D209: Data Mining I Task 1

Justin Huynh

Student ID: 012229514

M.S. Data Analytics

A1. Proposal of Question

My research question for this project is “What factors most significantly contribute to whether customers will churn or not?” We will use k-nearest neighbor (KNN) to answer our question.

A2. Defined Goal

The primary goal of this analysis is to develop a predictive model that can accurately classify whether a customer will churn or not based on their demographic, service usage, account information, etc. This model will help an organization identify key contributing factors to customer churn and enable them to proactively take action to improve retention rate. This analysis is an extension of the research question and analysis from D208 task 2 to further explore why customers churn.

B1: Explanation of Classification Method

The k-nearest neighbor (KNN) analyzes the select data set by classifying a data point based on the majority class among its k-nearest neighbors in the feature space. This is the mechanism of the instance-based learning algorithm. Another way it analyzes the data set is the KNN’s process of calculating the distance between the customer and all other customers in the training data set. It then selects the KNN, where k is a predefined integer, and assigns the class that is most common among these neighbors. An expected outcome is a classification of whether a customer will churn or not. The model leverages the similarity in feature space (e.g., age, income, monthly charge, etc.) to predict the likelihood of churn based on the behavior of similar customers in the training set.

B2: Summary of Method Assumption

One assumption of KNN is the assumption of feature space homogeneity. The KNN assumes that similar data points exist in close proximity in the feature space. This means that customers with similar features will have similar outcomes (churn or not churn). The algorithm relies on the idea that the decision boundary can be determined locally by examining a small subset of similar instances.

B3: Packages or Libraries List

For this analysis, we will be using Python and its libraries and packages. We will be using pandas for data manipulation and analysis due to its ability to provide structures like dataframes that help simplify cleaning, encoding, and splitting data sets. We'll also be using numpy due to its ability to perform numerical computations on large data sets that is essential for distance calculations in KNN. We'll also use scikit-learn since this machine learning library can implement model training, evaluation, and validation. Lastly, we'll use matplotlib and seaborn for creating visualizations, which will help us visualize data distributions, relationships, and evaluation metrics to better understand the model performance.

C1: Data Processing

The main goal of data preprocessing for KNN is to ensure that all features are in a suitable format for the algorithm to compute distances effectively. This involves normalizing numerical features to a common scale and encoding categorical variables into numeric values. This step is crucial for KNN as it relies on distance calculations in the feature space.

C2: Data Set Variables

These are the identified variables for our analysis:

- Numeric Variables
    - ‘Age’
    - ‘Income’
    - ‘MonthlyCharge’
    - ‘Bandwidth_GB_Year’
    - ‘Tenure
- Categorical Variables
    - ‘Gender’
    - ‘Contract’
    - ‘TechSupport’
    - ‘Churn’ (target variable)

C3: Steps for Analysis

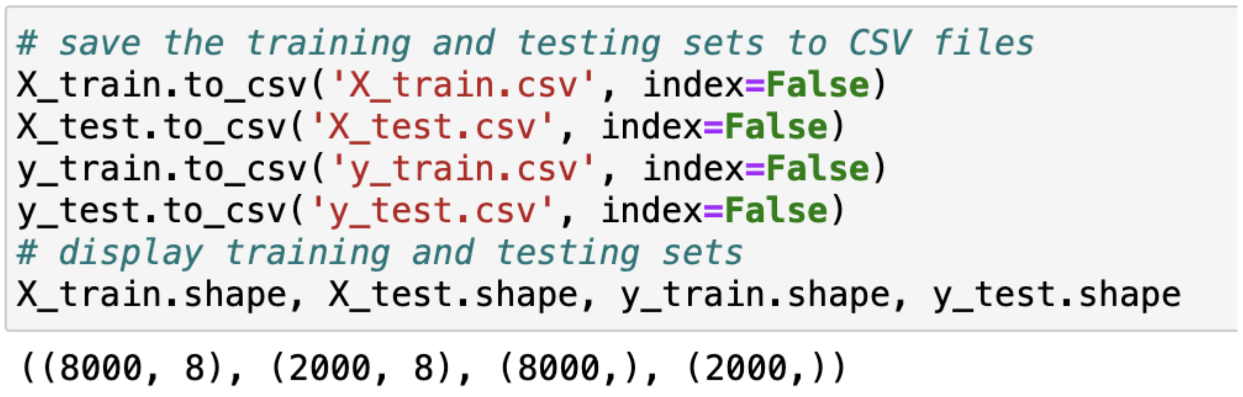
For our first step, we’ve loaded the ‘churn_clean.csv’ file into our notebook so we can display the data using pandas to verify the data loaded correctly. Next, we select only the relevant variables needed for our analysis and display the selected features for verification. We’ll then encode our categorical variables into numeric using LabelEncoder from sklearn library. We then normalize the data using StandardScaler from sklearn to ensure all features contribute equally to the distance calculations in the KNN algorithm. Lastly, we’ll split the data set into training and testing sets using train_test_split from sklearn once again. All these steps will be done before saving it to a csv file.

