**Western Governors University (WGU)**

**D209 Data Mining 1**

**Task 1: Classification Analysis**

**Natallia Zimnitskaya | ID: 012247127**

**Master of Science, Data Analytics**

**Part I: Research Question**

A1.  Can a patient’s readmission be predicted using a k-nearest neighbor (KNN) method? Predicting a patient’s readmission using KNN is highly relevant in healthcare settings. It helps reduce costs by identifying high-risk patients and implementing targeted interventions, improving patient care and satisfaction.

A2. One reasonable goal for this data analysis could be to 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. Analyzing these variables allows for identifying high-risk patients and implementing targeted interventions to reduce readmission rates.

**Part II: Method Justification**

B1. One of the most fundamental yet important machine learning classification algorithms is KNN, part of the supervised learning domain with extensive applications in pattern recognition, data mining, and intrusion detection. Because it is non-parametric, KNN is widely used in real-world scenarios (GeeksforGeeks, 2024b).

The basic idea of KNN classification is to predict the label of a data point by examining the ‘k’ closest labeled data points and assigning the most common label among them. This method relies on measuring similarity, often using Euclidean distance, to determine the classification. The variables in this dataset (Area, Age, Gender, ReAdmis, VitD\_levels, Initial\_admin, HighBlood, Stroke, Complication\_risk, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Services, Initial\_days, Additional\_charges) are considered as features, with ReAdmis being the target variable. The KNN algorithm is then trained on the dataset, finding the ‘k’ nearest neighbors for each data point. For a new patient, the model predicts the likelihood of readmission by looking at the ‘k’ nearest neighbors and taking a majority vote on their readmission status.

B2. One key assumption of the KNN classification method is that similar data points exist in proximity. This means that data points near each other in the feature space will likely have the same label. For example, in a healthcare dataset, patients with similar health metrics (like age, blood pressure, and cholesterol levels) are assumed to have similar outcomes, such as the likelihood of readmission. This assumption underpins the effectiveness of KNN in classifying new data points based on their nearest neighbors.

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

* **pandas**: for data manipulation and analysis
* **CategoricalDtype**: helps define categorical data types, which is useful for encoding categorical variables.
* **numpy**: supports large, multi-dimensional arrays and matrices and a collection of mathematical functions to operate on these arrays.
* **matplotlib.pyplot**: a plotting library for creating static, interactive, and animated visualizations in Python.
* **seaborn**: provides a high-level interface for drawing attractive and informative statistical graphics.
* **scipy**: contains modules for optimization, integration, interpolation, eigenvalue problems, and algebraic equations and is helpful for statistical analysis and hypothesis testing.
* **statsmodels.api**: provides classes and functions for estimating many different statistical models and conducting statistical tests and data exploration.
* **variance\_inflation\_factor:** helps detect multicollinearity in the dataset.
* **mosaic**: is a part of stats models; it is used for creating mosaic plots, which are useful for visualizing categorical data.
* **sklearn.model\_selection.train\_test\_split**: splits the dataset into training and testing sets, which is crucial for evaluating your model's performance.
* **sklearn.preprocessing**: provides functions for data preprocessing, such as scaling and encoding.
* **sklearn.feature\_selection.SelectKBest, f\_classif**: used for feature selection and identification.
* **sklearn.neighbors.KNeighborsClassifier**: implements the k-NN algorithm.
* **sklearn.model\_selection.GridSearchCV**: performs hyperparameter tuning to find the best parameters for a KNN model.
* **sklearn.metrics.confusion\_matrix**: evaluates the performance of the classification model by comparing predicted and actual values.
* **sklearn.metrics.roc\_auc\_score**: computes the Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
* **sklearn.metrics.roc\_curve**: plots the ROC curve, which helps visualize the classification model's performance.
* **sklearn.metrics.classification\_report**: generates a report showing the main classification metrics, comprehensively evaluating the model’s performance.

**Part III: Data Preparation**

C1. One crucial data preprocessing goal for the KNN classification method is normalizing numerical features. KNN relies on distance metrics (like Euclidean distance) to determine the similarity between data points. If the numerical features in the dataset, such as Age, VitD\_levels, Initial\_days, and Additional\_charges, have different scales, it can skew the distance calculations and negatively impact the model’s performance. Normalizing these features to a common scale ensures that each feature contributes equally to the distance computation, leading to more accurate and reliable predictions.

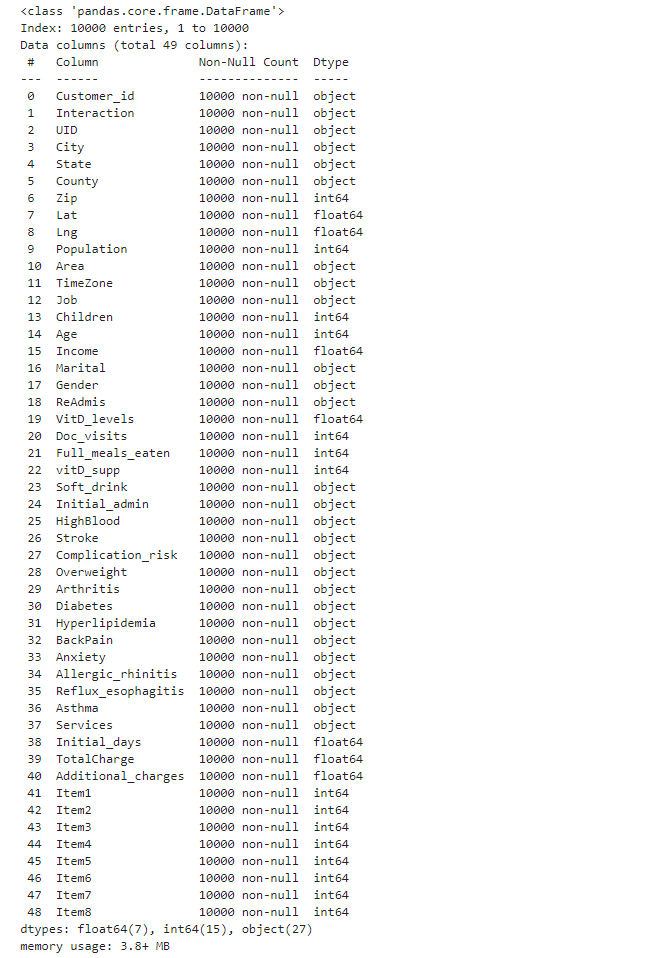
C2. All variables for my research question are listed in the tab below.

|  |  |  |  |
| --- | --- | --- | --- |
| 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.

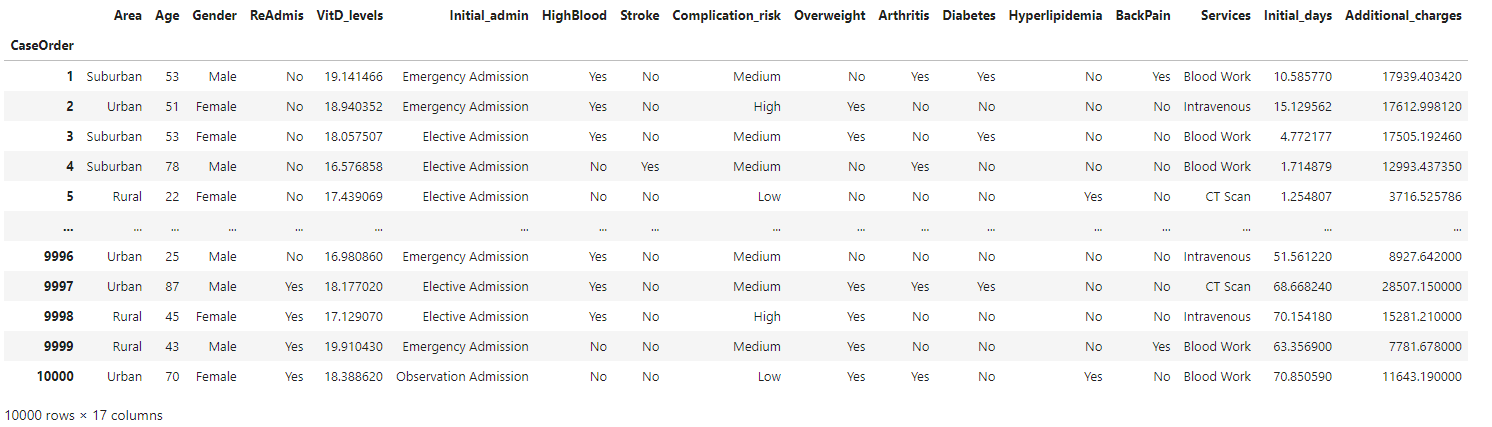
The first step is to import required packages and libraries, then using the pd.read\_csv() function, load the medical\_clean.csv file onto the Jupyter Notebook, and lastly, with df.info() function, get the information about the dataset.



