Data Mining I

Jason Willis

College of Information Technology,

Western Governors University

Dr. Eric Straw

February 19th, 2021

Table of Contents for Each Rubric

Part I: Research Question

Describe Purpose, Summarize Research Question and Define Objectives: 3

Part II: Method Justification

Summarize Classification Assumptions and List Python Libraries w/Justifications 7

Part III: Data Preparation

Describe Data Preprocessing, Summary Statistics, Visualizations and Code 8

Part IV: Analysis

Split Data Into Training/Test Data Sets w/Provided Files 15

Describe Analysis Technique w/Screenshots of Calculations 15

Provide Code Used for Classification Analysis from Previous Step 15

Part V: Data Summary and Implications

Explain Accuracy and AUC of Classification Model 16

Discuss Results and Implications of the Classification Analysis 16

Discuss ONE Limitation of the Data Analysis 16

Recommend a Course of Action on Hypothesis and Implications 16

Part VI: Demonstration

Video 17

Sources for Third-Party Code 17

Sources 18

Hospital Readmission Problem

For our chain of hospitals to lower readmission concerns, we need to identify patients who have increased risk of rehospitalization within a month of their release. According to Schuller (2020), non-obese adults were 21% less likely to be readmitted than obese adults. A readmission study by Gert, et. al. (2002) showed a correlation between longer initial hospital stays and readmission. Within the provided dataset, I'm leveraging these studies to help create my hypothetical question and shape my approach in finding potential patient groups with a statistically significant chance for readmission outcomes.

After viewing the provided medical_clean.csv data set and accompanying data dictionary, there seems to be some patient groupings which are aligned with the research mentioned above. For instance, the following patient data fields: Initial patient admin days, Total Charges, and Initial Says (inpatient) both caught my attention and were underscored by the research mentioned above. While my initial feelings towards these variables might make them feel related, are they?

A1 - Research Question

From information about previous patients who were readmitted, can we predict which patients are likely to be readmitted in the future?

A2 - Defined Goal

The goal of our analysis is to logically investigate the provided data set and, with evidence, support or reject the hypothesis. Some data will need to be converted from categorical to numerical data types prior to processing. Our objective is to see how, if at all, any patient's data correlate with potential readmission.

B1 – Explanation of Classification Method

According to Bruce (2020) "The K-Nearest Neighbors method classifies a record in accordance with how similar records are classified." As new records are introduced, they are compared to *k* records which represent similar features. The "classification" part of the process is figuring out the majority representation or grouping among similar records and assign this new record accordingly. Essentially, a decision boundary is created and as new data is input into the model, depending on where the data point is located in reference to the boundaries (based on the k value), they are compared and grouped accordingly.

B2 – Summary of Method Assumption

One assumption of the KNN model is on how the algorithm associate's similarity via distance (Euclidian). Meaning, the closer a data points proximity assumes stronger similarity, while the data points further away represent increased dissimilarity. Additionally, the scikit-learn API requires the data is either a NumPy array or Pandas Dataframe and that the features are continuous values without nulls.

B3 – Packages or Libraries List

The following Python libraries were used followed by their corresponding reason for use:

- Pandas – Used to import dataset and data analysis tasks.
- Numpy – Used for describing the data set and computing distances in KNN.
- Matplotlib – Used for viewing the testing and actual data as a scatter plot.
- Seaborn – Used for creating a heatmap when looking for null values in the original dataset and ggplot style graph matrix to help visualize univariate data.
- Sklearn – Used for preprocessing, model splitting and KNN tasks.

C1 – Data Preprocessing

A preprocessing goal achieved for this model was standardization, as seen below in Figure 1 using .scale() from sklearn. Standardization helps by formatting variables on similar scales (subtracting the mean and dividing the standard deviation) as to reduce variables from being a dominant influence to the model.

