**Jason Pena**

**WGU University**

D209 PERFORMANCE ASSESSMENT NVM2 TASK 1

CLASSIFICATION ANALYSIS

**Part I: Research Question**

1. Describe the purpose of this data mining report by doing the following:
2. Propose one question relevant to a real-world organizational situation that you will answer using one of the following classification methods:

* K-nearest neighbor (KNN)
* Naïve Bayes

Which customers are at high risk of churn? And which customer’s features/variables are most significant to churn? This question will be using the classification method: K-nearest neighbor (KNN)

1. Define one goal of the data analysis. Ensure that your goal is reasonable within the scope of the scenario and is represented in the available data.

The goal or objective of this analysis is to be able to predict in advance that a specific customer is likely to churn. "The churn rate, also known as the rate of attrition, is the rate at which customers stop doing business with an entity. It is most expressed as the percentage of service subscribers who discontinue their subscriptions within a given time period. (Frankenfield, 2020) Stakeholders in the company can benefit by this analysis by understanding more effectively which customers are likely to churn soon. The analysis will provide weight for decisions in marketing to improve services to customers with these characteristics and past user experiences.

**Part II: Method Justification**

1. Explain the reasons for your chosen classification method from part A1 by doing the following:
2. Explain how the classification method you chose analyzes the selected data set. Include expected outcomes.

“The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.”(Harrison, 2018) The algorithm stores all labels and classifies new labels by a “majority-vote” of its K-nearest neighbors. K is the number of nearest neighbors. The number of neighbors is the core deciding factor. KNN will find the most similar data points in the training data. A k number of data points will be chosen by the model. The dominant classes of the closest data points will suggest how a data point of interest should be classified. Test data will then be used to test the size of the models outcomes.

1. Summarize one assumption of the chosen classification method.

“The k-nearest neighbors algorithm is based on a very simple premise: That things that are close together have a lot in common.” (Harrison, 2018) One assumption is that the method assumes that the closest neighbors are similar enough to classify the data point of interest as the same.

1. List the packages or libraries you have chosen for Python or R, and justify how each item on the list supports the analysis.

* Matplot.lib.pylot
* Pandas
* NumPy
* Scikit-learn
* Seaborn

“NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.” (Numpy.org, 2021)

“Similar to Numpy, pandas deals primarily with data in 1-D and 2-D arrays; however, pandas handles them differently.“ (Educative.io, 2021)

“In pandas, 1-D arrays are referred to a series. A series is created through the pd.Series constructor, which has a lot of optional arguments. The most common argument is data, which specifies the elements of the series.” (Educative.io, 2021)

“A DataFrame is simply a 2-D array. It can be created through the pd.DataFrame constructor, which takes in essentially the same arguments as pd.Series. However, while a series could be constructed from a scalar (representing a single value Series), a DataFrame cannot.” (Educative.io, 2021)

PyLab is a procedural interface to the Matplotlib object-oriented plotting library. Matplotlib is the whole package; matplotlib.pyplot is a module in Matplotlib; and PyLab is a module that gets installed alongside Matplotlib. PyLab is a convenience module that bulk imports matplotlib.pyplot (for plotting) and NumPy (for Mathematics and working with arrays) in a single name space. (tutorialspoints.com, 2021)

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them. (Seaborn.pydata.org, 2021)

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. (tutorialspoints.com, 2021) The tools that will be used in the analysis from sklearn are: Datasets, KNeighborsClassifer, Train\_test\_split, cross\_val\_score, GridSearchCV, metrics, accuracy\_score, classification\_report, confusion\_matrix, preprocessing, StandardScaler, and Pipeline.

**Part III: Data Preparation**

1. Perform data preparation for the chosen data set by doing the following:
2. Describe one data preprocessing goal relevant to the classification method from part A1.

One data preprocessing goal relevant to the classification method from part A1 is encoding binary categorical variables into 0/1.

1. Identify the initial data set variables that you will use to perform the classification question from part A1. And classify each variable as continuous or categorical.

