Matthew Blessing
*Student ID**: 000507424***
NVM2 - Task 1, D209 - Data Mining

Performance Assessment Task 1: Classification Analysis - NVM2

Part I: Research Question

A1.
In this analysis, we will attempt to determine if we can accurately utilize a k-nearest neighbor (kNN) supervised machine learning algorithm to determine if it is possible to predict customer churn.

A2.
The goal of the analysis is to be able to accurately utilize a kNN algorithm to predict if a customer will churn based on their age, yearly bandwidth used, income, and length of tenure. If we can accurately predict customer churn, this can benefit the business by being able to accurately predict which customers may leave the service, allowing more accurate forecasting, in addition to knowing which customers to target for retention campaigns.

Part II: Method Justification

B1.
The kNN model attempts to classify data by looking at the K, which is a user-specified number of the closest labeled data points to the unlabeled data point, and having those labeled data points vote to decide how the unlabeled point should be labeled (*Classification - Supervised Learning*, n.d.).

We should expect that the kNN algorithm will provide us a prediction for each observed data point, and give us an overall accuracy score for the model.

B2.
Because k-Nearest Neighbor utilizes labeled data points to vote on how an unlabeled data point should be classified, the kNN model makes an assumption that similar things exist in close proximity, and assumes that similar data points will share similar data labels (*Lecture 2: K-Nearest Neighbors / Curse of Dimensionality*, n.d.).

B3.
3. List the packages or libraries you have chosen for Python or R, and justify how each item on the list supports the analysis.

For this analysis, we will be utilizing Python within a Jupyter notebook, along with the following libraries:

- *Pandas*
- *NumPy*
- *Matplotlib*
- *Seaborn*
- *SciKit-Learn*

Python is an object-oriented programming language that is extremely popular for data science due to it being a powerful, easy to learn language that is extremely expandable with a large library of data science packages, such as NumPy, SciPy, Pandas, and Matplotlib (*Advantages of Learning Python for Data Science*, 2018). These libraries easily allow users to implement classification, regression, machine learning and more on chosen data sets.

Seaborn and Matplotlib are imported primarily for their powerful visualization tools to show how data within our dataset is distributed, allowing us to easily produce histograms, as well as bar charts, scatterplots, and box plots.

SciKit-Learn is a powerful machine learning library for Python, that offers several classification, regression, and clustering algorithms (*SciKit-Learn: Machine Learning in Python — Scikit-Learn 1.0 Documentation*, n.d.).

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text (*Project Jupyter*, n.d.).

Part III: Data Preparation

C1.
To prepare our data for use in the kNN model, we first need to ensure that our variables that we have selected are numerical. Of the variables we have selected, only one, "Churn", is not numerical. We will need to convert the categorical variable of "Churn" to a binary variable in order to use this in our kNN model.

To accomplish this, we will execute the following code:

```
df.Churn.replace(('Yes', 'No'), (1, 0), inplace=True)
```

