**PERFORMANCE ASSESSMENT TASK 1:**

**D209 – Data Mining I**

Michael Lawson

Western Governors University

D209 – Data Mining I

Festus Elleh

03/20/2022

Michael C. Lawson | Student ID: #001479557 | Program: MS Data Analytics (5/1/21) |

Program Mentor: Emil Stoica | 859-265-1888 (EST) | [mlaws77@wgu.edu](mailto:mlaws77@wgu.edu)

**Part I: Research Question**

A.

1. One question and one classification method.

Is a patient going to be readmitted to the hospital within a month of release?

The k-nearest neighbor (KNN) classification method will be used.

2.  One goal of the data analysis.

The goal of the data analysis will be to classify patients by whether they are readmitted within a month or release from the hospital and to predict whether a patient will be readmitted. In the medical data set, the column ‘ReAdmis’ reveals which patients were readmitted, and KNN will label new data coming into the data set as a readmitted patient or not.

**Part II: Method Justification**

B.  1.  Explain how KNN analyzes and expected outcomes.

K-nearest neighbor is a supervised learning method used for classification and regression. Objects are compared to other objects with known classification. The objects with known classification are weighted by the distance from the object being considered. Finally, the algorithm determines how the incoming object should be labeled according to the selected number of nearest neighbors by their weight and count. In the medical data set, the ‘ReAdmis’ column will either be labeled with “0” for patients not readmitted or ”1” for readmits.

2.  One assumption of KNN.

KNN assumes that similar objects are near each other (Harrison, 2018).

3.  Python Packages.

**Libraries used**

1. **Pandas** – fast and powerful data analysis and manipulation library.
2. **NumPy** – wide range of math functions.
3. **Matplotlib**.**pyplot –** visualizations of data.
4. **sklearn** – efficient tools for predictive analysis. Most of the packages specific to this analysis will come from this library.
5. **Warnings.filterwarnings -**  loaded to remove filter warnings

**Packages of sklearn used**

* 1. Linear\_model
     1. LogisticRegression – creates a logistic regression model.
  2. Model\_selection
     1. Train\_test\_split – used to divide rows into training and test groups.
     2. GridSearchCV – used for hyperparameter tuning.
     3. Cross\_val\_score – used to measure the performance of the model across the entire data set by changing the objects selected for the test group.
  3. Neighbors
     1. KNeighborsClassifier – to perform the k-nearest neighbors function.
  4. Metrics (All self-explanatory. These are used to measure knn performance)
     1. Classification\_report
     2. Confusion\_matrix
     3. Accuracy\_score
     4. Roc\_curve
  5. Linear\_model
     1. Lasso – knn feature selection.
     2. Ridge – regularized regression.

 **Part III: Data Preparation**

C.  1.  One preprocessing goal.

Preprocessing for k-nearest neighbors will include changing all object data to numerical values. For instance, all the columns with the binary output of “No”, ”Yes” will be respectively changed to “0”, ”1”.

2.  Initial variables classified as categorical or continuous.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ReAdmis |  | Categorical | Additional\_charges | Continuous |
| Children |  | Categorical | Item1 | Categorical |
| Age |  | Categorical | Item2 | Categorical |
| Income |  | Continuous | Item3 | Categorical |
| VitD\_levels |  | Continuous | Item4 | Categorical |
| Doc\_visits |  | Categorical | Item5 | Categorical |
| Full\_meals\_eaten |  | Categorical | Item6 | Categorical |
| vitD\_supp |  | Categorical | Item7 | Categorical |
| Soft\_drink |  | Categorical | Item8 | Categorical |
| HighBlood |  | Categorical | Services\_Blood Work | Categorical |
| Stroke |  | Categorical | Services\_CT Scan | Categorical |
| Complication\_risk |  | Categorical | Services\_Intravenous | Categorical |
| Overweight |  | Categorical | Services\_MRI | Categorical |
| Arthritis |  | Categorical | Marital\_Divorced | Categorical |
| Diabetes |  | Categorical | Marital\_Married | Categorical |
| Hyperlipidemia |  | Categorical | Marital\_Never Married | Categorical |
| BackPain |  | Categorical | Marital\_Separated | Categorical |
| Anxiety |  | Categorical | Marital\_Widowed | Categorical |
| Allergic\_rhinitis |  | Categorical | Gender\_Female | Categorical |
| Reflux\_esophagitis |  | Categorical | Gender\_Male | Categorical |
| Asthma |  | Categorical | Gender\_Nonbinary | Categorical |
| Initial\_days |  | Continuous | Initial\_admin\_Elective Admission | Categorical |
| TotalCharge |  | Continuous | Initial\_admin\_Emergency Admission | Categorical |
|  |  |  | Initial\_admin\_Observation Admission | Categorical |

