Assessing the Long-term treatment outcome in HIV patients.

Celestine Kubuafor, Gopi Krishna Boppana, Ria Treesa Raju, Sameeksha Pulijala, and Salwa Moiz

Indiana University Purdue University Indianapolis, IN 46202, USA
ckubuafo@iu.edu, gboppana@iu.edu, rraju@iu.edu, sampulij@iu.edu, smoiz@iu.edu

Abstract. In countries where treatment is broadly accessible, combination antiretroviral medication (ART) has transitioned HIV from a deadly disease into a long-term illness. In contrast to the community as a whole, people with HIV nevertheless suffer a markedly lower degree of health-related well-being. This study intends to evaluate the relationships between patient characteristics, ART regimen types, and clinical stages of HIV with changes in viral load, opportunistic infections, suppression of CD4 count, and adherence to ART regimens. To identify every possible behaviour or pattern that may influence forthcoming therapies and improve patient outcomes, the study independently examined the data and compared the factors.

Keywords: HIV, Viral load , Quality of life, Opportunistic infections, Antiretroviral therapy.

1 Project Scope

1.1 Introduction

In nations where treatment is widely accessible, combination antiretroviral therapy (ART) has transformed HIV from a fatal disease into a chronic condition [1]. Although the majority of patients who are HIV-positive have virological control and immunological stability, nonetheless, people with HIV experience a significantly reduced health-related level of well-being compared to the normal population [2]. According to Sutini et al., the study showed 12.3 % of HIV patients experienced opportunistic infections, where the most common types were tuberculosis infection (43%), candidiasis (21%), and diarrhea (9%) [3]. Assessment of health related QoL (presence of opportunistic infections) together with therapeutic endpoints such as viral load, CD4 count, and adherence to an ART regimen is necessary for the evaluation of novel therapies and healthcare improvement initiatives [4]. So, our project aims to analyze the correlation of factors such as demographics, the regimen, and clinical stages with the CD4 count, viral load, presence of opportunistic infection and sustained adherence to ART regimens by comparing them independently.

1.2 Aim

"The aim of this study is to investigate the association between demographics, types of regimens, and clinical stages of HIV with viral load progression, opportunistic infections, suppression of CD4 count, and adherence to ART regimens in HIV-positive patients, and to determine whether there are any patterns or trends in the data that can inform future interventions and enhance patient outcomes."

1.3 Hypothesis

Based on our aim, our two hypotheses are below:
Null Hypothesis – The factors we considered and examined do not show any association with the viral load progression, opportunistic infections, suppression of CD4 count, and adherence to ART regimens in HIV-positive patients.
Alternate Hypothesis – The factors we considered and examined showed an association with the viral load progression, opportunistic infections, suppression of CD4 count, and adherence to ART regimens in HIV-positive patients.

1.4 Purpose

The purpose of this study is to find out how variables like demographics, types of regimens, and clinical stages of HIV are associated with viral load progression, presence of opportunistic infections, suppression of CD4 count, and adherence to ART regimen. According to Robbins et al, For HIV-positive people to have viral suppression and better quality of life, effective compliance to antiretroviral therapy (ART) is essential [5]. This study seeks to determine any potential patterns or trends in the data and provide insights that can inform future interventions and enhance the quality of life of HIV-positive patients.

2 Methodology

2.1 Steps of the Project

The primary objective of our study is to examine and assess the relationships between patient characteristics, ART regimen types, and clinical stages of HIV with changes in viral load, opportunistic infections, suppression of CD4 count, and adherence to ART regimens using database technologies like SQL and Python. Jupyter Notebook in Python and phpMyAdmin were the tools we used for this approach.
The methodology of our project involves 4 stages, which includes:
1. Data Collection
2. Data Extraction and Storage

3. Data Analysis
4. Data Visualization

2.2 Team Members and Responsibilities

In our project, we had a diverse team of members from different medical backgrounds, each with their own unique skill sets and responsibilities.
The following are the roles assigned to each member:

Table 1. Project Member's Responsibilities

Name	**Background**	**Roles and Responsibilities**
Celestine Kubuafor	Diploma in Information Management, new to python	- Data Collection - Machine Learning models
Gopi Krishna Boppana	Doctor of Pharmacy, experience with statistics, new to python	- Data Cleaning - Machine Learning models, Final Report
Ria Treesa Raju	Bachelor of dental surgery, new to python	- Data Analysis - Machine Learning models, Final Report
Sameeksha Pulijala	Doctor of Pharmacy, experience with statistics, new to python	- Statistics - Data Visualization, Project Presentation
Salwa Moiz	Bachelor of Medicine & Bachelor of Surgery, new to python	- Data Collection, Data Visualization - Data Visualization, and Final Report

2.3 Project Challenges

Initial Data Set Challanges: The project was successful, and we worked efficiently as a team. Despite this, we indeed faced some hurdles. We faced a challenge at the very first step as our data set contains 27289 rows and 46 attributes. This obstructed us from directly uploading the csv file to MySQL. So, to import it, we divided the data set into 4 zip files and created a table with the attributes and then uploaded it.
Data Cleaning Process Obstacles: Our data set contained some special characters which made our description and cleaning process difficult. It took considerable time to manually eliminate redundant characters throughout the data cleaning procedure.
Compromised Quality of Insights due to Erroneous Data: When analysis was done on data that was erroneous or incomplete, the quality of the insights suffered. Because of this, the analysis's findings were of poorer quality, which reduced their value in helping people make educated decisions.
Visualization Technique Difficulties: Due to the limited number of methods that worked with our data, it became difficult to locate a visualization technique that was appropriate for our dataset. We had to spend a lot of time looking for

appropriate techniques and testing with various choices. Even after experimenting with several visualization techniques, we had trouble settling on one that would successfully communicate the data we wished to portray.
Team Management and Coordination Issues: Organizing in-person team meetings and keeping track of assigned work proved difficult because one of our team members is an online student. We also encountered restrictions in our capacity to be creative with the results we hoped to obtain because of the dataset difficulties stated previously. Despite these difficulties, we tried our best to stay in regular contact with the online team member by using online conferences and team building software.

3 Data Collection

Using Kaggle, we have obtained our Quality of care in HIV clients' dataset.
https://www.kaggle.com/datasets/iogbonna/quality-of-care-dataset-for-hiv-clients

4 Data Extraction and Storage

4.1 Data Importing

Data importing is a critical step in any data analysis project. In our project, we imported a CSV file into a MySQL database using the following steps:

Uploaded the CSV file into SQL: The first step was to upload the CSV file into SQL. This was done to ensure that the data was in a format that could be easily imported into the MySQL database.

Installed MySQLdb with the help of pip: Installed MySQLdb with the help of pip: After uploading the CSV file into SQL, we installed MySQLdb with the help of pip. This is a Python interface for MySQL that allows us to connect to the MySQL database and perform various operations.

Imported the CSV file into the MySQL database: We established a connection to the MySQL database with the help of the connect() function. In this function, we specified the host, username, password, and database name. We then created a cursor object using the cursor() function. Finally, we executed the SQL query using execute() function to load the data into the database [cursor.execute()].

By importing the data into the MySQL database, we were able to easily manipulate and analyze it using various SQL queries. This allowed us to efficiently clean, preprocess, and analyze the data for our project.

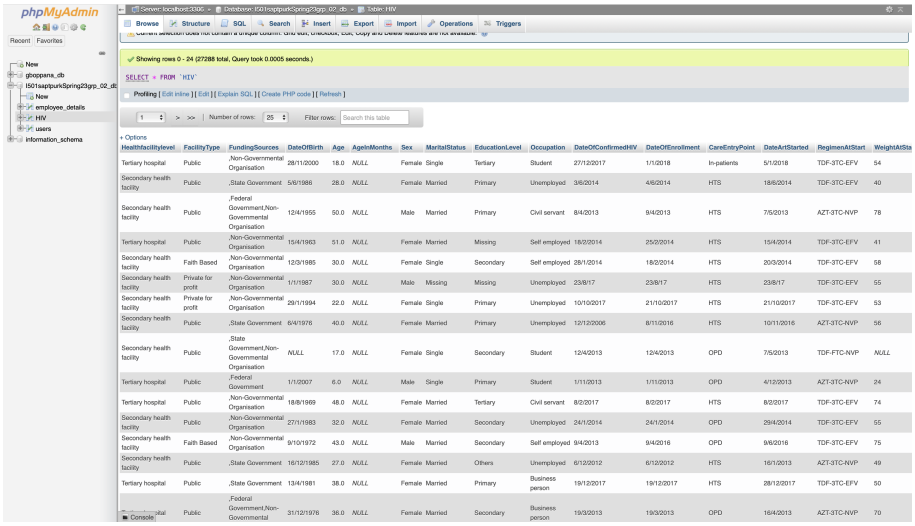


Fig. 1. Importing the dataset into SQL

4.2 Data Extraction

From the Quality-of-care dataset our initial task is to remove the unnecessary columns. After discussing our aims and objectives with the group, we decided to keep around 16 columns. Our aim was "To investigate the association between demographics, types of regimens, and clinical stages of HIV with viral load progression, opportunistic infections, suppression of CD4 count, and adherence to ART regimens in HIV-positive patients, and to determine whether there are any patterns or trends in the data that can inform future interventions and enhance patient outcomes."
The 16 attributes we selected are below:
1. Age
2. Sex
3. Marital Status
4. Education
5. Occupation
6. Regimen at start
7. Weight at start
8. Weight at last visit
9. Clinical stage at last visit
10. Tb status at last visit
11. CD4 at start
12. Any side effects
13. Opportunistic infection present at last visit
14. Viral load
15. Most recent CD4 count

16. ARV adherence latest level

4.3 Data Storage

After extracting the necessary columns, we stored our data in a pandas data frame.