Preprocessing - Scaling

```
1 from sklearn.preprocessing import scale
2 Xs = scale(X)
3 Xs_train, Xs_test, y_train, y_test = train_test_split(Xs, y, test_size=0.2, random_state=42)
4 knn_model_2 = knn.fit(Xs_train, y_train)
5 print('k-NN score for test set: %f' % knn_model_2.score(Xs_test, y_test))
6 print('k-NN score for training set: %f' % knn_model_2.score(Xs_train, y_train))
7 y_true, y_pred = y_test, knn_model_2.predict(Xs_test)
8 print(classification_report(y_true, y_pred))
```

```
k-NN score for test set: 0.979000
k-NN score for training set: 0.979000
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      1291
           1       0.97      0.97      0.97       709

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

Figure 1 - Preprocessing Goal: Scaling

C2 – Data Set Variables

A correlation matrix helped to prune the data frame to three variables used to perform the analysis for classification:

- Initial_days – continuous
- TotalCharge – continuous
- ReAdmis_Yes – continuous, (ReAdmis (categorical) converted to Bool)

```
correlation_matrix = pruned_df.corr()

print(correlation_matrix["ReAdmis_Yes"] > 0.5)
```

```
Initial_days     True
TotalCharge      True
ReAdmis_Yes      True
Name: ReAdmis_Yes, dtype: bool
```

Figure 2 - Correlation Matrix Showing Results Above 0.5

C3 Steps for Analysis

Initially, the dataset was loaded using pd.read_csv('medical_clean.csv') and a data frame was created. Some exploratory data analysis was performed to familiarize myself to the data, look for missing values and view data statistics using df.describe(). Since my target variable was initially a categorical data series, I converted this column to integers (0,1). I viewed univariate data using a ggplot style matrix and matplotlib. A correlation matrix identified highly correlated data with our target "ReAdmis_Yes" for classification. Next, this data set was reduced to contain only the highly correlated variables: Initial_days, TotalCharge and ReAdmis_Yes series (Figure 2), with a shape of 10,000 rows by 3 columns.

C4 – Cleaned Data Set

The cleaned data set was saved to "final_cleaned_dataset.csv".

D1 – Splitting the Data

Next, the data was split into training (80%) and testing sets (20%) to evaluate the model. Then, tuned for an optimal k of 46 nearest neighbors to compare (Figure 3). The random state

was set to 73 for reproducibility. Next, the data was fit to the training dataset X_train, and y_train and contains information needed to make predictions on new datapoints.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

knn = KNeighborsClassifier(n_neighbors=46)  # From Fit kNN Regression Below

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3,
    random_state=73,
    stratify=y)  # stratisfy reflects labels of data for both train and test
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred));
```

Figure 3 - Split and Tune Data for Optimal k

The following data sets are provided:

- medical_clean.csv (original format)
- final_cleaned_dataset.csv
    - These three columns have one target and two predictor series, this file is in a pre-split format
- X_test_data.csv
- X_train_data.csv
- y_test_data.csv
- y_train_data.csv

D2 – Output and Intermediate Calculations

Initial accuracy of the KNN model was 97.6%. After scaling the data (Figure 1), this increased to 98% accuracy. The classification report (precision, recall, f1-score) increased as did the knn.score() with scaling.

Originally, k = 20 was used but later k=2 was found to be an optimal. As a test, 3 of the closest neighbors to "test_data" are analyzed (using ".argsort()"), also seen in Figure 4. Next, the data is standardized to compare variables with a similar scale (Figure 5).

The Area Under the Curve was calculated next, as 99.1% (Figure 4). Classification predictions were "No" or 0 = 1907 while "Yes" or 1 was 1093 from the testing data. The Random Forest Classifier was fit and scored the scaled test data at 98%.

```
from sklearn.metrics import roc_curve
y_pred_prob = knn.predict_proba(Xs_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='KNN Regression')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('KNN ROC / AUC Curve')
plt.show();
```

