Shawn Wheeler

000364855

D209: Data Mining I

Task II

Table of Contents

[A1: Proposal of Question 3](#_Toc166681779)

[A2: Defined Goal 3](#_Toc166681780)

[B1: Explanation of Prediction Method 3](#_Toc166681781)

[B2: Summary of Method Assumption 3](#_Toc166681782)

[B3: Packages or Libraries Set 3](#_Toc166681783)

[C1: Data Preprocessing 4](#_Toc166681784)

[C2: Data Set Variables 4](#_Toc166681785)

[C3: Steps for Analysis 5](#_Toc166681786)

[C4: Cleaned Data Set 5](#_Toc166681787)

[D1: Splitting the Data 8](#_Toc166681788)

[D2: Output and Intermediate Calculations 8](#_Toc166681789)

[D3: Code Execution 12](#_Toc166681790)

[E1: Accuracy and MSE 15](#_Toc166681791)

[E2: Results and Implications 16](#_Toc166681792)

[E3: Limitations 16](#_Toc166681793)

[E4: Course of Action 16](#_Toc166681794)

[F: Panopto Recording 17](#_Toc166681795)

[G: Sources for Third-Party Code 17](#_Toc166681796)

[H: Sources 17](#_Toc166681797)

## **Part I: Research Question**

## A1: Proposal of Question

Can the length of initial days of a hospital stay be predicted along with important features by way of using a random forest method within the medical data set?

## A2: Defined Goal

The goal of this project is to attempt to predict the length of an initial hospital stay (Initial\_days) with the usage of a random forest. At the same time, I will attempt to determine and visualize the important features correlated to Initial\_days. Being able to develop a greater understanding of the potential reasons that affect the initial hospital stay could allow for cost reduction via better human resource management, for instance.

**Part II: Method Justification**

## B1: Explanation of Prediction Method

For this project, random forest was selected as the prediction method. A random forest is a collection of many singular decision trees which produce their own results averaged together. A decision tree is essentially a flowchart that uses outcomes of a number of sequential questions to arrive at a particular answer. Since this project requires a regression, the random forest will average all of the individual decision trees to arrive at a prediction. This allows for greater accuracy and additional mitigation of potential issues for a singular decision tree and will provide insight into the research question.

## B2: Summary of Method Assumption

An assumption of the random forest method is that as it uses a plethora of individual decision trees, the predictions from “the individual trees must have very low correlations.” (Charles, 2023). With higher correlation between trees, the model will have much lower performance and may be useless for attempting to learn more regarding the research question.

## B3: Packages or Libraries List

*Table of Libraries and Packages:*

|  |  |
| --- | --- |
| **Libraries / Packages** | **Usage** |
| pandas | Data manipulation |
| numpy | Mathematics operations |
| seaborn | Creating visualizations |
| matplotlib.pyplot | Creating visualizations |
| stats.models.stats.outliers\_influence | Calculating variance inflation factor (VIFs) |
| sklearn.feature\_selection | SelectKBest to determine appropriate features for model |
| sklearn.model\_selection |  |
| import train\_test\_split | Splitting data into training and testing data sets |
| import GridSearchCV | Determining best variables to use for k-NN model |
| sklearn.neighbors | KNeighborsClassifier to instantiate k-NN computations |
| sklearn.preprocessing | Assisting with calculations by preprocessing data |
| sklearn.metrics |  |
| import mean\_squared\_error | Calculating test/training mean squared error scores |
| import r2\_score | Calculating test/training r2 scores |
| sklearn.model\_selection |  |
| import train\_test\_split | Creating test and training data from input |
| Import GridSearchCV | Searching input to determine optimum values |
| warnings | Ignoring warnings for Panopto video |
| sklearn.ensemble.RandomForestRegressor | Random forest method to attempt to predict values |
| sklearn.tree.plot\_tree | Visualization of singular decision tree in random forest |

**Part III: Data Preparation**

## C1: Data Preprocessing

One preprocessing data preparation goal for this project is the creation of dummy variables. Dummy variables needed to be created in order to be able to use non-binary categorical data in the prediction. They will be created using the getdummies() method, then one of the options will be deleted in order to combat multicollinearity, the values changed to 1s and 0s, and finally, the data type changed to int. These steps allow for the elimination of unusable variables for the random forest method while still being able to make use of that same information for the project’s purposes.