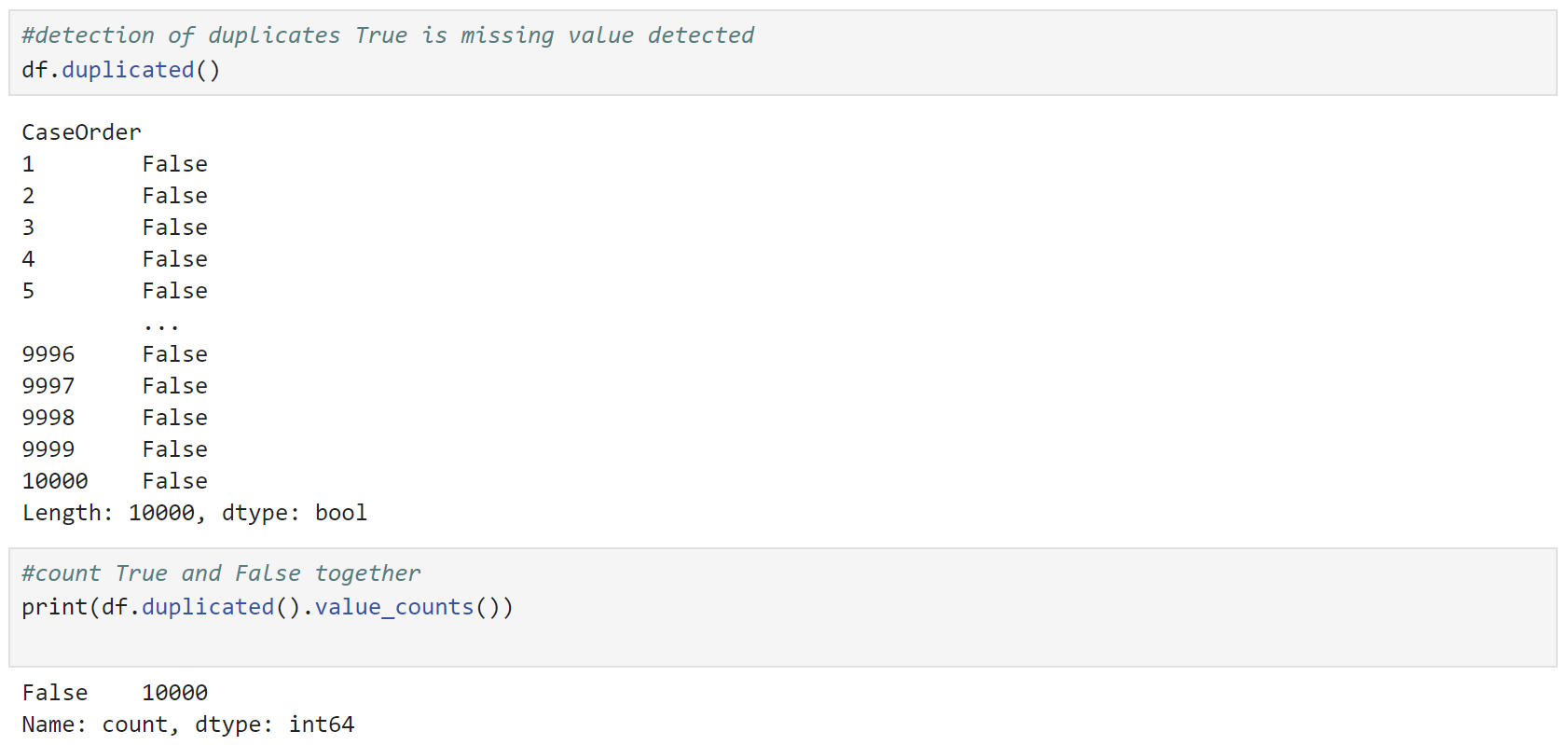


The next step is to drop variables unrelated to the research question ('Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'TimeZone', 'Job', 'Income', 'Children', 'Marital', 'Doc\_visits', 'Full\_meals\_eaten', 'vitD\_supp', 'Soft\_drink', 'Anxiety', 'Allergic\_rhinitis', '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.



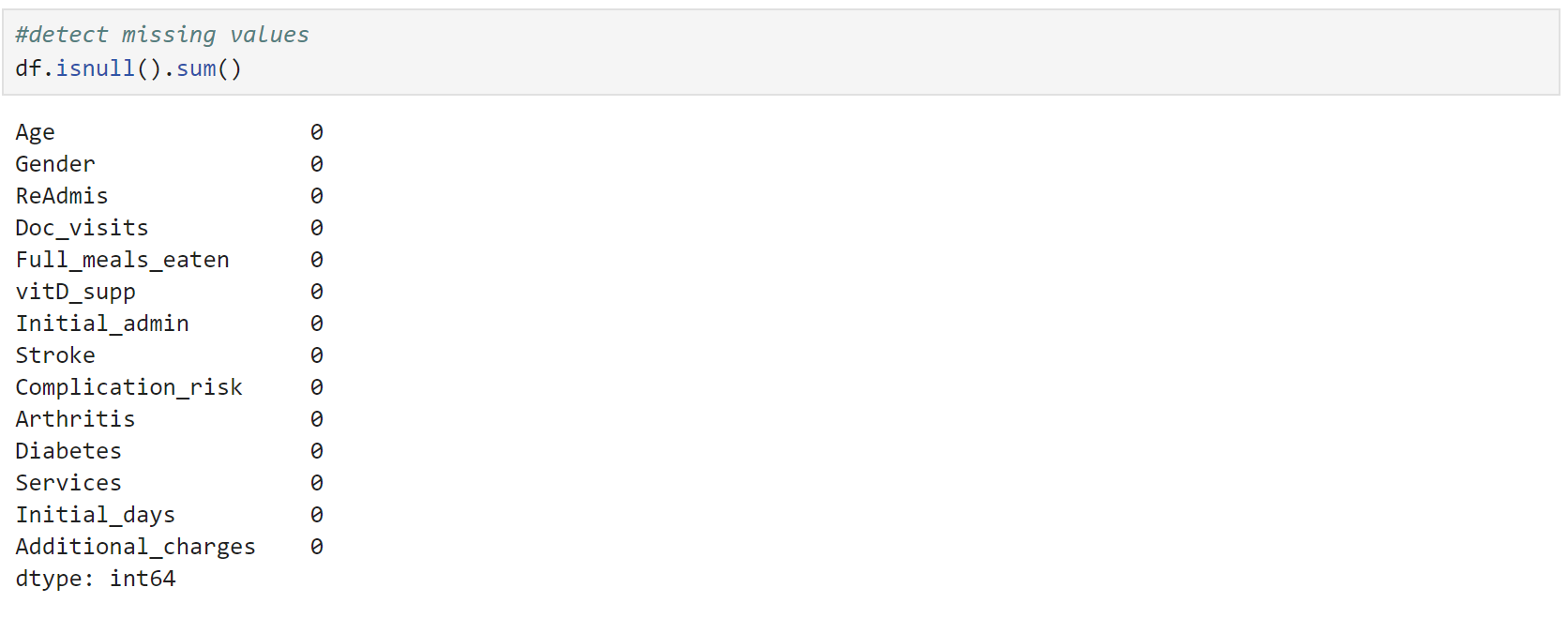


For the detection of duplicates, the duplicated() and duplicated().value\_counts() functions were used.



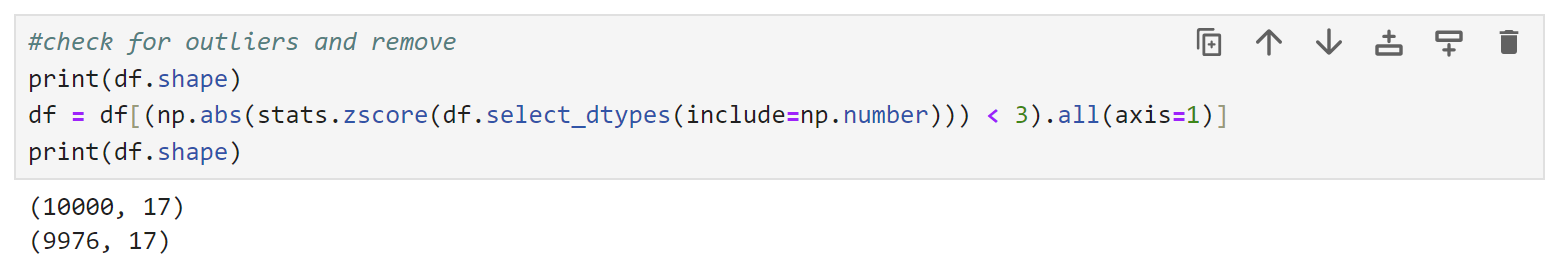
There are no duplicates in the dataset.

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

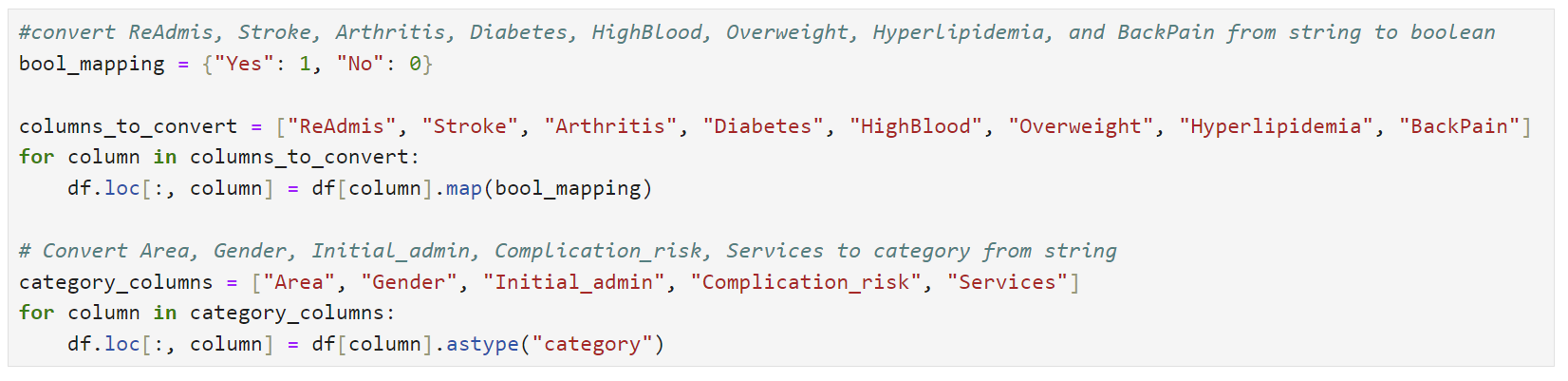


There are no missing values in the dataset.

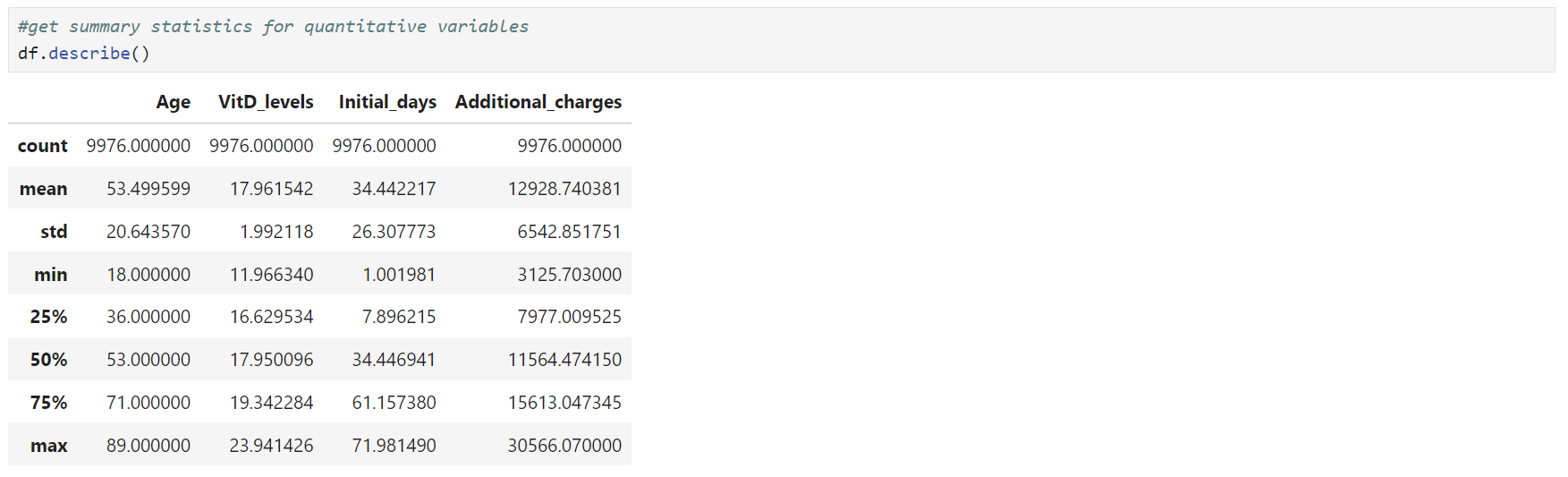
Next, using the z-score method, outliers were checked and removed.



