**Western Governors University (WGU)**

**D209 Data Mining 1**

**Task 2: Predictive Analysis**

**Natallia Zimnitskaya | ID: 012247127**

**Master of Science, Data Analytics**

**Part I: Research Question**

A1. Is it possible to predict the patient's readmission and identify key contributors using a decision tree? It helps reduce costs by identifying high-risk patients and implementing targeted interventions, improving patient care and satisfaction.

A2. One goal for this data analysis could be to create a decision tree that can accurately predict the likelihood of patient readmission within 30 days of discharge. This goal is well-supported by the available data, which includes variables such as Area, Age, Gender, VitD\_levels, Initial\_admin, HighBlood, Stroke, Complication\_risk, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Services, Initial\_days, and Additional\_charges. Decision trees are effective for this type of analysis because they can handle numerical and categorical data and provide clear, interpretable results highlighting the most important factors influencing readmission.

**Part II: Method Justification**

B1. A decision tree is a model used in machine learning that can handle both categorical and numerical data to make predictions based on data. It works like a flowchart, where each decision point, or “node,” represents a question about one of the features in the data (GeeksforGeeks, 2024c). For example, a node might ask if a patient’s age exceeds a certain number. Depending on the answer, the tree branches off into different paths, leading to further questions or a final prediction. The endpoints of these paths, called “leaves,” represent the predicted outcome, such as whether a patient will be readmitted to the hospital. Decision trees are popular because they are easy to understand and interpret, making it clear how different factors contribute to the final prediction. This analysis can help healthcare providers implement targeted interventions for high-risk patients, potentially reducing readmission rates.

B2. One important assumption when using decision trees is that the data should ideally have no outliers. Decision trees are sensitive to extreme values, which can disproportionately influence how the tree is constructed (Analytics Vidhya, 2024). Outliers can lead to splits that do not generalize to new data, reducing the model’s overall performance. To handle outliers effectively, it is crucial to preprocess the data by either removing or transforming these extreme values.

B3. List of the Python packages and libraries chosen for the analysis:

* **numpy**: Provides support for large, multi-dimensional arrays and matrices, essential for numerical operations and handling arrays.
* **pandas**: Crucial for data manipulation and analysis, offering data structures like DataFrames to efficiently handle and analyze structured data.
* **seaborn**: It builds on matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics, which are useful for visualizing data distributions and relationships.
* **matplotlib**: A comprehensive library for creating static, animated, and interactive visualizations, helping to understand data visually.
* **statsmodels**: Provides classes and functions for estimating many different statistical models and conducting statistical tests, essential for statistical analysis.
* **scipy**: Used for scientific and technical computing, including tasks such as optimization, integration, and interpolation.
* **sklearn.preprocessing:** This includes tools for preprocessing data, such as scaling and encoding, which are necessary steps before model building.
* **sklearn.model\_selection**: Provides tools for splitting data into training and testing sets, performing cross-validation, and tuning model parameters using techniques like **GridSearchCV**.
* **sklearn.feature\_selection**: Offers methods for selecting the most relevant features for the model, such as **SelectKBest** and **f\_classif**.
* **sklearn.tree**: Contains the **DecisionTreeClassifier** for building decision tree models and plot\_tree for visualizing them.
* **sklearn.metrics**: Provides functions for evaluating the performance of machine learning models, including confusion matrix, ROC-AUC score, and classification report.

**Part III: Data Preparation**

C1. One important data preprocessing goal relevant to using a decision tree is handling missing values (Analytics Vidhya, 2024). Missing data can lead to inaccurate splits and poor model performance. By addressing missing values, either through imputation (filling in missing values with mean, median, mode, or other methods) or removing rows/columns with missing data, we ensure that the decision tree can make accurate and reliable splits based on complete information.

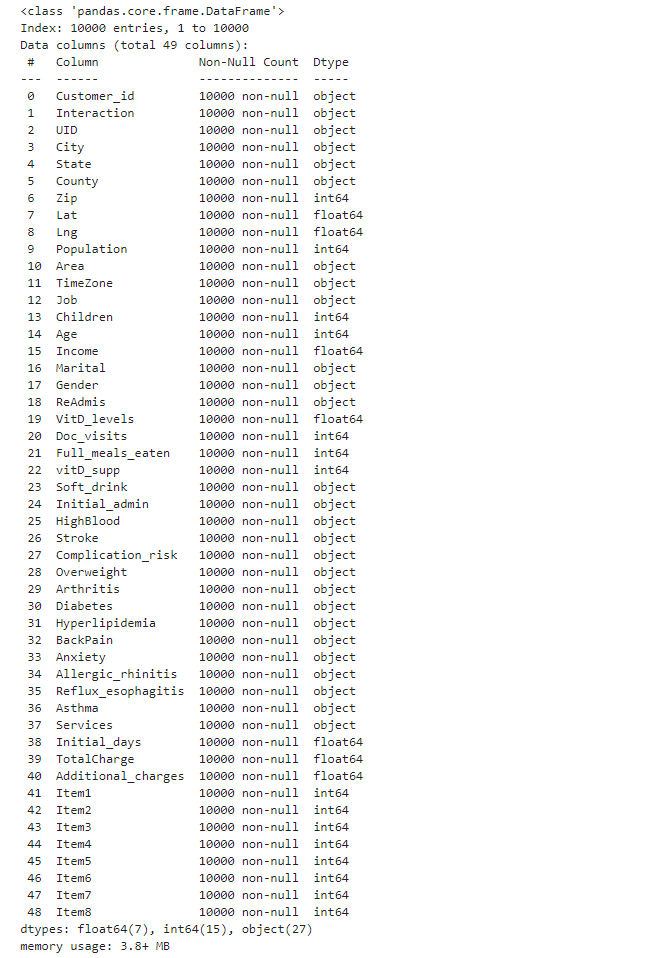
C2. All variables for my research question are listed in the tab below.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Data type | Description | Example (row #5) |
| Area | Categorical | Area type (rural, urban, suburban) | Rural |
| Age | Numeric | Patient's age | 22 |
| Gender | Categorical | Self-identification as male, female, or nonbinary | Female |
| ReAdmis | Categorical | Readmission within a month of release (yes, no) | No |
| VitD\_levels | Numeric | Vit D levels in ng/ml | 16.87052 |
| Initial\_admin | Categorical | Type of initial admission (emergency, elective, observation) | Elective Admission |
| HighBlood | Categorical | A patient has a high blood pressure (yes, no) | No |
| Stroke | Categorical | A patient has had a stroke (yes, no) | No |
| Complication\_risk | Categorical | Level of complication risk (high, medium, low) | Low |
| Overweight | Categorical | The patient is overweight (yes, no) | 0 |
| Arthritis | Categorical | A patient has arthritis (yes, no) | No |
| Diabetes | Categorical | A patient has diabetes (yes, no) | No |
| Hyperlipidemia | Categorical | A patient has hyperlipidemia (yes, no) | Yes |
| BackPain | Categorical | A patient has a chronic back pain (yes, no) | No |
| Services | Categorical | Services received while hospitalized (blood work, intravenous, CT scan, MRI) | CT scan |
| Initial\_days | Numeric | Number of days in the hospital | 1.254807 |
| Additional\_charges | Numeric | Charge for miscellaneous procedures | 3716.526 |

C3. The primary goals for data cleaning are to ensure the dataset is accurate, consistent, and ready for analysis. This involves handling duplicates and missing values, detection and treatment of outliers, correcting errors, and standardizing formats.

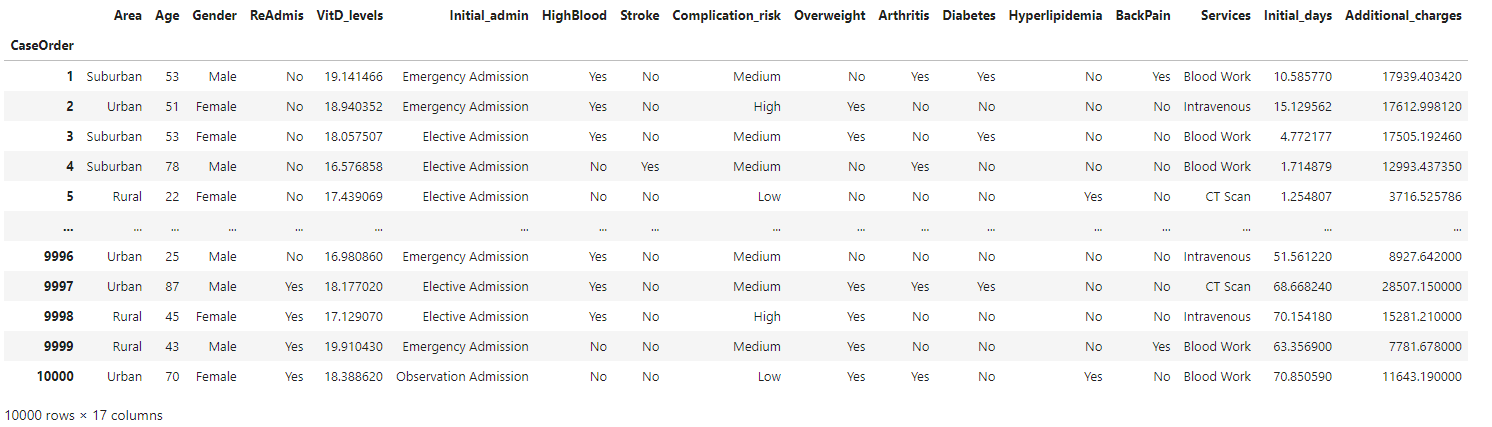
The first step is to import required packages and libraries, then using the pd.read\_csv() function, load the medical\_clean.csv file onto the Jupyter Notebook, and lastly, with df.info() function, get the information about the dataset.