4.4 Data Description

Our dataset contains 16 attributes out of which 12 are independent variables and 4 are dependent variables.
The following are independent and dependent variables.

Table 2. Description of Dataset Variables

Types of Variables	**Attributes**
Independent Variables	Age, Sex, Marital status, Education, Occupation, Regimen at start, Weight at start and at last visit, Clinical stage at last visit, Tbstatus at last visit, CD4 at start, Any side effects.
Dependent Variables	Opportunistic infection, Viral load, Most recent CD4 count, ARV adherence level

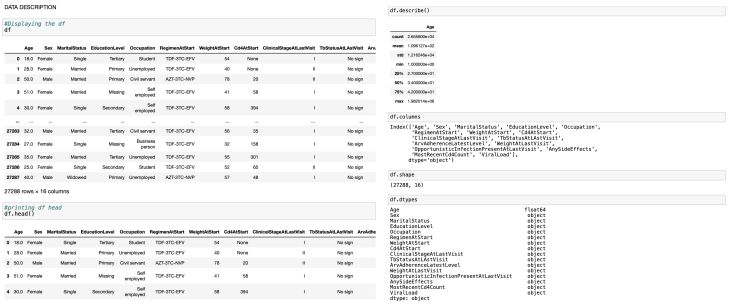


Fig. 2. Describing the attributes and instances selected for the project, displaying the first 5 rows, the columns present, shape of the data set, and data types

4.5 Data Cleaning

Data cleaning is a critical process that ensures that datasets are accurate and reliable for analysis. The presence of missing values, duplicates, and outliers can

have a significant impact on the validity and reliability of the insights derived from the data. Addressing these issues is essential for producing meaningful and valid conclusions from the dataset.

Checking and Replacing Null values: The first step in data cleaning involves identifying missing values and using an appropriate method to replace them. In our case, we identified the number of missing values in each attribute and used mode imputation for categorical variables and mean imputation for numerical variables. For the variables having more missing values, we replaced with imperative imputation. This helps to ensure that the dataset is as complete and accurate as possible, allowing for more meaningful insights to be derived from the data.

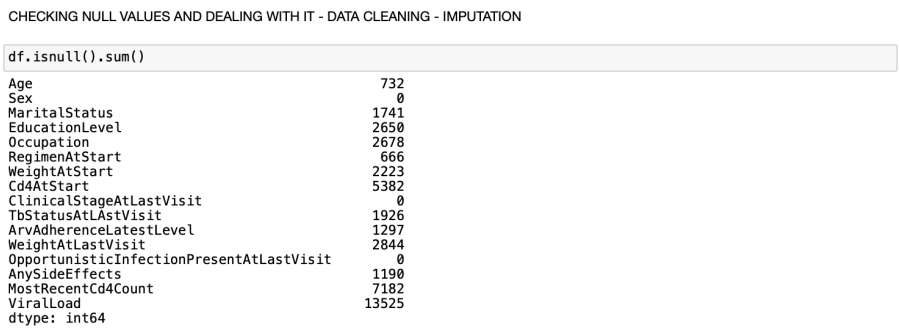


Fig. 3. Checking the null values present in the attributes

Fig. 4. Iterative Imputation to replace the null values present

Dealing with Duplicates: After addressing missing values, the next step is to remove duplicates from the dataset. This involves using a function like drop_duplicates() to identify and remove records with identical attribute values. This helps to ensure that the analysis is not skewed by repeated observations and that the insights derived from the data are more accurate and reliable.

Fig. 5. Checking the duplicates present and removing the duplicates

Handling Outliers: Finally, outliers are another issue that must be addressed during data cleaning. These are values that fall far outside the expected range for a given variable and can have a significant impact on statistical analyses. To address this issue, we used the interquartile range (IQR) to identify extreme values and then replaced them with more typical values using a capping method. This helps to ensure that the dataset is not skewed by extreme values and that the insights derived from the data are more accurate and reliable.

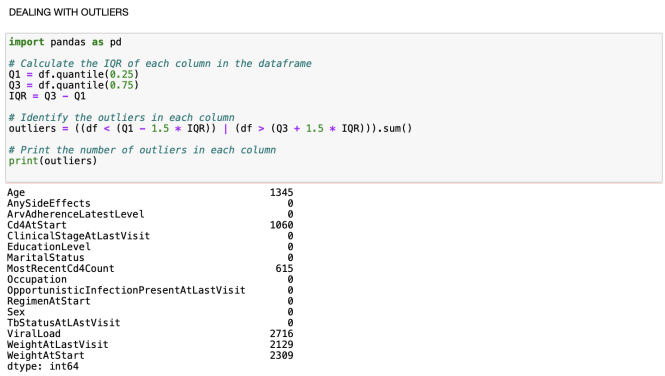


Fig. 6. Identifying the outliers present in the attributes

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np


# Create a figure with two subplots
fig, axs = plt.subplots(ncols=2, figsize=(10, 5))

# Plot boxplot of the MostRecentCd4Count column before outlier removal
sns.boxplot(x=df['MostRecentCd4Count'], ax=axs[0])
axs[0].set_title('Before Outlier Removal')

# Calculate the first and third quartiles of the MostRecentCd4Count column
Q1 = df['MostRecentCd4Count'].quantile(0.25)
Q3 = df['MostRecentCd4Count'].quantile(0.75)

# Calculate the interquartile range of the MostRecentCd4Count column
IQR = Q3 - Q1

# Define the lower and upper bounds of the "normal" range of the MostRecentCd4Count column
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Replace the outliers in the MostRecentCd4Count column with the capped values
df['MostRecentCd4Count'] = np.where(df['MostRecentCd4Count'] < lower_bound, lower_bound, df['MostRecentCd4Coun
df['MostRecentCd4Count'] = np.where(df['MostRecentCd4Count'] > upper_bound, upper_bound, df['MostRecentCd4Coun


# Plot boxplot of the MostRecentCd4Count column after outlier removal
sns.boxplot(x=df['MostRecentCd4Count'], ax=axs[1])
axs[1].set_title('After Outlier Removal')

# Set the plot title
plt.suptitle('Distribution of MostRecentCd4Count Column Before and After Outlier Removal')

# Show the plot
plt.show()
```

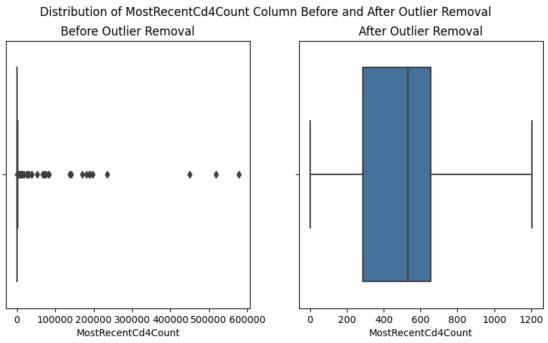


Fig. 7. Interquartile range method to detect the outliers and capping to replace the outliers. Box and Whisker plot to visualize after capping the outliers

5 Data Analysis

Statistics plays a vital role in analyzing and interpreting data. In this paper, we will discuss the statistics used in our project, including the use of descriptive statistics, normality tests, non-parametric tests, and the chi-square test.

Our first step in analyzing data is to use descriptive statistics to summarize and understand the data. In our project, we used the df.describe() function to generate descriptive statistics such as the mean, median, and standard deviation for each variable in our dataset. These descriptive statistics helped us to better understand the distribution of the data and identify potential issues such as outliers and skewed distributions.

DECSRIPTIVE STATISTICS - DATA ANALYSIS