## C2: Data Set Variables

*Variables Chosen to be Tested with SelectKBest:*

|  |  |
| --- | --- |
| **Independent Variables** | **Data Type** |
| Area | Categorical |
| Children | Continuous (Numerical) |
| Marital | Categorical |
| Gender | Categorical |
| Age | Continuous (Numerical) |
| Income | Continuous (Numerical) |
| ReAdmi | Categorical |
| VitD\_levels | Continuous (Numerical) |
| Doc\_visits | Continuous (Numerical) |
| Full\_meals\_eaten | Continuous (Numerical) |
| vitD\_supp | Continuous (Numerical) |
| Initial\_days | Continuous (Numerical) |
| Soft\_drink | Categorical |
| Initial\_admin | Categorical |
| HighBlood | Categorical |
| Stroke | Categorical |
| Complication\_risk | Categorical |
| Overweight | Categorical |
| Arthritis | Categorical |
| Diabetes | Categorical |
| Hyperlipidemia | Categorical |
| BackPain | Categorical |
| Anxiety | Categorical |
| Allergic\_rhinitis | Categorical |
| Reflux\_esophagitis | Categorical |
| Asthma | Categorical |
| Services | Categorical |
| Initial\_days | Continuous (Numerical) |
| Timely\_admission | Continuous (Numerical) |
| Timely\_treatment | Continuous (Numerical) |
| Timely\_visits | Continuous (Numerical) |
| Reliability | Continuous (Numerical) |
| Options | Continuous (Numerical) |
| Hours\_of\_treatment | Continuous (Numerical) |
| Courteous\_staff | Continuous (Numerical) |
| Active\_listening | Continuous (Numerical) |

## C3: Steps for Analysis

1. Load data.

df = pd.read\_csv(“C:/Users/Owner/medical\_clean.csv”)

1. Data exploration.

df.info()

1. Drop columns that will not be used for analysis.

df = df.drop(['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'TimeZone', 'Job', 'TotalCharge', 'Additional\_charges'], axis=1)

1. Check for missing data, duplicates, and examine boxplots for outliers. Outliers were detected but left alone due to the fact that random forest is quite robust against outliers. No missing data or duplicates were detected.

df.isnull().sum()

df.duplicated().value\_counts()

quant\_columns = ['Children', 'Age', 'Income', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten', 'vitD\_supp', 'Initial\_days', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8']

for column in df:

if column in quant\_columns:

plt.figure()

plt.gca().set\_title(column)

df.boxplot([column])

1. Rename columns for usability. VitD\_supp did not have an initial capital V as per the data dictionary and consistency, and the other columns were renamed in order to be better usable for examining data and results.

df = df.rename(columns={'vitD\_supp': 'VitD\_supp', 'Item1': 'Timely\_admission', 'Item2': 'Timely\_treatment', 'Item3': 'Timely\_visits', 'Item4': 'Reliability', 'Item5': 'Options', 'Item6': 'Hours\_of\_treatment', 'Item7': 'Courteous\_staff', 'Item8': 'Active\_listening'})

1. Re-express Booleans as numeric values

df['ReAdmis'] = df['ReAdmis'].map({'Yes': '1', 'No': '0'})

df['Soft\_drink'] = df['Soft\_drink'].map({'Yes': '1', 'No': '0'})

df['HighBlood'] = df['HighBlood'].map({'Yes': '1', 'No': '0'})

df['Stroke'] = df['Stroke'].map({'Yes': '1', 'No': '0'})

df['Overweight'] = df['Overweight'].map({'Yes': '1', 'No': '0'})

df['Arthritis'] = df['Arthritis'].map({'Yes': '1', 'No': '0'})

df['Diabetes'] = df['Diabetes'].map({'Yes': '1', 'No': '0'})

df['Hyperlipidemia'] = df['Hyperlipidemia'].map({'Yes': '1', 'No': '0'})

df['BackPain'] = df['BackPain'].map({'Yes': '1', 'No': '0'})

df['Anxiety'] = df['Anxiety'].map({'Yes': '1', 'No': '0'})

df['Allergic\_rhinitis'] = df['Allergic\_rhinitis'].map({'Yes': '1', 'No': '0'})

df['Reflux\_esophagitis'] = df['Reflux\_esophagitis'].map({'Yes': '1', 'No': '0'})

df['Asthma'] = df['Asthma'].map({'Yes': '1', 'No': '0'})