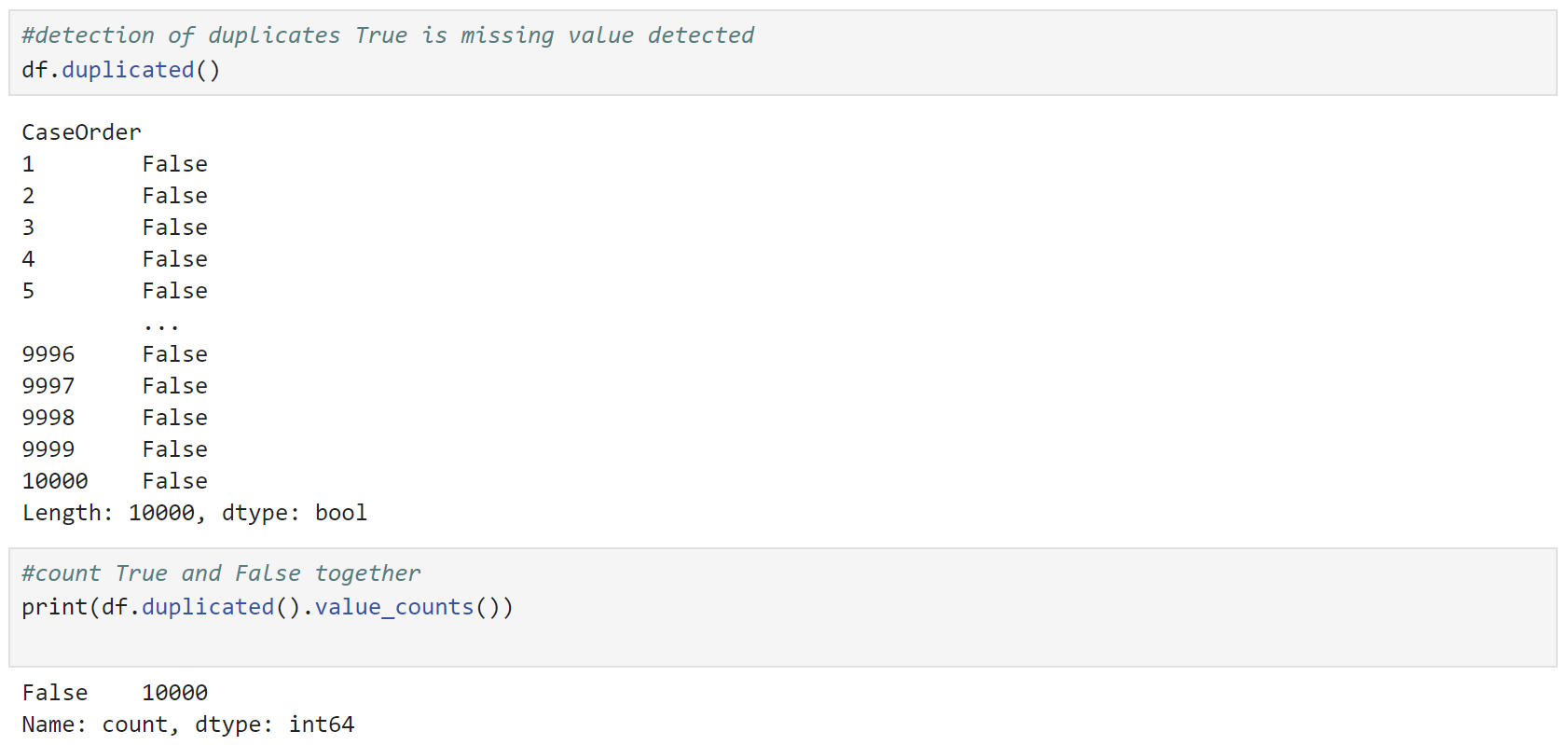


The next step is to drop variables unrelated to the research question ('Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'TimeZone', 'Job', 'Income', 'Children', 'Marital', 'Doc\_visits', 'Full\_meals\_eaten', 'vitD\_supp', 'Soft\_drink', 'Anxiety', 'Allergic\_rhinitis', 'Reflux\_esophagitis', 'Asthma', 'TotalCharge', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8') using the drop() function. With the pd.set\_option() function, visually inspect the dataset to facilitate exploration and spot problems.



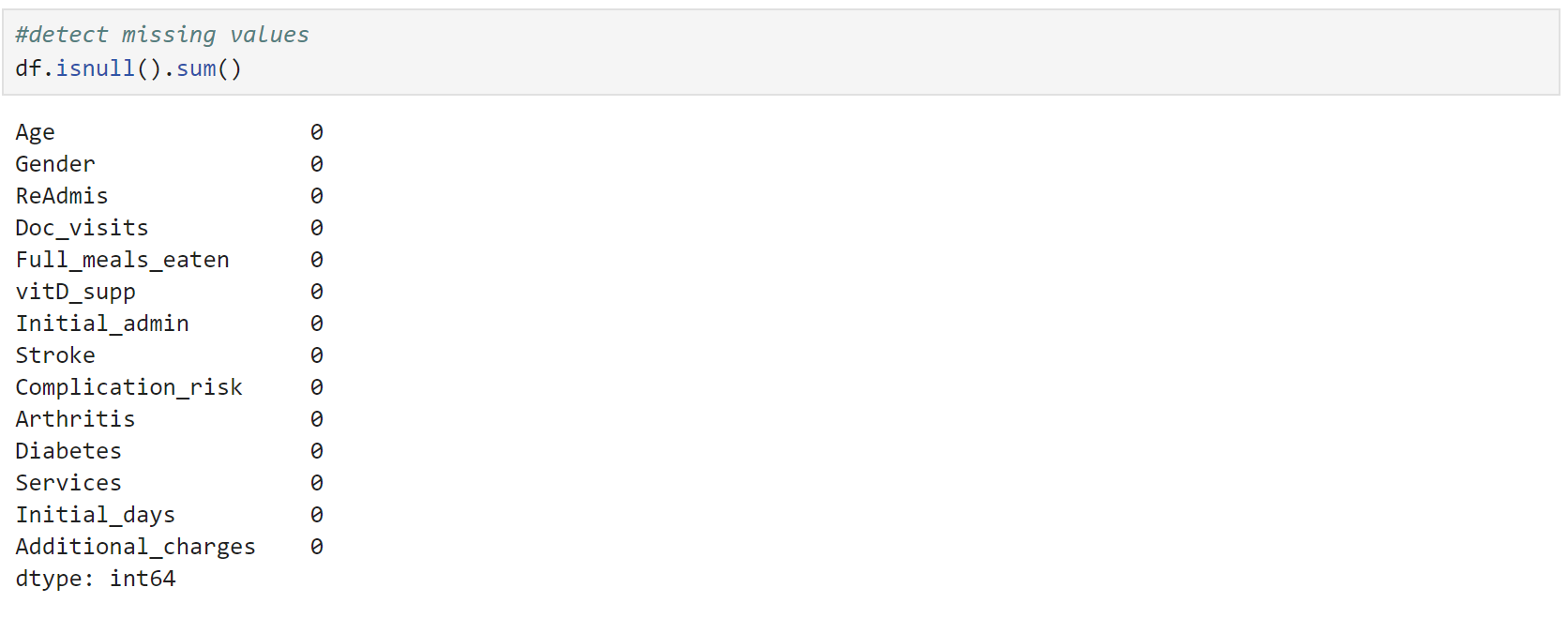


For the detection of duplicates, the duplicated() and duplicated().value\_counts() functions were used.



