A1) Research Question

“Which predictor variables most significantly relate to patients being readmitted to the hospital within 30 days”

A2) Objectives

The goal of this analysis is to utilize k-nearest neighbors (KNN) to identify what factors most significantly relate to patients being readmitted. The variables that will be analyzed are within the medical dataset provided. KNN classification will assist with creating an effective model to answer our research question.

B1) Classification Method and Outcomes

K-Nearest Neighbor predicts the classification of a data point by calculating the distance between input variable data points and a data point of interest utilizing Euclidean distance (Zhang, 2016). To answer our research question, the readmission variable will be utilized as the data point of interest and KNN will be utilized to calculate the nearest data points and their classifications. The data point of interest is classified by surveying the neighboring data points and establishing patterns to label the data point of interest. The expected outcome with our analysis and classification model is to achieve an accuracy score of at least 95 percent, giving us confidence in our model.

B2) K-Nearest Neighbor Assumptions

The primary assumption for a K-Nearest Neighbor model is the data points within a model that are in close proximity to each other are generally very similar. If this assumption is not present within a KNN model, the accuracy of classification for a data point of interest is prone to inaccuracy.

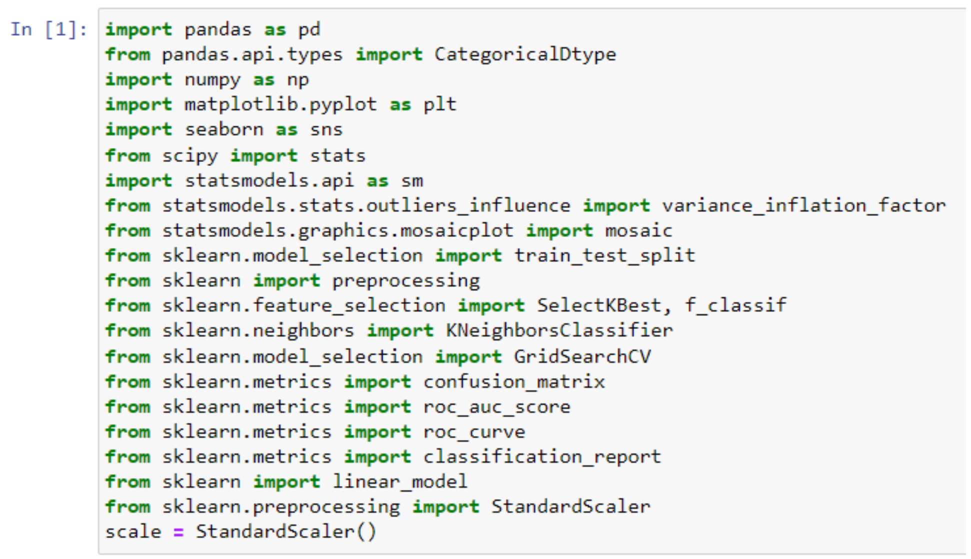
B3) Libraries and Packages

The programming language I utilized for this analysis is Python. Python is familiar to me as a programming tool and has simple syntax to utilize. There are also a large variety of libraries and packages that are beneficial for statistical analysis. Specifically, Python has specific tools for K-Nearest Neighbor classifications, which makes it a good choice for this analysis. I utilized a combination of matplotlib, seaborn, statmodels, sklearn, numpy, and pandas for my analysis. Sklearn in particular has many statistical libraries that are useful for classification, such as KNeighborsClassifier. I utilized these libraries due to my familiarity, ease of use, and statistically relevant packages.

