D209 Task 2

January 26, 2022

1 D209 Task 2: Predictive Modeling

1.1 Malcolm Mikkelsen

1.2 MSDA

1.3 D209: Data Mining 1

1.4 20 January, 2022

2 Part 1: Research Question

Which patients are at a high likelihood of readmission within the fine period? We will investigate this using a decision tree model. ## Goal of Analysis A goal of this analysis is to allow stakeholders to identify patients at high risk for readmission and to begin addressing the variables that most impact the outcome. # Part 2: Method Justification The decision tree classifier algorithm creates branches of if/else statements utilizing both categorical and continuous features. The objective of this is to predict a label, true/false, of whether a patient will be readmitted. Outcomes of this model are branches that represent attributes that allow the model to make its prediction at the end of each branch. ## Libraries and Methods

We will be using the following libraries for analysis:

Pandas Numpy Scikit.learn Seaborn

The first three libraries are needed in our data preparation and analysis steps. We will use Pandas and Numpy throughout the process to aggregate and manipulate the data into the proper format. Scikit.learn will be utilized heavily throughout the analysis phase. We will import and tune our model using methods from this library. Seaborn will help us to create more advanced visualizations.

3 Part 3: Data Preparation

3.1 One Goal of Data Prep

One goal that we will need to accomplish before our analysis is to create dummy variables that recast categorical predictors as numerical values. Our analysis does not accept strings such as 'male' or 'female', but rather needs them to be numerical such as 0 for males. ## Identify Variables for Analysis The features that we are using that are continuous are: • Age • Number of provider visits • Initial Stay length • Total charges • Additional Charges

The categorical predictors are: • Gender • Readmission status • Vitamin Supplementation • Soft drink intake • Stroke • High blood pressure • Complication risk • Overweight • Arthritis

• Diabetes • Back pain • Hyperlipidemia • Anxiety • Allergic Rhinitis • Reflux esophagitis • Asthma • Timely Admittance • Timely treatment • Timely visits • Reliability • Option presented • Hours of service • Active listening by the provider

Our target variable, readmission status, is categorical as well.

3.2 Explain Process

To prepare the data for processing we will follow these steps. First, we will import the data and all libraries and packages listed previously. Then we will drop any column that is not being used for this analysis. This is due to them not being a preexisting condition, or not reflecting the quality and length of their initial admission. Next, we will identify and address null values.For categorical variables we will assume that they were left blank because they do not apply to the patient, thus will be coded as 0. For continuous variables, we will replace the null with the mean value of the column. Lastly, we will save the data set in case of future analyses. In order to detect outliers we will create boxplots of the continous variables. We can see that while meals eaten has outliers, they are still within a normal range and will be left in for the analysis.

[72]:
```
# Import Data and libraries
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.metrics import mean_squared_error as MSE
# Uncleaned data importb
medData = pd.read_csv('medical_raw_data.csv')
```

[73]:
```
# See the columns and header of data
medData.columns
```

[73]:
```
Index(['Unnamed: 0', 'CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City',
       'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area',
       'Timezone', 'Job', 'Children', 'Age', 'Education', 'Employment',
       'Income', 'Marital', 'Gender', 'ReAdmis', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'VitD_supp', 'Soft_drink', 'Initial_admin',
       'HighBlood', 'Stroke', 'Complication_risk', 'Overweight', 'Arthritis',
       'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',
       'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Services',
       'Initial_days', 'TotalCharge', 'Additional_charges', 'Item1', 'Item2',
       'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

```
[74]: medData.head()
```

```
[74]:    Unnamed: 0  CaseOrder Customer_id                           Interaction  \
     0           1          1     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
     1           2          2     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
     2           3          3     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
     3           4          4     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
     4           5          5     C544523  5885f56b-d6da-43a3-8760-83583af94266

                                    UID          City State        County    Zip  \
     0  3a83ddb66e2ae73798bdf1d705dc0932           Eva    AL        Morgan  35621
     1  176354c5eef714957d486009feabf195      Marianna    FL       Jackson  32446
     2  e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD     Minnehaha  57110
     3  cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca  56072
     4  d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William  23181