Out of the 50 variables in the initial data set, 34 variables have been identified to be used in the classification analysis:

Continuous variables in the data set:

* Age (Age of customer as reported in sign-up information)
* Bandwidth\_GB\_Year (Data usage in gigabytes)
* Children (Number of children)
* Contacts (Number of times customer contacted technical support)
* Email (Number of emails sent out to customer)
* Income (Annual income of customer)
* MonthlyCharge (The amount charged to the customer monthly)
* Outage\_sec\_perweek (Average number of seconds per week of system outages in the customer’s neighborhood)
* Tenure (Number of months the customer has stayed with the provider)
* Yearly\_equip\_failure (The number of times customer’s equipment failed and had to be reset/replaced in the past year)

Categorical variables in the data set:

* Churn (If the customer is likely to leave)
* Contract (The contract term of the customer (month-to-month, one year, two year))
* Device Protection (Whether the customer has device protection add-on (yes, no))
* Gender (Customer self-identification as male, female, or nonbinary)
* InternetService (Customer’s internet service provider (DSL, fiber optic, None))
* Multiple (Whether the customer has multiple lines (yes, no))
* OnlineBackup (Whether the customer has an online backup add-on (yes, no))
* OnlineSecurity (Whether the customer has an online security add-on (yes, no))
* PaperlessBilling (Whether the customer has paperless billing (yes, no))
* Phone (Whether the customer has a phone service (yes, no))
* Port\_modem (Whether the customer has a portable modem (yes, no))
* StreamingMovies (Whether the customer has streaming movies (yes, no))
* StreamingTV (Whether the customer has streaming TV (yes, no))
* Tablet (Whether the customer owns a tablet such as iPad, Surface, etc. (yes, no))
* Techie (Whether the customer considers themselves technically inclined (based on customer questionnaire when they signed up for services) (yes, no))
* TechSupport (Whether the customer has a technical support add-on (yes, no))

Discrete variables in the data set:

* Item1: Timely response
* Item2: Timely fixes
* Item3: Timely replacements
* Item4: Reliability
* Item5: Options
* Item6: Respectful response
* Item7: Courteous exchange
* Item8: Evidence of active listening

1. Explain *each* of the steps used to prepare the data for the analysis. Identify the code segment for *each* step.

The data preparation goals and data manipulations will include:

* + - 1. Import tools and packages for analysis.
      2. Import the dataset into Python.
      3. Provide a name to identify my dataset. The naming convention I chose for my dataset is: Churn\_df.
      4. Evaluate the data structure to gain a better understanding of the variables and data types.
      5. Check for any missing data that could skew the model. Missing data will be inputted with measures of central tendency.
      6. Check for any misleading variable names and rename them.
      7. Create data manipulations to the dataset to use in analysis.
      8. Create visualizations to identify any outliers that could affect the model.
      9. Removes less meaningful categorical values from the dataset to provide a fully numerical dataframe to continue with the analysis.
      10. Extract and use prepped dataset for K-Nearest Neighbor Model.

|  |
| --- |
| Legend: |
| My headers for my code: GREY |
| Code: BLUE |
| My explanation: GREEN |

**Annotated Code with explanation of each step:**

(Step 1 of the data prep.)

# Standard Data Science Imports

import numpy as np

import pandas as pd

from pandas import Series, DataFrame

Here I am importing NumPy as pd and pandas as pd. I am also importing Series, and DataFrame

“NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.” (Numpy.org, 2021)

“Similar to Numpy, pandas deals primarily with data in 1-D and 2-D arrays; however, pandas handles them differently.“ (Educative.io, 2021)

“In pandas, 1-D arrays are referred to a series. A series is created through the pd.Series constructor, which has a lot of optional arguments. The most common argument is data, which specifies the elements of the series.” (Educative.io, 2021)

“A DataFrame is simply a 2-D array. It can be created through the pd.DataFrame constructor, which takes in essentially the same arguments as pd.Series. However, while a series could be constructed from a scalar (representing a single value Series), a DataFrame cannot.” (Educative.io, 2021)

# Visualization libraries

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

Next I will be importing the Visualization libraries.