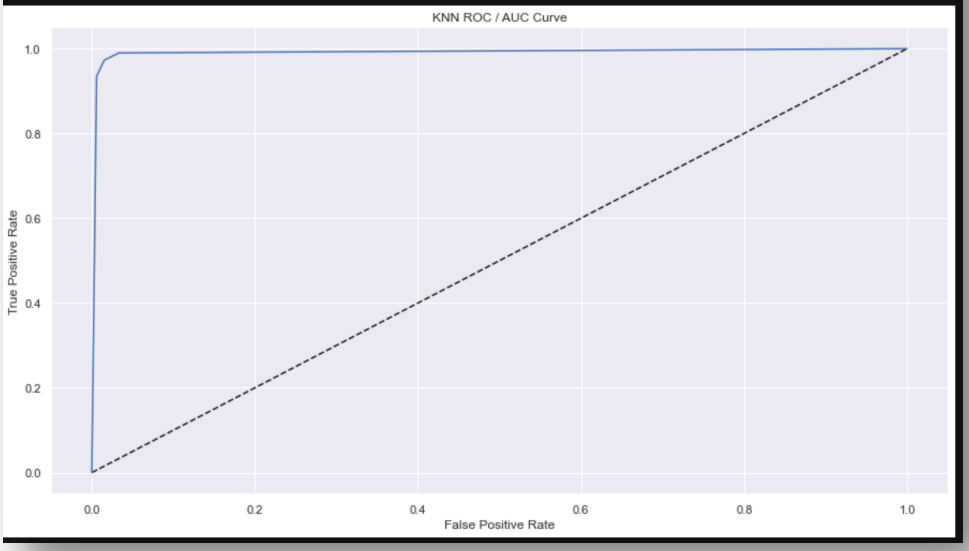
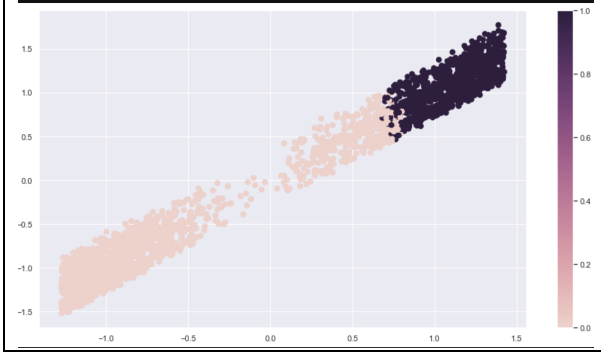


Figure 4 - KNN Set Up Using NumPy Arrays

Mean Absolute Error (MAE) and Mean Squared Errors (MSE) were calculated on train and test data at 0.0197 and 0.0138 respectively. Next, I plotted the fit of the model, using predicted and actual data from the split (Figure 7).

Predicted

```
# Predicted
import seaborn as sns
cmap = sns.cubehelix_palette(as_cmap=True)
f, ax = plt.subplots()
points = ax.scatter(
    Xs_test[:, 0], Xs_test[:, 1], c=test_preds, s=50, cmap=cmap
)
f.colorbar(points)
plt.show()
```



Actual

```
# Actual
import seaborn as sns
cmap = sns.cubehelix_palette(as_cmap=True)
f, ax = plt.subplots()
points = ax.scatter(
    Xs_test[:, 0], Xs_test[:, 1], c=y_test, s=50, cmap=cmap
)
f.colorbar(points)
plt.show()
```

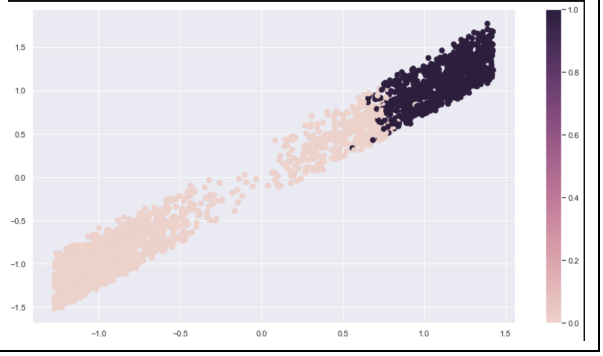


Figure 5 - Plotting Predicted and Actual Data

D3 – Code Execution

Code is located in the "JWillis_D209_Data_Mining_PA1.ipynb" document.

