the decision trees in the case of regression or using majority voting in the case of classification. Unlike bagging and random forest that develop each decision tree from a bootstrap sample of the training dataset, the Extra trees algorithm fits each decision tree on the whole training dataset. The Extra trees algorithm randomly samples the features at each split point of a decision tree (Géron, 2019).

The most important and unique characteristic of Extra trees is the random selection of a splitting value for a feature. Instead of calculating a locally optimal value using Gini or entropy to split the data, the algorithm randomly selects a split value. This makes the trees diversified and uncorrelated. As a result, the main advantage of Extra trees is the reduction in bias. This is because of sampling from the entire dataset during the construction of the trees. Different subsets of the data may introduce different biases in the results obtained, hence Extra trees prevent this by sampling the entire dataset. Another advantage of Extra trees is that they reduce variance. This is a result of the randomized splitting of nodes within the decision trees, hence the algorithm is not heavily influenced by certain features or patterns in the dataset.

Similarly, the Extra trees algorithm using the entropy and Gini impurity as the splitting criteria for classification problems are denoted as “ExtraTreesEntr” and “ExtraTreesGini” respectively, while the extra trees algorithm using MSE as the splitting criterion for regression problems are denoted as “ExtraTreesMSE.”

### 4.1.4 Boosting Tree Algorithms

The main drawback of decision trees is overfitting the training data. An alternative to decision trees is random forests by combining multiple decision trees via a technique called bagging (bootstrap aggregating). Although random forests have much better performance than decision trees, they cannot deal with mistakes (if any) created by their individual decision trees because of the parallel learning process. As a result, boosted-tree models are introduced to outperform decision trees and random forests by avoiding the above drawbacks.

In bagging, the algorithm makes samples with replacement, and the final prediction of RF is to aggregate the predictions of bootstrapped decision tree samples. Therefore, a random forest is a bagging ensemble method with independent trees. In contrast, boosting tree is an ensemble method in which boosted trees are created with a group of decision trees. In boosting, new trees are formed by considering the errors of trees in previous rounds. This type of learning is called sequential learning, whereas parallel computing is not ideal to perform. Several boosting algorithms have been developed in machine learning, such as gradient boosting (GBoost), extreme gradient boosting (XGBoost), light gradient boosting machine (LightGBM), and categorical boosting (CatBoost).

*GBoost*: GBoost is a decision tree ensemble learning algorithm similar to a random forest. GBoost iteratively trains an ensemble of shallow decision trees, with each iteration using the error residuals of the previous model to fit the next model. The final prediction is a weighted sum of all tree predictions. Random forest “bagging” minimizes the variance and overfitting, while GBoost “boosting” minimizes the bias and underfitting.

*XGBoost*: XGBoost, which stands for extreme gradient boosting, is a scalable implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing ML model performance and computational speed. With XGBoost, trees are built in parallel, instead of sequentially like GBoost. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

*LightGBM*: Light GBM (gradient-boosting machine) is a fast, distributed gradient-boosting framework that uses a tree-based learning algorithm (Ke et al., 2017). Light GBM excludes a significant proportion of data instances with small gradients, and only uses the rest to estimate the information gain by using gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB). LightGBM could speed up the training process of conventional GBoost by more than 20 times but with comparable accuracy. There are three types of LightGBM methods, “LightGBM,” “LightGBMXT,” and “LightGBMLarge,” using different hyperparameters. “LightGBM” runs with default hyperparameters; “LightGBMXT” enables extra trees hyperparameters; while “LightGBMLarge” runs with a custom configuration, which enables larger models but trains more slowly.

### 4.1.5 Neural Network Algorithms

Neural networks, also known as artificial neural networks (ANNs), are a subset of ML and are at the heart of deep learning algorithms. ANNs are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data are passed along to the next layer of the network.

The neural network algorithms executed with PyTorch and FastAI are denoted as “NeuralNetTorch” and “NeuralNetFastAI.”

