Matthew Blessing
*Student ID: **000507424***
NVM2 - Task 2, D212 - Data Mining II

OFM 3 - Performance Assessment Task 1: Clustering Techniques

Part I: Research Question

A1.
In this analysis we will explore if it is possible to utilize K-Means to perform customer segmentation on the 'Churn' dataset, determine how many clusters to segment the customers into, and determine their centroids.

A2.
The goal of the analysis will be to utilize K-Means clustering in an attempt to accurately cluster customers based on their income and monthly charge.

Part II: Technique Justification

B1.
K-Means is an unsupervised machine learning algorithm that attempts to divide data into a number of clusters, *k*. The algorithm looks at the data it is given, and attempts to segment the data into *k* clusters, where each observation is placed into a cluster with the closest mean (Clustering for Dataset Exploration - Unsupervised Learning, n.d.). The K-Means clustering model takes data points and groups them into a number of clusters, *k*. The algorithm's goal is to take data points as inputs, and determine to which cluster these data points belong by looking at each data point individually and measuring its distance between the data point and the number of centroids, *k* (*Clustering for Dataset Exploration - Unsupervised Learning*, n.d.). As an example of what K-Means could be used for, the algorithm could be utilized to determine a type of wine (Pinot Noir, Cabernet Sauvignon, etc) , purely based on input data such as alcohol content, phenol count, or flavanoid count.

We should expect that the algorithm will partition our selected variables into *k* clusters, (the number of which we will determine utilizing the elbow method, and identify each cluster's centroid, or the imaginary center of the cluster (Garbade, 2018).

B2.
One assumption of K-Means is that the clusters are of similar size. When the clusters are not spherical, the clusters will potentially not be as defined, in addition to the clusters not proving very useful in terms of gaining insight from the algorithm (All the Annoying Assumptions - Towards Data Science, 2019) .

B3.
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 ensure our model is as accurate as possible, we will want to ensure all of our chosen variables have the same number of observations.

We can do this by utilizing Pandas .info() function that will return all of the variables in our data frame, alert us to any variables that contain missing (null) values, and inform us of each variable's data type:

```
In [3]: #gets the data type, count, and data type of all variables in the data set
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
CaseOrder               10000 non-null int64
Customer_id             10000 non-null object
Interaction             10000 non-null object
UID                     10000 non-null object
City                    10000 non-null object
State                   10000 non-null object
County                  10000 non-null object
Zip                     10000 non-null int64
Lat                     10000 non-null float64
Lng                     10000 non-null float64
Population              10000 non-null int64
Area                    10000 non-null object
TimeZone                10000 non-null object
Job                     10000 non-null object
Children                10000 non-null int64
Age                     10000 non-null int64
Income                  10000 non-null float64
Marital                 10000 non-null object
Gender                  10000 non-null object
Churn                   10000 non-null object
Outage_sec_perweek      10000 non-null float64
Email                   10000 non-null int64
Contacts                10000 non-null int64
Yearly_equip_failure    10000 non-null int64
Techie                  10000 non-null object
Contract                10000 non-null object
Port_modem              10000 non-null object
Tablet                  10000 non-null object
InternetService         10000 non-null object
Phone                   10000 non-null object
Multiple                10000 non-null object
OnlineSecurity          10000 non-null object
OnlineBackup            10000 non-null object
DeviceProtection        10000 non-null object
TechSupport             10000 non-null object
StreamingTV             10000 non-null object
StreamingMovies         10000 non-null object
PaperlessBilling        10000 non-null object
PaymentMethod           10000 non-null object
Tenure                  10000 non-null float64
MonthlyCharge           10000 non-null float64
Bandwidth_GB_Year       10000 non-null float64
Item1                   10000 non-null int64
Item2                   10000 non-null int64
Item3                   10000 non-null int64
Item4                   10000 non-null int64
Item5                   10000 non-null int64
Item6                   10000 non-null int64
Item7                   10000 non-null int64
Item8                   10000 non-null int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

For cluster analysis, any rows with null values would be ignored by the clustering model, which could lead to an extremely inaccurate model. If we had null values in this dataset, we would likely need to replace those null values with the mean of the variable where the null value is present. However, as we can see in the above screenshot, our dataset seems fairly clean, with all variables having the same number of observations (10,000) in addition to all variables being free of any null values.

C2.

For this analysis, we want to select variables that the K-Means clustering algorithm should be able to use to determine segmentation clusters for our customers. We will want to select variables that will cluster customers into groups that can be used for marketing campaigns and other business initiatives within the company. In order to achieve this, we have selected the continuous variables 'Income' and 'MonthlyCharge' as our variables to determine if we can cluster these groups accurately so that we can get a better sense of how income relates to how many services customers pay for on a monthly basis. We will also be utilizing some other additional variables that would also be useful for the same purposes stated above.

- Income - Continuous variable
- MonthlyCharge - Continuous variable

Additional variables we will be splitting into their own data frame with Income and MonthlyCharge:

- Age - Continuous variable
- Bandwidth_GB_Year - Continuous variable
- Tenure - Continuous variable

C3.

To prepare our data for our analysis, we first import the libraries we will be utilizing:

