**Part I: Research Question**

1. Describe the purpose of this data mining report by doing the following:
2. Using a random forest regressor, can the length of a patient’s initial stay be predicted based on the information available at the tie of admittance? Information available at time of admission does not include data on a patient’s initial stay. The variables that fit these criteria include City, State, County, Zip, Lat, Lng, Population, Area, TimeZone, Job, Children, Age, Income, Marital, Gender, VitD\_levels, Soft\_drink, Initial\_admin, HighBlood, Stroke, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic\_rhinitis, Reflux\_esophagitis, and Asthma.
3. One goal of the analysis is to determine a patient’s complication risk based on available information at the time of admission. This classification will allow hospitals to better prioritize the care of admitted patients and potentially reduce readmission rates.

**Part II: Method Justification**

1. Explain the reasons for your chosen classification method from part A1 by doing the following:
2. Random forest regressors determine their predictions by averaging the prediction of multiple decision tree regressors. These decision trees determine their predictions by comparing the values of predictor variables to thresholds determined through the fitting of the model to the data. By combining the predictions of multiple models, the random forest can better predict the value of a patient’s Initial\_days variable.

If the model predicts the target variable perfectly in the test data, the mean squared error of the model’s predictions will be zero and the accuracy score provided by the RandomForestRegressor’s score method will be one. If the model produces a mean squared error or score far from these expected values, we can deduce that the data is not hierarchically organized by the predictor variables as expected.

1. Decision trees, and by extension random forests, assume that predictor variables, called features in the context of decision trees, are categorical or can discretized. Decision trees prefer features that are categorical over numerical features (Rawale, 2018).
2. In Python, we will use the Pandas library to create data frames, divide data, and join data. This library allows for easy access to data based on index, as well as selection of data based on values within variables. This library is a dependency of the modeling library as well. We use the Numpy library for computations, including mean, standard deviation, square rots, and more. This library also provides tools for working with multiple slices of the same Pandas data frame or series and is a dependency of the modeling library. The Sklearn library, specifically the preprocessing and ensemble classes, is included to encode categorical variables into numeric values, normalize numeric data, split the data into training and test data, and create a random forest regressor on the data.

**Part III: Data Preparation**

1. Perform data preparation for the chosen data set by doing the following:
2. Using the pandas get\_dummies method, the non-binary categorical variables will be expanded into multiple binary variables indicating the existence, or lack of, a value for each unique value in the variable. The parameter drop\_first will remove one of the value-variables being created as a value of zero in each of the other newly created variables implies a value of one in the final option. This process is often called one-hot encoding and results in n-1 variables for each non-binary category variable where n is the number of unique values for that variable. Next, the binary categorical variables will be encoded with values of "Yes" being encoded as one and "No" as zero. The ordinal categorical variables are manually encoded, assigning lower values to lower risk values or population densities and higher values to higher risk values or population densities. Because there are only three ordinal variables in our data set, this encoding was easily constructed as a dictionary and applied through the pandas replace method.
3. The following variables will be used as predictor variables:

City (Categorical), State (Categorical), County (Categorical), Zip (Continuous), Lat (Continuous), Lng (Continuous), Population (Continuous), Area (Categorical), TimeZone (Categorical), Job (Categorical), Children (Continuous), Age (Continuous), Income (Continuous), Marital (Categorical), Gender (Categorical), VitD\_levels (Continuous), Soft\_drink (Categorical), Initial\_admin (Categorical), HighBlood (Categorical), Stroke (Categorical), Overweight (Categorical), Arthritis, Diabetes (Categorical), Hyperlipidemia (Categorical), BackPain (Categorical), Anxiety (Categorical), Allergic\_rhinitis (Categorical), Reflux\_esophagitis (Categorical), and Asthma (Categorical).

1. We build the following data preparation steps from the preparation function first presented in Task One of Predictive Modeling.

The data must be divided into four data frames, one for numeric variables, another for non-binary categorical variables, one for binary categorical variables, and another for ordinal categorical variables. The numeric variables will have their outliers defined and offending observations dropped from the data set and the remaining values normalized with sklearn’s MinMaxScaler. Identification and removal of outliers is completed in the block of code highlighted in blue. Normalization is accomplished later in the process in the code highlighted in purple. The non-binary categorical variables will be one-hot encoded using the pandas library’s get\_dummies method, highlighted in orange below. The binary categorical variables will be encoded manually, such that values of "Yes" become one and values of "No" become zero. This task is accomplished in the for loop highlighted in yellow below. The ordinal categorical variables will be manually encoded so that values representing lesser risk or population density are given lower values while values indicating higher risk or population density are given higher values. The dictionary of values to encode and the replacement of values occurs in the code highlighted in green below. Finally, these four data frames are reunited in the code highlighted in red. The full code used to generate these transformations is provided below:

def meddata\_preprocessing(data\_num, data\_enc, data\_yn, target):

for i in range(len(data\_num.columns)):

mean, std = np.mean(data\_num.iloc[:,i]), np.std(data\_num.iloc[:,i])

upper, lower = mean + 3 \* std, mean - 3 \* std

drop = [inx for inx, x in enumerate(data\_num.iloc[:, i]) if x < lower or x > upper]

for d in drop:

if d in data\_num.index:

data\_num = data\_num.drop(d)