             Lat  …  TotalCharge  Additional_charges Item1 Item2 Item3  Item4  \
     0  34.34960  …  3191.048774        17939.403420     3     3     2      2
     1  30.84513  …  4214.905346        17612.998120     3     4     3      4
     2  43.54321  …  2177.586768        17505.192460     2     4     4      4
     3  43.89744  …  2465.118965        12993.437350     3     5     5      3
     4  37.59894  …  1885.655137         3716.525786     2     1     3      3

        Item5 Item6 Item7  Item8
     0      4     3     3      4
     1      4     4     3      3
     2      3     4     3      3
     3      4     5     5      5
     4      5     3     4      3

     [5 rows x 53 columns]
```

4 Drop Un-needed Columns and Rename Columns

```
[75]: # Drop columns not used for analysis
      medData = medData.drop(columns=['CaseOrder', 'Customer_id', 'Interaction',␣
       ↪'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area',␣
       ↪'Timezone', 'Job', 'Children','Initial_admin','Education', 'Services',␣
       ↪'Employment', 'Income', 'Marital'])
```

```
[76]: medData = medData.iloc[: , 1:]
```

```
[77]: # Rename survey columns to remember what they are
      medData.rename(columns = {'Item1': 'TimeAdmit',
                               'Item2': 'TimeTreat',
                               'Item3': 'TimeVisits',
                               "Item4": 'Reliability',
                               'Item5': 'Options',
                               'Item6': 'Hours',
                               'Item7': 'Staff',
                               'Item8': 'ActiveListen'},
                               inplace = True)
      medData.columns
```

```
[77]: Index(['Age', 'Gender', 'ReAdmis', 'VitD_levels', 'Doc_visits',
             'Full_meals_eaten', 'VitD_supp', 'Soft_drink', 'HighBlood', 'Stroke',
             'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
             'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
             'Reflux_esophagitis', 'Asthma', 'Initial_days', 'TotalCharge',
             'Additional_charges', 'TimeAdmit', 'TimeTreat', 'TimeVisits',
             'Reliability', 'Options', 'Hours', 'Staff', 'ActiveListen'],
            dtype='object')
```

5 Identify Null Values

```
[78]: # Adress Nulll Values
mdNull = medData.isnull().sum()
print(mdNull)
```

```
Age                   2414
Gender                   0
ReAdmis                  0
VitD_levels              0
Doc_visits               0
Full_meals_eaten         0
VitD_supp                0
Soft_drink            2467
HighBlood                0
Stroke                   0
Complication_risk        0
Overweight             982
Arthritis                0
Diabetes                 0
Hyperlipidemia           0
BackPain                 0
Anxiety                984
Allergic_rhinitis        0
Reflux_esophagitis       0
Asthma                   0
Initial_days          1056
TotalCharge              0
Additional_charges       0
TimeAdmit                0
```

```
TimeTreat             0
TimeVisits            0
Reliability           0
Options               0
Hours                 0
Staff                 0
ActiveListen          0
dtype: int64
```

[79]:

```
# Replace categorical nulls with 0
medData.Anxiety.fillna(0, inplace = True)
medData.Overweight.fillna(0, inplace = True)
medData.Allergic_rhinitis.fillna(0, inplace = True)
medData.Soft_drink.fillna(0, inplace = True)
#Replace remaining Null values with the mean value of the column
medData['Age'] = medData['Age'].fillna((medData['Age'].mean()))
medData['Initial_days'] = medData['Initial_days'].
 ↪fillna((medData['Initial_days'].mean()))