```
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 from a .csv to a Pandas dataframe:

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

We use Pandas .info() function to get an overview of our variables, number of observations, null values, and data types:

```
df.info()
```

We then drop any duplicate rows from the dataset:

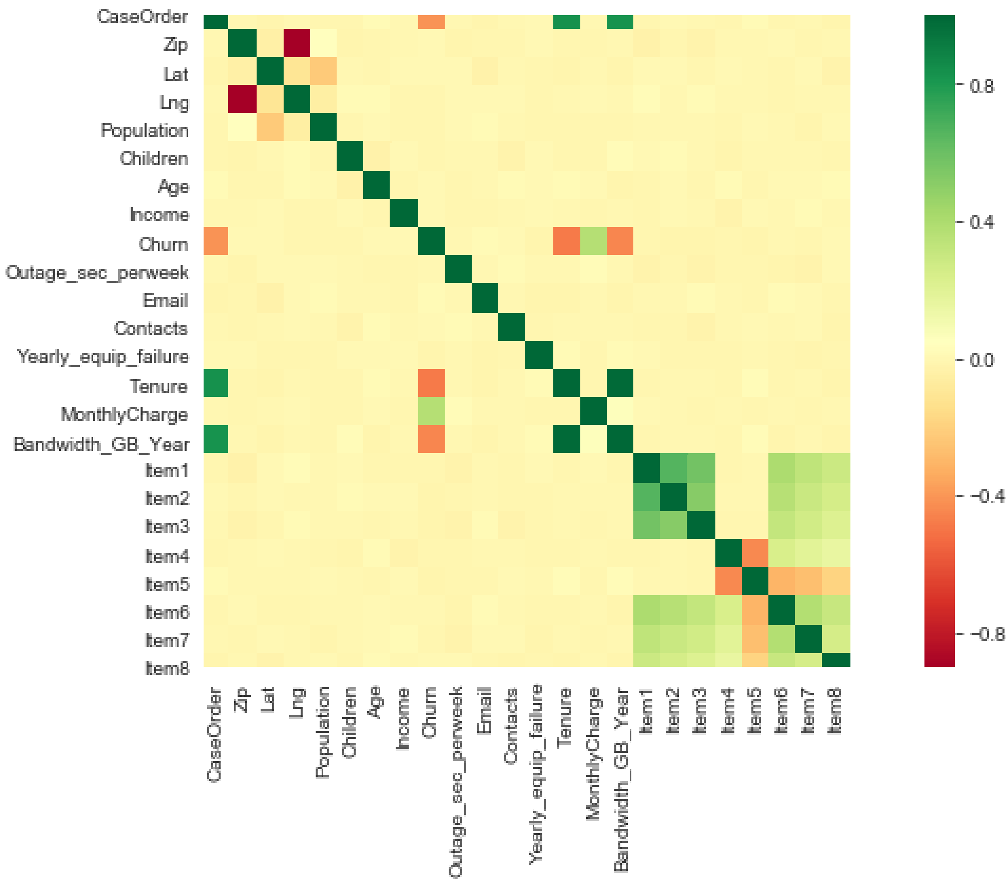
```
df.drop_duplicates()
```

We then double check the dataset for the sum of any null values per variable:

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

We then plot a heatmap to show positive and negative correlations within the dataset:

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



We then create a new dataframe containing only a handful of variables relevant to our analysis:

```
df1=df[["Age","Income","MonthlyCharge","Bandwidth_GB_Year","Tenure"]]
```

We then define X, which will be our variables selected for our K-Means model:

```
X=df1[["Income","MonthlyCharge"]]
```

We then check the top 5 rows of X to ensure it contains the variables we selected:

```
X.head(5)
```