- Pandas – provides a method to read and visualize data
- Numpy – utilized for operating with arrays and provides functions for meathematical calculations
- Seaborn – utilized for visually descriptive graphs and matrics
- Matplotlib – Provides tools for report and data visualization
- Scikit-learn – utilized for statistical modeling and machine learning

Figure 1

Packages and Libraries



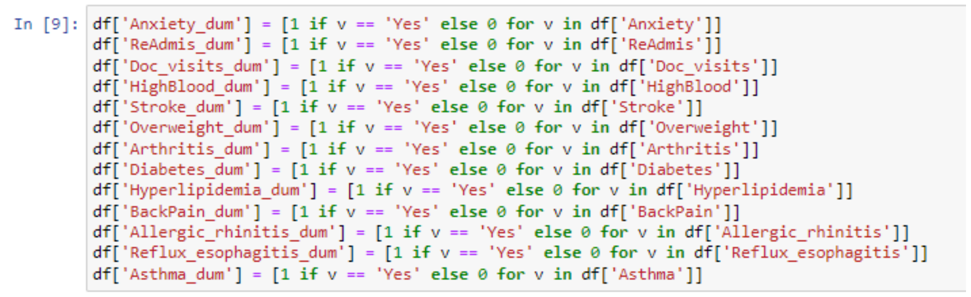
```
In [1]: import pandas as pd
        from pandas.api.types import CategoricalDtype
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from scipy import stats
        import statsmodels.api as sm
        from statsmodels.stats.outliers_influence import variance_inflation_factor
        from statsmodels.graphics.mosaicplot import mosaic
        from sklearn.model_selection import train_test_split
        from sklearn import preprocessing
        from sklearn.feature_selection import SelectKBest, f_classif
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import GridSearchCV
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import roc_auc_score
        from sklearn.metrics import roc_curve
        from sklearn.metrics import classification_report
        from sklearn import linear_model
        from sklearn.preprocessing import StandardScaler
        scale = StandardScaler()
```

C1) Data Preprocessing

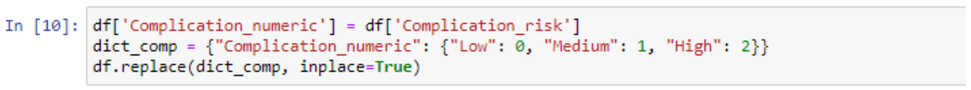
For the classification to perform properly, the data in the dataset needs to be transformed. Specifically, categorical or qualitative variables need to be transformed to quantitative values. This will be accomplished by utilizing dummy variables and one hot encoding.

Figure 2

Data Transformation



```
In [9]: df['Anxiety_dum'] = [1 if v == 'Yes' else 0 for v in df['Anxiety']]
        df['ReAdmis_dum'] = [1 if v == 'Yes' else 0 for v in df['ReAdmis']]
        df['Doc_visits_dum'] = [1 if v == 'Yes' else 0 for v in df['Doc_visits']]
        df['HighBlood_dum'] = [1 if v == 'Yes' else 0 for v in df['HighBlood']]
        df['Stroke_dum'] = [1 if v == 'Yes' else 0 for v in df['Stroke']]
        df['Overweight_dum'] = [1 if v == 'Yes' else 0 for v in df['Overweight']]
        df['Arthritis_dum'] = [1 if v == 'Yes' else 0 for v in df['Arthritis']]
        df['Diabetes_dum'] = [1 if v == 'Yes' else 0 for v in df['Diabetes']]
        df['Hyperlipidemia_dum'] = [1 if v == 'Yes' else 0 for v in df['Hyperlipidemia']]
        df['BackPain_dum'] = [1 if v == 'Yes' else 0 for v in df['BackPain']]
        df['Allergic_rhinitis_dum'] = [1 if v == 'Yes' else 0 for v in df['Allergic_rhinitis']]
        df['Reflux_esophagitis_dum'] = [1 if v == 'Yes' else 0 for v in df['Reflux_esophagitis']]
        df['Asthma_dum'] = [1 if v == 'Yes' else 0 for v in df['Asthma']]
```



```
In [10]: df['Complication_numeric'] = df['Complication_risk']
         dict_comp = {"Complication_numeric": {"Low": 0, "Medium": 1, "High": 2}}
         df.replace(dict_comp, inplace=True)
```