## 4.2 Modeling Results

Chandrika

### 4.2.1 LSTM

A graph with lines and numbers

Description automatically generated

Figure 2 Training loss of the LSTM model for training data

A graph of different colored lines

Description automatically generated

Figure 3 Training loss of the LSTM model for validation data

Table 1 Confusion matrix

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 80 | 0 |
| 1 | 26 | 0 |

Table 2 Evaluation matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | AUC |
| 0 | 0.75 | 1.00 | 0.86 | 0.5 |
| 1 | 0.00 | 0.00 | 0.00 |
| macro avg | 0.38 | 0.50 | 0.43 |
| weighted avg | 0.57 | 9.75 | 0/65 |

### 4.2.2 Random Forest Algorithms

Table 3 Confusion matrix (RandomForestEntr)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2012 | 65 |
| 1 | 317 | 268 |

Table 4 Classification report (RandomForestEntr)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.86 | 0.97 | 0.91 | 2077 |
| 1 | 0.8 | 0.46 | 0.58 | 585 |
| accuracy |  |  | 0.86 | 2662 |
| macro-avg | 0.83 | 0.71 | 0.75 | 2662 |
| weighted-avg | 0.85 | 0.86 | 0.84 | 2662 |

Table 5 Confusion matrix (RandomForestGini)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2010 | 67 |
| 1 | 315 | 270 |

Table 6 Classification report (RandomForestGini)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.86 | 0.97 | 0.91 | 2077 |
| 1 | 0.8 | 0.46 | 0.59 | 585 |
| accuracy |  |  | 0.86 | 2662 |
| macro-avg | 0.83 | 0.71 | 0.75 | 2662 |
| weighted-avg | 0.85 | 0.86 | 0.84 | 2662 |

|  |  |
| --- | --- |
| A graph with text overlay  Description automatically generated | A graph with text overlay  Description automatically generated |

Figure 4 Feature Importance for Predicting ADE (Random Forest Algorithms)

### 4.2.3 Extreme Random Tree Algorithms

Table 7 Confusion matrix (ExtraTreesGini)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2011 | 66 |
| 1 | 317 | 268 |

Table 8 Classification report (RandomForestGini)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.86 | 0.97 | 0.91 | 2077 |
| 1 | 0.8 | 0.46 | 0.58 | 585 |
| accuracy |  |  | 0.86 | 2662 |
| macro-avg | 0.83 | 0.71 | 0.75 | 2662 |
| weighted-avg | 0.85 | 0.86 | 0.84 | 2662 |

Table 9 Confusion matrix (ExtraTreesEntr)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2009 | 68 |
| 1 | 315 | 270 |

Table 10 Classification report (ExtraTreesEntr)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.86 | 0.97 | 0.91 | 2077 |
| 1 | 0.8 | 0.46 | 0.59 | 585 |
| accuracy |  |  | 0.86 | 2662 |
| macro-avg | 0.83 | 0.71 | 0.75 | 2662 |
| weighted-avg | 0.85 | 0.86 | 0.84 | 2662 |

|  |  |
| --- | --- |
|  |  |

Figure 5 Feature Importance for Predicting ADE (Extreme Random Tree Algorithms)

### 4.2.4 Boosting Tree Algorithms

Table 11 Confusion matrix (LightGBM)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2002 | 75 |
| 1 | 322 | 263 |

Table 12 Classification report (LightGBM)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.86 | 0.96 | 0.91 | 2077 |
| 1 | 0.78 | 0.45 | 0.57 | 585 |
| accuracy |  |  | 0.85 | 2662 |
| macro-avg | 0.82 | 0.71 | 0.74 | 2662 |
| weighted-avg | 0.84 | 0.85 | 0.84 | 2662 |

Table 13 Confusion matrix (LightGBMXT)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2014 | 63 |
| 1 | 334 | 251 |

Table 14 Classification report (LightGBMXT)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.86 | 0.97 | 0.91 | 2077 |
| 1 | 0.8 | 0.43 | 0.56 | 585 |
| accuracy |  |  | 0.85 | 2662 |
| macro-avg | 0.83 | 0.7 | 0.73 | 2662 |
| weighted-avg | 0.84 | 0.85 | 0.83 | 2662 |