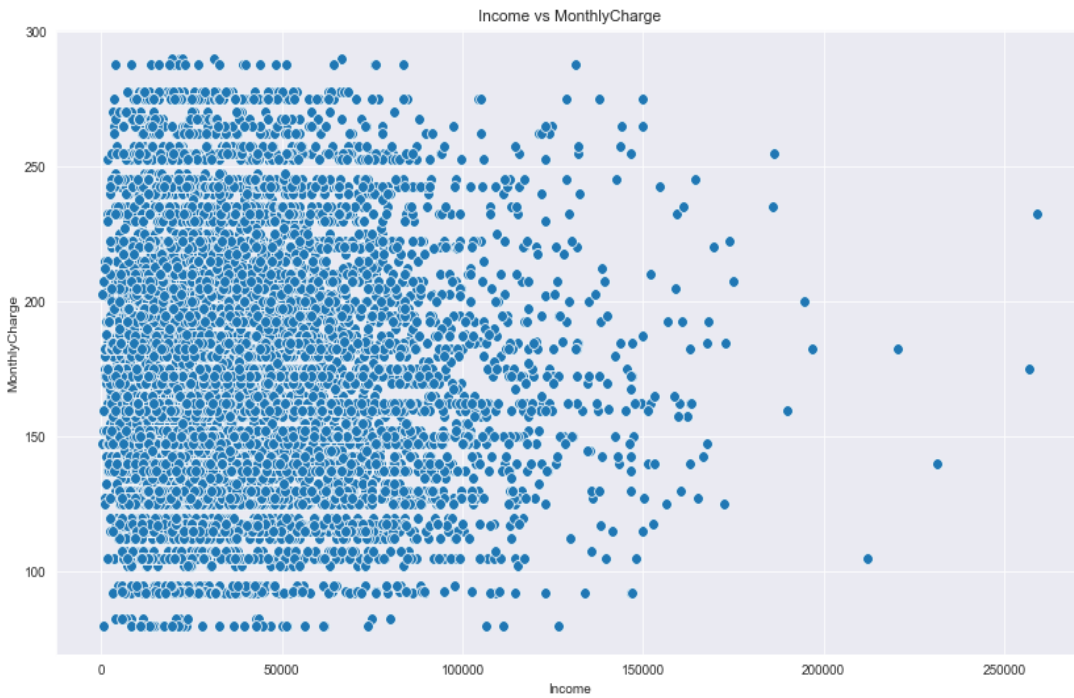
We then check the measures of center on both of our variables within X, just to get a sense of the data:

```
df.Income.describe()
df.MonthlyCharge.describe()
```

We then create a scatterplot of our two variables:

```
plt.figure(figsize=(14,8))
```

```
sns.scatterplot(x = 'Income',y = 'MonthlyCharge',  data = X  ,s = 60 )
plt.xlabel('Income')
plt.ylabel('MonthlyCharge')
plt.title('Income vs MonthlyCharge')
plt.show();
```



We then import packages from sklearn, including our pipeline, standardscaler, and K-Means model:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

Lastly, we export our cleaned dataset by utilizing Pandas to_csv() function:

```
df1.to_csv('D212_Task1_Dataset.csv')
```

C4.

Please see the attached ‘D212_Task1_Dataset.csv’ file.

Part IV: Analysis

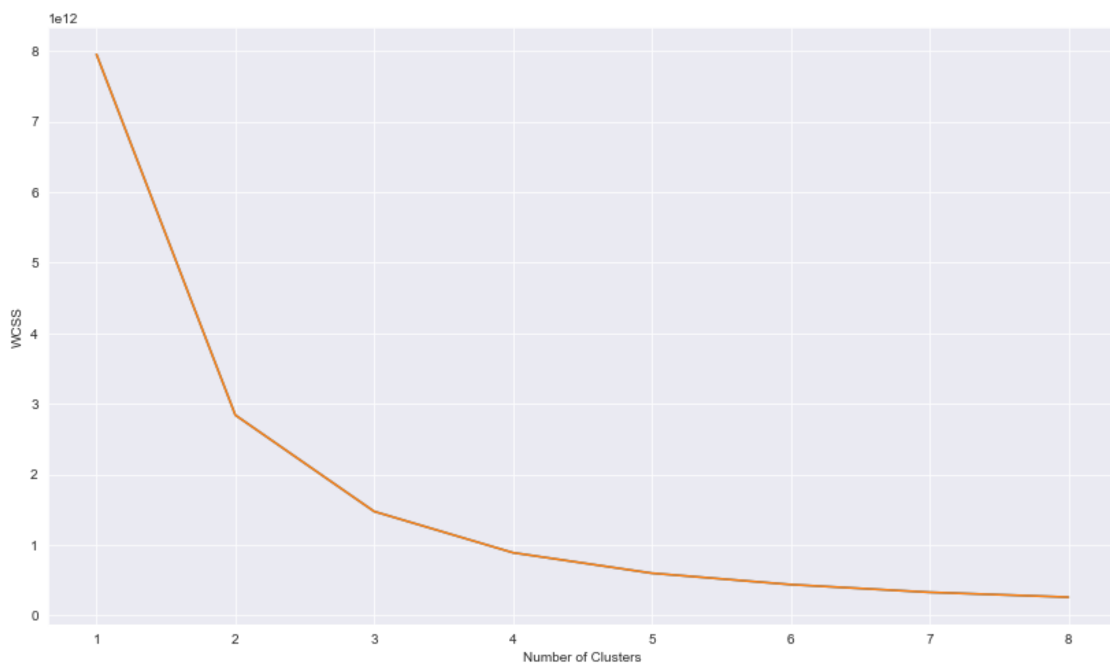
D1-D2.