C2) Initial Data Set Variables

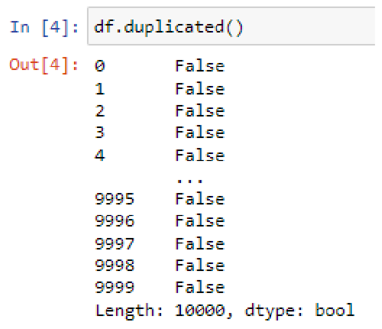
To answer our research question, we need to identify the variables that will assist in answering our question. The variable ReAdmis (categorical) is the dependent (target) variable, and the variables that we will be analyzing for classification are the independent (explanatory) variables. The independent or predictor variables include anxiety (categorical), doc visits (continuous), high blood pressure (categorical), stroke (categorical), overweight (categorical), arthritis (categorical), diabetes (categorical), initial days (continuous), hyperlipidemia (categorical), back pain (categorical), allergic rhinitis (categorical), reflux esophagitis (categorical), complication risk (categorical), and age (continuous).

C3) Data Preparation

Before we can analyze our data, we need to clean and prepare our dataset. I utilized the duplicated() function to detect duplicates. I then utilized the isnull() function to detect missing values. Finally, to detect outliers I created boxplots with the qualitative variables that will be included in the analysis. Upon completion of all these detection methods, I did not find any duplicates, missing values, or outliers that need to be treated.

Figure 3

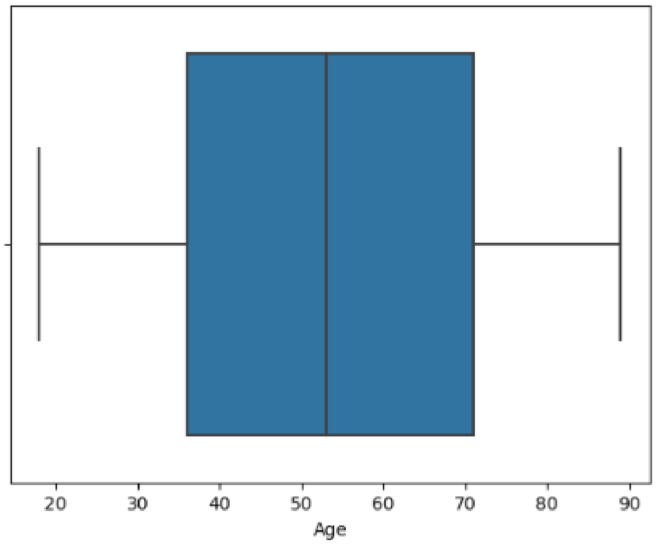
Detecting for Duplicates, Missing Values, and Outliers



```
In [4]: df.duplicated()

Out[4]: 0       False
        1       False
        2       False
        3       False
        4       False
                ...
        9995    False
        9996    False
        9997    False
        9998    False
        9999    False
        Length: 10000, dtype: bool
```

```
In [5]: df.isnull().sum()

Out[5]: CaseOrder             0
        Customer_id           0
        Interaction           0
        UID                   0
        City                  0
        State                 0
        County                0
        Zip                   0
        Lat                   0
        Lng                   0
        Population            0
        Area                  0
        TimeZone              0
        Job                   0
        Children              0
        Age                   0
        Income                0
        Marital               0
        Gender                0
        ReAdmis               0
        VitD_levels           0
        Doc_visits            0
        Full_meals_eaten      0
        vitD_supp             0
        Soft_drink            0
        Initial_admin         0
        HighBlood             0
        Stroke                0
        Complication_risk     0
        Overweight            0
        Arthritis             0
        Diabetes              0
        Hyperlipidemia        0
        BackPain              0
        Anxiety               0
        Allergic_rhinitis     0
        Reflux_esophagitis    0
        Asthma                0
        Services              0
        Initial_days          0
        TotalCharge           0
        Additional_charges    0
        Item1                 0
        Item2                 0
        Item3                 0
        Item4                 0
        Item5                 0
        Item6                 0
        Item7                 0
        Item8                 0
        dtype: int64
```