```
df.describe()
```

	Age	WeightAtStart	Cd4AtStart	WeightAtLastVisit	MostRecentCd4Count	ViralLoad
count	27093.000000	27093.000000	27093.000000	27093.000000	27093.000000	27093.000000
mean	34.985199	55.483224	341.149655	59.422035	497.990293	130.442791
std	13.806066	14.478059	212.418046	15.041646	262.939322	121.324662
min	3.000000	24.000000	0.000000	25.500000	0.000000	0.000000
25%	27.000000	48.000000	169.000000	51.000000	288.000000	20.000000
50%	35.000000	55.000000	335.000000	60.000000	530.000000	130.442791
75%	43.000000	64.000000	451.511158	68.000000	654.213839	130.442791
max	67.000000	88.000000	875.277894	93.500000	1203.534598	447.500000

Fig. 8. Descriptive statistics

5.1 Performing Normality Test

Statistical analysis plays a critical role in drawing valid conclusions from a dataset. In our project, we used the normaltest() function and visually inspected QQ plots to check for normality of our data. None of the variables followed a normal distribution, so we used non-parametric tests such as the Mann-Whitney U test and the Kruskal-Wallis test to analyze our data.

QQ PLOT - FOR CHECKING NORMALITY

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt


# Extract the columns as NumPy arrays
age_data = df['Age'].to_numpy()
weight_start_data = df['WeightAtStart'].to_numpy()
weight_last_data = df['WeightAtLastVisit'].to_numpy()
cd4_start_data = df['Cd4AtStart'].to_numpy()
Most_recent_CD4_count = df['MostRecentCd4Count'].to_numpy()
Viral_load = df['ViralLoad'].to_numpy()

# Create a 2 x 2 grid of subplots
fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(10, 8))

# Plot the QQ plot for age data
sm.qqplot(age_data, line='s', ax=axs[0, 0])
axs[0, 0].set_title('QQ Plot for Age Data')

# Plot the QQ plot for weight at start data
sm.qqplot(weight_start_data, line='s', ax=axs[0, 1])
axs[0, 1].set_title('QQ Plot for WeightAtStart Data')

# Plot the QQ plot for weight at last visit data
sm.qqplot(weight_last_data, line='s', ax=axs[0, 2])
axs[0, 2].set_title('QQ Plot for WeightAtLastVisit Data')

# Plot the QQ plot for CD4 at start data
sm.qqplot(cd4_start_data, line='s', ax=axs[1, 0])
axs[1, 0].set_title('QQ Plot for Cd4AtStart Data')

# Plot the QQ plot for CD4 at start data
sm.qqplot(Most_recent_CD4_count, line='s', ax=axs[1, 1])
axs[1, 1].set_title('QQ Plot for MostRecentCd4Count Data')

# Plot the QQ plot for CD4 at start data
sm.qqplot(Viral_load, line='s', ax=axs[1, 2])
axs[1, 2].set_title('QQ Plot for ViralLoad Data')


# Adjust the layout and display the plot
plt.tight_layout()
plt.show()
```

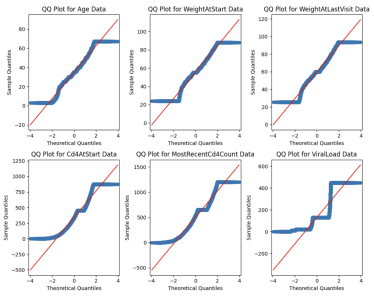
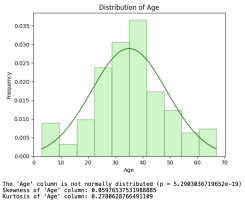


Fig. 9. Inferential Statistics using QQ Plot

NORMALITY TEST

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import normaltest, skew, kurtosis

# perform normality test on 'Age' column
stat, p = normaltest(df['Age'])

# calculate skewness and kurtosis of 'Age' column
sk = skew(df['Age'])
ku = kurtosis(df['Age'])

# plot histogram of 'Age' column
plt.hist(df['Age'], density=True, alpha=0.5, color='lightgreen', edgecolor = 'green')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')

# add a normal distribution curve to the plot for comparison
mu, sigma = df['Age'].mean(), df['Age'].std()
x = np.linspace(df['Age'].min(), df['Age'].max(), 100)
y = np.exp(-(x-mu)**2 / (2*sigma**2)) / (sigma * np.sqrt(2*np.pi))
plt.plot(x, y, color='green')

plt.show()

alpha = 0.05
if p < alpha:
    print("The 'Age' column is not normally distributed (p = {})".format(p))
else:
    print("The 'Age' column is normally distributed (p = {})".format(p))

print("Skewness of 'Age' column: {}".format(sk))
print("Kurtosis of 'Age' column: {}".format(ku))
```




```
The 'Age' column is not normally distributed (p = 5.290303671965 2e-19)
Skewness of 'Age' column: 0.8597653753138888 85
Kurtosis of 'Age' column: 0.2780628766493189
```

Fig. 10. Normality test to check the data distribution

5.2 Non Parametric Tests

We performed Mann Whitney tests between continuous independent and categorical dependent variables, and vice versa. We also used Kruskal Wallis tests to analyze continuous independent and categorical dependent variables that had more than two classes. The results of our Kruskal Wallis tests showed that there were no significant differences between CD4 at start and ARV adherence latest level. Additionally, our Mann Whitney tests revealed that there were no significant differences between age and opportunistic infections at the last visit, and no significant difference between any side effects and viral load.

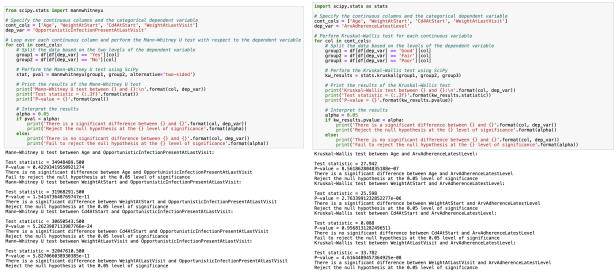


Fig. 11. Mann Whitney U test and Kruskal Wallis Test to check for association

5.3 Chi Square Test

We used the chi-square test to analyze the relationship between two categorical variables. Our chi-square test results indicated that Sex did not have a significant association with our categorical dependent variables, opportunistic infection at last visit and ARV adherence latest level.

Fig. 12. Chi-square Statistical test to check for association

5.4 Correlation Test

Finally, we performed Kendall tau correlation between continuous independent and dependent variables. Our results showed that weight at start, weight at last visit, and age had no correlation with most recent CD4 count, and weight at start, weight at last visit, age, and CD4 at start had no correlation with viral load. These findings provide insight into the relationships between various variables in our dataset.