# Adress Nulll Values
mdNull = medData.isnull().sum()
print(mdNull)
```

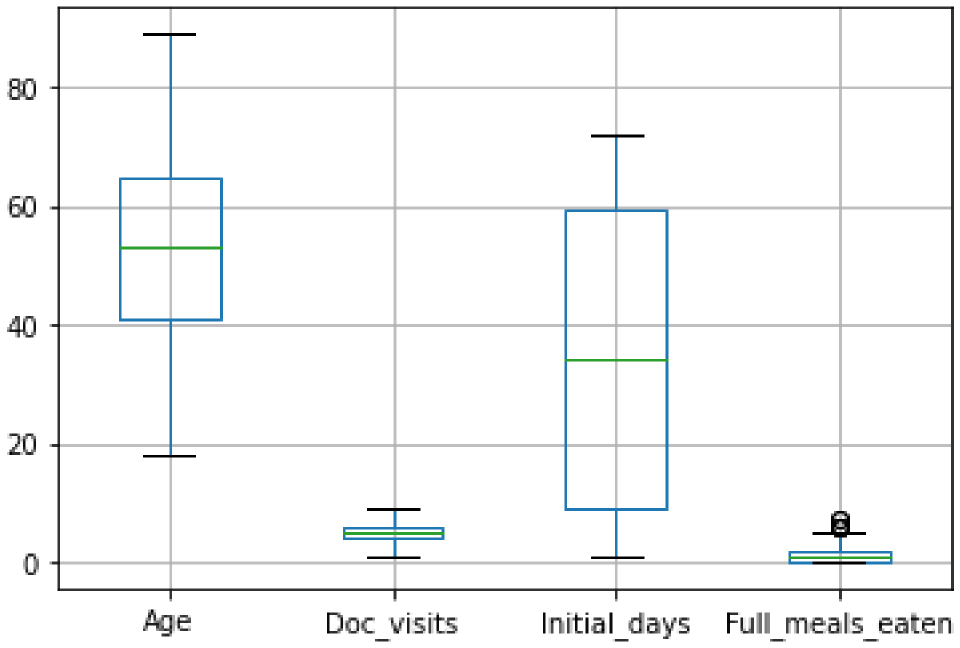
```
Age                   0
Gender                0
ReAdmis               0
VitD_levels           0
Doc_visits            0
Full_meals_eaten      0
VitD_supp             0
Soft_drink            0
HighBlood             0
Stroke                0
Complication_risk     0
Overweight            0
Arthritis             0
Diabetes              0
Hyperlipidemia        0
BackPain              0
Anxiety               0
Allergic_rhinitis     0
Reflux_esophagitis    0
Asthma                0
Initial_days          0
TotalCharge           0
Additional_charges    0
TimeAdmit             0
TimeTreat             0
```

```
TimeVisits          0
Reliability         0
Options             0
Hours               0
Staff               0
ActiveListen        0
dtype: int64
```

6 Outlier Detection

[80]:
```
#Using a boxplot to identify outliers in the rows most likely to contain them
medData.boxplot(['Age', 'Doc_visits', 'Initial_days', 'Full_meals_eaten'])
```

[80]:
```
<AxesSubplot:>
```



7 Create dummy variables

[81]:
```
# Get dummy variables for categorical
medDataDum = pd.get_dummies(medData, drop_first = True)
```

[82]:
```
medDataDum.columns
```

[82]:
```
Index(['Age', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'VitD_supp',
       'Overweight', 'Anxiety', 'Initial_days', 'TotalCharge',
       'Additional_charges', 'TimeAdmit', 'TimeTreat', 'TimeVisits',
       'Reliability', 'Options', 'Hours', 'Staff', 'ActiveListen',
       'Gender_Male', 'Gender_Prefer not to answer', 'ReAdmis_Yes',
       'Soft_drink_No', 'Soft_drink_Yes', 'HighBlood_Yes', 'Stroke_Yes',
       'Complication_risk_Low', 'Complication_risk_Medium', 'Arthritis_Yes',
       'Diabetes_Yes', 'Hyperlipidemia_Yes', 'BackPain_Yes',
       'Allergic_rhinitis_Yes', 'Reflux_esophagitis_Yes', 'Asthma_Yes'],
      dtype='object')
```

8 Move Target Variable to the End

[83]:
```
# Move Dummy Readmit to end
medDataPro = medDataDum[['Age', 'VitD_levels', 'Doc_visits',␣
 ↪'Full_meals_eaten', 'VitD_supp',
        'Overweight', 'Anxiety', 'Initial_days', 'TotalCharge',
        'Additional_charges', 'TimeAdmit', 'TimeTreat', 'TimeVisits',
        'Reliability', 'Options', 'Hours', 'Staff', 'ActiveListen',
        'Gender_Male', 'Gender_Prefer not to answer',
        'Soft_drink_No', 'Soft_drink_Yes', 'HighBlood_Yes', 'Stroke_Yes',
        'Complication_risk_Low', 'Complication_risk_Medium', 'Arthritis_Yes',
        'Diabetes_Yes', 'Hyperlipidemia_Yes', 'BackPain_Yes',
        'Allergic_rhinitis_Yes', 'Reflux_esophagitis_Yes', 'Asthma_Yes',␣
 ↪'ReAdmis_Yes']]
```

9 Save Data File

[84]:
```
# Save Data
medDataPro.to_csv('clean_data_T2_209.csv')
```

10 Part4: Analysis

The analysis begins by creating our feature and predicted variables. Then we will split our data into a test set and a training set. This allows us to test the model on labeled data that it has not seen before. We will then fit our model to the training data and use the fitted model to predict the labels for our test data. We will then use the predicted values and the actual values to determine the mean squared error. This will let us know how far from the actual value our model is. In the case of a categorical variable we expect it to be between 0 and 1, where close to 0 is more accurate and close to 1 is less accurate.

[85]:
```
# Assign Variables to features and target variable
X = medDataPro.drop('ReAdmis_Yes', axis = 1).values
y = medDataPro['ReAdmis_Yes'].values
```

11 Create Test and Train Data Sets

[86]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3,␣
↪random_state = 42)
```