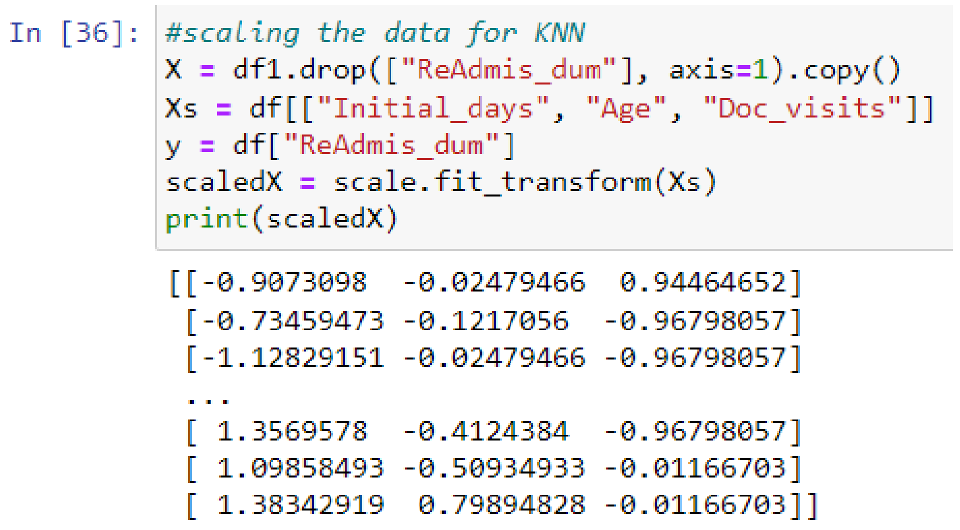
```
In [6]: boxplot=sns.boxplot(x= 'Age',data=df)
        df.plot.scatter(x = 'Age', y = 'ReAdmis')

Out[6]: <Axes: xlabel='Age', ylabel='ReAdmis'>
```



For an accurate k-NN classification, I needed to scale the continuous data set I inputed into the model. Below is the code I utilized to scale the data set.

Figure 4

Data Scaling



```
In [36]: #scaling the data for KNN
         X = df1.drop(["ReAdmis_dum"], axis=1).copy()
         Xs = df[["Initial_days", "Age", "Doc_visits"]]
         y = df["ReAdmis_dum"]
         scaledX = scale.fit_transform(Xs)
         print(scaledX)

         [[-0.9073098  -0.02479466  0.94464652]
          [-0.73459473 -0.1217056  -0.96798057]
          [-1.12829151 -0.02479466 -0.96798057]
          ...
          [ 1.3569578  -0.4124384  -0.96798057]
          [ 1.09858493 -0.50934933 -0.01166703]
          [ 1.38342919  0.79894828 -0.01166703]]
```