```
import seaborn as sns
import pandas as pd

# Calculate the Kendall correlation matrix using pandas
corr_matrix = df.corr(method='kendall')

# Create a heatmap using Seaborn
sns.heatmap(corr_matrix, annot=True, cmap = 'Reds')

# Display the plot
plt.show()
```

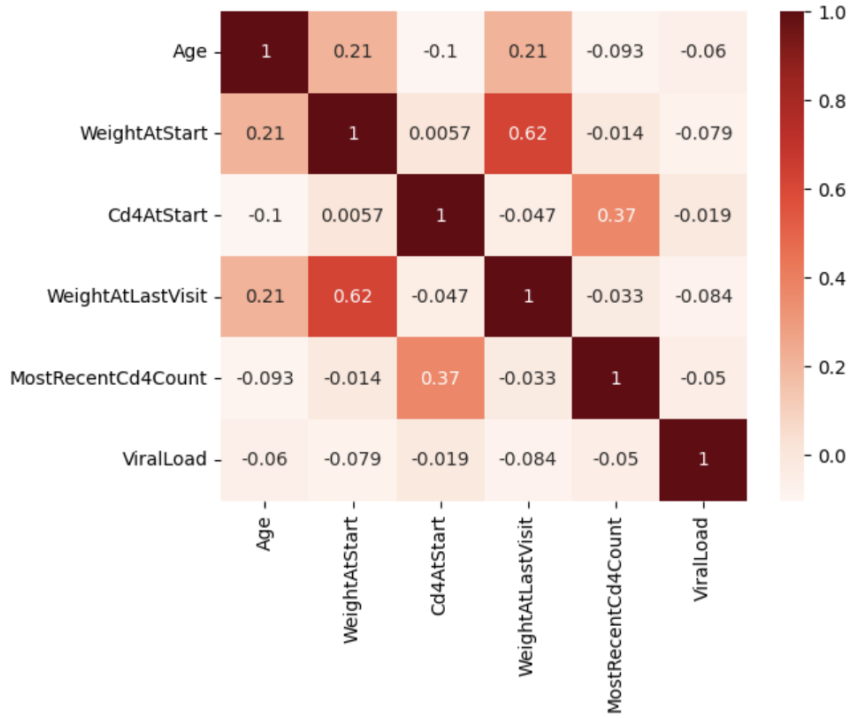


Fig. 13. Kendalls Tau Correlation matrix

6 Data Visualization

Data visualization is an important tool for gaining insights and presenting findings in a clear and concise manner. In our project, we used several types of data visualization techniques to explore and analyze our data.

Density Plot: We started by creating density plots to visualize the distribution of Most Recent CD4 Count and Viral Load. Density plots are useful for showing the shape of a distribution and for comparing the distributions of two or more variables.

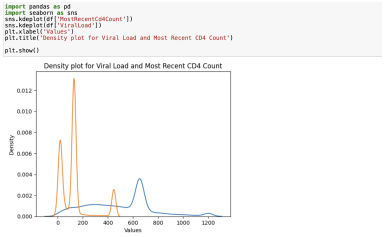


Fig. 14. Density plot depicting the relationship between MostRecentCD4 count and ViralLoad

Bar Chart: We also created bar charts to visualize the distribution of Age, Weight at Start, and Weight at Last Visit. Bar charts are an effective way to display categorical data and to compare the relative sizes of different categories.

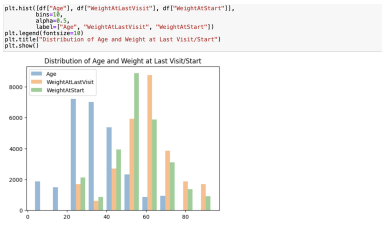


Fig. 15. Bar chart depicting the change of Weightatstart and WeightLastvst with the Age of the patients

Stacked Bar Chart: To explore the relationship between categorical variables, we used stacked bar charts to visualize the distribution of Education Level and ARV Adherence Level. Stacked bar charts allow us to see the distribution

of a categorical variable broken down by another categorical variable.

Fig. 16. Stacked bar Chart between adherence to ART regimen and Educational level

Violin Plot: Finally, we used a violin plot to visualize the distribution of Clinical Stage at Last Visit and Most Recent CD4 Count. Violin plots are useful for displaying the distribution of a continuous variable across different categories of a categorical variable.

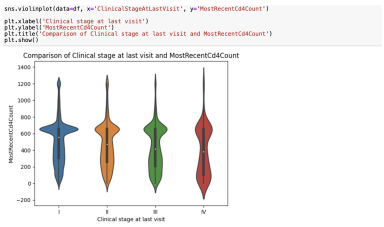


Fig. 17. A violin plot identifying the median of mostrecentcd4 count for each clinical stage of HIV at the last visit. The interquartile ranges for each stage can also be noted by the thick black line running through the center of each plot. For stage I, II, III the highest Cd4 count frequency was slightly higher than 600. Whereas for stage IV the most recent CD4 count showed the highest count between 600 and 700 as well as around 100.

Overall, data visualization played a crucial role in our project, helping us to better understand our data and to communicate our findings effectively to others.

7 Preprocessing before Machine Learning Models

Preprocessing is an essential step in any machine learning project. In our project, we followed a standard preprocessing pipeline that included data cleaning, feature selection, and feature engineering.

To start, we cleaned our data by removing any missing values and outliers. Next, we used statistical techniques such as correlation analysis and chi-square tests to select relevant features for our models. This process helped us to reduce the dimensionality of our data and select the most important features that would help us to predict the target variable.

7.1 Feature Selection

Based on the statistical tests performed we are selecting independent variables for specific dependent variable
Opportunistic infection -Marital status, Education,Occupation, Regimen at start, Weight at start and at last visit, Clinical stage at last visit, Tb status at last visit, CD4 at start,Any side effects.
Most recent CD4 count - Age, Sex, Marital status, Education, Occupation, Regimen at start, Weight at start and at last visit, Clinical stage at last visit, Tb status at last visit, CD4 at start, Any side effects.
Arv adherence level - Age, Marital status, Education, Occupation, Regimen at start, Weight at start and at last visit, Clinical stage at last visit, Tb status at last visit, CD4 at start, Any side effects.
Viral Load - Age, Sex, Marital status, Education, Occupation, Regimen at start, Weight at start and at last visit, Clinical stage at last visit, Tb status at last visit, CD4 at start.

7.2 Convertion of Categorical to Numerical variables

We converted all the categorical to numerical variables with the help of .replace() function.

Fig. 18. replacing all the categorical variables with numbers

7.3 Feature Engineering

After selecting the features, we performed feature engineering to prepare the data for modeling. We converted all categorical columns to numerical using replace() function. We applied a log transformation to some of our variables to normalize their distributions and reduce their skewness. We also used scaling techniques such as StandardScaler to rescale our data so that all the features have a similar range of values.

Fig. 19. Log transformation and Scaling the data and saving as a new file

By performing these preprocessing steps, we were able to create a clean and well-prepared dataset that was ready for modeling. This allowed us to train accurate and reliable machine learning models that could make predictions on new data with high accuracy.

8 Machine Learning Models

In our study, we applied machine learning algorithms to predict the outcome variable, which was categorical. We implemented several machine learning algorithms, including logistic regression, random forest, decision tree, and gradient boosting classifier. We evaluated the performance of these models by printing classification reports and calculating AUC-ROC scores.

Classification Models: For the outcome variables - Arv adherence latest level and Opportunistic infection present at last visit - We performed Classification models.
We performed SMOTE to correct the imbalance in classes present in the outcome variable.

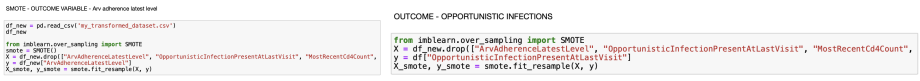


Fig. 20. Performing SMOTE on the new data set created

8.1 Logistic Regression

We split our dataset into a training set and a testing set using an 80/20 split. This allowed us to train our logistic regression model on a subset of the data and test its performance on a separate subset.
To ensure the reliability of our results, we used cross-validation with 10 folds. Cross-validation is a technique used to assess the performance of a machine learning model by splitting the data into several partitions, training the model on each partition, and evaluating its performance on the remaining data. This helps to reduce the risk of overfitting the model to the training data and improve its generalization to new data.
We evaluated the performance of our logistic regression model using various metrics, including the classification report, AUC ROC, and feature importance. The classification report provides a summary of the model's performance, including metrics such as precision, recall, and F1 score. AUC ROC is a metric that measures the ability of the model to distinguish between positive and negative classes.

LOGISTIC REGRESSION

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,cross_val_score, KFold
from sklearn.metrics import classification_report
import numpy as np
import scikitplot as skplt

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

lr = LogisticRegression()
# Perform 10-fold cross-validation
cv_scores = cross_val_score(lr, X_train, y_train, cv=KFold(n_splits=10, shuffle=True, random_state=42), scorin

# Print the mean and standard deviation of the cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean f1_macro score:", np.mean(cv_scores))

lr.fit(X_train, y_train)

# Print classification report for training set
y_train_pred = lr.predict(X_train)
print("Training Classification Report:")
print(classification_report(y_train, y_train_pred))

# Print classification report for test set
y_test_pred = lr.predict(X_test)
print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))


plt.rcParams['figure.figsize'] = [8,6]

predicted_probas = lr.predict_proba(X_test)
skplt.metrics.plot_roc(y_test, predicted_probas)

plt.show()
```

```
Cross-validation scores: [0.59934895 0.5930484  0.60522399 0.59757046 0.59886468 0.60040801
 0.59263055 0.60086159 0.58704199 0.6058159 ]
Mean f1_macro score: 0.5980814509881747
Training Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.80      0.68     19513
           1       0.68      0.42      0.52     19263

    accuracy                           0.61     38776
   macro avg       0.63      0.61      0.60     38776
weighted avg       0.63      0.61      0.60     38776

Test Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.80      0.66      4722
           1       0.69      0.42      0.52      4972

    accuracy                           0.61      9694
   macro avg       0.63      0.61      0.59      9694
weighted avg       0.63      0.61      0.59      9694
```

Fig. 21. Logistic Regression for ArvAdherenceLatestLevel and classification report of the model

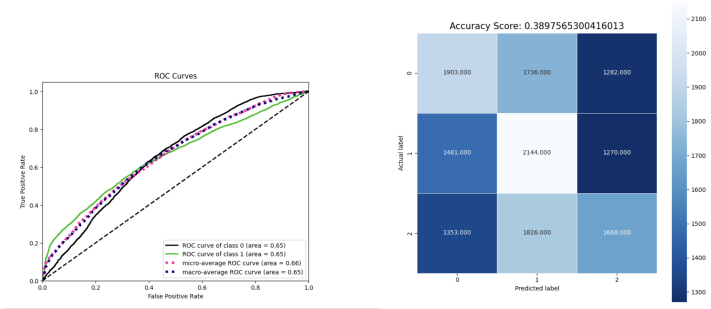
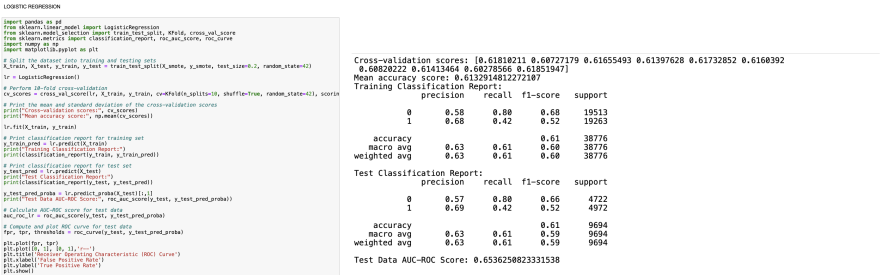