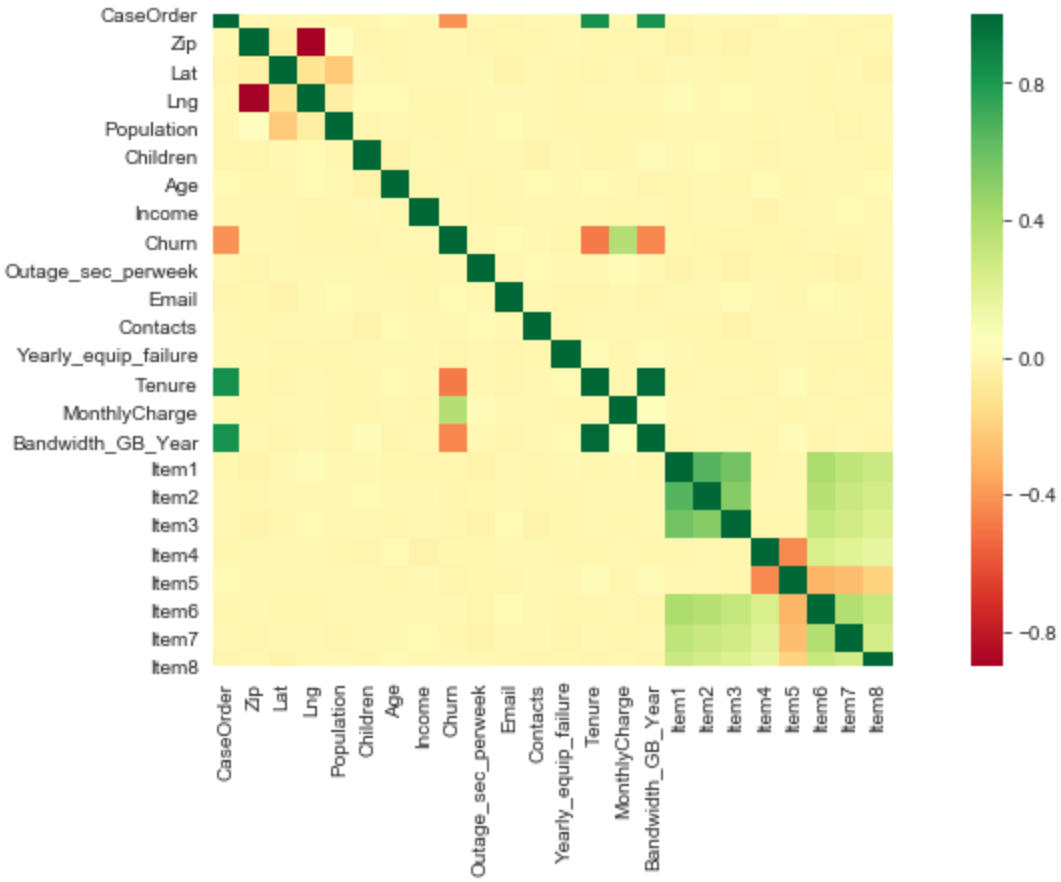
This will replace all variables in the "Churn" column, with a 1 for "Yes" and a 0 for "No".

C2.
While our dataset has several variables, a kNN model works best with a small number of inputs, but can become problematic with a large number of inputs (Brownlee, n.d.). To limit our variables, we have decided on four variables to be our feature variables, in addition to our target variable of Churn. These variables are of the following data types:

- Age - continuous variable
- Bandwidth_GB_Year - continuous variable
- Income - continuous variable
- Tenure - continuous variable
- Churn - categorical variable

The goal of the analysis is to determine if we can accurately determine if a new customer will churn, based on other customers who share a similar age, similar bandwidth used per year, similar income, and similar tenure.
By utilizing a correlation heatmap, we can also see that our selected variables are some of the only variables in our dataset that seem to have any correlation, be that positive, or negative, with our target variable, churn.

```
plt.figure(figsize=(14, 6))
sns.heatmap(df.corr(), square=True, cmap='RdYlGn');
```



C3.

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

To begin preparing the dataset for our analysis, we first import our Python libraries that we will use in the analysis:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import statsmodels.api as sm
sns.set_style('darkgrid')

#sets the jupyter notebook window to take up 90% width of the browser
window
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
```

We then read in the dataset .csv into Pandas data frame:

```
df = pd.read_csv('churn_clean.csv')
```

We then get a brief overview of the dataset and the variables it contains, by calling the .info() function on the data frame:

```
df.info()
```

We then ensure there are no duplicate rows in the data frame by utilizing the .drop_duplicates() function:

```
df.drop_duplicates()
```

After dropping any duplicate rows, we want to check to see if there are any null values in the dataset:

```
df.isnull().sum()
```

Luckily, there are no null values in this dataset. Now that our data frame appears to be clean, we will replace the values for the categorical variable, "Churn", by replacing "Yes'' with a 1 and "No" with a 0. To do this, we will utilize Pandas .replace() function:

```
df.Churn.replace(('Yes', 'No'), (1, 0), inplace=True)
```

Now that our data is clean and prepared, we will create a new data frame that contains only the variables we will want to use in our analysis:

```
df2 = df[['Age', 'Bandwidth_GB_Year','Income', 'Tenure', 'Churn']].copy()
```

Once our new data frame has been created, we will utilize Pandas .describe() function to get a quick look at measures of center and ensure that all of our variables have the same amount of observations, which they do (1,000):

```
df2.describe()
```

Lastly, we will export the cleaned and prepared dataset to a .csv file:

```
df2.to_csv('D209_Task1_Dataset.csv')
```

With these steps being completed, our data is now ready to be input into a kNN model.

C4.
Please see the attached 'D209_Task1_Dataset.csv'.

Part IV: Analysis

D1.
We begin preparing our data for use in the kNN model by first defining our feature and target arrays:

```
X = df2.drop('Churn', axis=1).values
```

```
y = df2['Churn'].values
```

We then import our model and scoring metrics from the SciKit Learn library:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score, train_test_split
```

We then split our data into training and testing data:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
```

D2-D3.

In splitting the training and testing data, we ensure to specify the size of the test set, which we have set to 0.2, or 20% of our data split to the training set while 80% is split to the testing set. We set a random_state as well, so that we can refine our model, if we so choose, and still maintain the same random split, allowing us to see if we can improve our classification accuracy. We also set stratify=y in order to maintain the proportion of each value of the y variable, in this case, 'Churn', in our training and testing datasets.

We then create a kNN classifier, and set the number of neighbors, k, to 6:

```
kNN = KNeighborsClassifier(n_neighbors=6)
```

All that is left is to fit the model to our data and look at how accurate our classifier has been with our data:

```
kNN.fit(X_train, y_train)
print(kNN.score(X_test,y_test))
```

```
0.73
```

And lastly, we will create a curve to visually represent how the classifier's accuracy would change based on the number of neighbors, and to ensure that we have not overfit or underfit our model:

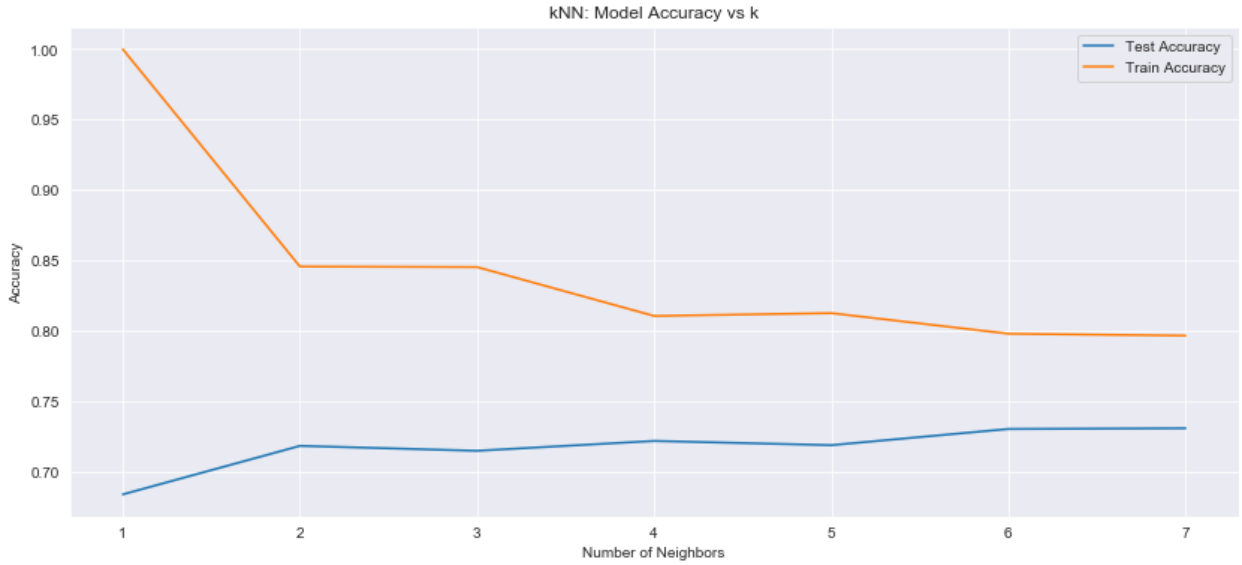
```
neighbors = np.arange(1, 8)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i, k in enumerate(neighbors):
    kNN = KNeighborsClassifier(n_neighbors=k)
    kNN.fit(X_train, y_train)

    train_accuracy[i] = kNN.score(X_train, y_train)
    test_accuracy[i] = kNN.score(X_test, y_test)


plt.figure(figsize=(14, 6))
plt.title('kNN: Model Accuracy vs k')
plt.plot(neighbors, test_accuracy, label = 'Test Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Train Accuracy')
```