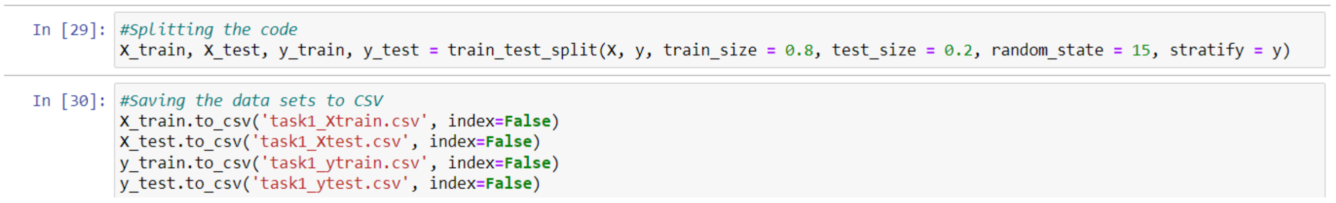
C4) Copy of Cleaned Data Set

A copy of the cleaned data set utilizing for this analysis is attached

D1) Data Splitting

Figure 5

Splitting the Data and Converting to CSV



```
In [29]: #Splitting the code
         X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, test_size = 0.2, random_state = 15, stratify = y)

In [30]: #Saving the data sets to CSV
         X_train.to_csv('task1_Xtrain.csv', index=False)
         X_test.to_csv('task1_Xtest.csv', index=False)
         y_train.to_csv('task1_ytrain.csv', index=False)
         y_test.to_csv('task1_ytest.csv', index=False)
```

D2) Data Analysis

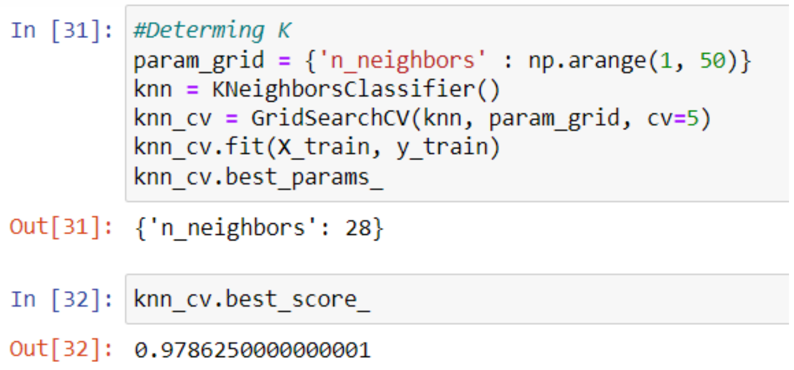
To run our analysis utilizing k-NN classification, we needed to determine what our k value will be. To do this, I utilized the GridSearchCV function to calculate an optimal k value. This calculation resulted in 28 neighbors. To ensure confidence, I ran a knn cv score test, which resulted in approximately 98 percent.

I then performed the KNN classifcation utilizing my k value of 28 and created a confusion matrix. The confusion matrix displayed the actual and false positive predictions for the KNN model. I then calculated the training and testing accuracy of the KNN model, which resulted in 97 and 98 percent respectfully. I also created a visualization for a ROC curve, followed by an accuracy score test for the AUC. I finished by running a summary statistic of our model. The code and calculations can be seen in the provided figures in D3.

D3) Data Analysis Code

Figure 6

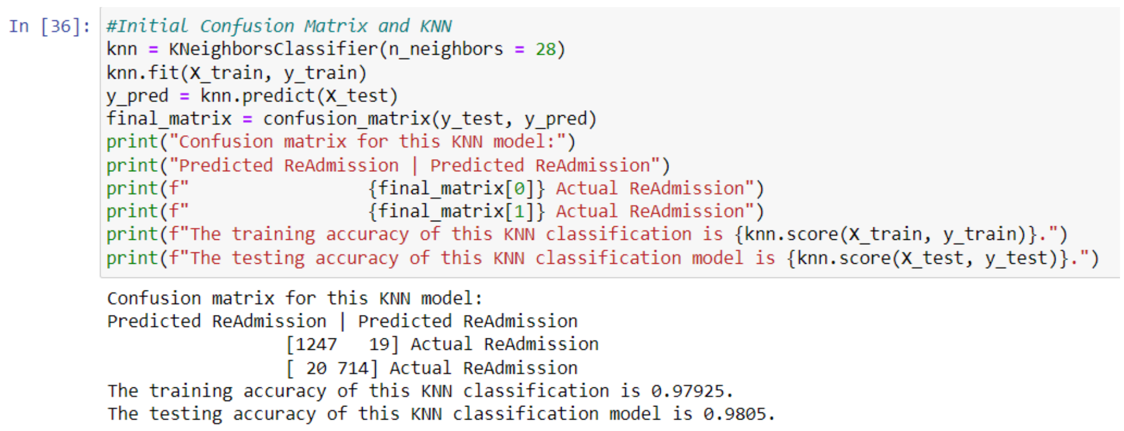
Determing K utilizing GridSearchCV



```
In [31]: #Determing K
         param_grid = {'n_neighbors' : np.arange(1, 50)}
         knn = KNeighborsClassifier()
         knn_cv = GridSearchCV(knn, param_grid, cv=5)
         knn_cv.fit(X_train, y_train)
         knn_cv.best_params_

Out[31]: {'n_neighbors': 28}

In [32]: knn_cv.best_score_

Out[32]: 0.9786250000000001
```