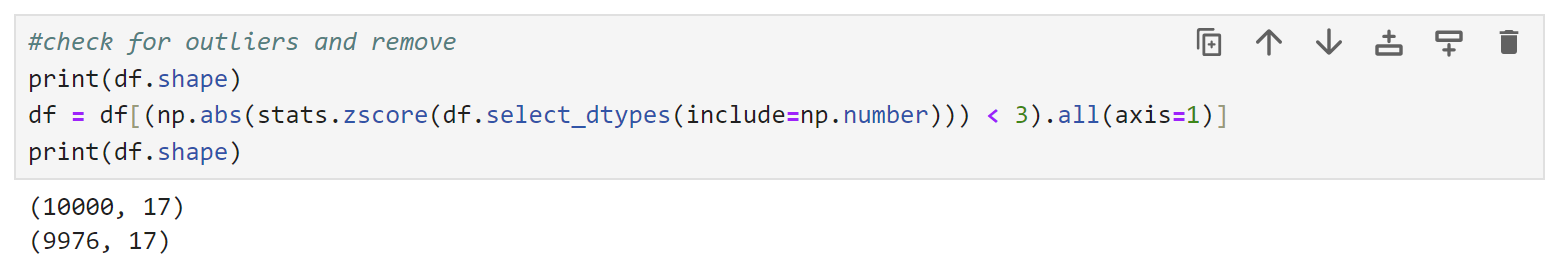
There are no duplicates in the dataset.

Next, the isnull().sum function was used to detect the missing values (for qualitative and quantitative variables).



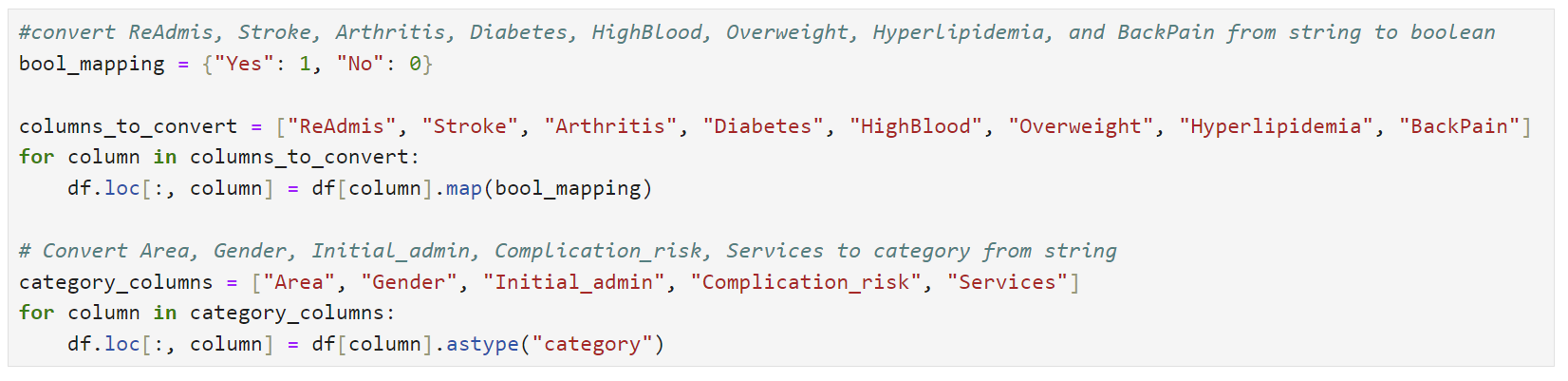
There are no missing values in the dataset.

Next, using the z-score method, outliers were checked and removed.

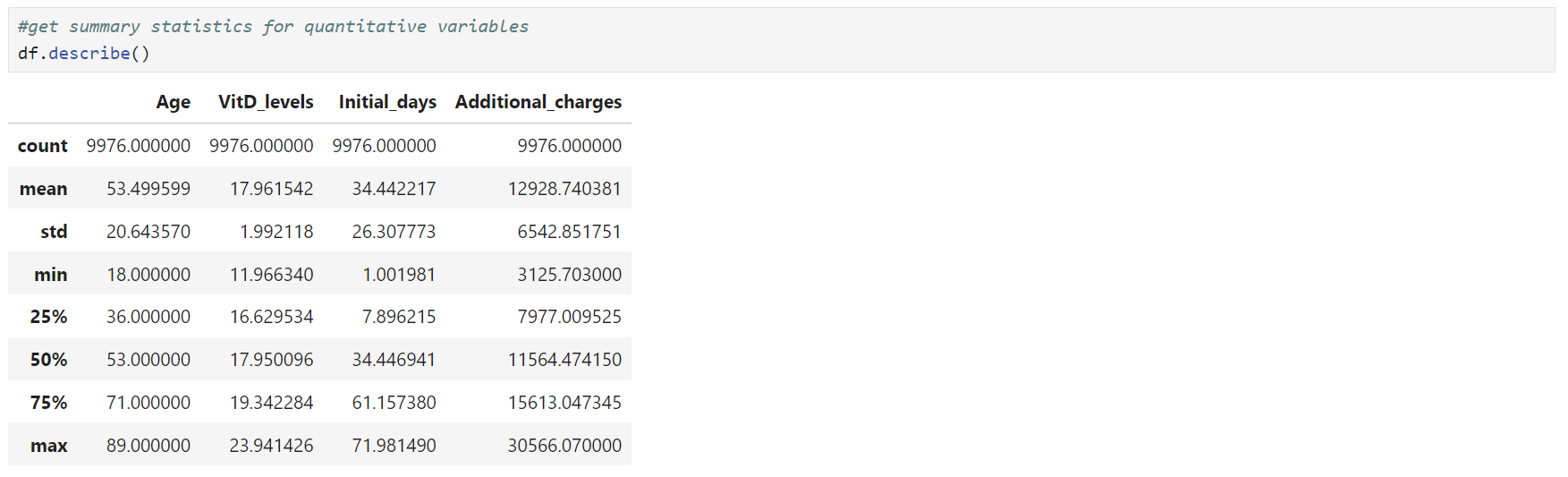


24 rows are removed from the dataset.

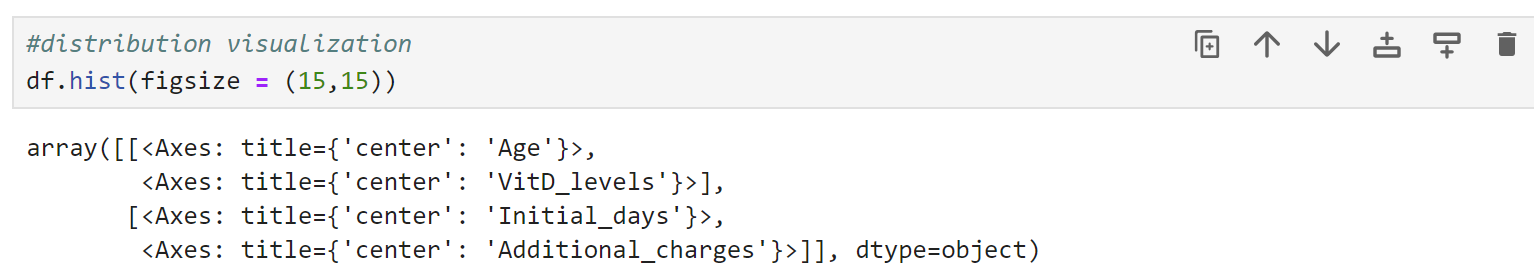
In the next step, I will convert ReAdmis, Stroke, Arthritis, Diabetes, HighBlood, Overweight, Hyperlipidemia, BackPain from string to boolean, and Area, Gender, Initial\_admis, Complicatiom\_risk, and Services to category.

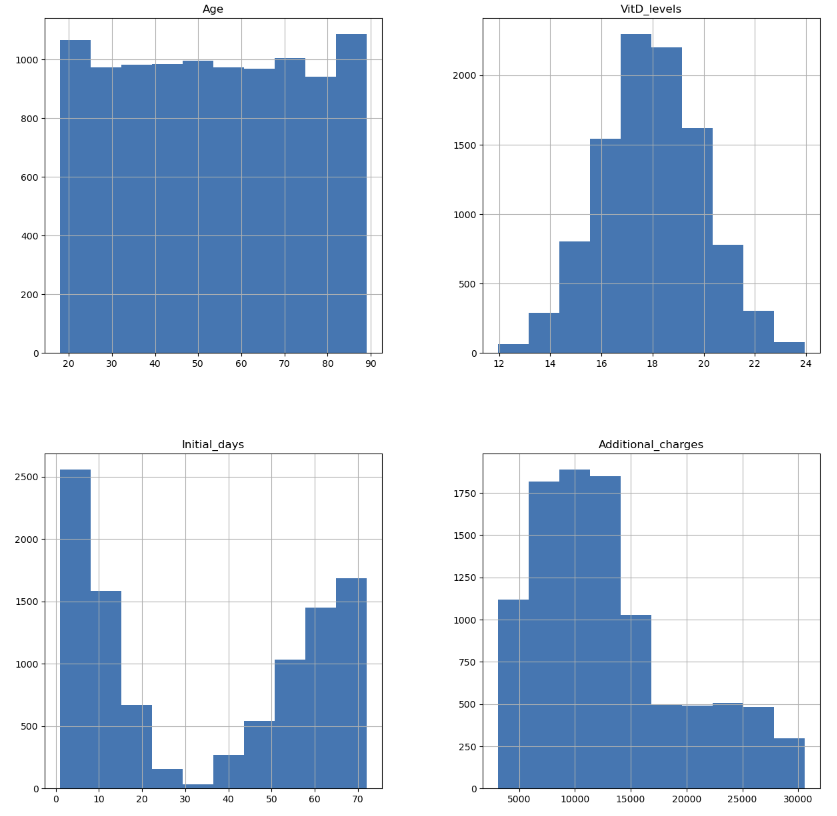


Next is data exploration with summary statistics for all continuous variables.