E1 – Accuracy and AUC

The Area Under the Curve or AUC measures performance of classification. This determination helps understand a measure of separability between possible outcome classes, e.g. a high AUC means predicted zeros have a higher probability of actually being zeros and therefore predicted ones have a higher probability of actually being ones. When looking at a visual, the more area under the curve, the more accurate the model. In our model graph (Figure 4) you can see this seems very good. How good? 99.13% (Figure 6)

```
1  from sklearn.metrics import roc_auc_score
2
3  y_pred_prob = knn.predict_proba(Xs_test)[:,1]
4  roc_auc_score(y_test, y_pred_prob)
```

```
0.9913121730018046
```

Figure 6 - AUC

E2 – Results and Implications

The model is highly accurate. Looking at Figure 5, you can see that each plotted point is a patient from the data set. As the patient's initial stay is longer and their total charge is higher, the chance that they are readmitted is closer to "Yes".

E3 – Limitation

One limitation is also a benefit: simplicity. KNN relies on established groupings and couldn't thrive on new and undefined data sets where groupings are unknown, e.g. an predicting undiscovered diseases.

E4 – Course of Action

Our model is highly predictive of patient readmissions rates. My recommendation would be to focus on researching and identifying patient's threshold criteria's for initial days and total charges as these predictor variables helped predict readmission with a high accuracy rate. By focusing on these predictors, we should be able to predict patients that will have a higher likelihood of readmission within 30 days.

F – Panopto Demonstration

Panopto video Will be uploaded once report is finalized.

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=5065042a-a228-4568-85f2-ae42001f8c66

G – Sources for Third-Party Code

- Help using Markdown: https://www.markdownguide.org/basic-syntax/
- Help to see ALL columns: https://stackoverflow.com/questions/24524104/pandas-describe-is-not-returning-summary-of-all-columns
- Help to create a better histogram design: https://mode.com/example-gallery/python_histogram/
- Matplotlib Help: https://matplotlib.org/2.1.2/api/_as_gen/matplotlib.pyplot.plot.html
- Multiple ways to conduct ANOVA: https://www.marsja.se/four-ways-to-conduct-one-way-anovas-using-python/
- Numpy Help: https://numpy.org/doc/stable/
- Pandas Help: https://pandas.pydata.org/docs/user_guide/index.html#user-guide
- Python Help: https://docs.python.org/3.9/library/index.html
- Scipy.stats Help: https://docs.scipy.org/doc/scipy/reference/tutorial/stats.html

References

Bowne-Anderson, H. (2016). Preprocessing in Data Science (Part 1): Centering, Scaling, and KNN *DataCamp*.com. https://www.datacamp.com/community/tutorials/preprocessing-in-data-science-part-1-centering-scaling-and-knn

Bruce, P., C., & Bruce, A (2020). Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python. O'Reilly Media. https://learning.oreilly.com/library/view/practical-statistics-for/9781492072935/titlepage01.html

Gert P Westert, Ronald J Lagoe, Ilmo Keskimäki, Alastair Leyland, Mark Murphy, An international study of hospital readmissions and related utilization in Europe and the USA, Health Policy, Volume 61, Issue 3, 2002, Pages 269-278, ISSN 0168-8510, https://doi.org/10.1016/S0168-8510(01)00236-6. (https://www.sciencedirect.com/science/article/pii/S0168851001002366)

Korstanje, J., The k-Nearest Neighbors (kNN) Algoritm in Python (2021). Real Python, https://realpython.com/knn-python/#knn-is-a-supervised-learner-for-both-classification-and-regression

Larose, D., C., & Larose, D., T. (2019). Data Science Using Python and R. Wiley. https://www.wiley.com/en-us/Data+Science+Using+Python+and+R-p-9781119526810

Schuller K. A. (2020). Is obesity a risk factor for readmission after acute myocardial infarction? *Journal of healthcare quality research*, *35*(1), 4–11. https://doi.org/10.1016/j.jhqr.2019.09.002