#One-hot encode categorical variables

data\_enc = pd.get\_dummies(data\_enc, prefix=data\_enc.columns, drop\_first=True)

for col in range(len(data\_yn.columns)):

for inx, val in enumerate(data\_yn.iloc[:, col]):

if val == 'Yes':

data\_yn.iloc[inx, col] = 1

else:

data\_yn.iloc[inx, col] = 0

#Determine levels of ordinal variables

scale\_mapper = {

"Area" : {

"Rural" : 0,

"Suburban" : 0.5,

"Urban" : 1

},

"Initial\_admin" : {

"Emergency Admission" : 1,

"Observation Admission" : 0.5,

"Elective Admission" : 0

},

"Complication\_risk" : {

"Low" : 0,

"Medium" : 0.5,

"High" : 1

}

}

#Replace values with numerical equivalents specified above

for col in data\_ord.columns:

data\_ord[col] = data\_ord[col].copy().replace(scale\_mapper[col])

mm = MinMaxScaler() #Instantiate the MinMaxScaler method

data\_num[data\_num.columns] = mm.fit\_transform(data\_num) #Normalize the data

#The use of an inner join preserves the dropping of rows performed on data\_num

#The data\_yn dataframe is converted to a numeric datatype, int32, before joining

data\_clean = data\_num.copy().join(data\_enc, how='inner').join(data\_yn.astype('int32'), how='inner').join(data\_ord, how='inner').join(data\_tar, how='inner')

return data\_clean

1. A copy of the cleaned data set is provided as data\_clean.csv.

**Part IV: Analysis**

1. Perform the data analysis and report on the results by doing the following:
2. The sklearn train\_test\_split method is used to divide the data set into training data and test data. We divide the data into the data sets X-train, X\_test, y\_train, and y\_test, where the prefix X represents a matrix of predictor variables, the prefix y represents a vector of target variables, the suffix train represents a dataset used for training, and the suffix test represents a dataset used for model validation and testing. We divide the data so that 80% of the data is retained in the training sets while 20% is reserved for testing by setting the test\_size parameter equal to 0.2. For the model to be reproducible, the random\_state parameter is set to 42. A copy of the divided data is provided in the csv files X\_train, X\_test, y\_train, and y\_test attached to this submission.
3. Sklearn’s RandomForestRegressor method is used to perform a random forest regression analysis. Random forest models are a form of ensemble analysis, meaning multiple individual models are all created, and their outputs combined to create a more accurate prediction. Random forest models are composed of multiple decision tree models, which are a serious of conditions that filter the prediction through a serious of true-or-false evaluations of the values of the predictor variables. The random forest regressor averages the predicted values of these decision trees.
4. X = data\_clean.drop(labels=['Initial\_days'], axis=1).copy()

y = data\_clean['Initial\_days'].copy()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

rf\_reg = RandomForestRegressor()

rf\_reg.fit(X\_train, y\_train)

pred = rf\_reg.predict(X\_test)

print(mean\_squared\_error(y\_test, pred))

rf\_reg.score(X\_test, y\_test)

**Part V: Data Summary and Implications**

1. Summarize your data analysis by doing the following:
2. The mean squared error (MSE) of the model is 761.423. As explained I Part A, we expect a well-performing model to have a MSE of zero. Such a large MSE indicates our model does a poor job fitting the predictor variables and cannot accurately predict the value of a patient’s Initial\_days variable. This hypothesis is further supported by the accuracy score of the model, -0.03099. A negative accuracy score implies that the model predicts the target variable so poorly that guessing a static value would better fit the data than the model does.
3. Because of the values of the MSE and accuracy score, it is clear that this model does a poor job modeling the data. This reality is not without its own revelations, however. Because decision trees attempt to organize data hierarchically, the poor performance of the model implies that the values of the predictor variables in the data set are not hierarchical in nature.
4. The performance of our model may be so poor because of the nature of the data provided to it. As discussed in Part B, the decision trees that make up the random forest regressor prefer predictor variables to be categorical, but a high number of numerical variables were provided.
5. Because of the model’s poor predictive performance, it is our recommendation that hospitals do not attempt to predict the length of a patient’s stay based on the available data at the time of admission. The lack of success for both this model and the model in Task One of this course which both used the data available at time of admission informs us that a patient’s length of stay and potential readmission do not depend on the condition of a patient prior to admission or to their lifestyle, but on the quality of care the patient receives. It would be better for hospitals to focus on providing quality care in a timely manner than trying to prioritize patients because of their preexisting conditions or lifestyle.

**Part VI: Demonstration**

1. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=e522c2f5-9fec-4c3d-b635-adb70162251d>

1. No outside sources were used to generate code for this project and all code was written personally by the student.
2. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Rawale, S. (2018, May 30). *Understanding decision tree, algorithm, drawbacks and advantages.* Medium. Retrieved October 1, 2021, from https://medium.com/@sagar.rawale3/understanding-decision-tree-algorithm-drawbacks-and-advantages-4486efa6b8c3.