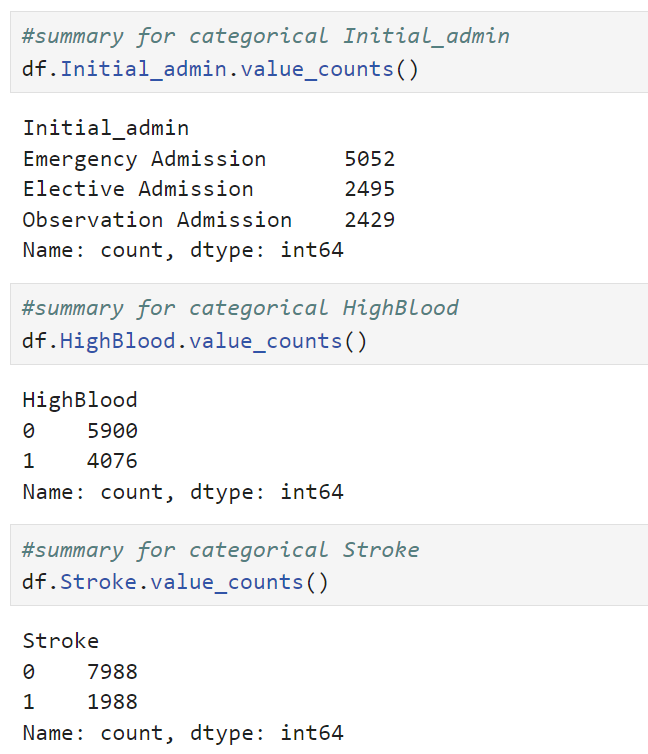
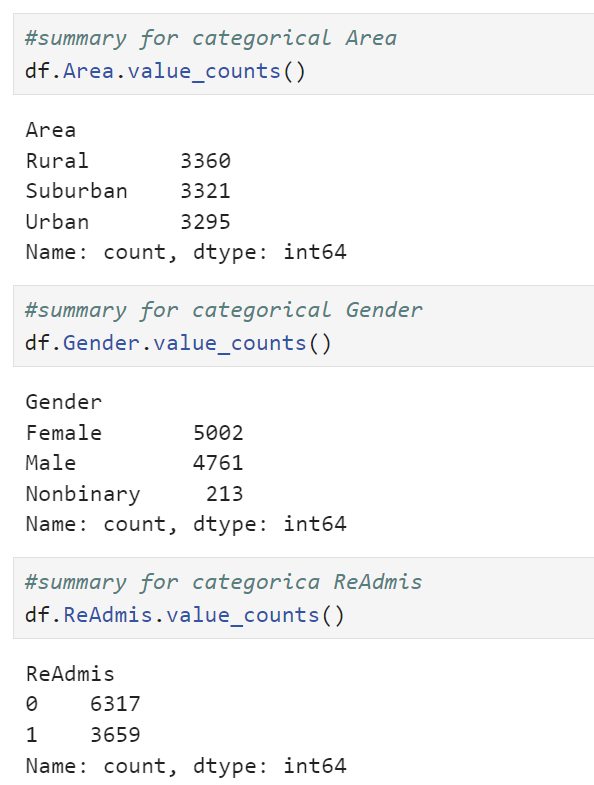


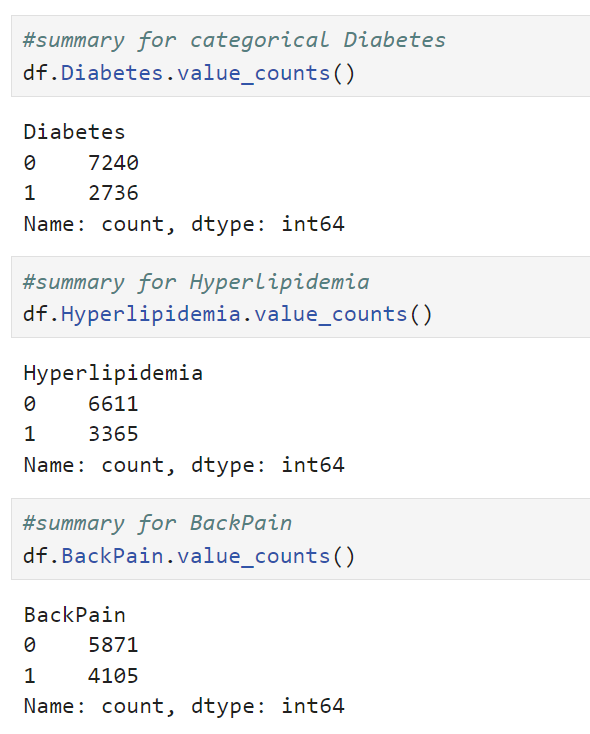
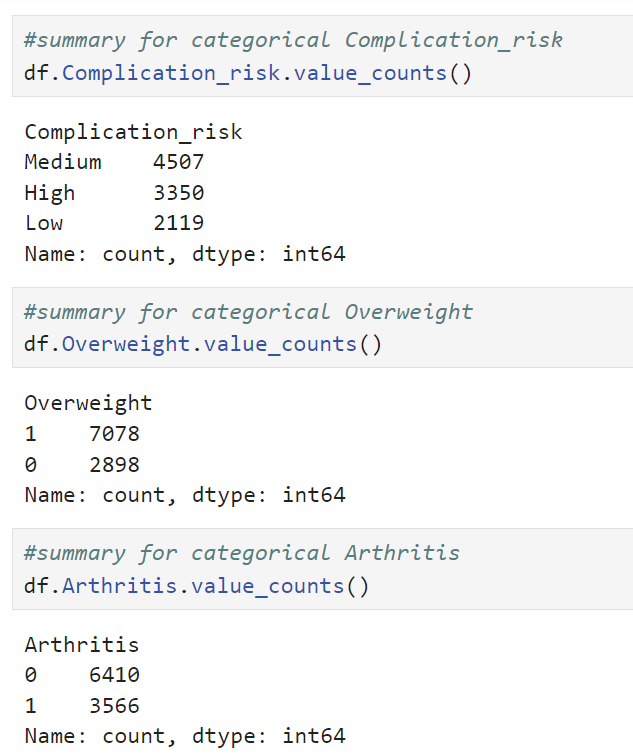
The average age of patients is about 53.5 years, ranging from 18 to 89. The average Vitamin D level is approximately 18, ranging from 12 to 24. Patients typically stay in the hospital for an average of 34.4 days, but this can vary widely from just over 1 day to nearly 72 days. The additional charges patients incur average around $12928, with a minimum of about $3126 and a maximum of approximately $30566.