1. Create heatmap to check for correlation in data to be used.

renamed\_heatmap\_columns = ['ReAdmis', 'Soft\_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis', 'Reflux\_esophagitis', 'Asthma', 'Children', 'Age', 'Income', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten', 'VitD\_supp', 'Initial\_days', 'Timely\_admission', 'Timely\_treatment', 'Timely\_visits', 'Reliability', 'Options', 'Hours\_of\_treatment', 'Courteous\_staff', 'Active\_listening']

fig, ax = plt.subplots(figsize=(10,10))

seaborn.heatmap(data=df[renamed\_heatmap\_columns].corr(), annot=False, cmap='coolwarm', ax=ax)

1. Creation of dummy variables from categorical variables.

# Create dummy variables and add to dataframe

# Area dummies

area\_dummies = pd.get\_dummies(df.Area, columns=['Rural', 'Urban'])

df.insert(1, 'Area\_rural', area\_dummies['Rural'])

df.insert(2, 'Area\_urban', area\_dummies['Urban'])

# Marital dummies

marital\_dummies = pd.get\_dummies(df.Marital, columns=['Divorced', 'Married', 'Widowed'])

df.insert(4, 'Marital\_divorced', marital\_dummies['Divorced'])

df.insert(5, 'Marital\_married', marital\_dummies['Married'])

df.insert(6, 'Marital\_never\_married', marital\_dummies['Never Married'])

# Gender dummies

gender\_dummies = pd.get\_dummies(df.Gender, columns=['Male', 'Female'])

df.insert(7, 'Gender\_male', gender\_dummies['Male'])

df.insert(8, 'Gender\_female', gender\_dummies['Female'])

# Initial\_admin dummies

initial\_admin\_dummies = pd.get\_dummies(df.Initial\_admin, columns=['Elective Admission', 'Emergency Admission'])

df.insert(25, 'Initial\_admin\_elective\_admission', initial\_admin\_dummies['Elective Admission'])

df.insert(25, 'Initial\_admin\_emergency\_admission', initial\_admin\_dummies['Emergency Admission'])

# Complication\_risk dummies

complication\_risk\_dummies = pd.get\_dummies(df.Complication\_risk, columns=['High', 'Low'])

df.insert(28, 'Complication\_risk\_high', complication\_risk\_dummies['High'])

df.insert(29, 'Complication\_risk\_low', complication\_risk\_dummies['Low'])

# Services dummies

services\_dummies = pd.get\_dummies(df.Services, columns=['Blood Work', 'Intravenous', 'CT Scan'])

df.insert(41, 'Services\_blood\_work', services\_dummies['Blood Work'])

df.insert(41, 'Services\_intravenous', services\_dummies['Intravenous'])

df.insert(41, 'Services\_ct\_scan', services\_dummies['CT Scan'])

# Map dummies to numerical values

df['Area\_rural'] = df['Area\_rural'].astype(int)

df['Area\_urban'] = df['Area\_urban'].astype(int)

df['Marital\_divorced'] = df['Marital\_divorced'].astype(int)

df['Marital\_married'] = df['Marital\_married'].astype(int)

df['Marital\_never\_married'] = df['Marital\_never\_married'].astype(int)

df['Gender\_male'] = df['Gender\_male'].astype(int)

df['Gender\_female'] = df['Gender\_female'].astype(int)

df['Initial\_admin\_elective\_admission'] = df['Initial\_admin\_elective\_admission'].astype(int)

df['Initial\_admin\_emergency\_admission'] = df['Initial\_admin\_emergency\_admission'].astype(int)

df['Complication\_risk\_high'] = df['Complication\_risk\_high'].astype(int)

df['Complication\_risk\_low'] = df['Complication\_risk\_low'].astype(int)

df['Services\_blood\_work'] = df['Services\_blood\_work'].astype(int)

df['Services\_intravenous'] = df['Services\_intravenous'].astype(int)

df['Services\_ct\_scan'] = df['Services\_ct\_scan'].astype(int)

# Drop variables used for dummies

df = df.drop(['Area', 'Marital', 'Gender', 'Initial\_admin', 'Complication\_risk', 'Services'], axis=1)

1. Convert data types as needed.

df[['ReAdmis', 'Soft\_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'Reflux\_esophagitis', 'BackPain', 'Anxiety', 'Allergic\_rhinitis', 'Asthma']] = df[['ReAdmis', 'Soft\_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'Reflux\_esophagitis', 'BackPain', 'Anxiety', 'Allergic\_rhinitis', 'Asthma']].astype(int)

1. Select best variables through usage of SelectKBest.

# SelectKBest process

X = df.drop(columns='Initial\_days')

y = df.Initial\_days

# SelectKBest and fit

skbest = SelectKBest(k='all')