See code attached in WGU_D209_Task_1.ipynb.

C4: Cleaned Data Set

A copy of 'prepared_data_d209_task1.csv' will be submitted with this task.

D1: Splitting the Data



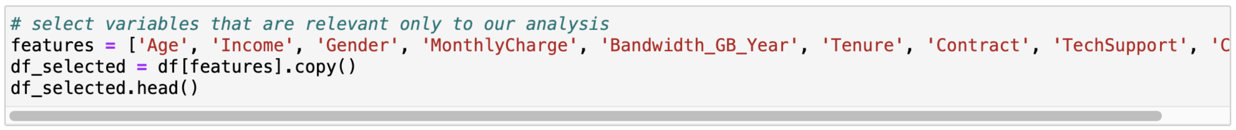
```
# save the training and testing sets to CSV files
X_train.to_csv('X_train.csv', index=False)
X_test.to_csv('X_test.csv', index=False)
y_train.to_csv('y_train.csv', index=False)
y_test.to_csv('y_test.csv', index=False)
# display training and testing sets
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((8000, 8), (2000, 8), (8000,), (2000,))
```

Copies of the training and testing sets will be submitted in csv files with this task. *See code attached in WGU_D209_Task_1.ipynb.*

D2: Output and Intermediate Calculations

For this analysis, we used the k-nearest neighbor (KNN) technique. We first encoded the categorical variables into numeric before we normalized the numeric features to ensure they all contribute equally to the distance calculations for KNN. For the algorithm, we used KNeighborsClassifier from the scikit-learn library. The algorithm calculates the distance between a data point and its neighbors and assigns the data point to the class most common among its k-nearest neighbors. We then evaluate the model performance using classification report and confusion matrix.