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them. (Seaborn.pydata.org, 2021)

#Scikit-learn

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

# Scikit-learn

import sklearn

from sklearn import datasets

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn import metrics

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import roc\_auc\_score

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

Then I will now import the Scikit-learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. (tutorialspoints.com, 2021)

The K Neighbors Classifier is implementing the K – nearest neighbors vote.

The train split test splits arrays or matrices into random train and test subsets.

The cross val score evaluates a score by cross-validation

The GridSearchCV is an exhaustive search over specified parameter values for an estimator. Important members are fit.

The Accuracy Score produces the accuracy classification score.

The classification report builds a text report showing the main classification metrics.

The confusion matrix computes a confusion matrix that evaluates the accuracy of a classification.

The Roc\_Auc\_Score computes the area under the receiver operating characteristic curve (ROC AUC) from prediction scores.

The Standard Scaler standardizes features by removing the mean and scaling to unit variance.

Pipeline applies a list of transforms and a final estimator.

(Step 2 and 3 of the data prep.)

# Loading the data set into Pandas dataframe

churn\_df = pd.read\_csv(r'C:\Users\Hydraconix\Desktop\DATA\churn\_clean.csv')

Here I am importing the dataset and naming my dataset churn\_df.

(Step 4 of the data prep.)

# Examining fist five records of dataset

churn\_df.head()

# Viewing DataFrame descriptive information

churn\_df.info

# Getting an overview of descriptive stats

churn\_df.describe()

# Getting data types of features

churn\_df.dtypes

I have evaluated the data structure for a better understanding of the variables and data types and will now move on to checking for null values.

(Step 5 of the data prep.)

#Checking for null data

Churn\_df.isnull()

There is no null values so I will no move on to checking for any misleading variables and executing data manipulations if needed.

(Step 6 and 7 of the data prep.)

# Renaming the last 8 Survey Columns for better description of variables

churn\_df.rename(columns = {'Item1' : 'TimelyResponse',

'Item2' : 'Fixes' ,

'Item3' : 'Replacements' ,

'Item4' : 'Reliability' ,

'Item5' : 'Options' ,

'Item6' : 'Respectfulness' ,

'Item7' : 'Courteous' ,

'Item8' : 'Listening'},

inplace=True)

I have renamed the survey columns so that we can get a better understanding when examining the variables.

# Converting ordinal categorical data into numeric variables

churn\_df['DummyInternetService'] = churn\_df.InternetService.map({'None' : 0, 'DSL' : 1, 'Fiber Optic' : 2})

churn\_df['DummyContract'] = churn\_df.Contract.map({'Month-to-month' : 0, 'One year' : 1, 'Two Year' : 2})

churn\_df['DummyGender'] = churn\_df.Gender.map({'Nonbinary' : 0, 'Male' : 1, 'Female' : 2})

I have converted the ordinal categorical data into numeric columns so that we can use these variables in the classification analysis.

Step 8 of the data prep.

# Histograms of continuous variables

churn\_df[['Age', 'Bandwidth\_GB\_Year', 'Children', 'Contacts', 'Email', 'Income', 'MonthlyCharge', 'Outage\_sec\_perweek', 'Tenure', 'Yearly\_equip\_failure', 'DummyInternetService', 'DummyContract', 'DummyGender']].hist()

plt.savefig('churn\_pyplot.jpg')

plt.tight\_layout()

Diagram

Description automatically generated

The visualizations of central tendency have revealed normal distribution for: Outage\_sec\_perweek, Email, and MonthlyCharge. When analyzing the visualization for Bandwidth\_GB\_Year and Tenure, the histograms displays a bimodal distributions.