Fig. 22. ROC curve for the model and the confusion matrix depicting the actual and the predicted values



```
LOGISTIC REGRESSION

import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import classification_report, roc_auc_score, roc_curve
import numpy as np
import matplotlib.pyplot as plt

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

lr = LogisticRegression()

# Perform 10-fold cross-validation
cv_scores = cross_val_score(lr, X_train, y_train, cv=KFold(n_splits=10, shuffle=True, random_state=42), scoring

# Print the mean and standard deviation of the cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean accuracy score:", np.mean(cv_scores))

lr.fit(X_train, y_train)

# Print classification report for training set
y_train_pred = lr.predict(X_train)
print("Training Classification Report:")
print(classification_report(y_train, y_train_pred))

# Print classification report for test set
y_test_pred = lr.predict(X_test)
print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))

y_test_pred_proba = lr.predict_proba(X_test)[:,1]
print("Test Data AUC-ROC Score:", roc_auc_score(y_test, y_test_pred_proba))

# Calculate AUC-ROC score for test data
auc_roc_lr = roc_auc_score(y_test, y_test_pred_proba)

# Compute and plot ROC curve for test data
fpr, tpr, thresholds = roc_curve(y_test, y_test_pred_proba)

plt.plot(fpr, tpr)
plt.plot([0, 1], [0, 1],'r--')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```

```
Cross-validation scores: [0.61810211 0.60727179 0.61655493 0.61397628 0.61732852 0.6160392
 0.60820222 0.61413464 0.60278566 0.61851947]
Mean accuracy score: 0.6132914812272107
Training Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.80      0.68     19513
           1       0.68      0.42      0.52     19263

    accuracy                           0.61     38776
   macro avg       0.63      0.61      0.60     38776
weighted avg       0.63      0.61      0.60     38776

Test Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.80      0.66      4722
           1       0.69      0.42      0.52      4972

    accuracy                           0.61      9694
   macro avg       0.63      0.61      0.59      9694
weighted avg       0.63      0.61      0.59      9694

Test Data AUC-ROC Score: 0.6536250823331538
```

Fig. 23. Logistic Regression for Opportunisticinfectionlastvisit and classification report of the model

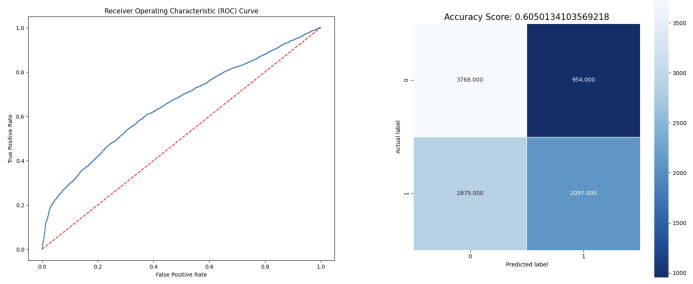


Fig. 24. Confusion Matrix and AUC-ROC Curve of the logistic regression model

8.2 Decision Tree Classifier

In our project, we used the Decision Tree Classifier algorithm to build a model to predict the outcome variable. We performed an 80-20 split on our data to train and test our model respectively. Additionally, we used cross-validation with 10 folds to ensure the robustness and accuracy of our model.

DECISION TREE CLASSIFIER

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.metrics import classification_report
import numpy as np
import scikitplot as skplt

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

dt = DecisionTreeClassifier()

# Perform 10-fold cross-validation
cv_scores = cross_val_score(dt, X_train, y_train, cv=KFold(n_splits=10, shuffle=True, random_state=42), scorin

# Print the mean and standard deviation of the cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean f1_macro score:", np.mean(cv_scores))

dt.fit(X_train, y_train)

# Print classification report for training set
y_train_pred = dt.predict(X_train)
print("Training Classification Report:")
print(classification_report(y_train, y_train_pred))

# Print classification report for test set
y_test_pred = dt.predict(X_test)
print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))

plt.rcParams['figure.figsize'] = [10,8]

predicted_probas = dt.predict_proba(X_test)
skplt.metrics.plot_roc(y_test, predicted_probas)

plt.show()
```

```
Cross-validation scores: [0.85559429 0.85449594 0.84547152 0.86123766 0.86658605 0.84680854
 0.83337445 0.85607267 0.84214547 0.86329526]
Mean f1_macro score: 0.8525801847783728
Training Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     19513
           1       1.00      0.99      1.00     19263

    accuracy                           1.00     38776
   macro avg       1.00      1.00      1.00     38776
weighted avg       1.00      1.00      1.00     38776

Test Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.86      4722
           1       0.87      0.88      0.87      4972

    accuracy                           0.87      9694
   macro avg       0.87      0.87      0.87      9694
weighted avg       0.87      0.87      0.87      9694
```

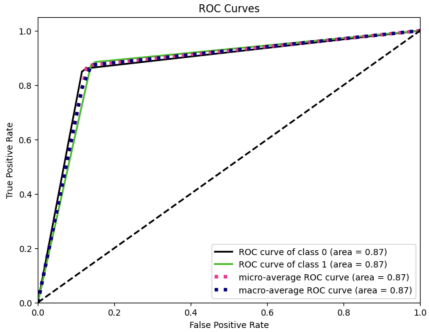


Fig. 25. Decision tree classifier model for ArvAdherenceLatestLevel and the classification report and the ROC curve for the model

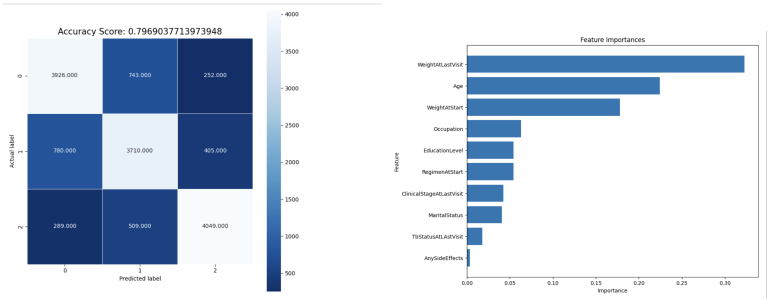
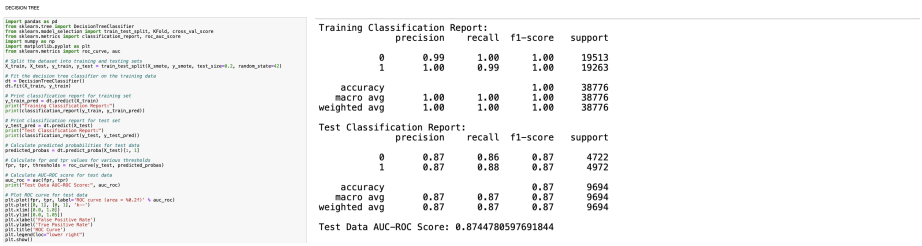


Fig. 26. Confusion Matrix and Feature importance of the model



```
DECISION TREE
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import classification_report, roc_auc_score
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

# Fit the decision tree classifier on the training data
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

# Print classification report for training set
y_train_pred = dt.predict(X_train)
print("Training Classification Report:")
print(classification_report(y_train, y_train_pred))

# Print classification report for test set
y_test_pred = dt.predict(X_test)
print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))

# Calculate predicted probabilities for test data
predicted_probas = dt.predict_proba(X_test)[:, 1]

# Calculate fpr and tpr values for various thresholds
fpr, tpr, thresholds = roc_curve(y_test, predicted_probas)

# Calculate AUC-ROC score for test data
auc_roc = auc(fpr, tpr)
print("Test Data AUC-ROC Score:", auc_roc)

# Plot ROC curve for test data
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_roc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.show()
```

```
Training Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     19513
           1       1.00      0.99      1.00     19263

    accuracy                           1.00     38776
   macro avg       1.00      1.00      1.00     38776
weighted avg       1.00      1.00      1.00     38776

Test Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.87      4722
           1       0.87      0.88      0.87      4972

    accuracy                           0.87      9694
   macro avg       0.87      0.87      0.87      9694
weighted avg       0.87      0.87      0.87      9694

Test Data AUC-ROC Score: 0.8744780597691844
```

Fig. 27. Decision tree classifier model for Opportunisticinfectionlastvisit and the classification report and the ROC curve for the model

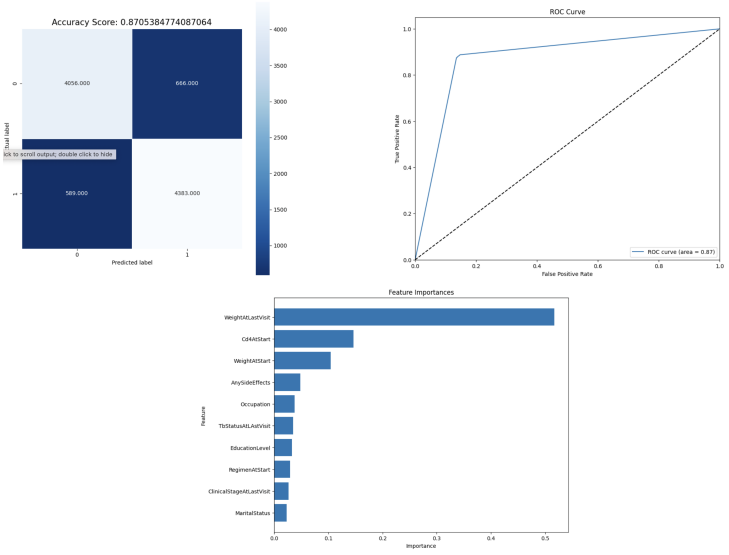


Fig. 28. Confusion Matrix and AUC-ROC Curve and feature importance of Decision tree classifier model

8.3 Random Forest Classifier

In this study, we applied the random forest classifier to predict the outcome variable using an 80-20 train-test split and performed 10-fold cross-validation to evaluate the model's performance. Random forest is an ensemble learning method that constructs multiple decision trees at training time and outputs the class that is the mode of the classes of the individual trees.