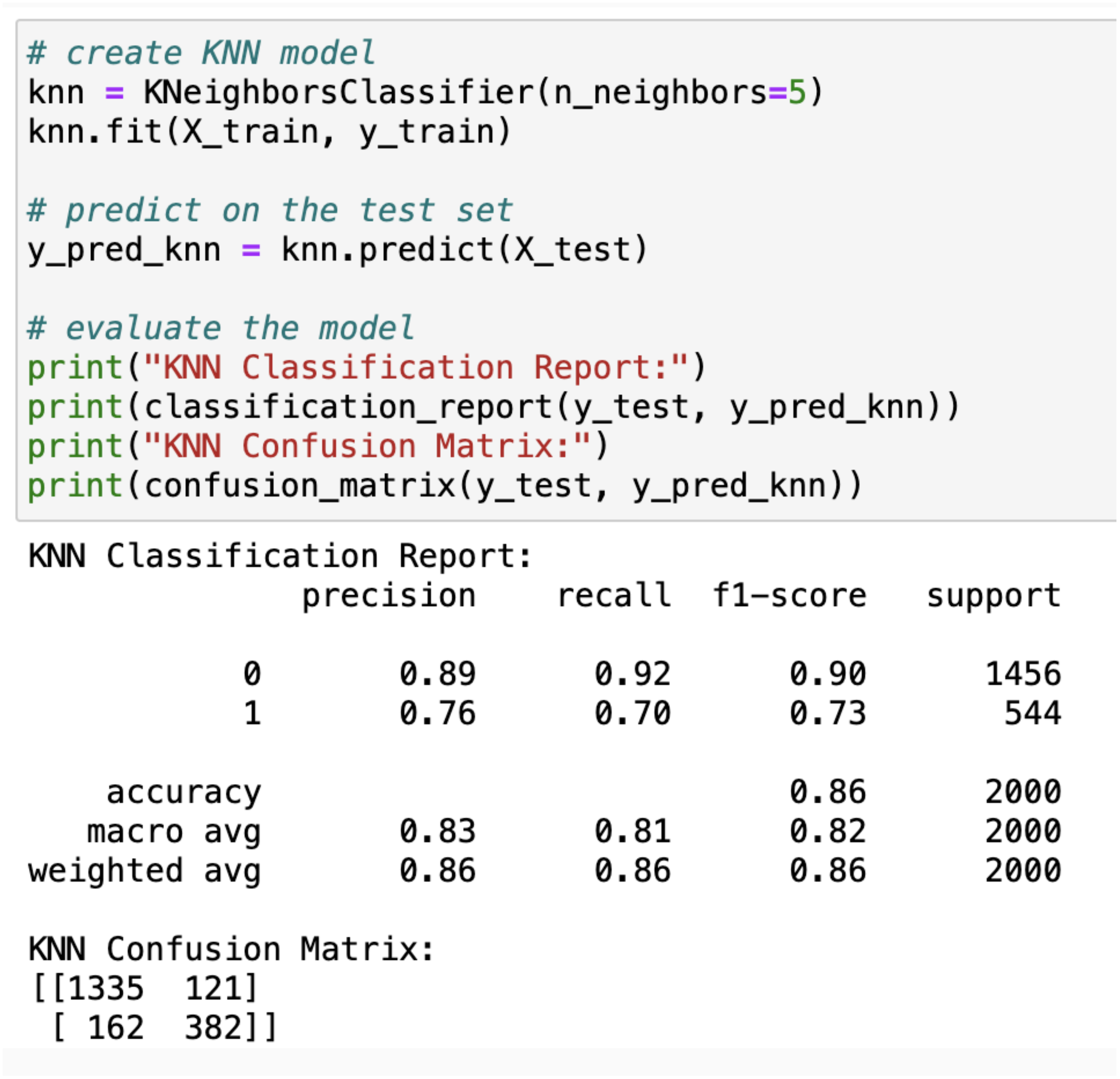


```
# select variables that are relevant only to our analysis
features = ['Age', 'Income', 'Gender', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Tenure', 'Contract', 'TechSupport', 'C
df_selected = df[features].copy()
df_selected.head()
```

```
# convert relevant categorical variables to numerical using one-hot encoding
label_encoder = LabelEncoder()
df_selected['Gender'] = label_encoder.fit_transform(df_selected['Gender'])
df_selected['Contract'] = label_encoder.fit_transform(df_selected['Contract'])
df_selected['TechSupport'] = label_encoder.fit_transform(df_selected['TechSupport'])
df_selected['Churn'] = label_encoder.fit_transform(df_selected['Churn'])
df_selected.head()
```