# A scatterplot to get an idea of correlations between potentially related variables

sns.scatterplot(x=churn\_df['MonthlyCharge'], y=churn\_df['Churn'], color='green')

plt.show()

A picture containing text

Description automatically generated

# A scatterplot to get an idea of correlations between potentially related variables

sns.scatterplot(x=churn\_df['Outage\_sec\_perweek'], y=churn\_df['Churn'], color='green')

plt.show()

A picture containing chart

Description automatically generated

# A scatterplot to get an idea of correlations between potentially related variables

sns.scatterplot(x=churn\_df['Tenure'], y=churn\_df['Churn'], color='green')

plt.show()

Chart

Description automatically generated with low confidence

When looking at the visualizations, we do not see any positive linear correlations in the continuous variables. I will now move on to the categorical variables.

Step 8 in the data prep. (cont.)

# I will now set the plot style to ggplot

plt.style.use('ggplot')

# Countplots of categorical variables

plt.figure()

sns.countplot(x='DeviceProtection', hue='Churn', data=churn\_df, palette='RdBu')

plt.xticks([0,1], ['No', 'Yes'])

plt.show()

Chart, bar chart

Description automatically generated

plt.figure()

sns.countplot(x='Multiple', hue='Churn', data=churn\_df, palette ='RdBu')

plt.xticks([0,1],['No','Yes'])

plt.show()

Chart, bar chart

Description automatically generated

plt.figure()

sns.countplot(x='Techie', hue='Churn', data=churn\_df, palette ='RdBu')

plt.xticks([0,1],['No','Yes'])

plt.show()

Chart, bar chart

Description automatically generated

plt.figure()

sns.countplot(x='TechSupport', hue='Churn', data=churn\_df, palette ='RdBu')

plt.xticks([0,1],['No','Yes'])

plt.show()

Chart, bar chart

Description automatically generated

Just like the continuous variables, the categorical variables show no indication of any potential linear relationships

Step 8 of the data prep. (cont.)

# A scatter matrix of the discrete variables for high level overview of potential relationships & distributions

churn\_discrete = churn\_df[['Churn', 'TimelyResponse', 'Fixes', 'Replacements', 'Reliability', 'Options', 'Respectfulness', 'Courteous', 'Listening']]

pd.plotting.scatter\_matrix(churn\_discrete, figsize = [30, 30])

Chart

Description automatically generated

# An individual scatterplot for viewing relationship of key financial feature against target variable

sns.scatterplot(x = churn\_df['TimelyResponse'], y = churn\_df['Churn'], color='red')

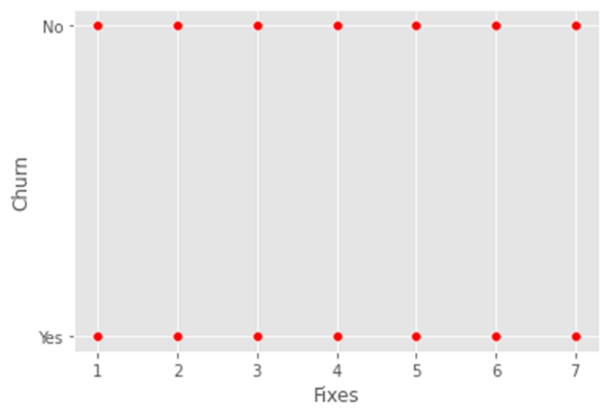
plt.show()

Chart

Description automatically generated

sns.scatterplot(x = churn\_df['Fixes'], y = churn\_df['Churn'], color='red')

plt.show()



sns.scatterplot(x = churn\_df['Replacements'], y = churn\_df['Churn'], color='red')

plt.show()

Chart

Description automatically generated

The discrete variables show no indications of positive linear relationships in the visualizations. So now I will now move onto manipulating the data so that we can use the variables in the classification analysis.

Step 7 of the data prep. (cont.)