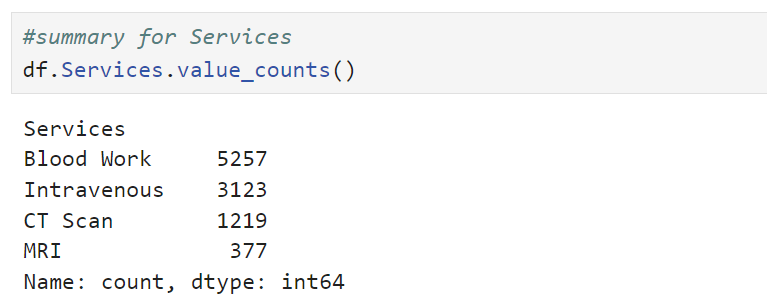




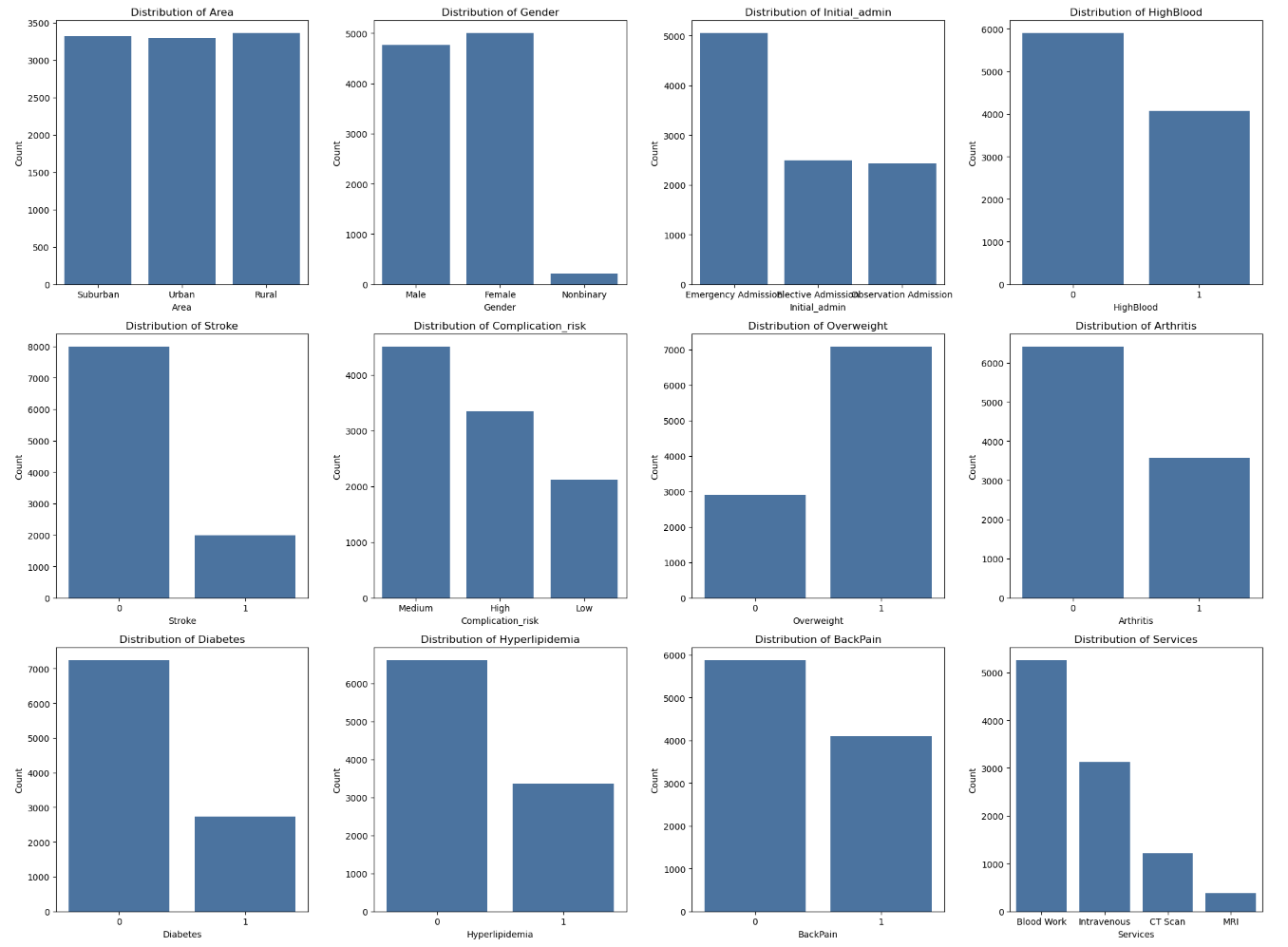
Here is a summary for categorical variables:









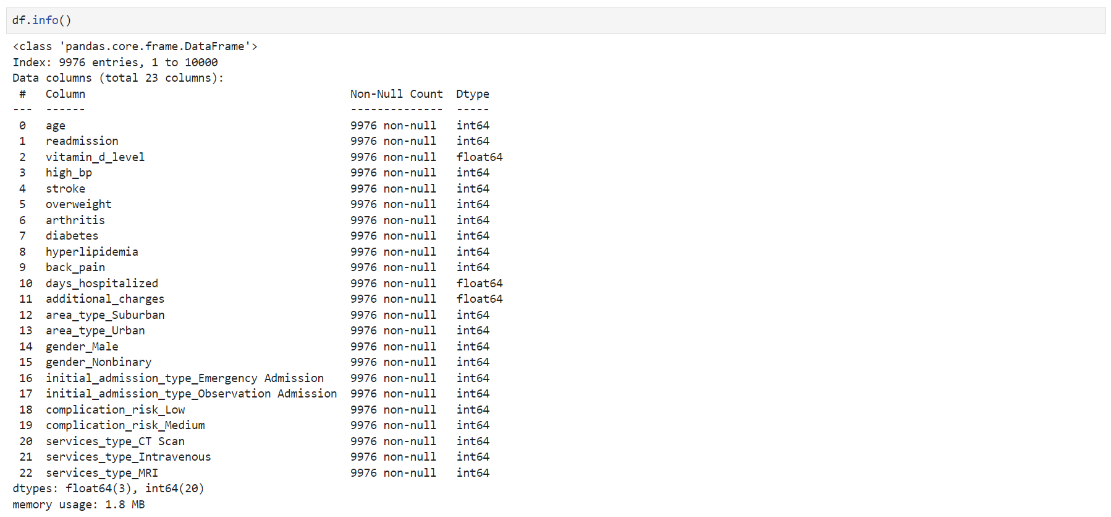


Next, I will rename columns to appropriate Pythonic names ('area\_type', 'age', 'gender', 'readmission', 'vitamin\_d\_level', 'initial\_admission\_type', 'high\_bp', 'stroke', 'complication\_risk', 'overweight', 'arthritis', 'diabetes', 'hyperlipidemia', 'back\_pain', 'services\_type', 'days\_hospitalized', 'additional\_charges')



One-hot encoding was performed using the pd.get\_dummies() function. In my decision tree model, I used drop\_first=True when creating dummy variables to simplify the model and improve interpretability (*The Decision Tree Classifier & Supervised Classification Models*, 2022). By dropping the first category, I reduced the number of dummy variables, which helps in making the decision tree easier to understand. Although decision trees are not affected by multicollinearity, this approach avoids redundancy. Additionally, it helps manage the dataset size and computational efficiency, which are important practical considerations for this analysis. Using df\_regress.replace({True: 1, False: 0}), I converted boolean values to numeric (1 and 0) and applied pd.to\_numeric to convert all columns to numeric.





C4. The prepared df dataset was saved to a new csv file. Please see the attached “D209task2\_clean.csv” file.

******

**Part IV: Analysis**

D1. The data was split into training (80%) and testing (20%) sets. This is to ensure the model has similar accuracy when predicting unseen data.



Please see the attached ‘task2\_Xtrain.csv’, ‘task2\_Xtest.csv’, ‘task2\_ytrain.csv, and ‘task2\_ytest.csv’ files.