RANDOM FOREST CLASSIFIER

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.metrics import classification_report
import numpy as np
import scikitplot as skplt
import matplotlib.pyplot as plt

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

rf = RandomForestClassifier()
# Perform 10-fold cross-validation
cv_scores = cross_val_score(dt, X_train, y_train, cv=KFold(n_splits=10, shuffle=True, random_state=42), scorin

# Print the mean and standard deviation of the cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean f1_macro score:", np.mean(cv_scores))

rf.fit(X_train, y_train)

# Print classification report for training set
y_train_pred = rf.predict(X_train)
print("Training Classification Report:")
print(classification_report(y_train, y_train_pred))

# Print classification report for test set
y_test_pred = rf.predict(X_test)
print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))


plt.rcParams['figure.figsize'] = [10,8]

predicted_probas = rf.predict_proba(X_test)
skplt.metrics.plot_roc(y_test, predicted_probas)

plt.show()
```

```
Cross-validation scores: [0.85481928 0.84829374 0.84054977 0.85942168 0.86914744 0.8519728
 0.83363408 0.85426842 0.84678866 0.85891138]
Mean f1_macro score: 0.8517807242475282
Training Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     19513
           1       1.00      0.99      1.00     19263

    accuracy                           1.00     38776
   macro avg       1.00      1.00      1.00     38776
weighted avg       1.00      1.00      1.00     38776

Test Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.89      0.89      4722
           1       0.90      0.89      0.90      4972

    accuracy                           0.89      9694
   macro avg       0.89      0.89      0.89      9694
weighted avg       0.89      0.89      0.89      9694
```

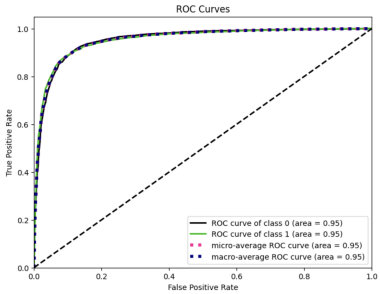


Fig. 29. Random Forest Classifier model for ArvAdherenceLatestLevel and the classification report and the ROC curve for the model

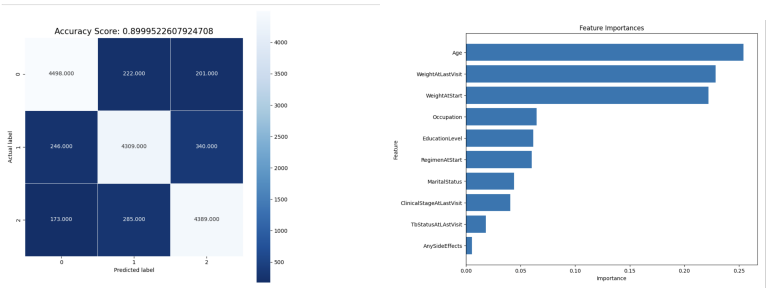
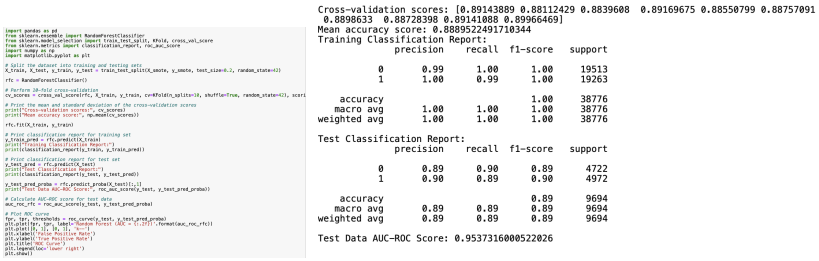


Fig. 30. Confusion Matrix and the feature importance of Random Forest classifier



```
Cross-validation scores: [0.89143889 0.88112429 0.8839608  0.89169675 0.88550799 0.88757091
 0.8898633  0.88728398 0.89141088 0.89966469]
Mean accuracy score: 0.888952249170344
Training Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     19513
           1       1.00      0.99      1.00     19263

    accuracy                           1.00     38776
   macro avg       1.00      1.00      1.00     38776
weighted avg       1.00      1.00      1.00     38776

Test Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.90      0.89      4722
           1       0.90      0.89      0.90      4972

    accuracy                           0.89      9694
   macro avg       0.89      0.89      0.89      9694
weighted avg       0.89      0.89      0.89      9694

Test Data AUC-ROC Score: 0.9537316000522026
```

Fig. 31. Random Forest classifier model for Opportunisticinfectionlastvisit and the classification report and the ROC curve for the model

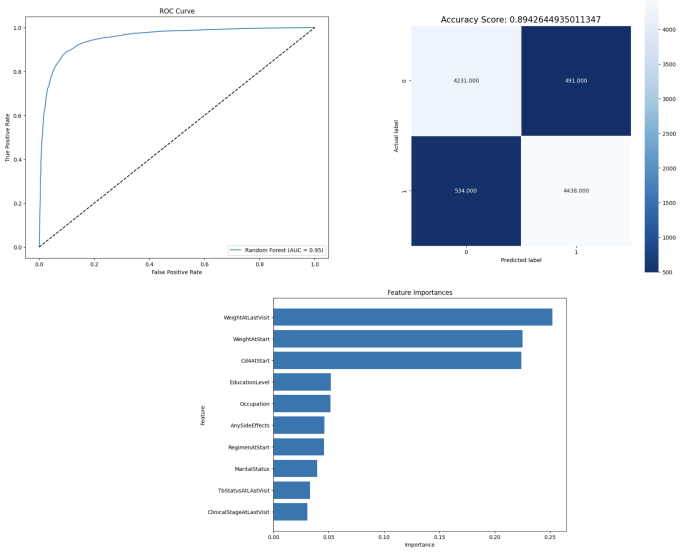


Fig. 32. Confusion Matrix and feature importance for the Random Forest Classifier model

8.4 Gradient Boosting Classifier

We chose gradient boost classifier because it works by iteratively adding decision trees to the model, with each subsequent tree attempting to correct the errors made by the previous tree. This process continues until a specified number of trees have been added or until the model is no longer improving.

GRADIENT BOOST CLASSSIFIER

```
import pandas as pd
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.metrics import classification_report
import numpy as np
import scikitplot as skplt

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

gb = GradientBoostingClassifier()

# Perform 10-fold cross-validation
cv_scores = cross_val_score(dt, X_train, y_train, cv=KFold(n_splits=10, shuffle=True, random_state=42), scorin

# Print the mean and standard deviation of the cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean f1_macro score:", np.mean(cv_scores))

gb.fit(X_train, y_train)

# Print classification report for training set
y_train_pred = gb.predict(X_train)
print('Training Classification Report:')
print(classification_report(y_train, y_train_pred))

# Print classification report for test set
y_test_pred = gb.predict(X_test)
print('Test Classification Report:')
print(classification_report(y_test, y_test_pred))

plt.rcParams['figure.figsize'] = [10,8]

predicted_probas = gb.predict_proba(X_test)
skplt.metrics.plot_roc(y_test, predicted_probas)

plt.show()
```

```
Cross-validation scores: [0.85713431 0.84801854 0.84107708 0.86278727 0.86501797 0.85017611
 0.83698709 0.85710592 0.8468143  0.86509959]
Mean f1_macro score: 0.8529418178661647
Training Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.83      0.80     19513
           1       0.81      0.74      0.77     19263

    accuracy                           0.79     38776
   macro avg       0.79      0.79      0.79     38776
weighted avg       0.79      0.79      0.79     38776

Test Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.82      0.79      4722
           1       0.82      0.74      0.78      4972

    accuracy                           0.78      9694
   macro avg       0.79      0.78      0.78      9694
weighted avg       0.79      0.78      0.78      9694
```

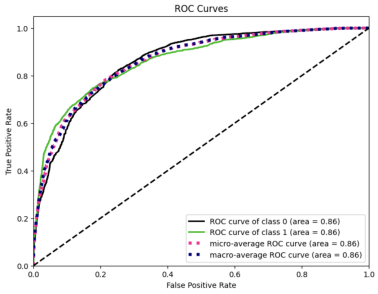


Fig. 33. Gradient Boost Classifier model for ArvAdherenceLatestLevel and the classification report and the ROC curve for the model

Regression Models: For the outcome variables - Most recent CD4 count and Viral Load we performed regression models.

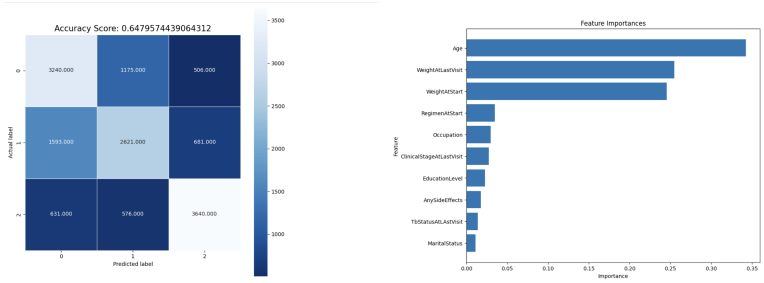


Fig. 34. Confusion Matrix and feature importance for the gradient Boost classifier model