# Converting binary categorical variables to numeric variables

churn\_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn\_df['Churn']]

churn\_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn\_df['Techie']]

churn\_df['DummyPort\_modem'] = [1 if v == 'Yes' else 0 for v in churn\_df['Port\_modem']]

churn\_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn\_df['Tablet']]

churn\_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn\_df['Phone']]

churn\_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in churn\_df['Multiple']]

churn\_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in churn\_df['OnlineSecurity']]

churn\_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in churn\_df['OnlineBackup']]

churn\_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in churn\_df['DeviceProtection']]

churn\_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in churn\_df['TechSupport']]

churn\_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in churn\_df['StreamingTV']]

churn\_df['DummyStreamingMovies'] = [1 if v == 'Yes' else 0 for v in churn\_df['StreamingMovies']]

churn\_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in churn\_df['PaperlessBilling']]

The new dummy variables have been created so we can use them in the machine learning algorithm. The variables need to be in a numeric variables for the algorithm to work.

Step 10 of the data prep.

#Drop original categorical features from dataframe for further analysis

churn\_df = churn\_df.drop(columns=['Churn', 'Contract', 'DeviceProtection', 'Gender', 'InternetService', 'Multiple' , 'OnlineBackup', 'OnlineSecurity', 'PaperlessBilling', 'Phone', 'Port\_modem', 'StreamingMovies', 'StreamingTV', 'Tablet', 'Techie', 'TechSupport'])

#Remove the other less meaningful categorical variables from dataset to provide fully numerical dataframe for further analysis

churn\_df = churn\_df.drop(columns=['Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Marital', 'PaymentMethod'])

I have removed the original categorical features and the other less meaningful categorical variables from the dataset so that we use the dataset into ML algorithm.

Step 11 of the data prep.

# Provide a copy of the prepared data set

churn\_df.to\_csv(r'C:\Users\Hydraconix\Desktop\'churn\_prepared\_log.csv')

An exported copy of the prepared dataset for the classification analysis.

**Part IV: Analysis**

1. Perform the data analysis and report on the results by doing the following:

Split the data into training and test data sets and provide the file(s).

# Re-read fully numerical prepared dataset

churn\_df = pd.read\_csv(r'C:\Users\Hydraconix\Desktop\'churn\_prepared\_log.csv')

Here I am importing the prepped dataset.

# Set predictor features & target variable

X = churn\_df.drop('DummyChurn', axis=1).values

y = churn\_df['DummyChurn'].values

Creating the predictor features & target variables for training and test sets

# Create training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 1)

Created training and test sets, test size is 20% of the data, training test is 80%, and the random state is set to 1 for reproducibility.

# Export X\_train dataset

X\_train\_df = pd.DataFrame(X\_train)

X\_train\_df.to\_csv(r'C:\Users\Hydraconix\Desktop\X\_train.csv')

# Export X\_test dataset

X\_test\_df = pd.DataFrame(X\_test)

X\_test\_df.to\_csv(r'C:\Users\Hydraconix\Desktop\X\_test.csv')

# Export y\_train dataset

y\_train\_df = pd.DataFrame(y\_train)

y\_train\_df.to\_csv(r'C:\Users\Hydraconix\Desktop\y\_train.csv')

# Export y\_test dataset

y\_test\_df = pd.DataFrame(X\_test)

y\_test\_df.to\_csv(r'C:\Users\Hydraconix\Desktop\y\_test.csv')

I have now exported all of the training and test data sets into csv files.

Describe the analysis technique you used to appropriately analyze the data. Include screenshots of the intermediate calculations you performed.

Once I have created my training and test data sets, I then fit the data sets into the model and create a new array called: y\_pred. The analysis technique I used to appropriately analyze the data is by computing the accuracy score of the K nearest neighbors model. Once the accuracy score is determined, the next step is to see if we can create a more accurate model if the data sets are scaled. This is to ensure we have analyzed the model as much as possible.

Here included are the annotated code and screenshots of my calculations performed:

# Initializing KNN model

knn = KNeighborsClassifier(n\_neighbors = 7)

Initializing ML model with nearest neighbors set to 7.