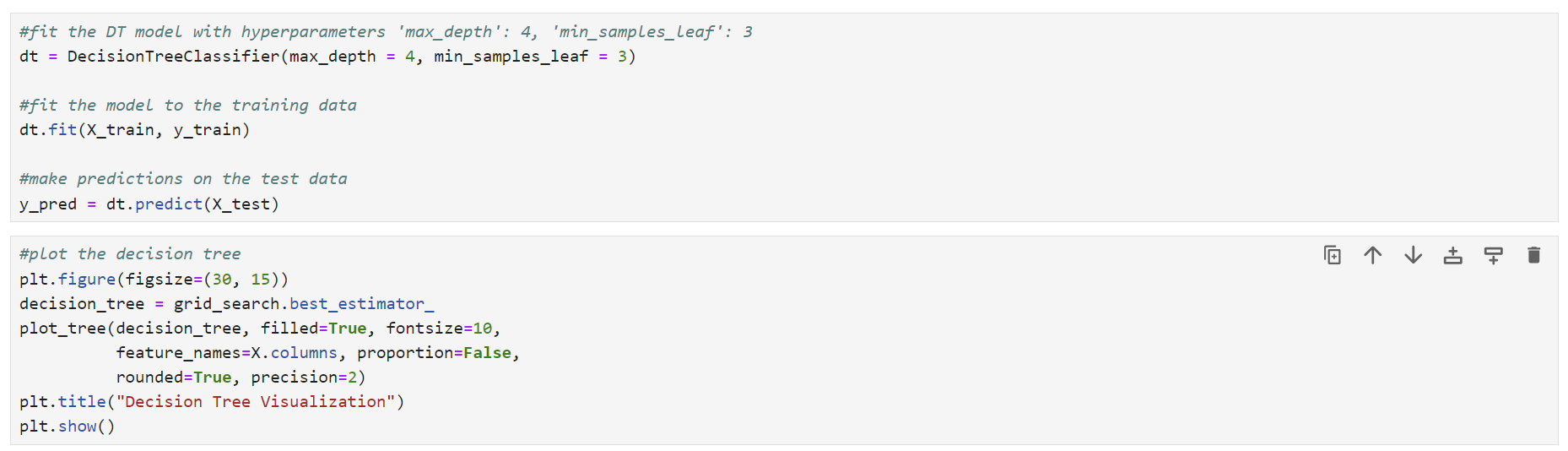


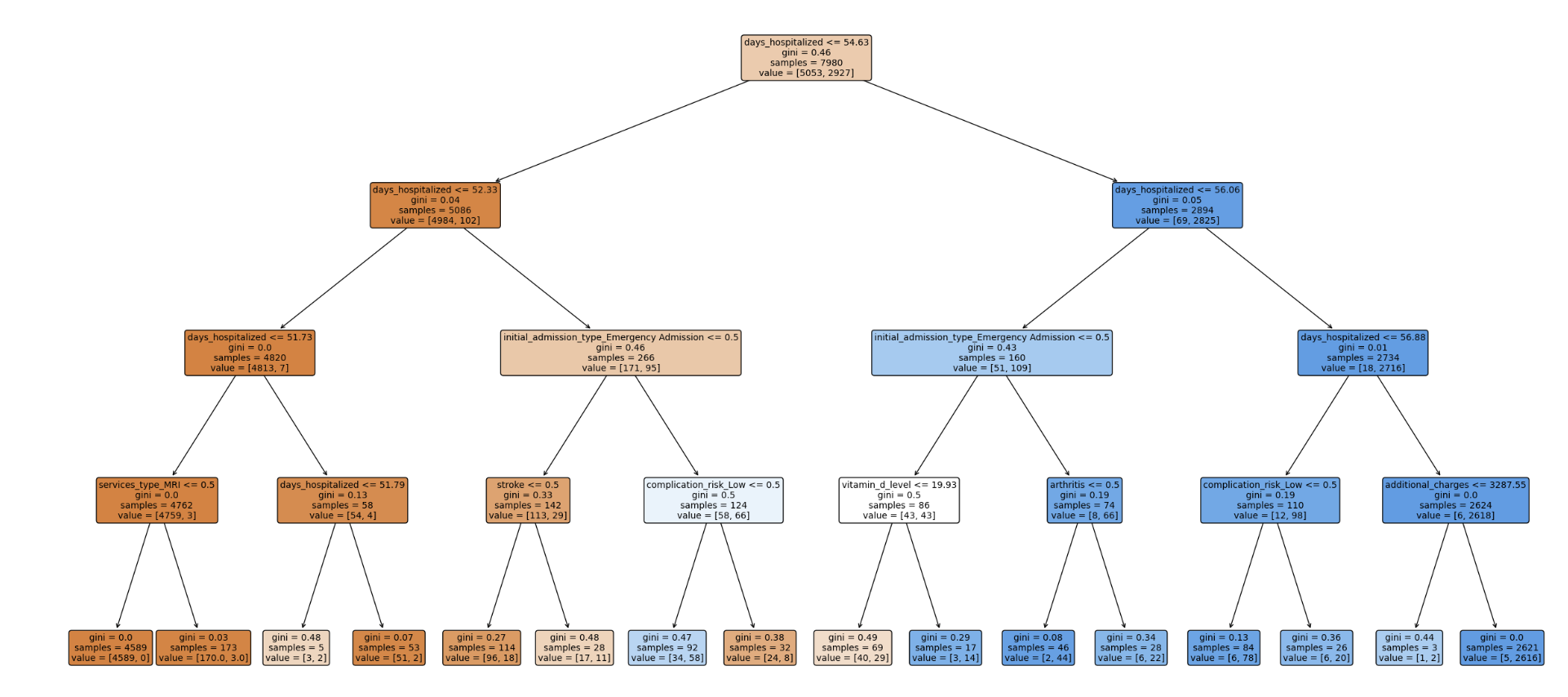
D2. To create a decision tree classification model using the DecisionTreeClassifier, I need to identify the optimal parameters for max\_depth and min\_samples\_leaf. The max\_depth parameter controls the number of levels the tree can grow, with greater depth increasing complexity and the risk of overfitting. The min\_samples\_leaf parameter sets the minimum number of samples required to split a leaf node, ensuring a certain number of samples remain on both sides of a split. GridSearchCV will evaluate the performance of each configuration and identify the set of parameters along with the calculation of the model's accuracy and AUC.

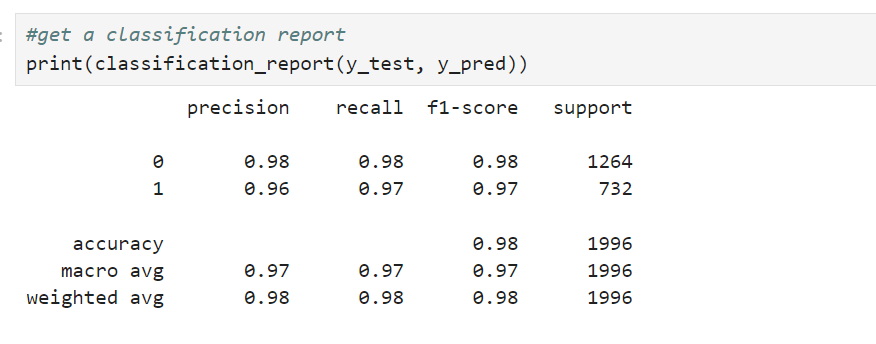


The best hyperparameters found were max\_depth of 4 and min\_samples\_leaf of 3. The model’s performance was evaluated, resulting in an accuracy of 0.9754509018036072 and an AUC of 0.9745797883378294. These metrics indicate that the model is highly effective at distinguishing between classes, making it a robust choice for the classification task.

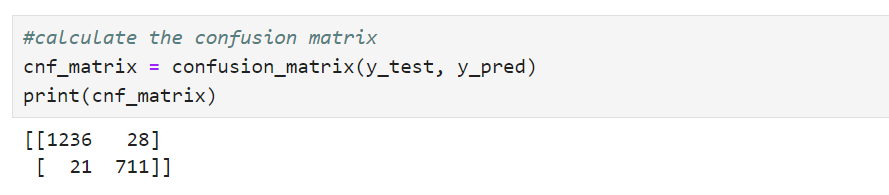
The decision tree model was subsequently trained using the identified parameters, and the decision tree plot was generated.