```
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.show();
```



We then print our classifier's accuracy score, and the area under the curve score:

```
print(kNN.score(X_test,y_test))

from sklearn.metrics import roc_auc_score
y_pred_prob = kNN.predict_proba(X_test)[:,1]

print((roc_auc_score(y_test, y_pred_prob)))

0.7551719933256321
```

Part V: Data Summary and Implications

E1.

As we can see from our printing of the kNN score, the classifier was accurate 73% of the time, which is good, but certainly not great. Looking at the area under the curve, or AUC, which is defined as the measure of the ability of a classifier to distinguish between classes. An AUC of 1 implies that the classifier is perfectly able to distinguish between classes (Bhandari, 2020). Our model shows an AUC score of 0.76, which is not bad for a first pass at attempting to predict churn based on our selected variables.

E2.

Our model's 73% accuracy score would suggest that there is room to improve the classifier. It is possible that utilizing a different number of neighbors would improve the classifier's accuracy, however, the 'Model Accuracy vs. k' plot we generated suggests that anything beyond 4-6 neighbors is actually creating an overly simplified model that underfits the data. It is also possible that our selected predictor variables do not allow the classifier to

more accurately predict churn. Whether that is due to poor feature selection with weak correlation to churn, or just simply not enough variables for the classifier to more accurately be able to classify a customer who will churn is something that would need to be explored with additional model revisions.

E3.

A limitation of this analysis is that we have only tried one pass at a kNN classifier with this dataset. While this analysis may prove somewhat useful, no model is immediately perfect, and it would be beneficial to iterate on this model to improve the accuracy and AUC scores. It could very well be that this analysis does not utilize enough, or the best, variables in order to produce a more accurate classification model. Of course, it is also possible that it is not possible to create a highly accurate classification model with the given data.

E4.

As a recommendation to stakeholders in the business for this situation, it would be recommended to pursue multiple iterations of this model in order to increase the accuracy and AUC scores. Ideally, it would be a great benefit to complete a more thorough feature selection process where we can investigate and identify features that have a strong correlation to customer churn. By determining which features, and how many neighbors best fit our model, we can provide a more accurate model that will create value for the business and key stakeholders.

Part VI: Demonstration

Please see the attached Panopto video demonstration.

References

Advantages of Learning Python for Data Science. (2018, March 16). BSD MAG. Retrieved October 14, 2021, from https://bsdmag.org/advantages-of-learning-python-for-data-science/

Bhandari, A. (2020, July 20). *AUC-ROC Curve in Machine Learning Clearly Explained*. Analytics Vidhya. Retrieved October 14, 2021, from https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

Brownlee, J. (n.d.). *K-Nearest Neighbors for Machine Learning*. Machine Learning Mastery. Retrieved October 17, 2021, from https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/

Classification - Supervised learning. (n.d.). DataCamp. Retrieved October 14, 2021, from https://campus.datacamp.com/courses/supervised-learning-with-scikit-learn/

kNN Classification using Sklearn Python. (n.d.). DataCamp. Retrieved October 14, 2021, from https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Lecture 2: k-nearest neighbors / Curse of Dimensionality. (n.d.). Cornell University. Retrieved October 14, 2021, from https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html

Project Jupyter. (n.d.). Project Jupyter. Retrieved October 14, 2021, from https://jupyter.org/

SciKit-Learn: machine learning in Python — scikit-learn 1.0 documentation. (n.d.). SciKit-Learn. Retrieved October 14, 2021, from https://scikit-learn.org/stable/