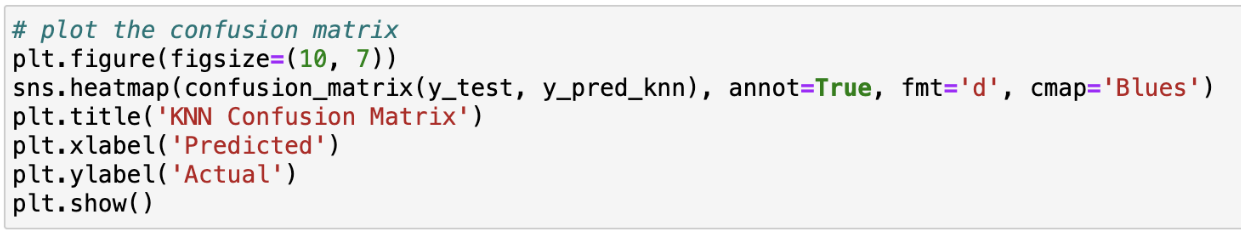
```
# normalize numeric variables
numeric_features = ['Age', 'Income', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Tenure']
scaler = StandardScaler()
df_selected[numeric_features] = scaler.fit_transform(df_selected[numeric_features])
df_selected.head()
```

```
# split data into training and testing sets
X = df_selected.drop('Churn', axis=1)
y = df_selected['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# display training and testing sets
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

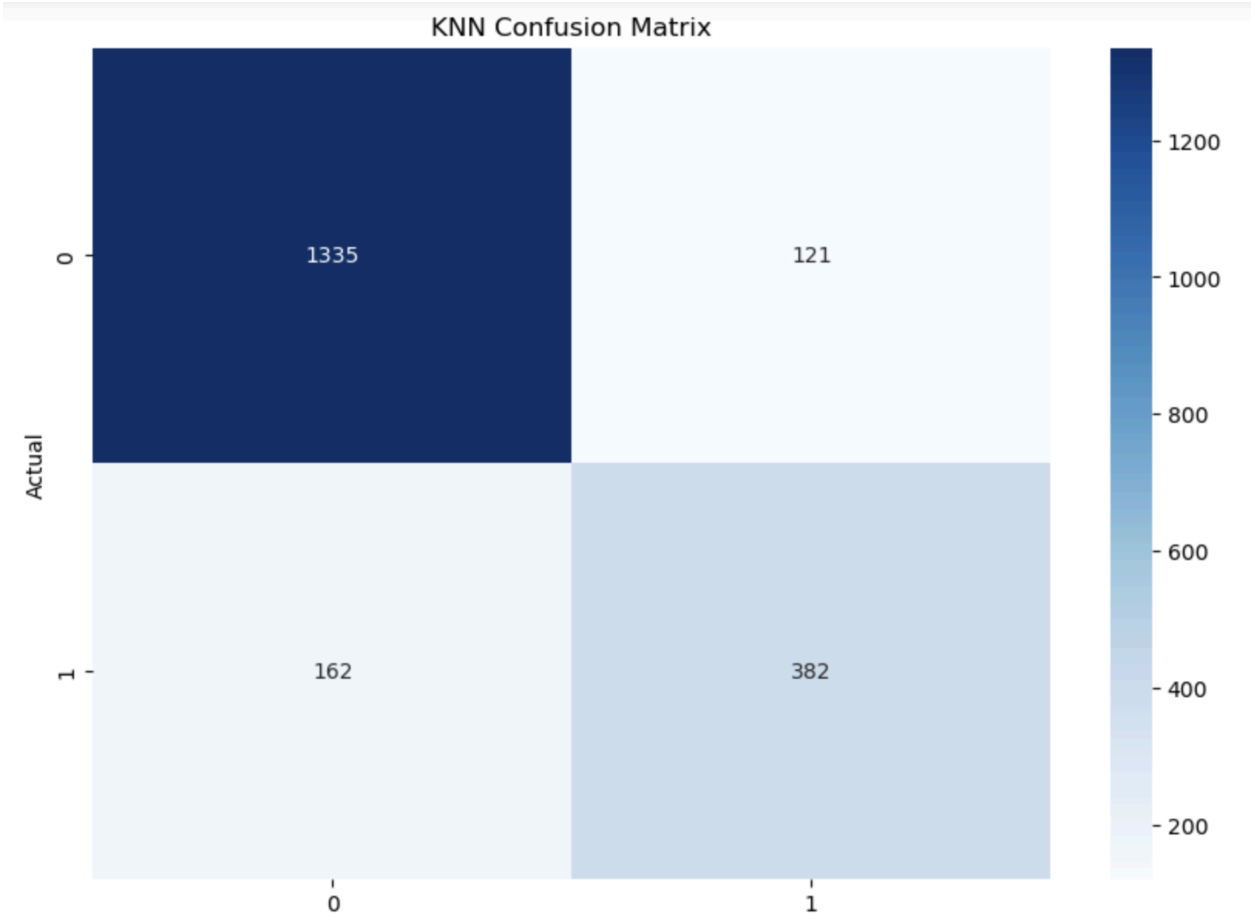