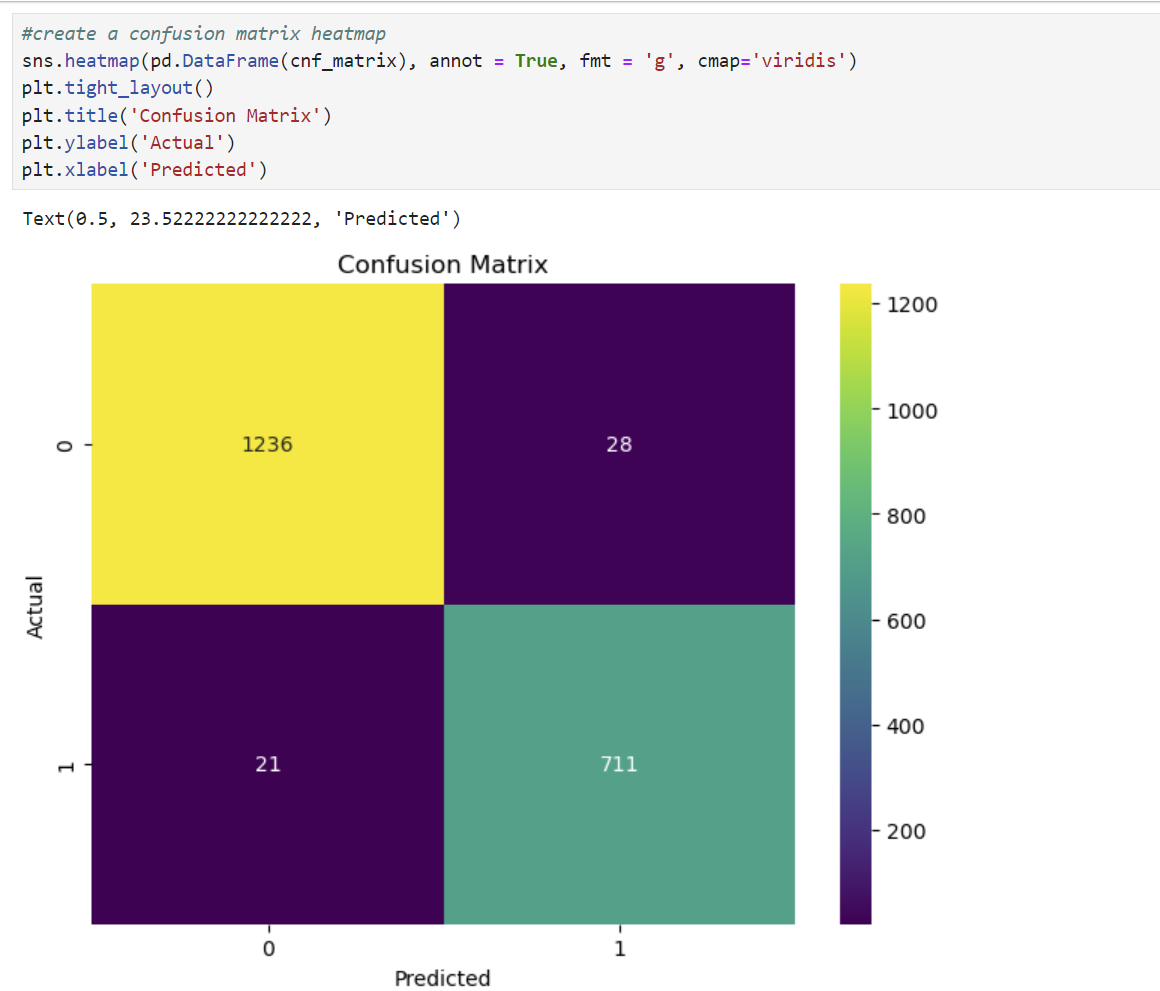




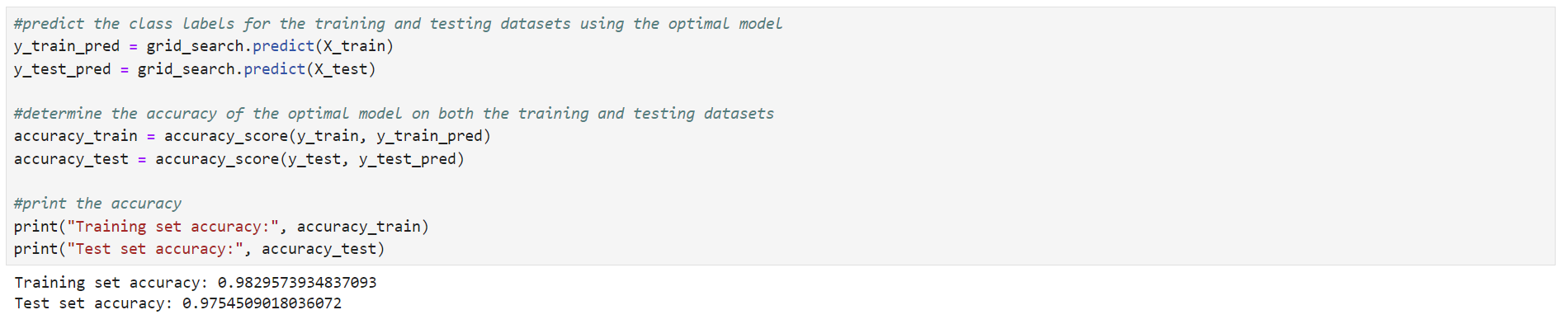
A classification report was subsequently utilized to see how well the model performed. 

For class 0, the precision is 0.98, meaning 98% of the predicted class 0 instances are correct. The recall is 0.98, indicating that 98% of the actual class 0 instances were correctly identified. The f1-score, which balances precision and recall, is 0.98. For class 1, the precision is 0.96, the recall is 0.97, and the f1-score is 0.97. The model's overall accuracy is 0.9755, predicting 97.55% of the instances.

A confusion matrix was subsequently generated. It is a valuable tool for assessing the performance of a binary classification model and understanding the types of errors the model is making. 

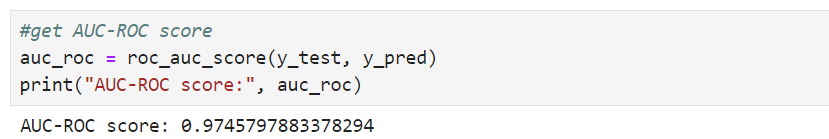


The confusion matrix revealed that the model correctly classified 1236 instances as negative and incorrectly classified 28 instances as positive. Additionally, the model incorrectly classified 21 instances as negative while correctly identifying 711 as positive.

Calculation of the accuracy of training and test sets. 

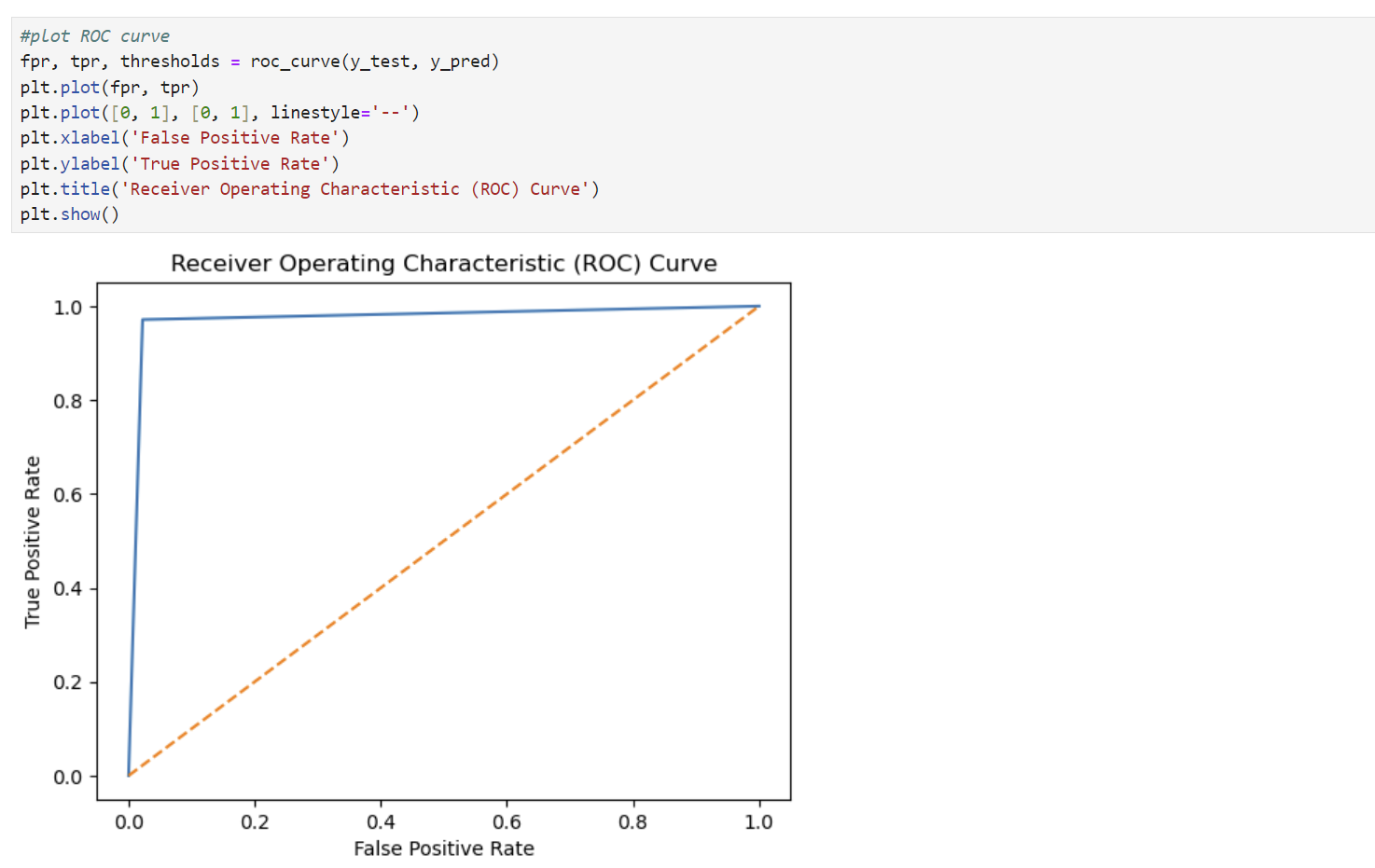
The model achieved an accuracy of 0.9829573934837093 on the training set, meaning it correctly predicted approximately 98.3% of the instances. This high accuracy suggests that the model has learned the patterns in the training data very well. On the test set, the model’s accuracy was slightly lower at 0.9754509018036072, indicating that it correctly predicted about 97.70% of the instances. This slight drop in accuracy on the test set is expected and shows that the model generalizes well to new, unseen data, maintaining a high level of performance.

The AUC score is a way to measure how good a classification model is at distinguishing between different classes. An AUC score of 1 indicates perfect classification, while a score of 0.5 suggests no better performance than random guessing (Bobbitt, 2021b).



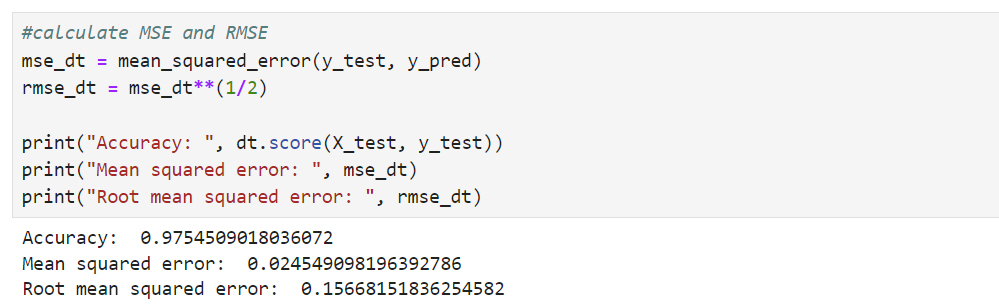
An AUC-ROC score of 0.9745797883378294 indicates that the model has a very high ability to differentiate between the positive and negative classes. The closer the score is to 1, the better the model predicts true positives and negatives while minimizing false positives and false negatives.