X\_new = skbest.fit\_transform(X,y)

# Determining p-values / listing < .05

p\_values = pd.DataFrame({'Feature': X.columns, 'p\_value': skbest.pvalues\_}).sort\_values('p\_value')

p\_values[p\_values['p\_value'] < 0.05]

# Variables to keep (<.05 p-value)

features\_to\_keep = p\_values['Feature'][p\_values['p\_value'] < 0.05]

print(features\_to\_keep)

## C4: Cleaned Data Set

*See Attached .csv:* cleand209datatask2.csv

**Part IV: Analysis**

## D1: Splitting the Data

The data was split into training data and testing data using train\_test\_split() from sklearn.model\_selection at a ratio of 80/20 as seen below. The .csv for each output data set is also attached.

*Split data:*

X\_train, X\_test, y\_train, y\_test= train\_test\_split(X, y, train\_size=.8, test\_size=.2, random\_state=12)

*Attached .csvs:* X\_train.csv, X\_test.csv, Y\_train.csv, Y\_test.csv

pd.DataFrame(X\_train).to\_csv('X\_train.csv')

pd.DataFrame(X\_test).to\_csv('X\_test.csv')

pd.DataFrame(y\_train).to\_csv('y\_train.csv')

pd.DataFrame(y\_test).to\_csv('y\_test.csv')

## D2: Output and Intermediate Calculations

During the process of preparing the data, a heatmap was created to analyze correlation between the potential variables. No combinations showed a correlation high enough to warrant further action.

*Heatmap:*

A screen shot of a computer screen

Description automatically generated

While still preparing the data, non-binary categorical variables were converted into dummy variables. These variables were Area, Marital, Gender, Initial\_admin, Complication\_risks, and Services. At that point, the SelectKBest method was utilized to choose variables. Variables with a p-value less than 0.05 were used for further analysis.

*SelectKBest:*

*A screenshot of a medical form

Description automatically generated*

Next, data was split into 80% training data and 20% testing data to be used for model creation and determining scores. Afterwards, the model was created and trained by instantiating RandomForestGenerator and performing GridSearchCV() for hyperparametric refinement.

*Random forest model fitting:*

*A screenshot of a computer

Description automatically generated*

After preparing that data, a visualization was created of an individual decision tree that was used in the random forest analysis. The screenshot is difficult to make much sense of due to its size, but it is an accurate representation of what was used in this random forest model. Many similar trees were averaged to attempt to predict Initial\_days. The depth of the tree, 11, can be observed which is the cause of the huge amount of data points.

*Singular Decision Tree:*

*A diagram of a structure

Description automatically generated*

*Depth of visual/individual tree*:



Then, the random forest model previously created was examined to determine the best parameters to be used. Also, the mean square error (MSE), root mean square error (RMSE), and R-Squared values for the training set were revealed at the same time.

*Scores for training data:*

A close up of numbers

Description automatically generated

Similarly, the testing data was checked for its prediction accuracy.

*Scores for testing data:*

**

Finally, the model was used to determine the rankings of importance of the selected features that were used in the model. A visualization of those rankings is included, as well.

*Sorted importance scores:*

*A screenshot of a computer program

Description automatically generated*

*Bar graph of comparison of importance of features:*

A graph with a red line

Description automatically generated

## D3: Code Execution

*See attached code:* d209Task2Complete.ipynb

*See code below:*

# Create dummy variables and adding to dataframe

# Area dummies

area\_dummies = pd.get\_dummies(df.Area, columns=['Rural', 'Urban'])

df.insert(1, 'Area\_rural', area\_dummies['Rural'])

df.insert(2, 'Area\_urban', area\_dummies['Urban'])

# Marital dummies

marital\_dummies = pd.get\_dummies(df.Marital, columns=['Divorced', 'Married', 'Widowed'])

df.insert(4, 'Marital\_divorced', marital\_dummies['Divorced'])

df.insert(5, 'Marital\_married', marital\_dummies['Married'])

df.insert(6, 'Marital\_never\_married', marital\_dummies['Never Married'])

# Gender dummies

gender\_dummies = pd.get\_dummies(df.Gender, columns=['Male', 'Female'])

df.insert(7, 'Gender\_male', gender\_dummies['Male'])

df.insert(8, 'Gender\_female', gender\_dummies['Female'])

# Initial\_admin dummies

initial\_admin\_dummies = pd.get\_dummies(df.Initial\_admin, columns=['Elective Admission', 'Emergency Admission'])

df.insert(25, 'Initial\_admin\_elective\_admission', initial\_admin\_dummies['Elective Admission'])

df.insert(25, 'Initial\_admin\_emergency\_admission', initial\_admin\_dummies['Emergency Admission'])