Table 15 Confusion matrix (LightGBMLarge)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2042 | 35 |
| 1 | 377 | 208 |

Table 16 Classification report (LightGBMLarge)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.84 | 0.98 | 0.91 | 2077 |
| 1 | 0.86 | 0.36 | 0.5 | 585 |
| accuracy |  |  | 0.85 | 2662 |
| macro-avg | 0.85 | 0.67 | 0.71 | 2662 |
| weighted-avg | 0.85 | 0.85 | 0.82 | 2662 |

Table 17 Confusion matrix (XGBoost)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2026 | 51 |
| 1 | 441 | 144 |

Table 18 Classification report (XGBoost)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.85 | 0.98 | 0.91 | 2077 |
| 1 | 0.82 | 0.39 | 0.53 | 585 |
| accuracy |  |  | 0.85 | 2662 |
| macro-avg | 0.83 | 0.68 | 0.72 | 2662 |
| weighted-avg | 0.84 | 0.85 | 0.82 | 2662 |

Table 19 Confusion matrix (CatBoost)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2044 | 33 |
| 1 | 441 | 144 |

Table 20 Classification report (CatBoost)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.82 | 0.98 | 0.9 | 2077 |
| 1 | 0.81 | 0.25 | 0.38 | 585 |
| accuracy |  |  | 0.82 | 2662 |
| macro-avg | 0.82 | 0.62 | 0.64 | 2662 |
| weighted-avg | 0.82 | 0.82 | 0.78 | 2662 |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Figure 6 Feature Importance for Predicting ADE (Boosting Tree Algorithms)

### 4.2.5 Neural Network Algorithms

Table 21 Confusion matrix (NeuralNetFastAI)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 2044 | 33 |
| 1 | 441 | 144 |

Table 22 Classification report (NeuralNetFastAI)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.83 | 0.97 | 0.89 | 2077 |
| 1 | 0.72 | 0.31 | 0.43 | 585 |
| accuracy |  |  | 0.82 | 2662 |
| macro-avg | 0.78 | 0.64 | 0.66 | 2662 |
| weighted-avg | 0.81 | 0.82 | 0.79 | 2662 |

A graph with text overlay

Description automatically generated

Figure 7 Feature Importance for Predicting ADE (Neural Network Algorithms)

## 4.3 AUC Score

Table 23 The AUC values for all the ML models

|  |  |  |
| --- | --- | --- |
| model | score\_train | score\_val |
| ExtraTreesGini | 0.8801 | 0.7986 |
| ExtraTreesEntr | 0.8797 | 0.7972 |
| RandomForestEntr | 0.8792 | 0.7943 |
| RandomForestGini | 0.8787 | 0.7916 |
| LightGBM | 0.8675 | 0.7876 |
| LightGBMXT | 0.8649 | 0.7855 |
| LightGBMLarge | 0.8554 | 0.7751 |
| XGBoost | 0.8538 | 0.7712 |
| CatBoost | 0.8018 | 0.7394 |
| NeuralNetFastAI | 0.7898 | 0.7393 |

ExtraTreesGini" refers to an Extra Trees model, which is an ensemble learning method. The "Gini" likely indicates that the model uses Gini impurity as a criterion for making splits in the individual trees.

This score of 0.8801 indicates the performance of the model on the training dataset. A score closer to 1 suggests the model has learned well from the training data.

The score of 0.7986 represents the performance of the model on a validation dataset. This is a measure of how well the model is likely to perform on unseen data.

"ExtraTreesEntr" suggests an Extra Trees model, which is an ensemble learning method. The "Entr" likely signifies that the model uses Entropy as a criterion for making splits in the individual trees. The score of 0.8797 indicates the performance of the model on the training dataset. A score closer to 1 suggests that the model fits the training data well. The score of 0.7972 represents the performance of the model on a validation dataset. This provides an estimate of how the model might perform on new, unseen data.