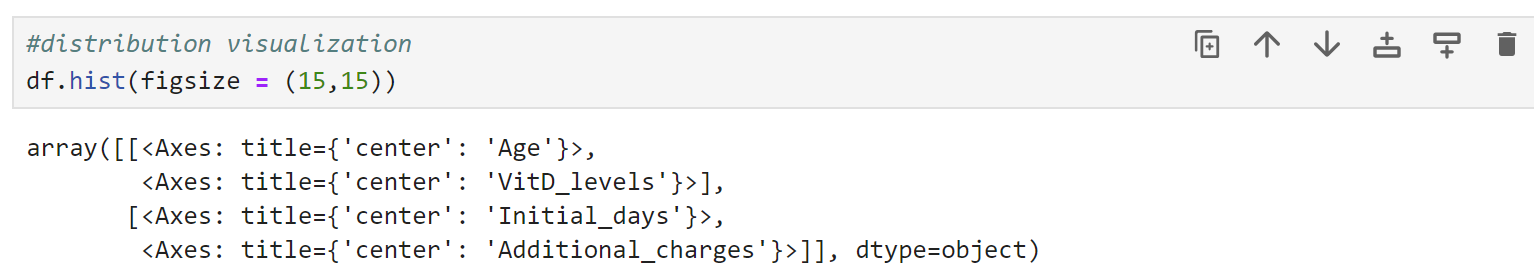
In the next step, I will convert ReAdmis, Stroke, Arthritis, Diabetes, HighBlood, Overweight, Hyperlipidemia, BackPain from string to boolean, and Area, Gender, Initial\_admis, Complicatiom\_risk, and Services to category.

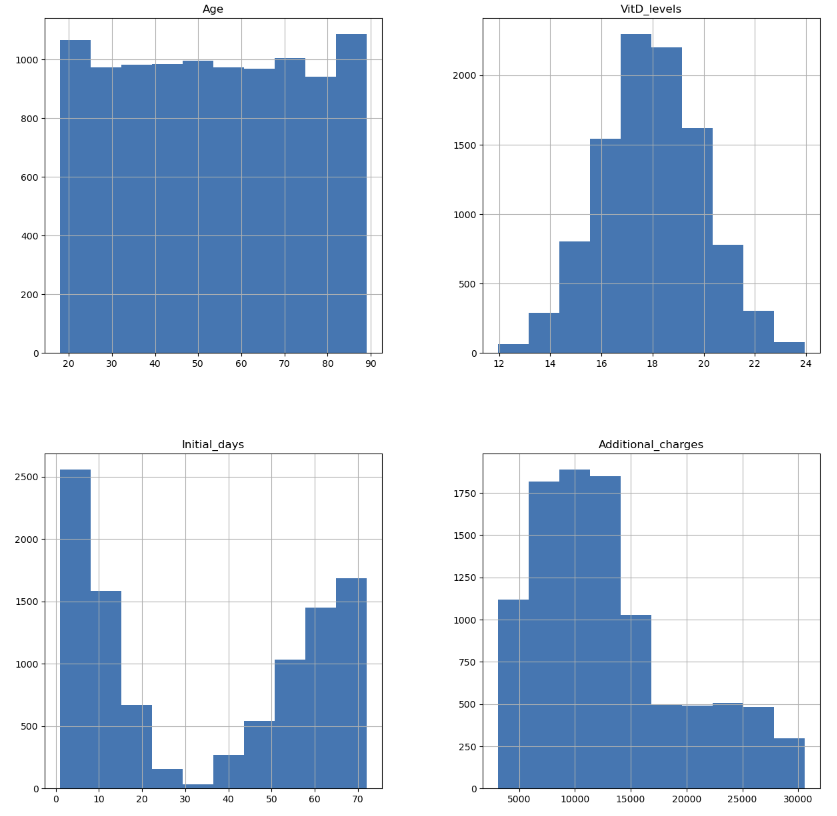


Next is data exploration with summary statistics for all continuous variables.

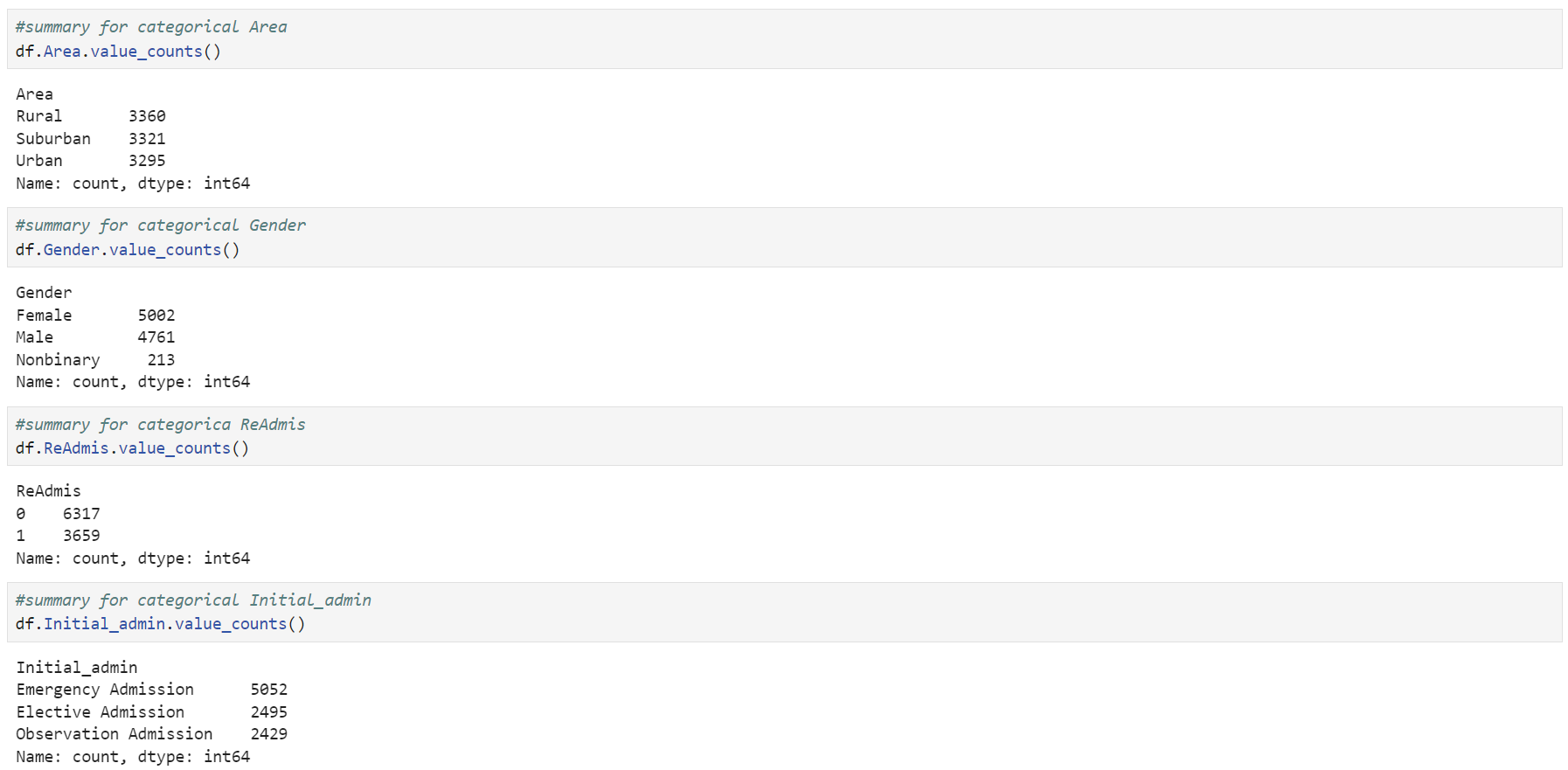


The average age of patients is about 53.5 years, ranging from 18 to 89. The average Vitamin D level is approximately 18, ranging from 12 to 24. Patients typically stay in the hospital for an average of 34.4 days, but this can vary widely from just over 1 day to nearly 72 days. The additional charges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