Figure 7

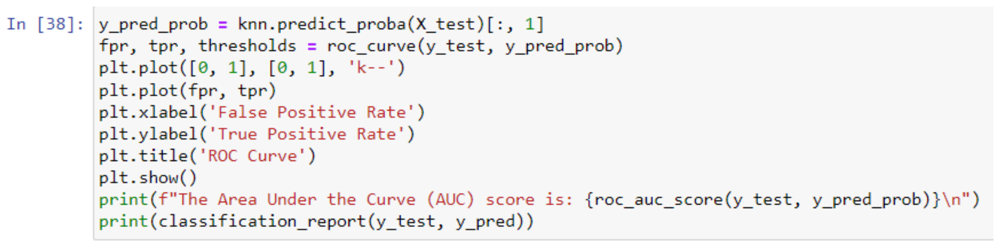
Confusion Matrix and Accuracy Scores



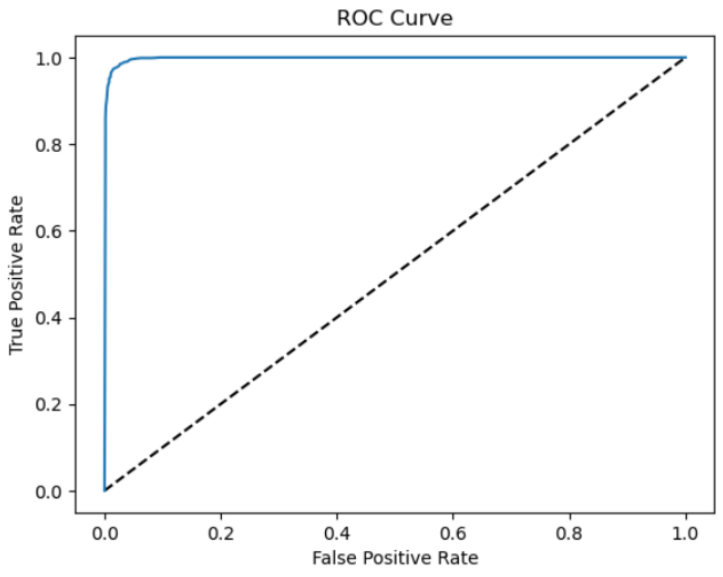
```
In [36]: #Initial Confusion Matrix and KNN
         knn = KNeighborsClassifier(n_neighbors = 28)
         knn.fit(X_train, y_train)
         y_pred = knn.predict(X_test)
         final_matrix = confusion_matrix(y_test, y_pred)
         print("Confusion matrix for this KNN model:")
         print("Predicted ReAdmission | Predicted ReAdmission")
         print(f"                 {final_matrix[0]} Actual ReAdmission")
         print(f"                 {final_matrix[1]} Actual ReAdmission")
         print(f"The training accuracy of this KNN classification is {knn.score(X_train, y_train)}.")
         print(f"The testing accuracy of this KNN classification model is {knn.score(X_test, y_test)}.")

         Confusion matrix for this KNN model:
         Predicted ReAdmission | Predicted ReAdmission
                          [1247   19] Actual ReAdmission
                          [ 20 714] Actual ReAdmission
         The training accuracy of this KNN classification is 0.97925.
         The testing accuracy of this KNN classification model is 0.9805.
```