# Complication\_risk dummies

complication\_risk\_dummies = pd.get\_dummies(df.Complication\_risk, columns=['High', 'Low'])

df.insert(28, 'Complication\_risk\_high', complication\_risk\_dummies['High'])

df.insert(29, 'Complication\_risk\_low', complication\_risk\_dummies['Low'])

# Services dummies

services\_dummies = pd.get\_dummies(df.Services, columns=['Blood Work', 'Intravenous', 'CT Scan'])

df.insert(41, 'Services\_blood\_work', services\_dummies['Blood Work'])

df.insert(41, 'Services\_intravenous', services\_dummies['Intravenous'])

df.insert(41, 'Services\_ct\_scan', services\_dummies['CT Scan'])

# Map dummies to numerical values

df['Area\_rural'] = df['Area\_rural'].astype(int)

df['Area\_urban'] = df['Area\_urban'].astype(int)

df['Marital\_divorced'] = df['Marital\_divorced'].astype(int)

df['Marital\_married'] = df['Marital\_married'].astype(int)

df['Marital\_never\_married'] = df['Marital\_never\_married'].astype(int)

df['Gender\_male'] = df['Gender\_male'].astype(int)

df['Gender\_female'] = df['Gender\_female'].astype(int)

df['Initial\_admin\_elective\_admission'] = df['Initial\_admin\_elective\_admission'].astype(int)

df['Initial\_admin\_emergency\_admission'] = df['Initial\_admin\_emergency\_admission'].astype(int)

df['Complication\_risk\_high'] = df['Complication\_risk\_high'].astype(int)

df['Complication\_risk\_low'] = df['Complication\_risk\_low'].astype(int)

df['Services\_blood\_work'] = df['Services\_blood\_work'].astype(int)

df['Services\_intravenous'] = df['Services\_intravenous'].astype(int)

df['Services\_ct\_scan'] = df['Services\_ct\_scan'].astype(int)

# Separate variables

X = df[['Initial\_admin\_emergency\_admission', 'Marital\_divorced', 'Gender\_male', 'Gender\_female', 'Diabetes', 'Initial\_admin\_elective\_admission', 'ReAdmis', 'Services\_ct\_scan', 'Anxiety', 'Stroke', 'Soft\_drink']]

y = df['Initial\_days']

# Split Data

X\_train, X\_test, y\_train, y\_test= train\_test\_split(X, y, train\_size=.8, test\_size=.2, random\_state=12)

# Output to .csvs

pd.DataFrame(X\_train).to\_csv('X\_train.csv')

pd.DataFrame(X\_test).to\_csv('X\_test.csv')

pd.DataFrame(y\_train).to\_csv('y\_train.csv')

pd.DataFrame(y\_test).to\_csv('y\_test.csv')

D2: Perform RandomForestRegressor

# Identify optimal value for parameters using hyperparemetric tuning

# (Course Materials, 2024)

parameters = {"n\_estimators": [10,50,100], "max\_features": [2,3,4], "max\_depth": [8,None]}

# Instantiate the RandomForestRegressor

forest = RandomForestRegressor(random\_state=22)

# perform grid search

gridsearch = GridSearchCV(forest, parameters)

# fit model on training data

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

# Visual

# (Plonski, 2020)

# Fit classifier

forest.fit(X,y)

# Visualize an individual tree

plt.figure(figsize=(5,5))

plot\_tree(forest.estimators\_[0], feature\_names=X.columns, filled=True)

plt.show

print("Depth for Random Forest Visual: ", forest.estimators\_[0].tree\_.max\_depth)

# Values

# Check scores from training data

y\_train\_pred = gridsearch.predict(X\_train)

print("Best params: ", gridsearch.best\_params\_)

print("Training - MSE: ", gridsearch.best\_score\_)

print("Training - RMSE: ", (gridsearch.best\_score\_)\*\*.5)

print("Training - R-squared: ", r2\_score(y\_train, y\_train\_pred))

# Check prediction accuracy for testing data

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

print("Testing - MSE: ", mean\_squared\_error(y\_test, y\_pred))

print("Testing - RMSE: ", mean\_squared\_error(y\_test, y\_pred)\*\*.5)

print("Testing - R-squared: ", r2\_score(y\_test, y\_pred))