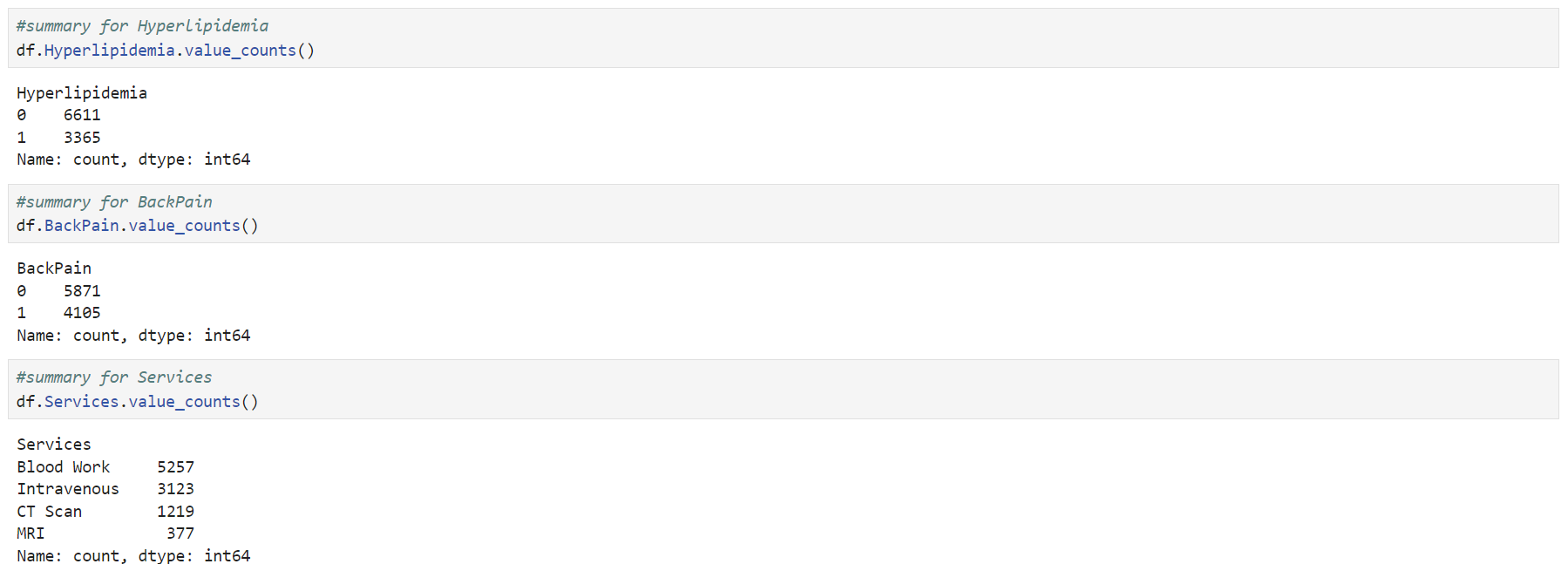


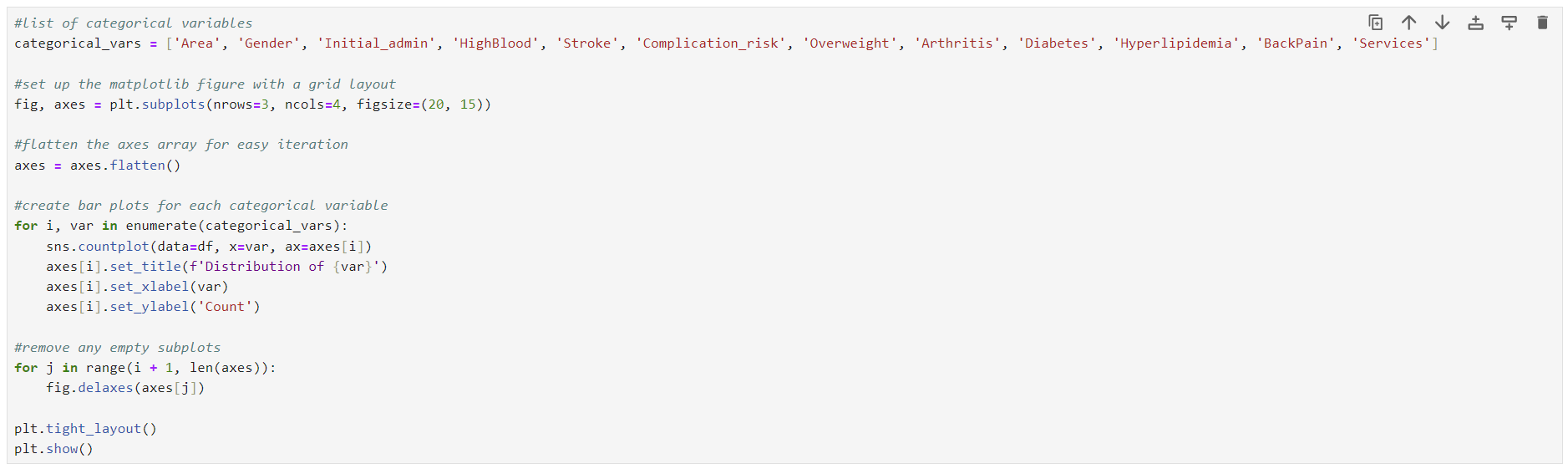
Here is a summary for categorical variables:

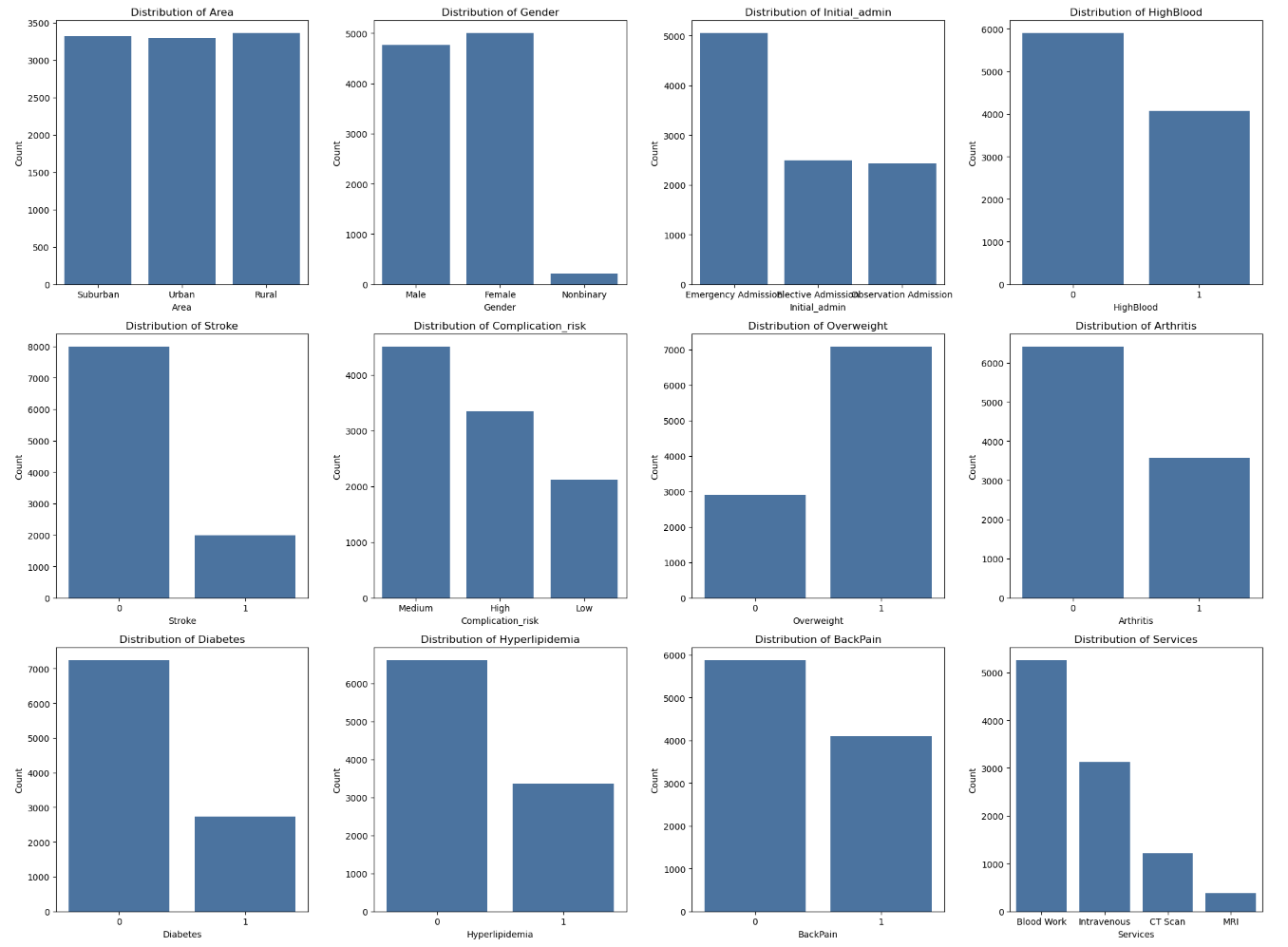




****





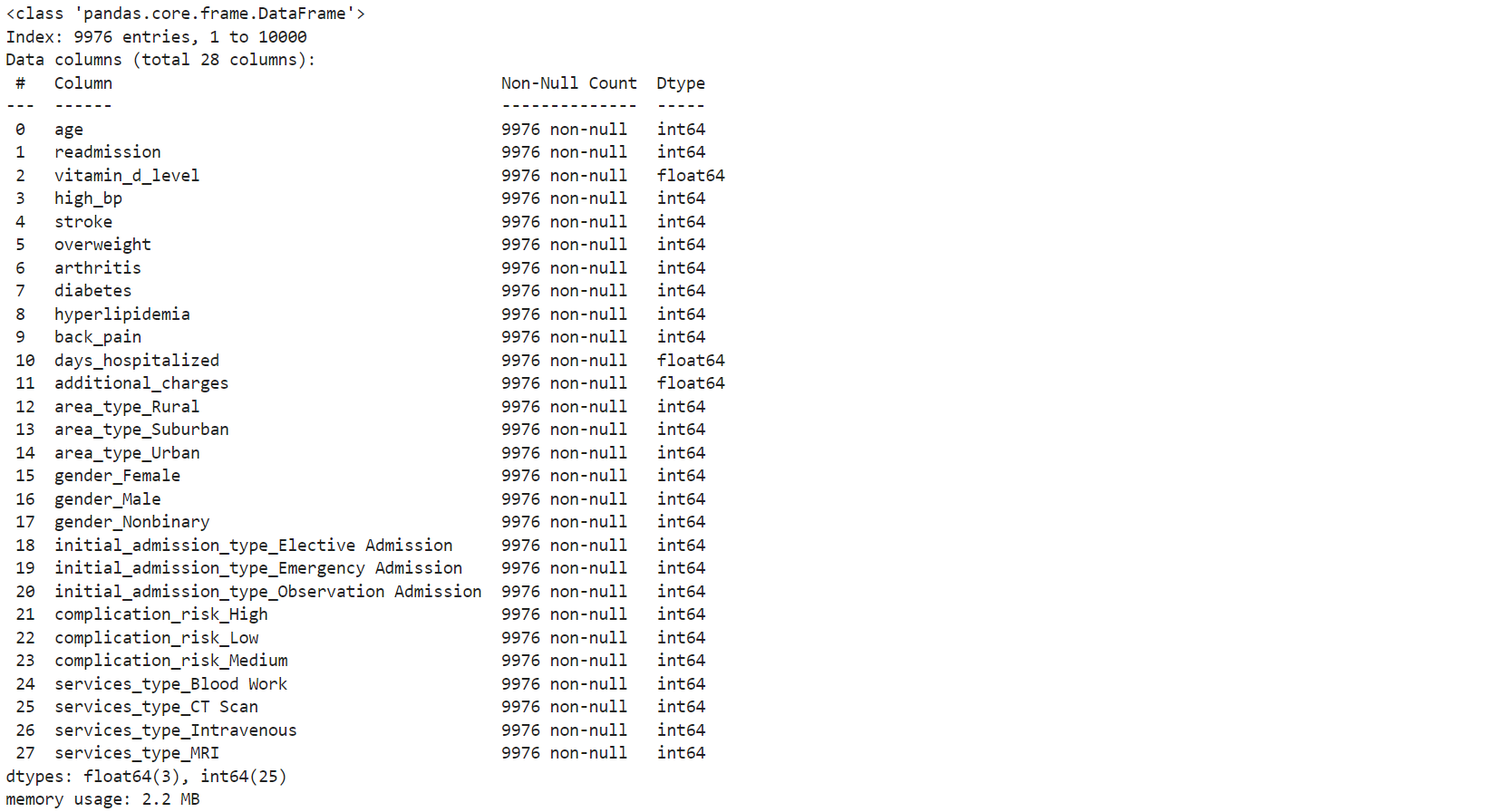


Next, I will rename columns to appropriate Pythonic names ('area\_type', 'age', 'gender', 'readmission', 'vitamin\_d\_level', 'initial\_admission\_type', 'high\_bp', 'stroke', 'complication\_risk', 'overweight', 'arthritis', 'diabetes', 'hyperlipidemia', 'back\_pain', 'services\_type', 'days\_hospitalized',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