We begin our K-Means analysis by determining how many clusters (*k*) we will need to utilize for our model. We can do this by utilizing the elbow method , which will iterate through our data using a range of clusters to determine their inertia, which measures how well the dataset has been measured by our model.

We create a graph for our elbow method by using a WCSS model, which stands for Within Cluster Sum of Squared Errors:

```
wcss=[]
for i in range(1,9):
  km=KMeans(n_clusters=i)
  km.fit(X)
  wcss.append(km.inertia_)

plt.figure(figsize=(14,8))
plt.plot(range(1,9),wcss)
plt.plot(range(1,9),wcss)
plt.xlabel("Number of Clusters")
plt.xticks(np.arange(1,9,1))
plt.ylabel("WCSS")
plt.show();
```



As we can see in the elbow method chart above, our ideal number of clusters is 4, as beyond 4 clusters the WCSS diminishes very slowly.

We then set up our feature scaler, define our number of clusters, and create a pipeline, so that our K-Means model first scales our data before applying the K-Means model.

```
scaler = StandardScaler()
kmeans=KMeans(n_clusters=4)
pipeline = make_pipeline(scaler, kmeans)
```

We then fit our model to our feature variables, X:

```
kmeans.fit(X)
```

We then predict on X to determine our labels, y:

```
y=kmeans.predict(X)
```

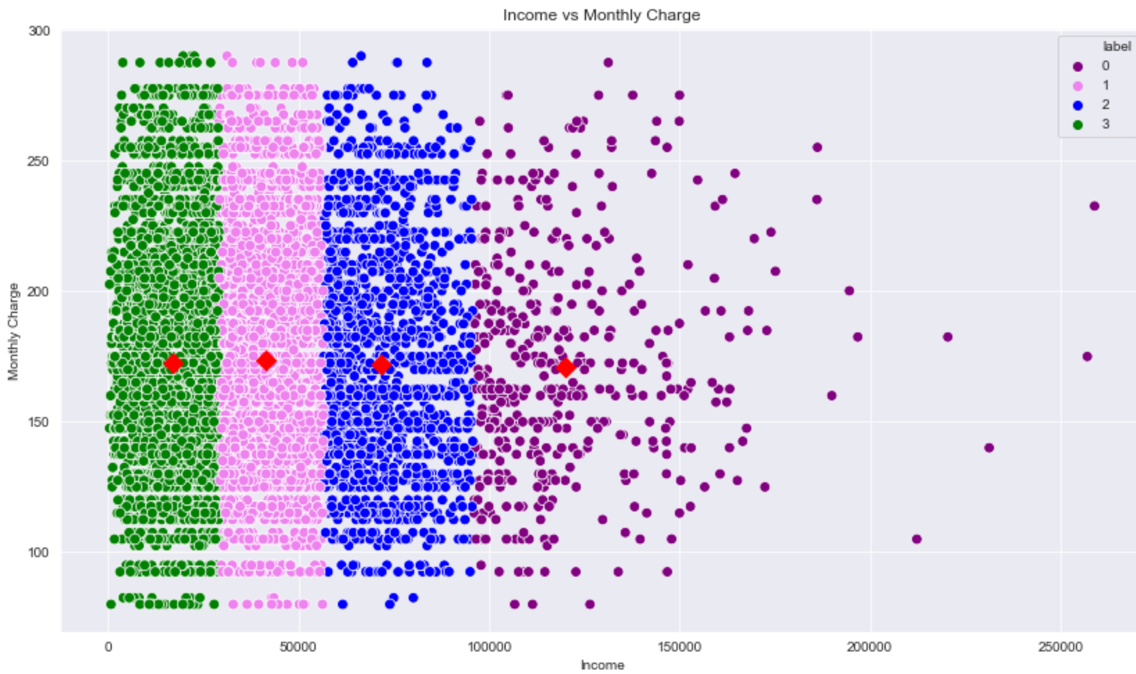
We then add the predicted labels, y, back to our data frame, df1, and check to ensure the column has been added by utilizing Pandas .head() function:

```
df1["label"] = y
df1.head()
```