# Fit data to KNN model

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

Fitting the X training set and y training set into the model to create prediction for y\_pred.

Output:

KNeighborsClassifier(n\_neighbors=7)

# Predict outcomes from test set

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

New outcome has been created.

# Export y\_pred dataset

y\_pred\_df = pd.DataFrame(y\_pred)

y\_pred\_df.to\_csv(r'C:\Users\Hydraconix\Desktop\y\_pred.csv')

I have now exported the y\_pred dataset to a csv file.

# Print initial accuracy score of KNN model

print('Initial accuracy score KNN model: ', accuracy\_score(y\_test, y\_pred))

Output:

Initial accuracy score KNN model: 0.715

# Compute classification metrics

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

A picture containing text, outdoor, meter, device

Description automatically generated

It appears that the accuracy of our model is not as accurate as we want it to be. I will now model the scale to see if I can obtain a higher accuracy score.

# Set steps for pipeline object

steps = [('scaler', StandardScaler()),

('knn', KNeighborsClassifier())]

# Initiate pipeline

pipeline = Pipeline(steps)

# Split dataframe

X\_train\_scaled, X\_test\_scaled, y\_train\_scaled, y\_test\_scaled = train\_test\_split(X, y, test\_size = 0.2, random\_state = 1)

# Scale dateframe with pipeline object

knn\_scaled = pipeline.fit(X\_train\_scaled, y\_train\_scaled)

# Predict from scaled dataframe

y\_pred\_scaled = pipeline.predict(X\_test\_scaled)

# Print new accuracy score of scaled KNN model

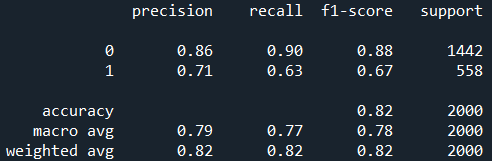
print('New accuracy score of scaled KNN model: {:0.3f}'.format(accuracy\_score(y\_test\_scaled, y\_pred\_scaled)))

Output:

New accuracy score of scaled KNN model: 0.825

# Compute classification metrics after scaling

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



#Confusion\_matrix & generate results

cf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(cf\_matrix)

Output:

[[1204 238]

[ 333 225]]

# Visual confusion matrix

group\_names = ['True Neg', 'False Pos', 'False Neg', 'True Pos']

group\_counts = ["{0:0.0f}".format(value) for value in cf\_matrix.flatten()]

group\_percentages = ["{0:.2%}".format(value) for value in cf\_matrix.flatten()/np.sum(cf\_matrix)]

labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in zip(group\_names,group\_counts,group\_percentages)]

labels = np.asarray(labels).reshape(2,2)

sns.heatmap(cf\_matrix, annot=labels, fmt='', cmap='Blues')

A picture containing timeline

Description automatically generated

Provide the code used to perform the classification analysis from part D2.

Please see annotated code above.

**Part V: Data Summary and Implications**

1. Summarize your data analysis by doing the following:
2. Explain the accuracy and the area under the curve (AUC) of your classification model.

# Set up parameters grid

param\_grid = {'n\_neighbors': np.arange(1, 50)}

# Re-initializing KNN for cross validation

knn = KNeighborsClassifier()

# Initializing GridSearch cross validation

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

# Fit model to

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

# Print best parameters

print('Best parameters for this KNN model: {}'.format(knn\_cv.best\_params\_))

Output:

Best parameters for this KNN model: {'n\_neighbors': 6}

# Generate model best score

print('Best score for this KNN model: {:.3f}'.format(knn\_cv.best\_score\_))

Output:

Best score for this KNN model: 0.735

# Fit it to the data

knn\_cv.fit(X, y)

# Compute predicted probabilities: y\_pred\_prob

y\_pred\_prob = knn\_cv.predict\_proba(X\_test)[:,1]

# Compute and print AUC score

print("The Area under curve (AUC) on validation dataset is: {:.4f}".format(roc\_auc\_score(y\_test, y\_pred\_prob)))