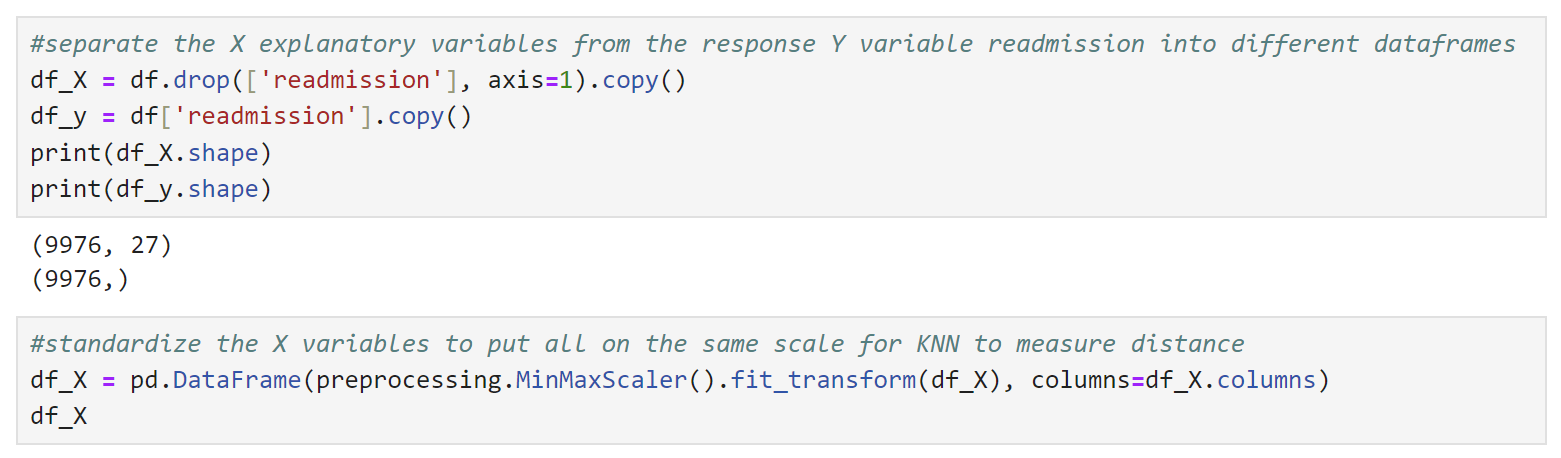
One-hot encoding was performed using the pd.get\_dummies() function. I didn’t use “drop\_first=True” because, for the KNN algorithm, using it with variables with more than two levels of categories can cause information loss and create problems for distance calculations. I converted boolean values to numeric (1 and 0) and applied pd.to\_numeric to convert all columns to numeric.