# (Course Materials, 2024)

# Create a pd.Series of features importances

importances = pd.Series(data=gridsearch.best\_estimator\_.feature\_importances\_, index=X\_train.columns)

# Sort importances

importances\_sorted = importances.sort\_values()

print(importances\_sorted)

# Draw a horizontal barplot of importances\_sorted

importances\_sorted.plot(kind='barh', color='red')

plt.title('Features Importances')

plt.xlabel('Importance')

plt.ylabel('Features')

plt.show()

**Part V: Data Summary and Implications**

## E1: Accuracy and MSE

Since this was a regression model used, evaluation of accuracy and fit will be conducted by considering the mean square error (MSE), the root mean square error (RMSE), and the R-Squared values for the testing and training data. There is also a table included of the values for each shown below. The MSE describes a calculation where a lower value indicates a more accurate model for prediction purposes. In this case, the testing data had an MSE of 205.088, which is very high, while the training data showed a value of 0.723, which is much closer to 0; therefore, the training model is more accurate based upon that singular metric.

The root mean square error is just the square root of the MSE. It is another measure of accuracy for a regression model with a continuous dependent variable. A lower value for RMSE again indicates a superior model with regards to prediction accuracy. In this instance, the testing data showed an RMSE of 14.321 while the training showed 0.851. Again, this value is much lower and therefore is further proof of the training model having better accuracy in its predictions.

R-Squared was also calculated for the training and testing data. Whereas the other two factors considered focused on accuracy of predictions, R-Squared refers to the model fit of the data itself. A higher value is considered a better fit, and the range is from 0 – 1. For these, the training value for R-Squared was .74 whereas the testing value was .702 which means that the testing model was by a small margin a better model fit; however, they are very close.

In conclusion, based on MSE and RMSE, the training model has far greater predictive accuracy. R-squared, however, shows a slightly worse fit for the training data versus the testing data.

*Table:*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Mean Square Error (MSE)** | **Root Mean Square Error (RMSE)** | **R-Squared** |
| **Training** | 0.723 | 0.851 | 0.74 |
| **Testing** | 205.088 | 14.321 | 0.702 |

## E2: Results and Implications

For this project, a random forest model was created to attempt to answer the research question of whether it’s possible to predict Initial\_days with the given data set while also identifying features that are most important to that model. SelectKBest was used initially to select independent variables with a p-value less than 0.05. Once the variables were selected, the data was split into training and testing data for the random forest model.

After creating and querying the random forest model’s scores, it had an R-squared value of 0.702 which is 70% of its maximum possible value which while being a slightly better fit for the data, is not a great value. The RMSE and MSE, 14.321 and 205.088 respectively, were far away from 0, as well, indicating poor predictive performance. The optimal parameters for the random forest model were also calculated at max\_depth=8, max\_features=4, and n\_estimators=50. Finally, features were compared for level of importance with ReAdmis accounting for 97.3% of the predictive importance among the 11 variables.

The implications of the results of this project would be that the model requires more refinement to have practical use. All of its performance indicators demonstrate poor results. There was an improvement in model fit, but there would need to be more training data and usable variables to input to be able to augment the MSE, RMSE, and R-Squared values further.

## E3: Limitations

One limitation that was encountered during the course of this project is the computational demand is very high. Even with the limited data set of 10,000 rows, it is noticeable in comparison to other methods and processes that have been performed in the course of this program, especially when attempting to visualize the data. At a larger scale which would make the model more usable, it would have a great deal of computational expense.

## E4: Course of Action

Looking at the random forest model that was created, it’s hard to draw any actionable insight from it. With a poor model performance, the best course of action would be to acquire more data overall for a larger sample and to create new variables for which to gather additional data that may relate to the length of an initial hospital stay. After these are collected to an acceptable amount, this project could be performed again in an attempt to gain further insight into predictions of Initial\_days and how the various features affect it.

**Part VI: Demonstration**

## F: Panopto Recording

*See Panopto Link:*

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=dd878ccb-ab95-4fb8-8bc8-b17601180b1d

## G: Sources for Third-Party Code

Course Materials

Płoński, P. (2020, June 29). *How to visualize a single Decision Tree from the Random Forest in Scikit-Learn (Python)?* MLJAR. https://mljar.com/blog/visualize-tree-from-random-forest/

## H: Sources

Charles, Chibuike. (2023, June 2). *Random Forest; Assumptions, Advantages, Disadvantages and Applications.* Medium. https://medium.com/@chibuike.odo.c/random-forest-assumptions-advantages-disadvantages-and-applications-2881f4ea14b6