We then plot our feature variables, along with their clusters and centroids:

```
plt.figure(figsize=(14,8))
sns.scatterplot(x = 'Income',y = 'MonthlyCharge',hue="label",
        palette=['purple','violet','blue','green'], legend='full',data = df1  ,s = 60 )

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'Red', label =
'Centroids', marker='D')
plt.xlabel('Income')
plt.ylabel('Monthly Charge')
plt.title('Income vs Monthly Charge')
plt.show();
```



Part V: Data Summary and Implications

E1-E2.
Our four cluster K-Means model has created four clusters of customers based on Income and MonthlyCharge, and has successfully defined those clusters' centroids. However, our clusters are not tightly clustered around their respective centroids, rather the four clusters have been defined more as bands than spherical clusters. This is likely due to clustering techniques working best when data is in distinct, relatively uniform clusters, which our data is not. The plot with 'Income' and 'MonthlyCharge' shows that our data is not a uniform distribution, and it is definitely not spherical.

Keeping in mind that the K-Means model makes the assumption that the variance of the distribution of each variable is spherical, we can already see that this is not the case in our data. While the centroids may be accurately defined for each band, it is likely that the model, when applied to the variables we have chosen for this analysis, is not accurate. As we stated above when discussing assumptions of K-Means, when the clusters are not spherical, the clusters will potentially not be as defined, in addition to the clusters not proving very useful in terms of gaining insight from the algorithm (All the Annoying Assumptions - Towards Data Science, 2019). It would appear as though this rings true, and while we have defined centroids, we are unable to truly gain any useful insight from the application of K-Means to our selected variables from this dataset.

E3.
A limitation of this analysis is that our dataset does not appear to be well suited to a K-Means clustering analysis. While K-Means should be extremely useful for customer segmentation, when looking at Income and MonthlyCharge, we can see that nearly all customers across all income levels have wildly varying monthly charges. In a realistic dataset, one would likely expect to see lower income customers paying lower monthly charges, due to being unable to afford expensive monthly charges for services. However, in this dataset, we can see that even the lowest income customers have some of the most expensive monthly charges, while higher income customers have lower monthly charges.

This could imply that our data is either full of outliers or, if it is accurate, looking at Income and MonthlyCharge to cluster customers will not provide any beneficial insight into what services different customers will pay for based on their income.

E4.
It is worth noting that our dataset shows very few correlations, which could be seen in our correlation heatmap above, which may make finding insights or relationships within the data difficult. As a recommendation to stakeholders in the business for this situation, it would be recommended to pursue multiple iterations of this model with different variables. It would also be recommended to perform a more detailed and thorough exploratory data analysis of the dataset to determine a useful business question that *can* be answered with the given data before exploring a question.

Part VI: Demonstration
Please see 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/

All the Annoying Assumptions - Towards Data Science. (2019, August 26). Medium. Retrieved October 21, 2021, from https://towardsdatascience.com/all-the-annoying-assumptions-31b55df246c3

Clustering for dataset exploration - Unsupervised Learning. (n.d.). DataCamp. Retrieved October 21, 2021, from https://campus.datacamp.com/courses/unsupervised-learning-in-python/clustering-for-dataset-exploration

Garbade, M. J. (2018, September 13). *Understanding K-means Clustering in Machine Learning*. Medium. Retrieved October 21, 2021, from https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

K-means clustering is not a free lunch. (2015, January 16). Variance Explained. Retrieved October 27, 2021, from http://varianceexplained.org/r/kmeans-free-lunch/

Majumder, P. (2021, May 25). *K-Means clustering with Mall Customer Segmentation*. Analytics Vidhya. Retrieved October 21, 2021, from https://www.analyticsvidhya.com/blog/2021/05/k-means-clustering-with-mall-customer-segmentation-data-full-detailed-code-and-explanation/

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/

(*K-Means Clustering Is Not a Free Lunch*, 2015)