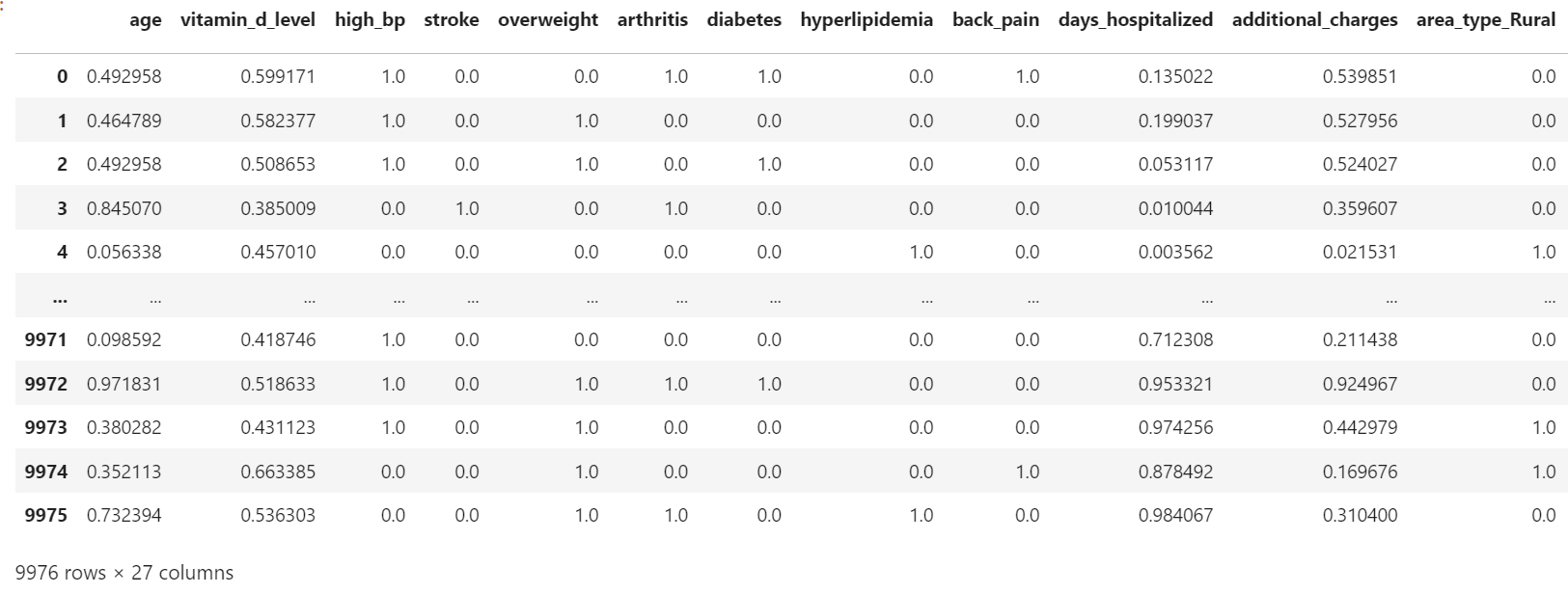


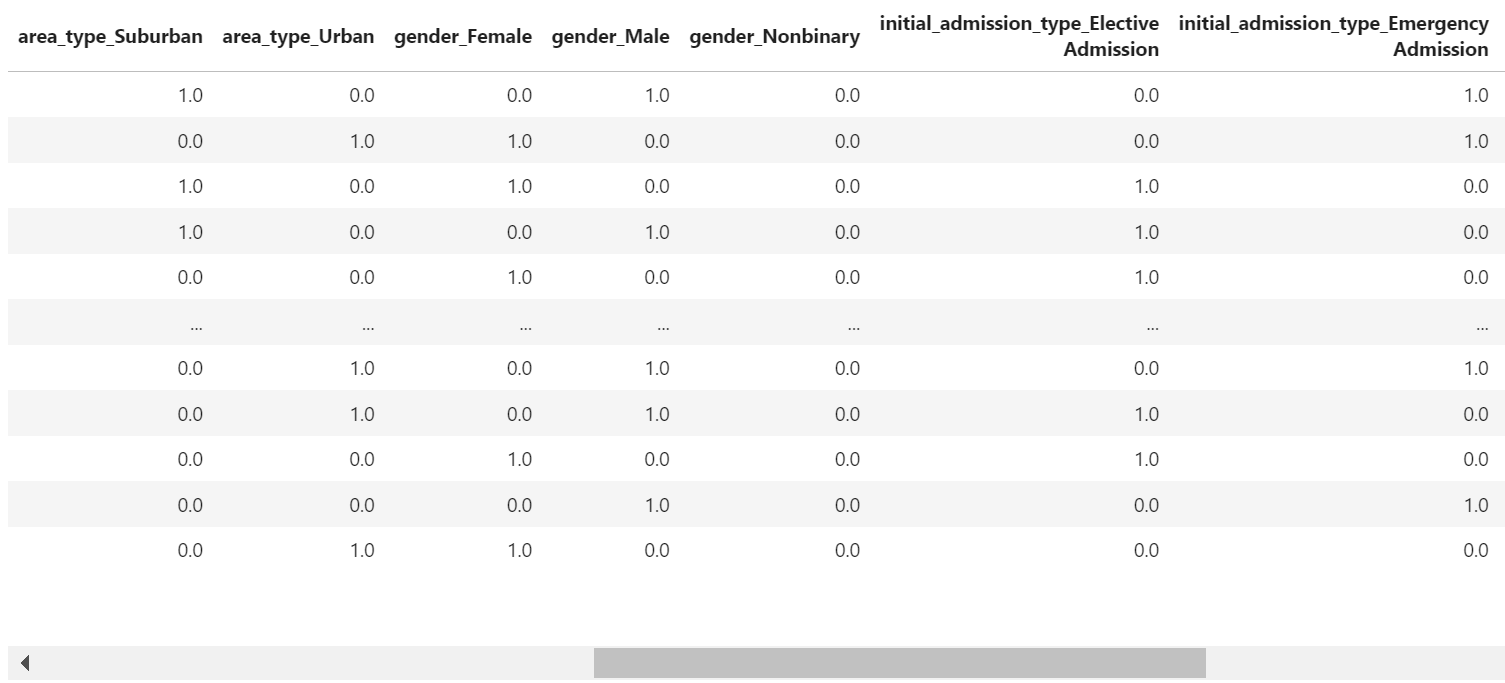


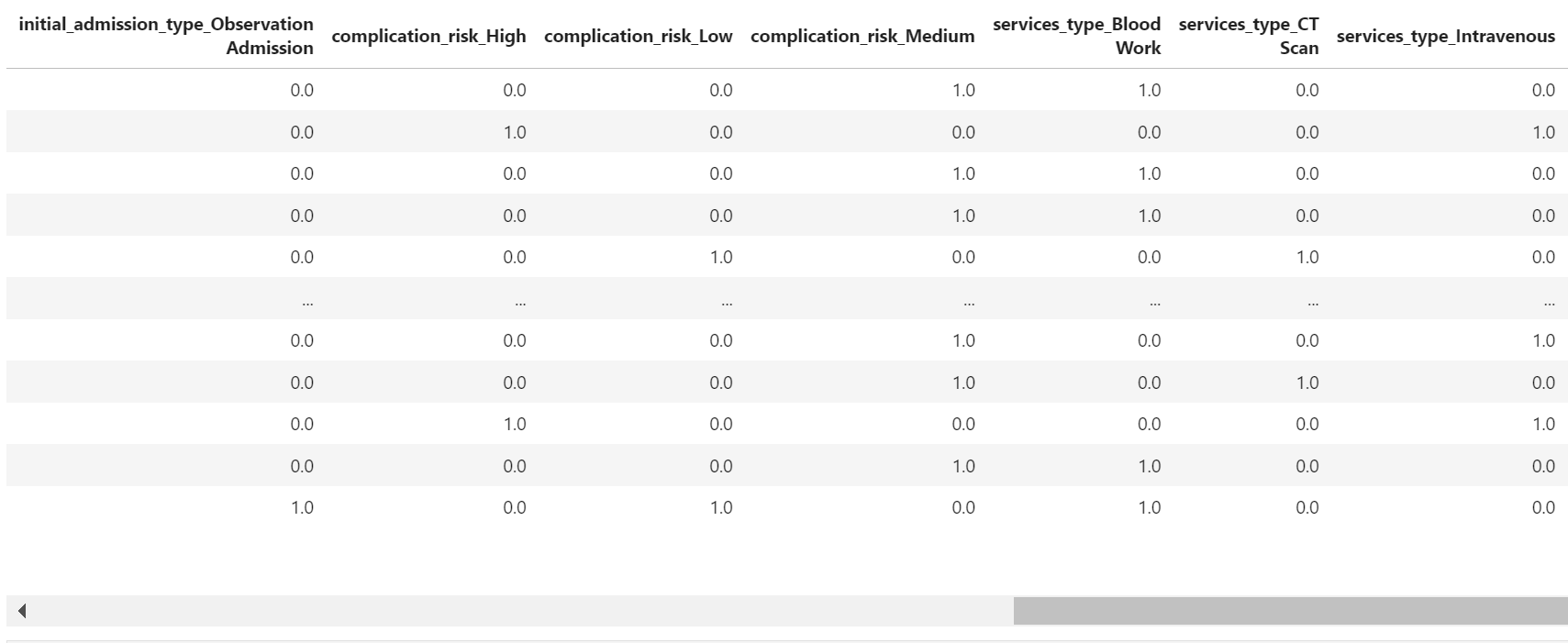
Dataset standardization is crucial for the KNN algorithm, which relies on distance metrics to determine the similarity between data points. If the features have different scales (e.g., age ranging from 18 to 89 and additional\_charges ranging from 3126 to 30566), the larger scale features will dominate the distance calculations, leading to biased results.

First, I will separate the X explanatory variables from the response y variable ‘readmission’ into different data frames. Then, I will standardize the X variables using the MinMaxScaler() function.

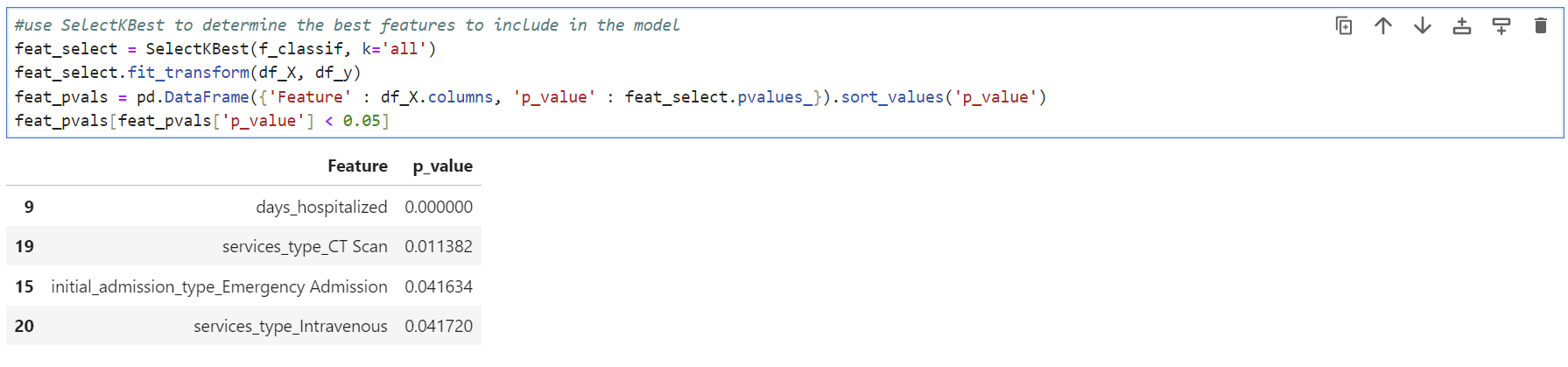








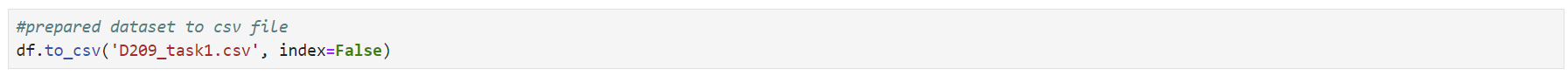


Using the SelectKBest() function helps identify the most important features to include in the model, evaluating each feature in the dataset and assigning a p-value, which indicates the feature’s significance in predicting the target variable. 

Next, I will check VIFs for multicollinearity issues among these variables.

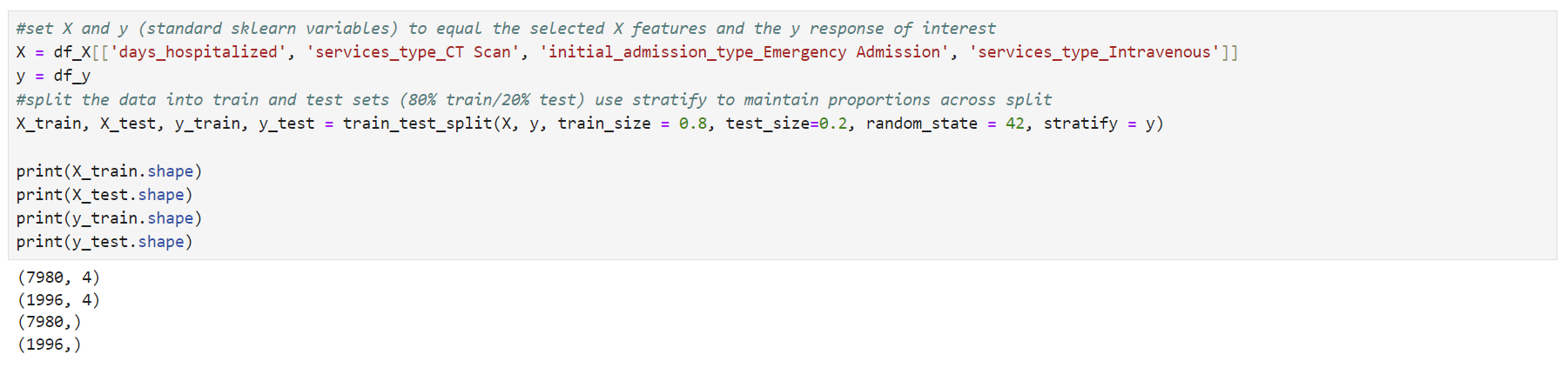


C4. The prepared df dataset was saved to a new csv file. Please see the attached “D209\_task1.csv” file.

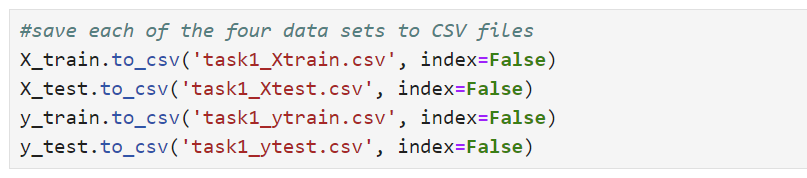
******

**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 ‘task1\_Xtrain.csv’, ‘task1\_Xtest.csv’, ‘task1\_ytrain.csv, and ‘task1\_ytest.csv’ files.

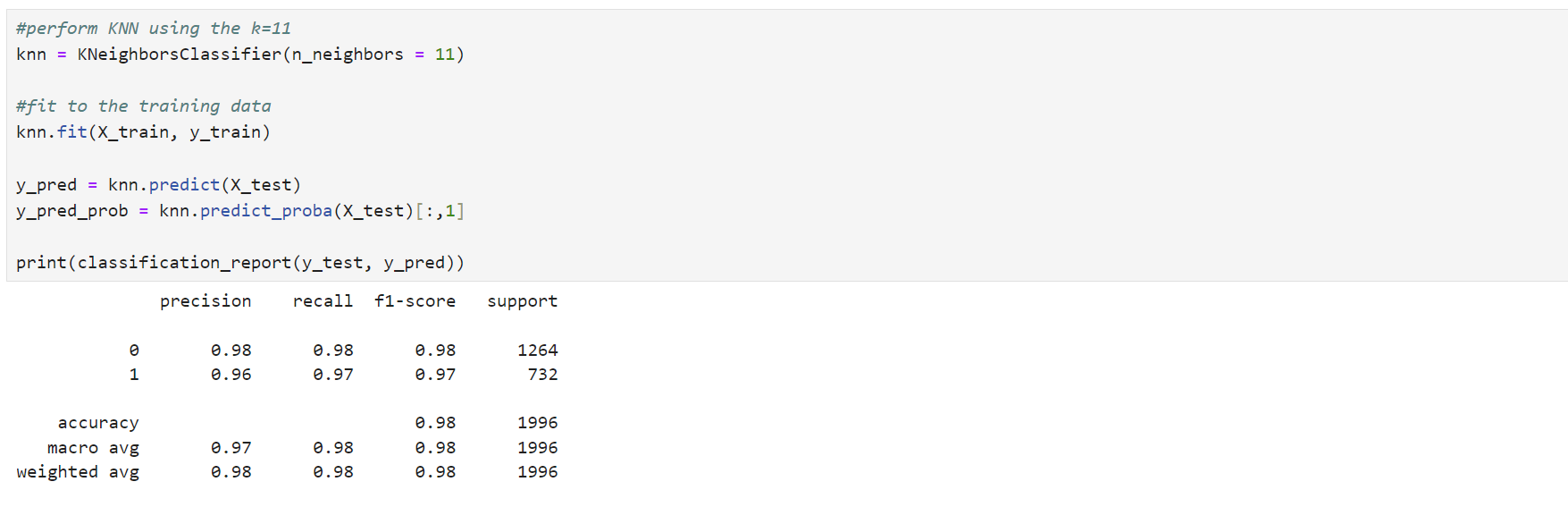


D2. To perform a KNN classification, it’s essential to determine the optimal value of k. Using GridSearchCV, we can test multiple values for k and select the one that yields the most accurate model (*K-Nearest Neighbors (KNN) Classification With Scikit-Learn*, 2023). Once the best k value is identified, we can proceed with the KNN classification. GridSearchCV trains the KNN model with each k value and evaluates its performance using cross-validation. It then selects the k value that gives the best performance.