3.  Preparation steps and code.

*# Import all possible packages useful for KNN analysis*

**import** numpy **as** np

**import** pandas **as** pd

**import** sklearn

**import** matplotlib.pyplot **as** plt

**from** sklearn.model\_selection **import** train\_test\_split, GridSearchCV, cross\_val\_score

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn **import** metrics

**from** sklearn.metrics **import** classification\_report, confusion\_matrix, accuracy\_score, roc\_curve

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.linear\_model **import** Lasso, Ridge

**import** warnings

warnings**.**filterwarnings("ignore")

*#Import data set from hard drive*

med **=** pd**.**read\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D208\medical\_clean.csv", skiprows**=**0, delimiter**=**",")

*#Examine the list of variables, their data type and the shape of the data*

med**.**info()

*#print the header*

med**.**head()

*#change the values of the ordinal categorical variable to numeric ranks*

med['Complication\_risk']**.**unique()

med['Complication\_risk'] **=** med['Complication\_risk']**.**map({'Low':0,'Medium':1,'High':2})

*#change all binary categorical values to 0,1*

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

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

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

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

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

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

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

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

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

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

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

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

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

*#Create dummy columns for the Services column*

med **=** pd**.**get\_dummies(med, prefix**=**['Services','Marital','Gender', 'Initial\_admin'], prefix\_sep**=**'\_', dummy\_na**=False**, columns**=**["Services","Marital","Gender","Initial\_admin"])

*#Check whether dummy columns were created*

print(med**.**keys())

*#Create Data and Target selections for X and y*

X **=** med**.**drop(axis**=**1, columns**=**['CaseOrder', 'Interaction', 'Customer\_id', 'UID', 'State','County', 'Zip', 'City', 'Lat', 'Lng', 'Population', 'TimeZone', 'Area', 'Job', 'ReAdmis',])

y **=** med['ReAdmis']

4.  Cleaned data set.

*#Export prepared data*

med**.**to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D209\medical\_prepared\_D209\_T2.csv")

**Part IV: Analysis**

D.  Perform data analysis and report results:

1.  Split data and provide the files.

*#Split data for 70% training, 30% test*

X\_train,X\_test,y\_train,y\_test**=**train\_test\_split(X,y,test\_size**=**0.3, random\_state**=**944, stratify**=**y)

#Provide split data files

Training\_file = pd.concat([X\_train,y\_train])

Test\_file = pd.concat([X\_test,y\_test])

Training\_file.to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D209\D209\_T1\_training\_data.csv")

Test\_file.to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D209\D209\_T1\_test\_data.csv")

2.  Analysis technique.

The k-nearest neighbors method analyzes the data and predicts which category that object belongs to. The accuracy of the KNN ran on “ReAdmis” was 92%.

A confusion matrix revealed that 2,755 predictions were true to the objects actual category and 245 predictions were false.

Table

Description automatically generated

KNN Accuracy: 0.9183333333333333

The ROC curve reveals how accurate the KNN model is visually.

Chart, line chart

Description automatically generated

The output of the area under the curve (AUC) using cross-validation revealed that KNN is very strong at predicting true positives.



Using Lasso for feature selection revealed that no single explanatory variable is best at predicting which patients will be readmitted.

A picture containing table

Description automatically generated

Hyperparameter tuning revealed that 3 is the best parameter to use for ‘n\_neighbors’ in KNN.

Graphical user interface, text, application

Description automatically generated

Plotting the accuracy of test and train on the same graph against varying parameters for neighbors gave visual confirmation of previous method.

Graphical user interface, application

Description automatically generated

3. Code

*#Perform KNN*

knn **=** KNeighborsClassifier(n\_neighbors**=**5)

knn**.**fit(X\_train,y\_train)

y\_pred **=** knn**.**predict(X\_test)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test,y\_pred))

#Create logistic regression model

logreg **=** LogisticRegression()

logreg**.**fit(X\_train,y\_train)

y\_pred **=** logreg**.**predict(X\_test)

*#ROC curve*

y\_prob **=** logreg**.**predict\_proba(X\_test)[:,1]

fpr,tpr, thresholds **=** roc\_curve(y\_test,y\_prob)

plt**.**plot([0,1],[0,1],'k--')

plt**.**plot(fpr,tpr,label**=**'Log Reg')

plt**.**ylabel('True + Rate')

plt**.**xlabel('False + Rate')

plt**.**title('Log Reg ROC Curve')

plt**.**show();

logreg**.**predict\_proba(X\_test)[:,1]

