D209 Task 1

January 25, 2022

1 D209 Task 1 Multiple Regression for Predictive Modeling

1.1 Malcolm Mikkelsen

1.2 MSDA

1.3 D209: Predictive Modeling

1.4 20 January, 2022

2 Part 1: Research Question

2.1 1A: Purpose of Data Mining Report

Using the K-Nearest Neighbors method, can we create a model using preexisting conditions and features from their admission, to predict which patients are likely to readmit to the hospital within 30 days of discharge.

2.1.1 Goal of Analysis

This analysis aims to create a model that, with a high level of accuracy, predicts which patients will be readmitted. This will allow our organization to create an action plan to reduce readmissions, thus reducing the fines associated with them.

3 Part 2: Method Justification

3.1 Reason for Using K-Nearest Neighbors Method

By using K-Nearest Neighbor, we expect to be able to have a tuned model that predicts whether a patient is likely to be readmitted to the hospital within the period that results in a fine. This is accomplished by the model comparing the (n) number of nearest data points and selecting an output based on those. One assumption of this model is that similar samples exist near each other within our data set (Grant, 2019).

3.2 Libraries and Packages Used

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.

4 Part 3: Data Preparation

4.1 Perform Data Preparation

One goal for preprocessing will be to create dummy variables for all categorical data. Using the get_dummies method from Pandas allows me to call one line of code that will replace all categorical columns with a numerical value (i.e. 1 for yes, 0 for no). This allows me to use these data points within the KNN model.

The features that we are using that are continuous are: • Age • Vitamin D Levels • 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

4.1.1 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.

5 Import libraries and data

[116]:
```
# Import Data and libraries
import seaborn as sns
import pandas as pd
import numpy as np
import sklearn
from sklearn import datasets
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import auc
```

[117]:
```
medData = pd.read_csv('medical_raw_data.csv')
```

[118]:
```
medData.head()
```

[118]:
```
   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]
```

[119]:
```
medData.columns
```

[119]:
```
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')
```

6 Dropping columns and renaming variables

[120]:
```python
# 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'])
```

[121]:
```python
# Drop unnamed column
medData = medData.iloc[: , 1:]
```

[122]:
```python
# Check remaining columns
medData.columns
```

[122]:
```
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', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5',
       'Item6', 'Item7', 'Item8'],
      dtype='object')
```

[123]:
```python
# 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
```

[123]:
```
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')
```

7 Identify and Adress null values

[124]:
```
# 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
```

[125]:
```
# 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
```

8 Create dummy variables

[126]:
```
# Get dummy variables for categorical
medDataDum = pd.get_dummies(medData, drop_first = True)
```

[127]:
```
medDataDum.columns
```

[127]:
```
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')
```

[128]:
```
# 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

[129]:
```
# Save Data
medDataPro.to_csv('clean_data_209.csv')
```

10 Create dataframes for analysis

[130]:
```
# Create variables for knn
X = medDataPro.drop('ReAdmis_Yes', axis = 1).values
y = medDataPro['ReAdmis_Yes'].values
```

11 Part 4: Perform Data Analysis

11.1 Explain Analysis

For these analyses, we first split the data into testing and training data. This allows us to have labeled test data to run the model with and identify the accuracy of the model. Withholding some data allows us to make sure the model functions on data it has not seen before. Then we import our classifier and create a parameter grid. The parameter grid is used on the grid search cv method, which tests different n values within a range to identify which n has the highest accuracy. For this model that is 19, meaning the model will use the 19 nearest data points to predict the readmission of the patient. Once we have this, we fit the model to our training data and use the fitted model to predict the readmission status of our test data.

12 Split data and provide files

[131]:
```
# Split test and training sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3,␣
↪random_state = 42)
```

[135]:
```
#Create KNN model
knn = KNeighborsClassifier()
knn_cv.fit(X_train,y_train)
```

[135]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,
9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])})
```

[102]:
```
#Create KNN model
knn = KNeighborsClassifier()
# We need to find the best n for the model
param_grid = {'n_neighbors': np.arange(1,50)}
knn_cv = GridSearchCV(knn, param_grid, cv = 5)
knn_cv.fit(X_train,y_train)

print(knn_cv.best_params_)
```

```
{'n_neighbors': 19}
```

[136]:
```
# Create the final model using computed n_neighbors
Finknn = KNeighborsClassifier(n_neighbors = 19)
```