In this analysis, the optimal hyperparameter for the KNN classifier is n\_neighbors = 11. With this setting, the model achieves a mean score of 0.9815. This mean score reflects the average performance of the model across all folds of the cross-validation process, indicating how well the model performs with the best set of hyperparameters. This high mean score suggests that the model is highly accurate and reliable in predicting patient readmissions.

Next, I will fit the KNN model with k=11 since it is the optimal parameter for the KNN.

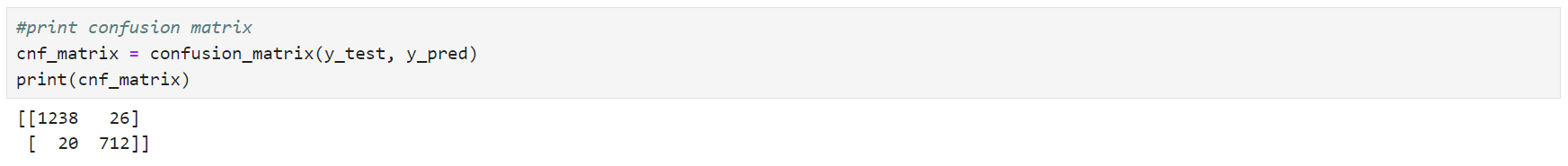


The classification report shows that for patients not readmitted (label 0), the model has a precision, recall, and f1-score of 0.98, meaning it is very accurate. For patients who were readmitted (label 1), the precision is 0.96, recall is 0.97, and f1-score is 0.97, indicating the model is also highly effective for this group. The macro average and weighted average scores are also high, showing consistent performance across both classes.

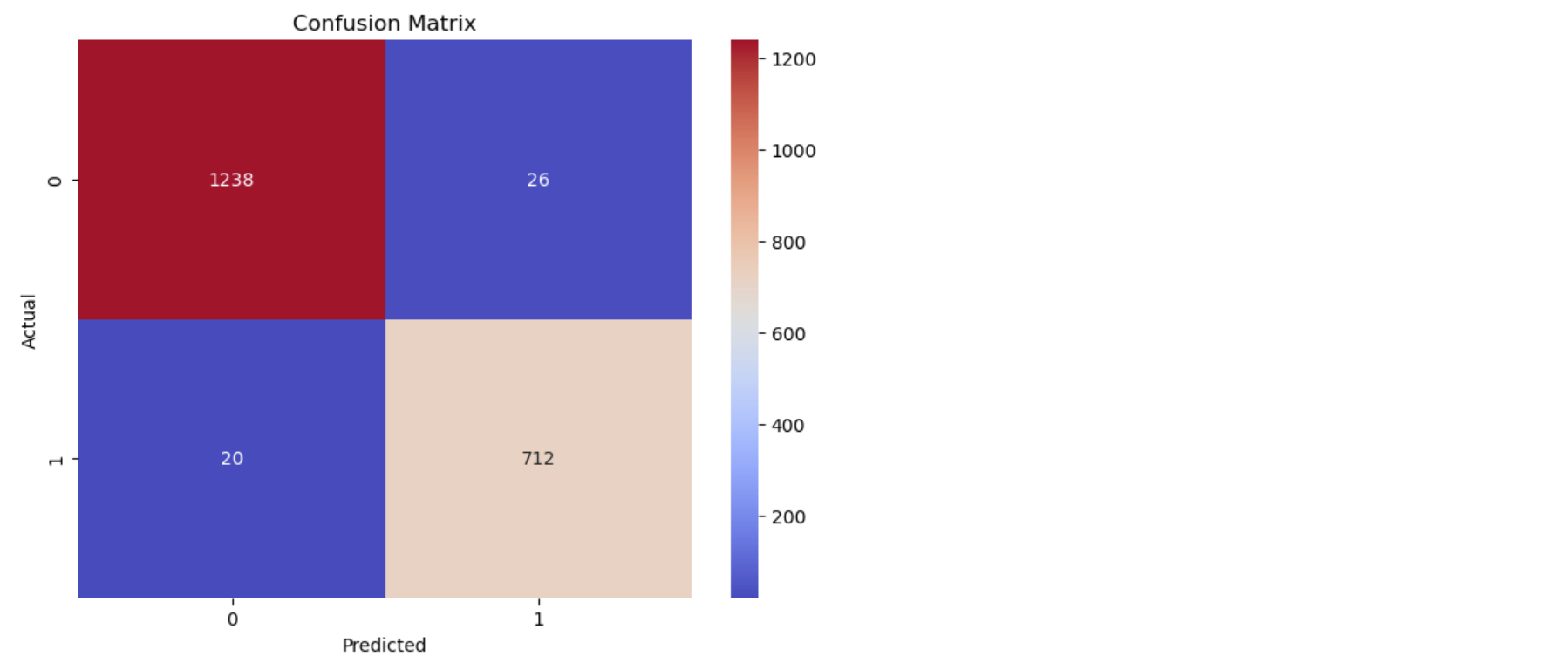
Next, the model accuracy is calculated.

The accuracy is 0.977, meaning it correctly predicts patient readmissions about 97.7% of the time.

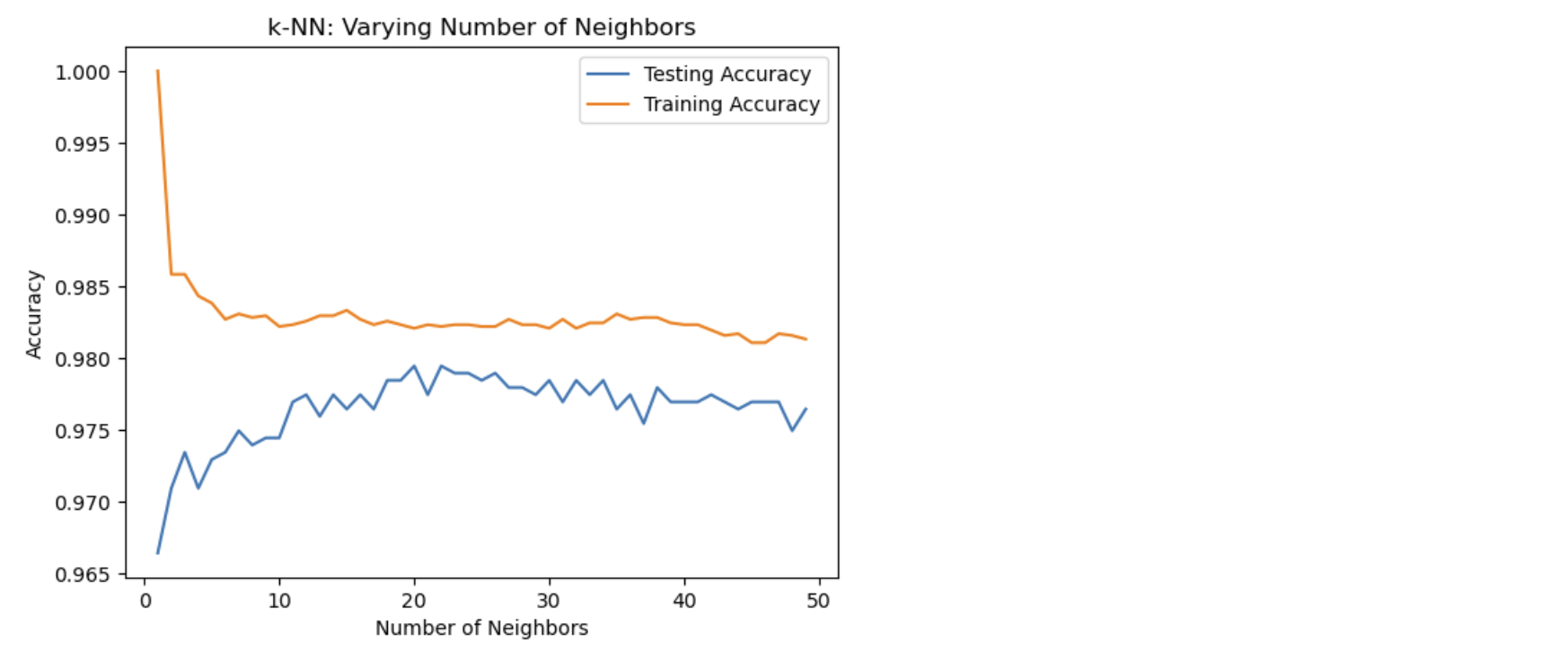
Next is a confusion matrix calculation and visualization.

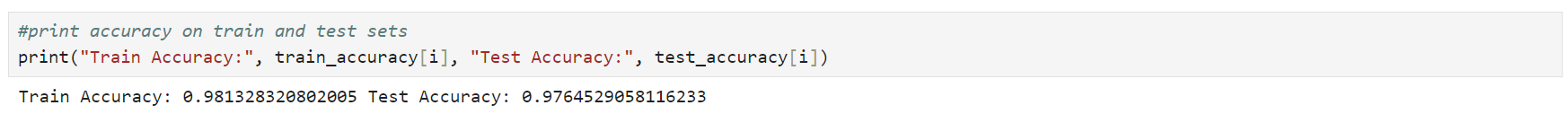


The confusion matrix summarizes the model’s performance by showing the number of correct and incorrect predictions. The model correctly predicted 1238 patients who were not readmitted (TN) and 712 patients who were readmitted (TP). It incorrectly predicted 26 patients as readmitted when not (FP) and 20 patients as not readmitted when they were (FN).









The model’s training accuracy is 0.9813, meaning it correctly predicts patient readmissions about 98.2% of the time on the training data. The test accuracy is 0.9765, indicating that the model maintains high accuracy (97.7%) when applied to new, unseen data. This close alignment between training and test accuracy suggests that the model generalizes well and does not overfit the training data.

The “k-NN: Varying Number of Neighbors” illustrates how the accuracy of a KNN classifier changes as the number of neighbors varies. The x-axis represents the number of neighbors, ranging from 0 to 50, while the y-axis shows the accuracy, ranging from 0.965 to 1.000. As the number of neighbors increases, the training accuracy decreases while the testing accuracy initially increases and stabilizes. This pattern helps identify the optimal number of neighbors that balances the model’s performance on training and test data, ensuring it generalizes well to new, unseen data.

D3. Please see the attached “D209Task1.pynb” file with annotated code.