Output:

The Area under curve (AUC) on validation dataset is: 0.7889

# Compute cross-validated AUC scores: cv\_auc

cv\_auc = cross\_val\_score(knn\_cv, X, y, cv=5, scoring='roc\_auc')

# Print list of AUC scores

print("AUC scores computed using 5-fold cross-validation: {}".format(cv\_auc))

1. Discuss the results and implications of your classification analysis.

After scaling the model, the accuracy has improved from 0.71 to 0.83 and precision from 0.40 to 0.72. The area under the curve is a score at 0.7889

1. Discuss one limitation of your data analysis.

"When using the k-nearest neighbors algorithm you have the ability to change k, potentially yielding dramatically different results. You choose the value of k by trying different values and testing the prediction capabilities of the model. This means you must develop, validate, and test several models" (Grant, pg. 1).

What this means to our analysis here is that the relatively arbitrary choice of k = 7 nearest neighbors might yield dramatically different results if we chose a different k number of neighbors. This can change the outcome of the analysis, also it appears that the cross-validation grid search takes a long time to compute.

1. Recommend a course of action for the real-world organizational situation from part A1 based on your results and implications discussed in part E2.

It is vital that stakeholders within the company understand the predictor variables used in the analysis created a relatively low accuracy score with results showing 0.83 after scaling. The recommended course of action would be to analyze the features that are appear in common with those leaving the company and attempt to reduce the probability of this happening in the future. Offering to improve services such as replacements or increasing online backup can improve the customer’s experiences.

**Part VI: Demonstration**

1. Video Link: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=f3ad3ccf-432f-440d-9a40-ae2e014e6a8c>
2. Record the web resources used to acquire the data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

K Nearest Neighbors:

Sklearn.neighbors.kneighborsclassifier. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

Train\_split\_test:

Sklearn.model\_selection.train\_test\_split. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>

GridSearchCV:

Sklearn.model\_selection.GRIDSEARCHCV. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>

Accuracy\_score:

Sklearn.metrics.accuracy\_score. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html>

Classification\_Report:

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html>

Roc\_auc\_score:

Sklearn.metrics.roc\_auc\_score. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html>

Pipeline:

Sklearn.pipeline.pipeline. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>

Cross\_val\_Score:

Sklearn.model\_selection.cross\_val\_score. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html>

Confusion Matrix:

Sklearn.metrics.confusion\_matrix. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>

Standard Scaler

Sklearn.preprocessing.StandardScaler. scikit. (n.d.). Retrieved February 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

1. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

*An introduction to seaborn*. - seaborn 0.11.2 documentation. (n.d.). Retrieved November 20, 2021, from <https://seaborn.pydata.org/introduction.html>

Frankenfield, J. (2021, May 19). Churn rate. Investopedia. Retrieved October 24, 2021, from <https://www.investopedia.com/terms/c/churnrate.asp>

Grant, P. (2019, July 21). Introducing K-nearest neighbors. Medium. Retrieved December 29, 2021, from <https://towardsdatascience.com/introducing-k-nearest-neighbors-7bcd10f938c5>

Harrison, O. (2019, July 14). Machine learning basics with the K-nearest neighbors algorithm. Medium. Retrieved December 12, 2021, from <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>

Matplotlib - PyLab module. (n.d.). Retrieved November 20, 2021, from <https://www.tutorialspoint.com/matplotlib/matplotlib_pylab_module.htm>

Scikit learn - introduction. (n.d.). Retrieved November 20, 2021, from <https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.html>

*What is NumPy?*.- NumPy v1.21 Manual. (n.d.). Retrieved November 20, 2021, from <https://numpy.org/doc/stable/user/whatisnumpy.html>

*What is Pandas in python? Educative.io. (n.d.). Retrieved November 20, 2021, from* [*https://www.educative.io/edpresso/what-is-pandas-in-python*](https://www.educative.io/edpresso/what-is-pandas-in-python)