```
import pandas as pd
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import classification_report, roc_auc_score
import numpy as np
import matplotlib.pyplot as plt

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

gb = GradientBoostingClassifier()

# Perform 10-fold cross-validation
cv_scores = cross_val_score(gb, X_train, y_train, cv=KFold(n_splits=10, shuffle=True, random_state=42), scoring

# Print the mean and standard deviation of the cross-validation scores
print('Cross-validation scores:', cv_scores)
print('Mean accuracy score:', np.mean(cv_scores))

gb.fit(X_train, y_train)

# Print classification report for training set
y_train_pred = gb.predict(X_train)
print("Training Classification Report:")
print(classification_report(y_train, y_train_pred))

# Print classification report for test set
y_test_pred = gb.predict(X_test)
print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))

y_test_pred_proba = rfc.predict_proba(X_test)[:,1]
print("Test Data AUC-ROC Score:", roc_auc_score(y_test, y_test_pred_proba))

# Calculate AUC-ROC score for test data
auc_roc_rfc = roc_auc_score(y_test, y_test_pred_proba)

# Plot ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_test_pred_proba)
plt.plot(fpr, tpr, label='Gradient Boost (AUC = %.2f)' %auc_roc_rfc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

```
Cross-validation scores: [0.78390923 0.76560083 0.78803507 0.7877772  0.79190304 0.78390923
 0.78488522 0.77688935 0.77327831 0.78514315]
Mean accuracy score: 0.7821330633323326
Training Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.83      0.80     19513
           1       0.81      0.74      0.77     19263

    accuracy                           0.79     38776
   macro avg       0.79      0.79      0.79     38776
weighted avg       0.79      0.79      0.79     38776

Test Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.82      0.79      4722
           1       0.82      0.74      0.78      4972

    accuracy                           0.78      9694
   macro avg       0.79      0.78      0.78      9694
weighted avg       0.79      0.78      0.78      9694

Test Data AUC-ROC Score: 0.9537316000522026
```

Fig. 35. Gradient Boost Classifier model for Opportunisticinfectionlastvisit and the classification report for the model

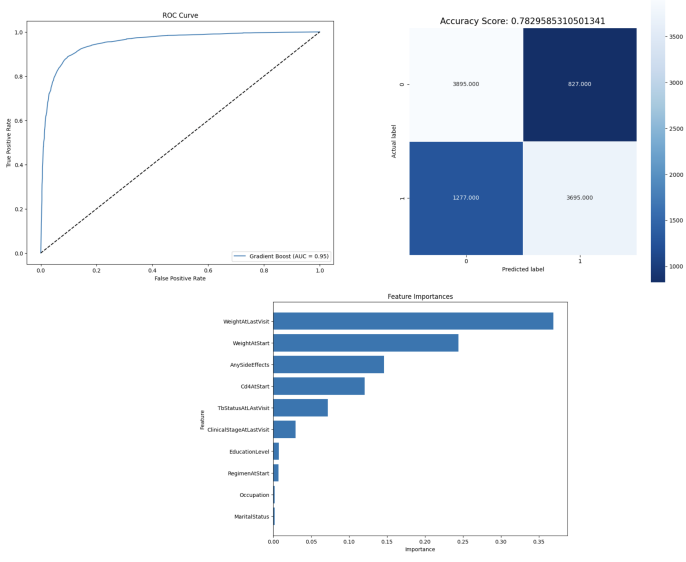
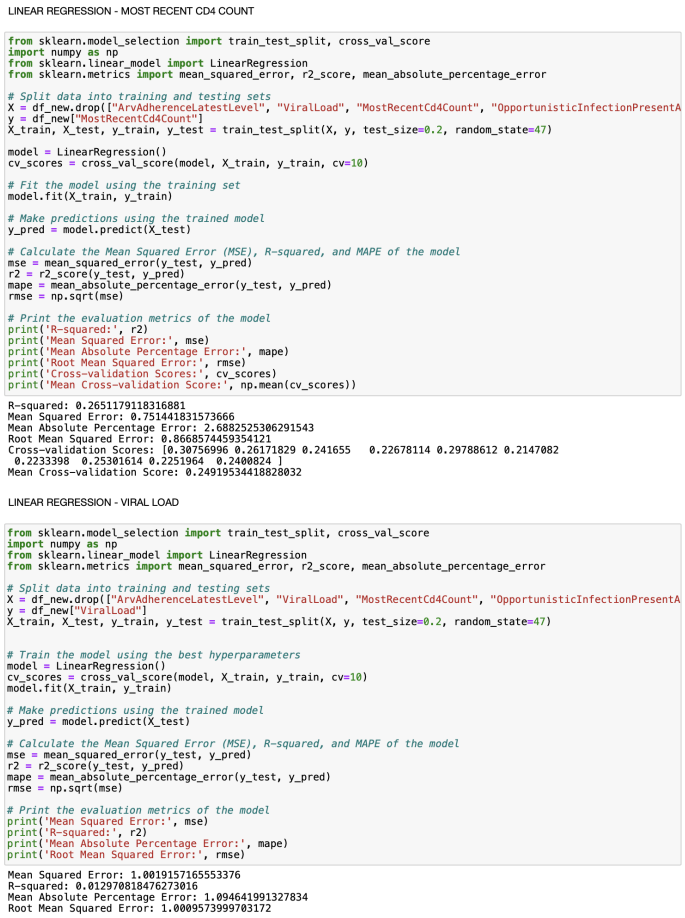


Fig. 36. ROC curve, confusion matrix and feature importance of gradient boost classifier

8.5 Linear Regression

We performed linear regression for outcome variables by splitting the data into training and testing data sets and with 10 folds cross validation and checked for model accuracy.



LINEAR REGRESSION - MOST RECENT CD4 COUNT

```
from sklearn.model_selection import train_test_split, cross_val_score
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

# Split data into training and testing sets
X = df_new.drop(["ArvAdherenceLatestLevel", "ViralLoad", "MostRecentCd4Count", "OpportunisticInfectionPresentA
y = df_new["MostRecentCd4Count"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

model = LinearRegression()
cv_scores = cross_val_score(model, X_train, y_train, cv=10)

# Fit the model using the training set
model.fit(X_train, y_train)

# Make predictions using the trained model
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE), R-squared, and MAPE of the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Print the evaluation metrics of the model
print('R-squared:', r2)
print('Mean Squared Error:', mse)
print('Mean Absolute Percentage Error:', mape)
print('Root Mean Squared Error:', rmse)
print('Cross-validation Scores:', cv_scores)
print('Mean Cross-validation Score:', np.mean(cv_scores))
```

```
R-squared: 0.2651179118316881
Mean Squared Error: 0.751441831573666
Mean Absolute Percentage Error: 2.6882525306291543
Root Mean Squared Error: 0.8668574459354121
Cross-validation Scores: [0.30756996 0.26171829 0.241655   0.22678114 0.29788612 0.2147082
 0.2233398  0.25301614 0.2251964  0.2400824 ]
Mean Cross-validation Score: 0.24919534418828032
```

LINEAR REGRESSION - VIRAL LOAD

```
from sklearn.model_selection import train_test_split, cross_val_score
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

# Split data into training and testing sets
X = df_new.drop(["ArvAdherenceLatestLevel", "ViralLoad", "MostRecentCd4Count", "OpportunisticInfectionPresentA
y = df_new["ViralLoad"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)


# Train the model using the best hyperparameters
model = LinearRegression()
cv_scores = cross_val_score(model, X_train, y_train, cv=10)
model.fit(X_train, y_train)

# Make predictions using the trained model
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE), R-squared, and MAPE of the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Print the evaluation metrics of the model
print('Mean Squared Error:', mse)
print('R-squared:', r2)
print('Mean Absolute Percentage Error:', mape)
print('Root Mean Squared Error:', rmse)
```

```
Mean Squared Error: 1.0019157165553376
R-squared: 0.012970818476273016
Mean Absolute Percentage Error: 1.094641991327834
Root Mean Squared Error: 1.0009573999703172
```

Fig. 37. Linear regression model for MostRecentCd4 count and viralload depicting the MSE, RMSE, MAPE and R-squared for the model

8.6 Decision tree Regressor

We performed Decision tree Regressor for outcome variables by splitting the data into training and testing data sets and with 10 folds cross validation and

checked for model accuracy.