#run GridSearchCV to find best number of k

param\_grid = {'n\_neighbors': np.arange(1, 50)}

knn = KNeighborsClassifier()

knn\_cv = GridSearchCV(knn, param\_grid)

knn\_cv.fit(X\_train, y\_train)

print('The best parameters for this model: {}'.format(knn\_cv.best\_params\_))

#calculate the mean score (for the best parameter setting found by the GridSearch)

mean\_score = knn\_cv.cv\_results\_['mean\_test\_score'][knn\_cv.best\_index\_]

print('Mean score: {}'.format(mean\_score))

#perform KNN using the k=11

knn = KNeighborsClassifier(n\_neighbors = 11)

#fit to the training data

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

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

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

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

#print accuracy of the model

print("The accuracy of the model is: ", knn.score(X\_test, y\_test))

#print confusion matrix

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

print(cnf\_matrix)

#use seaborn heatmap to visualize the confusion matrix

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

plt.tight\_layout()

plt.title('Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.savefig('matrix1.jpg')

#model complexity curve

neighbors = np. arange(1, 50)

train\_accuracy = np.empty (len (neighbors))

test\_accuracy = np.empty (len (neighbors))

#loop over different values of k

for i, k in enumerate (neighbors):

#setup a k-NN Classifier with k neighbors: knn

knn = KNeighborsClassifier (n\_neighbors=k)

#fit the classifier to the training data

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

#compute accuracy on the training set

train\_accuracy[i] = knn.score (X\_train, y\_train)

#compute accuracy on the testing set

test\_accuracy[i] = knn. score (X\_test, y\_test)

#generate plot

plt.title ('k-NN: Varying Number of Neighbors')

plt.plot(neighbors, test\_accuracy, label = 'Testing Accuracy')

plt.plot (neighbors, train\_accuracy, label = 'Training Accuracy')

plt.legend()

plt.xlabel('Number of Neighbors')

plt.ylabel('Accuracy')

plt.show()

#print accuracy on train and test sets

print("Train Accuracy:", train\_accuracy[i], "Test Accuracy:", test\_accuracy[i])

#generate AUC score and print

y\_pred\_prob = knn.predict\_proba(X\_test)[:, 1]

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

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

plt.plot(fpr, tpr)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

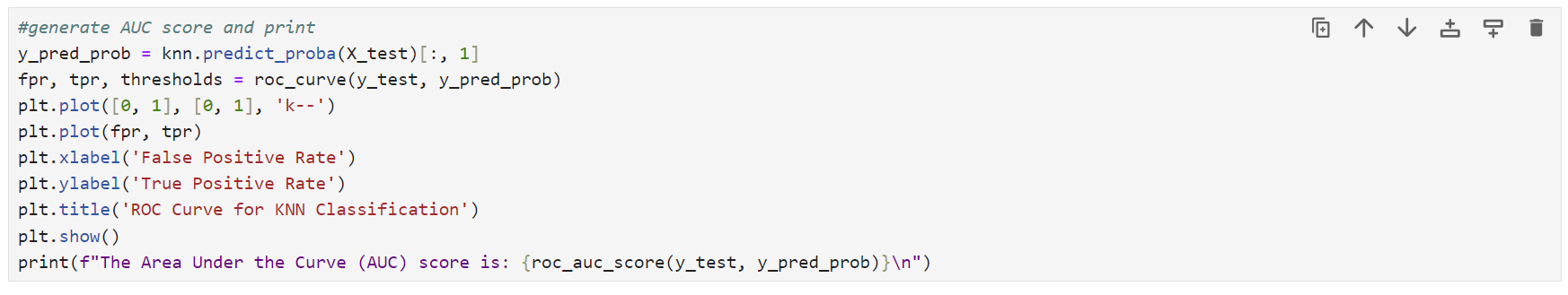
plt.title('ROC Curve for KNN Classification')

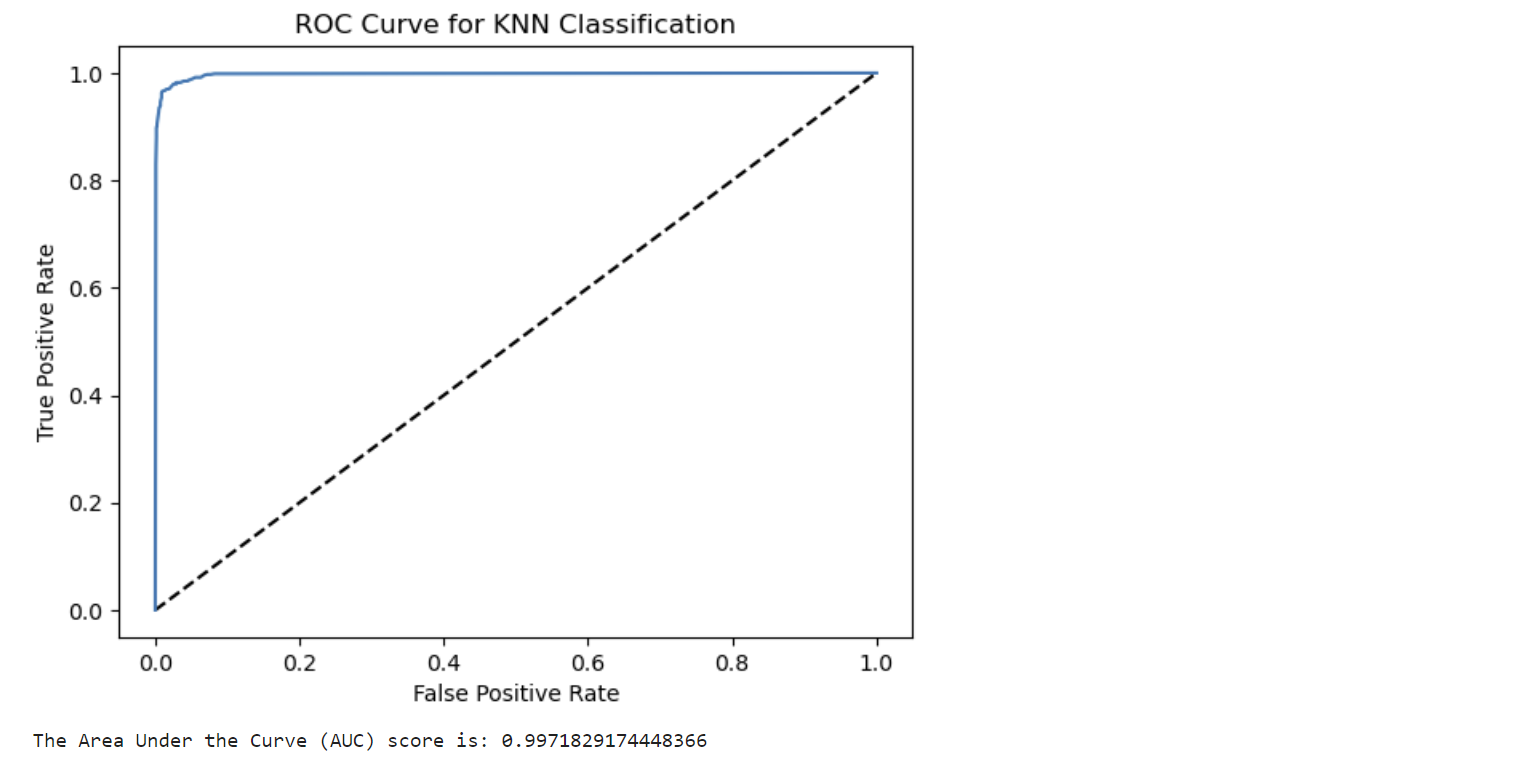
plt.show()

print(f"The Area Under the Curve (AUC) score is: {roc\_auc\_score(y\_test, y\_pred\_prob)}\n")

**Part V: Data Summary and Implications**

E1. 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 of 0.997 means this KNN model is almost perfect at distinguishing between the classes. This high score indicates that the model is very effective at predicting the correct class for new data points, making it a reliable tool for classification tasks.

The ROC curve is a graphical representation that illustrates the performance of a binary classification model by plotting the TP rate (sensitivity) against the FP rate (1-specificity) at various threshold settings. In this graph, the curve starts at the bottom left corner (0,0), rises sharply towards the top left corner (0,1), and then moves rightwards towards the top right corner (1,1). This shape indicates that the KNN model performs exceptionally well, quickly achieving a high TP Rate with a low FP Rate.

E2. The classification analysis using the KNN model focused on predicting the likelihood of patient readmission. Applying the SelectKBest method with the f\_classif function, I identified four significant features: days\_hospitalized, services\_type\_CT Scan, initial\_admission\_type Emergency Admission, and services\_type Intravenous. These features had p-values less than 0.05, indicating their strong contribution to the model’s predictive power and VIFs lower than 2, suggesting no multicollinearity. The KNN model, optimized with 11 neighbors, achieved an accuracy of 97.7%, with precision and recall scores of 0.98 and 0.97, respectively, for the two classes. The confusion matrix reveals that this model performs exceptionally well in predicting patient readmissions. It correctly identified 712 readmissions and 1238 non-readmissions. There were only 26 instances where the model incorrectly predicted readmissions for non-readmitted patients, and 20 instances failed to predict readmissions for readmitted patients. The AUC score was 0.997, reflecting its excellent ability to distinguish between classes. The confusion matrix further confirmed the model’s robustness, showing minimal misclassifications.

E3. One limitation of this data analysis is the potential for data imbalance. If the dataset has significantly more instances of one class than the other (non-readmissions vs readmissions), the model might become biased towards the majority class. This can lead to high accuracy but poor performance in correctly predicting the minority class. Although metrics like AUC help understand the model’s performance, they might not fully capture the challenges of imbalanced data. Addressing this limitation could involve resampling, using different evaluation metrics, or applying algorithms designed to handle imbalanced datasets.

E4. This KNN model demonstrated exceptional performance in predicting patient readmissions, achieving a high accuracy of 97.7% and an outstanding AUC score of 0.997. These results underscore the model’s reliability and effectiveness, making it a valuable tool for data-driven decision-making in healthcare. Based on these results, healthcare organizations can take several proactive steps. To reduce readmissions, they can implement targeted interventions and personalized care plans for high-risk patients. Efficient resource allocation, such as optimizing hospital beds and staff, can be achieved by focusing on those most likely to be readmitted. Adjusting admission and discharge policies and adopting best practices for managing conditions associated with high readmission rates can improve outcomes. Continuous monitoring and updating of the KNN model will ensure its accuracy and relevance. Additionally, educational programs and patient engagement initiatives can help patients understand the importance of follow-up care, medication adherence, and lifestyle changes, ultimately leading to better health outcomes and reduced readmissions.

**Part VI: Demonstration**

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

G. Sources of Third-Party Code

GeeksforGeeks. (2023, December 14). *How to find the optimal value of K in KNN*. GeeksforGeeks. https://www.geeksforgeeks.org/how-to-find-the-optimal-value-of-k-in-knn/?ref=header\_outind

*K-Nearest Neighbors (KNN) classification with Scikit-Learn*. (2023, February). www.datacamp.com. Retrieved September 5, 2024, from https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn

*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/

H. Sources

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. (2024b, July 15). *KNeArest Neighbor(KNN) algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/k-nearest-neighbours/#

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

*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