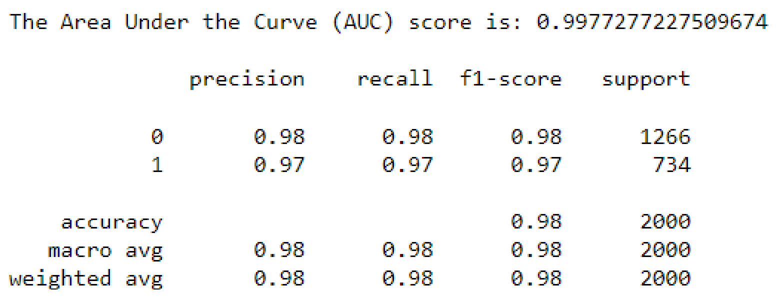
Figure 8

ROC Curve Visualization and Classification Report (Korstanje, 2022)



```
In [38]: 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')
         plt.show()
         print(f"The Area Under the Curve (AUC) score is: {roc_auc_score(y_test, y_pred_prob)}\n")
         print(classification_report(y_test, y_pred))
```





```
The Area Under the Curve (AUC) score is: 0.9977277227509674

              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1266
           1       0.97      0.97      0.97       734

    accuracy                           0.98      2000
   macro avg       0.98      0.98      0.98      2000
weighted avg       0.98      0.98      0.98      2000
```

E1) AUC of Classification Model

As mentioned above, I determined the best parameters for the k value by utilizing the GridSearchCV function. I then ran an accuracy score for this parameter, which resulted in approximately 0.98 or 98 percent. I also performed accuracy tests for our KNN classification model with the training and testing values, this resulted in approximately 97 and 98 percent.

Within our ROC KNN model, there is a central diagonal line that represents a 50 percent classification rate. The Area Under the Curve (AUC) score is a percentage of the model's area beneath the curve that is created by the KNN classification. The higher the percentage equates to a more accurate model. The resulting AUC score of our classification model was approximately 99.8 percent.

E2) Results and Implications

The accuracy scores that were mentioned in E1 provide confidence in our ROC KNN classification model. With a precision score of 98 percent, the model shows it can accurately predict if a patient is readmitted within 30 days of discharge 98 percent of the time with the predictor variables provided. The accuracy scores that resulted from indicates this model as a strong classifier of data that can produce accurate resuts based on our input variables with confidence.

E3) Limitations

One of the main limitations of K-Nearest Neighbor classification is the number of neighbors, or k. Though there are calculations to determine the best k value, there can be large inaccuracies depending on the k value chosen. Intuitively, most would believe that a larger k value would result in more accurate classification as there are more neighbors to classify the data point of interest. However, this may not be the case as a larger number of data points increases the sensitivity of the classification (Guo et al., 2003).

E4) Recommendations

Though our ROC KNN model returned very high accuracy scores, it is difficult to provide recommendations to stakeholders based only on our analysis. The KNN classification does not measure for a causal relationship, it classifies similar factors of input data points that are a certain distance from our data point of interest. I also believe that many of the limitations to our model are due to the data set that I utilized. As I have mentioned in previous courses, I do not believe there are an adequate quality of variables to accurately determine the significant factors for the readmission of patients. I would recommend for stakeholders to add more variables that can better predict what classification of factors that impact the readmission of patients.

G) Third-Party Code References

Korstanje, J. (2022, September 1). *The K-nearest neighbors (knn) algorithm in Python*. Real Python. https://realpython.com/knn-python/

H) References

Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K. (2003). KNN Model-Based Approach in Classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003. Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-39964-3_62

Zhang Z. (2016). Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, *4*(11), 218. https://doi.org/10.21037/atm.2016.0