12 Create Decision Tree

[87]:
```
# Instantiate model
dt = DecisionTreeRegressor(max_depth = 10, min_samples_leaf = 0.1, random_state␣
↪= 42)

# Fit Model to train data
dt.fit(X_train, y_train)

# predict values
y_pred = dt.predict(X_test)
```

13 Model Evaluation and MSE Calculations

[88]:
```
dt_mse = MSE(y_test, y_pred)
print('MSE is:', dt_mse)

# Calculate root mean square
dt_rmse = dt_mse ** (1/2)

print('RMSE is: ', dt_rmse)
```

```
MSE is: 0.03617185272935844
RMSE is:  0.19018899213508242
```

14 Part 5: Data Summary and Implications

The MSE for our model was ~ 0.04. This indicates a low rate of error for the model, supporting its predictive power.

The implications of the model are that we can accurately predict whether a patient is likely to be readmitted based on their pre-existing conditions and survey question responses. The stakeholder will benefit from this by being able to follow up with these patients through appointments or other outreach to mitigate the factors that are leading to readmission.

For example, one possible course of action would be to address areas that are consistently being marked as areas of opportunities on the survey.

15 One Model Limitation

One limitation of the model is that they are significantly less effective in predicting continuous variables. This limits what variables we can reliably predict using this model type (Corporate

Finance, 2021).

16 Annotations

Chantal D. Larose, & Daniel T. Larose. (2019). Data Science Using Python and R. Wiley.

Chelaru, M. (1966, October 1). Implementing roc curves for K-nn machine learning algorithm using Python and Scikit learn. Stack Overflow. Retrieved January 21, 2022, from https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-algorithm-using-python-and-sci

Grant, P. (2019, July 21). Introducing K-nearest neighbors. Medium. Retrieved January 19, 2022, from https://towardsdatascience.com/introducing-k-nearest-neighbors-7bcd10f938c5#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%20is%20a%20common%20classification

Finance, C. (2021, September 2). Decision tree. Corporate Finance Institute. Retrieved January 27, 2022, from https://corporatefinanceinstitute.com/resources/knowledge/other/decision-tree/

Narkhede, S. (2021, June 15). Understanding AUC - roc curve. Medium. Retrieved January 21, 2022, from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Pandas Development Team. (2008). pandas.DataFrame.drop — pandas 1.3.0 documentation. Pandas. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.

Python - seaborn.residplot() method. GeeksforGeeks. (2020, August 17). Retrieved December 26, 2021, from https://www.geeksforgeeks.org/python-seaborn-residplot-method/

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Shin, T. (2021, December 4). Understanding multicollinearity and how to detect it in Python. Medium. Retrieved December 18, 2021, from https://towardsdatascience.com/everything-you-need-to-know-about-multicollinearity-2f21f082d6dc