```python
# create KNN model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# predict on the test set
y_pred_knn = knn.predict(X_test)

# evaluate the model
print("KNN Classification Report:")
print(classification_report(y_test, y_pred_knn))
print("KNN Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_knn))
```

```
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.92      0.90      1456
           1       0.76      0.70      0.73       544

    accuracy                           0.86      2000
   macro avg       0.83      0.81      0.82      2000
weighted avg       0.86      0.86      0.86      2000

KNN Confusion Matrix:
[[1335  121]
 [ 162  382]]
```



```python
# plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(confusion_matrix(y_test, y_pred_knn), annot=True, fmt='d', cmap='Blues')
plt.title('KNN Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



See code attached in WGU_D209_Task_1.ipynb.

D3: Code Execution

See code attached in WGU_D209_Task_1.ipynb.

E1: Accuracy and AUC

To evaluate the KNN classification model, we use the metrics accuracy and AUC (Area Under the Curve) to look at the model performance. The accuracy metric indicates the proportion of correctly classified instances out of the total instances. It is calculated as (True Positives + True Negatives) / (Total Instances). For this model, the accuracy is 0.86, meaning the model correctly classifies 86% of the instances. The AUC represents the model's ability to distinguish between classes. AUC ranges from 0 to 1, with 1 indicating perfect discrimination and 0.5 indicating no discrimination. For this model, an AUC of 0.90 indicates excellent model performance, meaning the model can effectively differentiate between customers who will churn and those who will not.

See code attached in WGU_D209_Task_1.ipynb.

E2: Results and Implications

The accuracy of the KNN model is at 86%, indicating that the model correctly classifies 86% of the instances while the AUC value of 0.90 shows the model's ability to distinguish between churned and non-churned customers is very high. The classification report shows that the precision metric has a value of 0.89 for non-churned customers (class 0) and 0.76 for churned customers (class 1), the recall metric has a value of 0.92 for non-churned customers and 0.70 for churned customers, and the F1-score value of 0.90 for non-churned customers and 0.73 for churned customers. The implications mean that when the model is predicting when a customer will not churn, it is correct 89% of the time and identifies 92% of non-churned customers. The model also has moderate precision and recall when predicting churned customers, with it being correct 76% of the time and identifying 70% of the churned customers. The weighted average precision of the model performance of 86% indicates a balanced performance across both classes.

E3: Limitation

One limitation of the KNN technique is the algorithm's performance is sensitive to the choice of k (number of neighbors) and the presence of outliers in the data. The KNN also doesn’t perform well with high-dimensional data because the distance metric becomes less meaningful in higher dimensions. These factors can affect the model's ability to accurately classify churned and non-churned customers. Using the KNN technique can also be expensive for large data sets since it requires storing and comparing all data points during prediction.

E4: Course of Action

A course of action an organization can take based on the results is implementing targeted retention campaigns by using the model's performance to identify customers who are at high risk of churning and create campaigns specifically to retain these customers by offering discounts, personalized promos, enhancing customer support etc. The organization should also regularly monitor and update their model by updating it with new data so the accuracy and relevance of the performance is maintained.

F: Panopto Video

The URL link will also be submitted in the Performance Assessment task submission.

G: Sources of Third Party Code

1. "K-Nearest Neighbor (KNN) Algorithm" Retrieved from https://www.geeksforgeeks.org/k-nearest-neighbours/ l
2. "Building a k-Nearest-Neighbors (KNN) model with Scikit-learn" Retrieved from https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a
3. "ConfusionMatrixDisplay" Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html

H: Web Sources

1. “Confusion Matrix in Machine Learning.” Retrieved from https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
2. “D208: Predictive Modeling Task 2” Retrieved from WGU_D208_Task2_Justin_Huynh.pdf