[138]:
```
#fit the model to the data
Finknn.fit(X_train, y_train)
#predict test values
y_pred = Finknn.predict(X_test)
#Find the accuracy
print('Model accuracy score for finished KNN model is: ',accuracy_score(y_test,␣
↪y_pred))
```

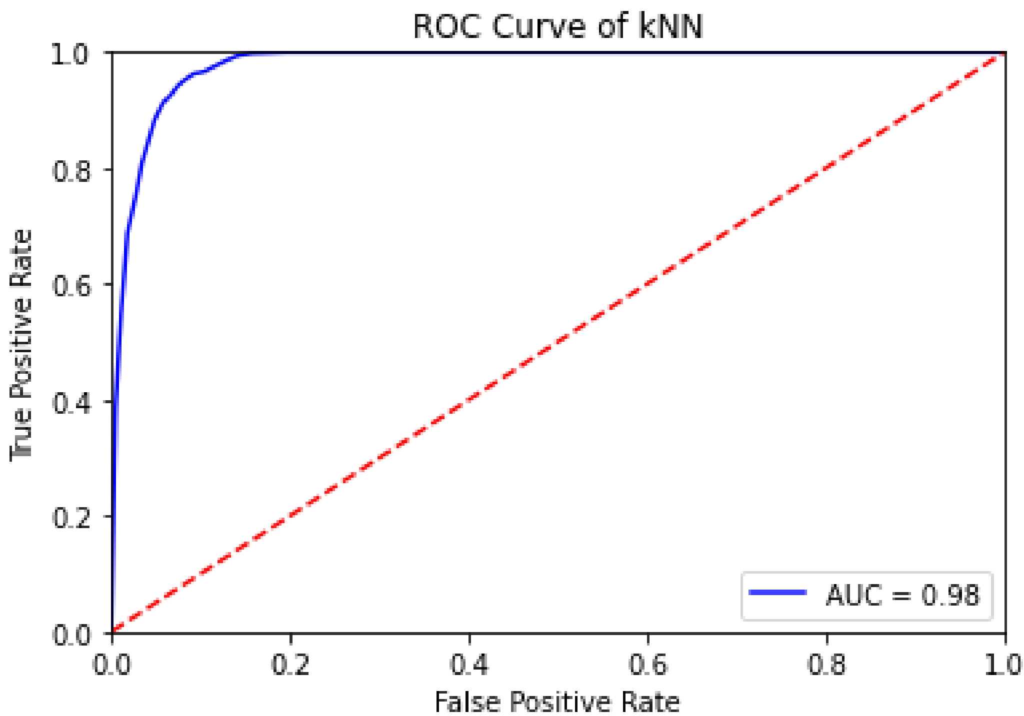
```
Model accuracy score for finished KNN model is:  0.9313333333333333
```

13 AUC-ROC Curve

[139]:
```
y_scores = Finknn.predict_proba(X_test)
fpr, tpr, threshold = roc_curve(y_test, y_scores[:, 1])
roc_auc = auc(fpr, tpr)

plt.title('Auc')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
```

```
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of kNN')
plt.show()
# code utilized here from stack overflow user Mahai Chelaru (Chelaru, 2020)
```



14 Part 5: Summary

14.1 Explain accuracy and area under the curve

The accuracy of our model is .93, and our AUC-ROC curve shows an AUC of 0.98. The AUC provides us an indication of our model's ability to distinguish between patients who are readmitted, vs those who are not. The closer to 1 that we are, the better our model is at detecting patients who were truly readmitted (Narkhede, 2021). For the KNN model, our accuracy is the proportion of correctly predicted outputs, versus the actual output. We can predict correctly 93% of the time whether a patient was going to be readmitted or not.

14.2 One limitation

One limitation of our model is a large amount of processing and memory we need to house our training data. The more data we collect, the more computationally expensive the modeling will get. For example, due to hardware limitations, it took several hours to fully run the model on a dataset of only 10,000 columns. This could become cost-prohibitive if we wish to increase the training data size.

14.3 Results and Implications

Our model strongly implies that the features utilized in this analysis should be addressed as possible by providers during the patient's initial admission. I would recommend that we create education for patients around controllable predictors or predictors that they can work to reduce. For instance, soft drink consumption is a predictor the patient can directly control, and high blood pressure is a predictor that a provider and patients can work to address through treatment. The model also implies that the patients perceived satisfaction with their provider, as shown by the survey questions that were utilized in the analysis, have an impact on whether a patient is readmitted within 30 days. I would recommend that on top of education and addressing preexisting conditions, a study is conducted of which survey questions have the lowest mean score, and that the providers work to increase those categories going forward.

15 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

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