*#Calculate cross validation score*

cv\_scores **=** cross\_val\_score(logreg,X,y,cv**=**5, scoring**=**'roc\_auc')

print(cv\_scores)

*#Calculate KNN accuracy*

print('KNN Accuracy:', accuracy\_score(y\_test,y\_pred))

*#Use Lasso for feature selection*

sns**.**set(rc **=** {'figure.figsize':(22,6)})

names **=** X**.**columns

lasso **=** Lasso(alpha**=**0.43, normalize**=True**)

lasso\_coef **=** lasso**.**fit(X,y)**.**coef\_

\_**=**plt**.**plot(range(len(names)), lasso\_coef)

\_**=**plt**.**xticks(range(len(names)), names, rotation**=**60)

\_**=**plt**.**ylabel('Coefficients')

plt**.**show()

*#Regularized regression score*

ridge**=** Ridge(alpha**=**0.1, normalize**=True**)

ridge**.**fit(X\_train,y\_train)

ridge\_pred **=** ridge**.**predict(X\_test)

ridge**.**score(X\_test,y\_test)

*#find the best number for ‘n-neighbors’*

param\_grid **=** {'n\_neighbors': np**.**arange(1,46)}

knn\_cv **=** GridSearchCV(knn, param\_grid, cv**=**5)

knn\_cv**.**fit(X\_train,y\_train)

knn\_cv**.**best\_params\_

*#find the best possible cross validation score*

knn\_cv**.**best\_score\_

*# Create arrays for accuracy of train and test*

neighbors **=** np**.**arange(1, 53)

train\_accuracy **=** np**.**empty(len(neighbors))

test\_accuracy **=** np**.**empty(len(neighbors))

*# Loop various values of k*

**for** i, k **in** enumerate(neighbors):

*# Create k-NN classifier*

knn **=** KNeighborsClassifier(n\_neighbors**=**k)

*# Fit the classifier to the training data*

knn**.**fit(X\_train, y\_train)

*#Training set accuracy*

train\_accuracy[i] **=** knn**.**score(X\_train, y\_train)

*#Testing set accuracy*

test\_accuracy[i] **=** knn**.**score(X\_test, y\_test)

*# Compare train and test data by plotting*

plt**.**title('KNN: Varying Number of Neighbors')

plt**.**plot(neighbors, test\_accuracy, label **=** 'Testing Accuracy')

plt**.**plot(neighbors, train\_accuracy, label **=** 'Training Accuracy')

plt**.**legend()

plt**.**xlabel('Number of Neighbors')

plt**.**ylabel('Accuracy')

plt**.**show()

*#Export prepared data*

med**.**to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D209\medical\_prepared\_D209\_T2.csv")

**Part V:**

E.  Summarize analysis:

1.  Accuracy and AUC.

The classification model has an accuracy of 92%, meaning the model incorrectly labels objects 8% of the time.

The ROC plots the true positive predictions against the false positive predictions at various thresholds. The curve arches up further in the direction of true positives as the model gets stronger. This creates a larger area under the curve; therefore, the model is better as the area under the ROC curve increases.

The AUC was calculated using 5-fold cross-validation with 5 outputs: 1,1,1, 0.9978079, & 0.95907749. The output represents percentages, with 1 being 100% and a perfect model. The cross-validation method examines the entire data set by changing the test data until all objects have been included in the test data at least once. This KNN model is strong with three test sets sending back 100% AUC.

2.  Results.

The model is very strong, but not 100% accurate in predicting whether a patient is a readmitted patient.

3.  One limitation of data analysis.

The k-nearest neighbors method requires a parameter be used for how many neighbors are compared to the object. When this parameter is changed, the model can change dramatically in accuracy.

4.  Recommend a course of action.

The model is strong enough to look further into what variables are predicting whether a patient is a readmit. Several different KNN models should be created to find whether readmits can be categorized using less information. The goal of the continued analysis would be to find more ways of predicting which patients will be readmitted and help reduce associated patient complications and fines.

**Part VI: Demonstration**

F.  Panopto video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=20dcf62a-9440-4ba6-9bb0-ae5f0008514f

**References**

Harrison, Onel (Sep 10, 2018). *Machine Learning Basics with the K-Nearest Neighbors Algorithm.*

https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761#:~:text=The%20KNN%20algorithm%20assumes%20that,of%20a%20feather%20flock%20together.%E2%80%9D&text=Notice%20in%20the%20image%20above,are%20close%20to%20each%20other.

Datacamp (2022) *Supervised Learning with scikit-learn.* Hyperparameter tuning with GridSearchCV. https://campus.datacamp.com/courses/supervised-learning-with-scikit-learn/fine-tuning-your-model?e x=10