The ROC curve was created to illustrate the performance of a binary classification model by plotting the TP rate (sensitivity) against the FP rate (1-specificity) at various threshold settings.



This shape indicates that the model performs exceptionally well, quickly achieving a high TP rate with a low FP rate.

Calculation of MSE and RMSE for model prediction. The MSE measures the average squared difference between the actual and predicted values, and the RMSE is the square root of the MSE and provides a measure of the average error in the same units as the target variable.



D3. Please see the attached “D209\_Task2.ipybn” file with annotated code.

#define the hyperparameter grid

param\_grid = {

'max\_depth' : [2, 3, 4, 5],

'min\_samples\_leaf' : [1, 2, 3, 4]

}

#create a decision tree classifier

clf = DecisionTreeClassifier()

#perform grid search using 5-fold cross-validation

grid\_search = GridSearchCV(clf, param\_grid, cv=5)

#train the model on the training data

grid\_search.fit(X\_train, y\_train)

#predict the class labels of the testing data using the best model

y\_pred = grid\_search.predict(X\_test)

#calculate the accuracy and AUC of the best model

accuracy = accuracy\_score(y\_test, y\_pred)

auc = roc\_auc\_score(y\_test, y\_pred)

#print accuracy and auc

print("Accuracy:", accuracy)

print("AUC:", auc)

#print the best hyperparameters

print("Best hyperparameters:", grid\_search.best\_params\_)

#fit the DT model with hyperparameters 'max\_depth': 4, 'min\_samples\_leaf': 3

dt = DecisionTreeClassifier(max\_depth = 4, min\_samples\_leaf = 3)

#fit the model to the training data

dt.fit(X\_train, y\_train)

#make predictions on the test data

y\_pred = dt.predict(X\_test)

#plot the decision tree

plt.figure(figsize=(30, 15))

decision\_tree = grid\_search.best\_estimator\_

plot\_tree(decision\_tree, filled=True, fontsize=10,

feature\_names=X.columns, proportion=False,

rounded=True, precision=2)

plt.title("Decision Tree Visualization")

plt.show()

#get a classification report

print(classification\_report(y\_test, y\_pred))

#calculate the confusion matrix

cnf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(cnf\_matrix)

#create a confusion matrix heatmap

sns.heatmap(pd.DataFrame(cnf\_matrix), annot = True, fmt = 'g', cmap='viridis')

plt.tight\_layout()

plt.title('Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

#predict the class labels for the training and testing datasets using the optimal model

y\_train\_pred = grid\_search.predict(X\_train)

y\_test\_pred = grid\_search.predict(X\_test)

#determine the accuracy of the optimal model on both the training and testing datasets

accuracy\_train = accuracy\_score(y\_train, y\_train\_pred)

accuracy\_test = accuracy\_score(y\_test, y\_test\_pred)

#print the accuracy

print("Training set accuracy:", accuracy\_train)

print("Test set accuracy:", accuracy\_test)

#get AUC-ROC score

auc\_roc = roc\_auc\_score(y\_test, y\_pred)

print("AUC-ROC score:", auc\_roc)

#plot ROC curve

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)

plt.plot(fpr, tpr)

plt.plot([0, 1], [0, 1], linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.show()

#calculate MSE and RMSE

mse\_dt = mean\_squared\_error(y\_test, y\_pred)

rmse\_dt = mse\_dt\*\*(1/2)

print("Accuracy: ", dt.score(X\_test, y\_test))

print("Mean squared error: ", mse\_dt)

print("Root mean squared error: ", rmse\_dt)

**Part V: Data Summary and Implications**

E1. The accuracy of my prediction model is 0.9754509018036072, which means that the model correctly predicts approximately 97.6% of the instances in the test set. This high accuracy indicates that the model is performing very well. The MSE is 0.024549098196392786 and measures the average squared difference between the actual and predicted values. A lower MSE indicates better model performance, and this model's errors are quite small.

E2. This decision tree prediction analysis demonstrates great results, with an accuracy of 97.6% and an AUC-ROC score of 0.98, indicating high model effectiveness. The model’s precision, recall, and f1-scores for both classes are high (Class 0 has a precision of 0.98, recall of 0.98, and f1-score of 0.98, and Class 1 has a precision of 0.96, recall of 0.97, and f1-score of 0.97). This reflects balanced performance in distinguishing between the two classes. The best hyperparameters identified are a max depth of 4 and a minimum of 3 samples per leaf, which helps avoid overfitting while capturing necessary data complexity. The low MSE 0.024549098196392786 and RMSE 0.15668151836254582 further confirm the model’s accuracy and reliability. This robust and well-optimized decision tree model provides valuable insights for further analysis and decision-making.

E3. One limitation of this data analysis is the potential for overfitting despite the model’s high accuracy and performance metrics. While the decision tree model with a max depth of 4 and a minimum of 3 samples per leaf helps mitigate this risk, the model might still capture noise in the training data, leading to less generalizable results on new, unseen data. It is particularly important to consider whether the dataset is sufficiently large or diverse, as the model’s performance might degrade when applied to different populations or conditions not represented in the training set. Regular validation and testing on varied datasets can help address this limitation.

E4. Based on this decision tree analysis, predicting patient readmission and identifying key contributors is possible.

Here are some recommended courses of action for real-world organizational situations:

* Integrating the decision tree model into a hospital's information system to predict real-time patient readmissions. This can help identify high-risk patients early and take preventive measures.
* Monitor the model's performance and update it with new data to ensure its accuracy and relevance.
* Since days\_hospitalized is highly significant, consider developing targeted interventions for patients with longer hospital stays. This could include enhanced post-discharge follow-up and support.
* Patients admitted through emergency services are at higher risk. Implementing specialized care plans for these patients could reduce readmission rates.
* Using the insights from the model to create personalized care plans that address the specific needs of high-risk patients.
* Strengthening post-discharge support systems, such as follow-up calls, home visits, and telehealth services, to ensure patients adhere to their care plans and recover well.
* Continuously collecting and analyzing data to refine the model and identify any new trends or factors that may emerge over time.

**Part VI: Demonstration**

F. Please see the attached link to the Panopto video.

G. Sources of Third-Party Code

*Model validation in Python*. (n.d.). [Video]. datacamp.com. https://app.datacamp.com/learn/courses/model-validation-in-python

Prabhakaran, S. (n.d.). *Train Test split – How to split data into train and test for validating machine learning models?* www.machinelearningplus.com. Retrieved September 1, 2024, from https://www.machinelearningplus.com/machine-learning/train-test-split/

*Redirecting*. (n.d.-p). https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/\_layouts/15/Doc.aspx?sourcedoc=%7B3BDA644B-A08B-48B3-9F30-80D67738E48E%7D&file=D209%20Data%20Mining%201%20Task%202%20Cohort.pptx&action=edit&mobileredirect=true

*The decision Tree Classifier & Supervised Classification models*. (2022, May 6). blog.resolvingpython.com. Retrieved September 9, 2024, from https://blog.resolvingpython.com/02-the-decision-tree-classifier-supervised-classification-models#heading-dummy-variables

H. Sources

Analytics Vidhya. (2024, September 6). *What is Decision Tree Algorithm?* https://www.analyticsvidhya.com/decision-tree-algorithm/

Bobbitt, Z. (2021b, August 9). *How to interpret a ROC curve (With Examples)*. Statology. https://www.statology.org/interpret-roc-curve/

*Data Mining i*. (n.d.). datacamp.com. Retrieved August 26, 2024, from https://app.datacamp.com/learn/custom-tracks/custom-data-mining-i

GeeksforGeeks. (2024a, March 15). *Decision tree in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree-introduction-example/?ref=header\_outind

GeeksforGeeks. (2024c, April 16). *How to tune a Decision Tree in Hyperparameter tuning*. GeeksforGeeks. https://www.geeksforgeeks.org/how-to-tune-a-decision-tree-in-hyperparameter-tuning/

GeeksforGeeks. (2024e, May 17). *Decision tree*. GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree/?ref=header\_outind

Prabhakaran, S. (n.d.). *Train Test split – How to split data into train and test for validating machine learning models?* www.machinelearningplus.com. Retrieved September 1, 2024, from https://www.machinelearningplus.com/machine-learning/train-test-split/

*Redirecting*. (n.d.-o). https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/\_layouts/15/Doc.aspx?sourcedoc=%7BCCC8A210-DF7F-48A8-A947-558BCA95729F%7D&file=D209%20Data%20Mining%201%20Task%201%20Cohort.pptx&action=edit&mobileredirect=true

*Redirecting*. (n.d.-p). https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/\_layouts/15/Doc.aspx?sourcedoc=%7B3BDA644B-A08B-48B3-9F30-80D67738E48E%7D&file=D209%20Data%20Mining%201%20Task%202%20Cohort.pptx&action=edit&mobileredirect=true

*Www.geeksforgeeks.org*. (2024, July 24). Retrieved August 29, 2024, from https://www.geeksforgeeks.org/confusion-matrix-machine-learning/