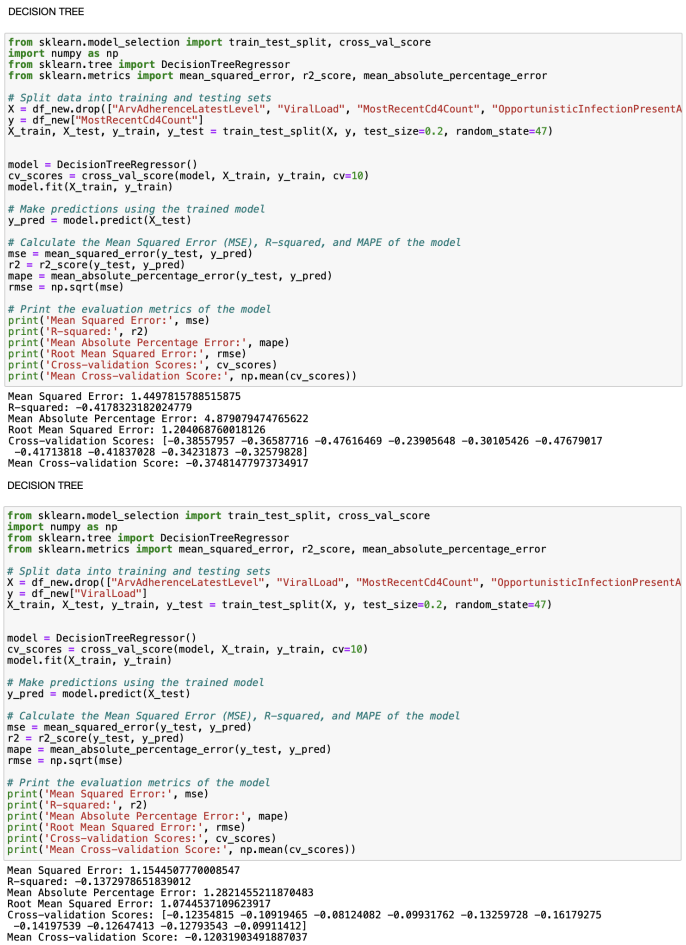


Fig. 38. Decision tree regression model for MostRecentCd4 count and viralload depicting the MSE, RMSE, MAPE and R-squared for the model

8.7 Random Forest Regressor

We performed Random Forest Regressor for outcome variables by splitting the data into training and testing data sets and with 10 folds cross validation and checked for model accuracy.

RANDOM FOREST

```
from sklearn.model_selection import train_test_split, cross_val_score
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

# Split data into training and testing sets
X = df_new.drop(["ArvAdherenceLatestLevel", "ViralLoad", "MostRecentCd4Count", "OpportunisticInfectionPresentA
y = df_new["MostRecentCd4Count"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)


model = RandomForestRegressor()
cv_scores = cross_val_score(model, X_train, y_train, cv=10)
model.fit(X_train, y_train)

# Make predictions using the trained model
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE), R-squared, and MAPE of the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Print the evaluation metrics of the model
print('Mean Squared Error:', mse)
print('R-squared:', r2)
print('Mean Absolute Percentage Error:', mape)
print('Root Mean Squared Error:', rmse)
print('Cross-validation Scores:', cv_scores)
print('Mean Cross-validation Score:', np.mean(cv_scores))
```

```
Mean Squared Error: 0.8702916913466755
R-squared: 0.14888718117673916
Mean Absolute Percentage Error: 3.456537662598192
Root Mean Squared Error: 0.9328942551793722
Cross-validation Scores: [0.17709303 0.16207268 0.07574007 0.16873059 0.19373021 0.09236586
 0.12092809 0.14982216 0.10292608 0.16351874]
Mean Cross-validation Score: 0.14069275074890436
```

RANDOM FOREST

```
from sklearn.model_selection import train_test_split, cross_val_score
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

# Split data into training and testing sets
X = df_new.drop(["ArvAdherenceLatestLevel", "ViralLoad", "MostRecentCd4Count", "OpportunisticInfectionPresentA
y = df_new["ViralLoad"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)


model = RandomForestRegressor()
cv_scores = cross_val_score(model, X_train, y_train, cv=10)
model.fit(X_train, y_train)

# Make predictions using the trained model
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE), R-squared, and MAPE of the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Print the evaluation metrics of the model
print('Mean Squared Error:', mse)
print('R-squared:', r2)
print('Mean Absolute Percentage Error:', mape)
print('Root Mean Squared Error:', rmse)
print('Cross-validation Scores:', cv_scores)
print('Mean Cross-validation Score:', np.mean(cv_scores))
```

```
Mean Squared Error: 1.0706055412481066
R-squared: -0.05469840791196012
Mean Absolute Percentage Error: 1.2138588567588402
Root Mean Squared Error: 1.0347007012890765
Cross-validation Scores: [-0.05979767 -0.05164663 -0.0298059  -0.03790782 -0.05557538 -0.06860553
 -0.05181814 -0.05595853 -0.05652479 -0.02825381]
Mean Cross-validation Score: -0.0495894202302515
```

Fig. 39. Random forest regression model for MostRecentCd4 count and viralload depicting the MSE, RMSE, MAPE and R-squared for the model

8.8 Gradient Boosting Regressor

We performed Gradient Boosting Regressor for outcome variables by splitting the data into training and testing data sets and with 10 folds cross validation and checked for model accuracy.

GRADIENT BOOSTING

```
from sklearn.model_selection import train_test_split, cross_val_score
import numpy as np
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

# Split data into training and testing sets
X = df_new.drop(["ArvAdherenceLatestLevel", "ViralLoad", "MostRecentCd4Count", "OpportunisticInfectionPresentA
y = df_new["MostRecentCd4Count"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

model = GradientBoostingRegressor()
cv_scores = cross_val_score(model, X_train, y_train, cv=10)
model.fit(X_train, y_train)

# Make predictions using the trained model
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE), R-squared, and MAPE of the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Print the evaluation metrics of the model
print('Mean Squared Error:', mse)
print('R-squared:', r2)
print('Mean Absolute Percentage Error:', mape)
print('Root Mean Squared Error:', rmse)
print('Cross-validation Scores:', cv_scores)
print('Mean Cross-validation Score:', np.mean(cv_scores))
```

```
Mean Squared Error: 0.7512943672120832
R-squared: 0.26526212647808806
Mean Absolute Percentage Error: 2.831321669925598
Root Mean Squared Error: 0.866772384892414
Cross-validation Scores: [0.3103769  0.27762065 0.24321765 0.27332385 0.30565989 0.21163581
 0.24613861 0.2752422  0.21916109 0.266594  ]
Mean Cross-validation Score: 0.26289706527980516
```

GRADIENT BOOSTING

```
from sklearn.model_selection import train_test_split, cross_val_score
import numpy as np
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

X = df_new.drop(["ArvAdherenceLatestLevel", "ViralLoad", "MostRecentCd4Count", "OpportunisticInfectionPresentA
y = df_new["ViralLoad"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

model = GradientBoostingRegressor()
cv_scores = cross_val_score(model, X_train, y_train, cv=10)
model.fit(X_train, y_train)

# Make predictions using the trained model
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE), R-squared, and MAPE of the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Print the evaluation metrics of the model
print('Mean Squared Error:', mse)
print('R-squared:', r2)
print('Mean Absolute Percentage Error:', mape)
print('Root Mean Squared Error:', rmse)
print('Cross-validation Scores:', cv_scores)
print('Mean Cross-validation Score:', np.mean(cv_scores))
```

```
Mean Squared Error: 1.0000191031430443
R-squared: 0.014839251871488046
Mean Absolute Percentage Error: 1.1249501340471002
Root Mean Squared Error: 1.0000095515259064
Cross-validation Scores: [0.009664   0.02104916 0.01605704 0.01763693 0.03157062 0.01446245
 0.01981584 0.02203569 0.01336473 0.01804735]
Mean Cross-validation Score: 0.018370381274380433
```

Fig. 40. Gradient boosting regression model for MostRecentCd4 count and viralload depicting the MSE, RMSE, MAPE and R-squared for the model

8.9 Stacking Method

As the individual models didn't performed well. We opted for Stacking method to find out the accuracy of the meta model.

Fig. 41. Stacking method for mostRecentCD4 model to improve the model performance

Fig. 42. Stacking method for ViralLoad model to improve the model performance

9 Summary Findings

We reject the null hypothesis i.e.) there is an association between demographics, types of regimens, and clinical stages of HIV with viral load progression, opportunistic infections, suppression of CD4 count, and adherence to ART regimens in HIV-positive patients.

For the outcome variables - Arv adherence latest level and Opportunsitic infection present at last visit - Random Forest Classifier performed well based on Accuracy and AUC-ROC curves.

For the outcome variables - Most recent CD4 count and Viral Load - Gradient Boosting and Linear Regression performed well based on R2 and MAPE. Stacking gave good results when compared to the individual models for the outcome variables - Most recent CD4 count and Viral Load.

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017