## 4.4 Feature Importance

Chandrika

Table 24 Feature importance summary for all ML models

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Variable | Extra  TreesGini | Extra  TreesEntr | Random  ForestEntr | Random  ForestGini | Light  GBM | Light  GBMXT | Light  GBMLarge | XG  Boost | Cat  Boost | Neural  NetFastAI |
| race\_cd | 1 | 1 | 1 | 1 | 1 | 1 | 0.87 | 1 | 1 | 1 |
| est\_age | 0.90 | 0.88 | 0.86 | 0.86 | 0.91 | 0.98 | 1.00 | 0.88 | 0.81 | 0.55 |
| primary\_diag\_cd\_new | 0.75 | 0.74 | 0.77 | 0.77 | 0.88 | 0.74 | 0.72 | 0.77 | 0.56 | 0.98 |
| cms\_low\_income\_ind | 0.72 | 0.71 | 0.66 | 0.65 | 0.70 | 0.79 | 0.74 | 0.53 | 0.20 | 0.71 |
| sex\_cd | 0.68 | 0.69 | 0.66 | 0.66 | 0.67 | 0.64 | 0.77 | 0.57 | 0.67 | 0.45 |
| cms\_disabled\_ind | 0.39 | 0.39 | 0.35 | 0.36 | 0.41 | 0.47 | 0.50 | 0.44 | 0.16 | 0.66 |
| ade\_diagnosis | 0.09 | 0.09 | 0.10 | 0.10 | 0.16 | 0.15 | 0.13 | 0.08 | 0.08 | 0.15 |
| nausea\_treat\_ind | 0.05 | 0.05 | 0.05 | 0.05 | 0.04 | 0.04 | 0.04 | 0.05 | 0.04 | 0.05 |
| fatigue\_diagnosis | 0.03 | 0.03 | 0.02 | 0.02 | 0.01 | 0.02 | 0.02 | 0.03 | 0.01 | 0.09 |
| nausea\_diagnosis | 0.02 | 0.02 | 0.02 | 0.02 | 0.00 | 0.01 | 0.01 | 0.02 | 0.00 | 0.07 |
| diarrhea\_diagnosis | 0.01 | 0.02 | 0.02 | 0.02 | 0.00 | 0.01 | 0.04 | 0.01 | 0.01 | 0.05 |
| constipation\_diagnosis | 0.01 | 0.01 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.01 | 0.00 | 0.05 |
| diarrhea\_treat\_ind | 0.01 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 |
| anticoag\_ind | 0.01 | 0.01 | 0.01 | 0.01 | 0.00 | 0.00 | 0.02 | 0.01 | 0.00 | 0.03 |
| seizure\_diagnosis | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.02 |
| hyperglycemia\_diagnosis | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 |
| ddi\_ind | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 |
| seizure\_treat\_ind | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Feature importance refers to a set of techniques that assign a score to input features based on how useful or valuable they are in predicting the target variable in a machine learning model. The idea is to understand which features (or variables) drive the predictions most and which are less relevant. The values seem to represent the importance or influence of each variable in the model's predictions:

A value of 1 indicates maximum importance in that model. A value closer to 0 suggests that the variable has little to no importance for that model's predictions. The variable race , est\_age are very important across almost all models, and with values around 0.9 or higher in most cases.



Need some sentences for 4.2 and 4.3 (Chandrika)

Missing value: fill all missing value with 0

Oncology has witnessed noteworthy advancements in treatments, yet cancer remains a prominent cause of mortality in the US, claiming around 600,000 lives annually. A key medication, Osimertinib, an oral tyrosine kinase inhibitor, is employed for early-stage non-small cell lung cancer patients with a specific EGFR mutation. While its efficacy is commendable, doubling survival rates and reducing recurrence by 80% when adhered to, its associated side effects, including nausea, fatigue, and high blood glucose, present challenges. Roughly 25% of Humana members discontinue Osimertinib within six months due to these side effects. Leveraging data analytics can aid in identifying at-risk members, promoting medication adherence, and enhancing the quality of life for oncology patients.