Data Mining I - D209
Task 1
Eric D. Otten
Student ID: 011399183

A1. Propose one question relevant to a real-world organizational situation that you will answer using one of the following classification methods.

My research question, "Which customer factors contribute most to a customer's decision to churn?" is highly relevant to any organization, particularly in competitive industries such as telecommunications. Understanding the key drivers of customer churn can help a company implement targeted interventions to improve customer retention, which is often more cost-effective than acquiring new customers. To answer my research question, I will be using the telecommunications churn data found in churn_cleaned.csv and I will be using the KNN classification method to perform my analysis.

By identifying specific factors that lead to churn, the organization can tailor its products, services, and customer interactions to better meet the needs and preferences of its customers. This approach can enhance customer satisfaction and loyalty, leading to increased lifetime value of customers and sustained revenue streams.

Additionally, insights derived from churn analysis can inform strategic decisions around marketing, customer service, and product development. For instance, if certain services or billing issues are identified as major churn influencers, the organization can prioritize resolving these issues. More specifically, analyzing the customer factors which contribute most heavily to churn can be leveraged in the following ways:

1. **Marketing strategies:** Data identifying why customers churn can guide more personalized marketing efforts. For example, if young customers are found to churn due to lack of digital engagement, the company can introduce targeted social media campaigns or app-based interactions.

2. **Service optimization:** Understanding specific features or services that lead to churn allows a company to refine or eliminate underperforming services or to introduce new features that meet customers' needs more effectively.

3. **Price adjustments:** Insights from churn analysis can inform adjustments in pricing strategies to align better with what different customer segments value and are willing to pay for.

4. **Customer Segmentation**: Analyzing churn can help refine customer segmentation by identifying characteristics of at-risk customers, allowing for proactive engagement and retention strategies.

Overall, reducing churn rates not only boosts profitability but also strengthens the organization's market position and brand reputation.

A2. 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.

My goal with this data analysis is to improve upon the linear regression model completed in D208. The goal of using KNN in this context is to capture the complexities and non-linear relationships that might exist between customer attributes and their decision to churn, which may not have been adequately modeled by linear regression. For example:

A linear regression model assumes a linear relationship between the dependent and independent variables. This can be limiting if the true relationship is complex or nonlinear. On the other hand, KNN makes no assumptions about the underlying data distribution.

Linear regression predicts a continuous output, which is ideal for tasks like estimating real values (e.g., predicting prices or age). KNN is typically used for classification problems where the output is a category, such as 'churn' or 'no churn'. It identifies the class of a data point by a majority vote of its neighbors, with the data point being assigned to the class most common among its k nearest neighbors.

Lastly, linear regression can be sensitive to outliers, which can significantly influence the slope and intercept of the regression line. KNN is sensitive to the scale of the data and requires normalization for all features to contribute equally, especially because it uses distances between data points to determine their similarity.

The goal of using KNN in this context is to capture the complexities and non-linear relationships that might exist between customer attributes and their decision to churn, which may not have been adequately modeled by linear regression. KNN can be particularly useful if the decision to churn is influenced more by the similarity of customer profiles than by an underlying linear trend across individual variables. By evaluating the closeness of various customer instances, KNN could potentially provide a more nuanced understanding of churn patterns based on real-world data distributions.

Using KNN could also help address specific challenges where the linear model's assumptions don't hold true, providing a model that may be better suited for predicting categorical outcomes like churn.

B1. Explain how the classification method you chose analyzes the selected data set. Include expected outcomes.

To address my research question, “Which customer factors contribute most to a customer’s decision to churn?”, we can use the K-Nearest Neighbors (KNN) classification method to identify patterns and influential features in the dataset. By analyzing the characteristics of customers who have churned (i.e., canceled their subscription to the internet service provider,) such as their demographics, service usage, and interaction with customer support, we can discern which attributes are most heavily correlated with churn. KNN is a non-parametric, instance-based learning algorithm that classifies a data point based on how its neighbors are classified.

The KNN algorithm works as follows:

1. **Choose the number of neighbors (k):** This is a user-defined constant that determines the number of nearest neighbors to include in the majority voting process.
2. **Calculate the distance:** For each data point in the test set, the algorithm calculates the distance (commonly Euclidean distance) between the test point and all the points in the training set.
3. **Sort and select neighbors:** The algorithm sorts the distances in ascending order and selects the k-nearest neighbors to the test point.
4. **Majority vote:** The algorithm assigns the test point to the class that is most common among its k-nearest neighbors. (Classification: K Nearest neighbours.)

In the context of customer churn analysis:

- **Training Phase:** The KNN algorithm will be trained on the historical data of customer attributes (e.g., demographics, service usage, customer support interactions) and their churn status (churn or no churn).
- **Prediction Phase:** For each new customer, the algorithm will calculate the distance to all existing customers in the training set, identify the k-nearest neighbors, and predict the churn status based on the majority class of these neighbors.

By applying KNN to the churn dataset, we expect to:

1. **Identify Influential Factors:** Determine which customer attributes are most indicative of churn by observing the common characteristics of the neighbors in the same class.
2. **Predict Churn:** Accurately predict whether a new customer will churn based on the attributes of similar customers.

3. **Gain Insights into Customer Behavior:** Gain insights into patterns of customer behavior that contribute to churn, which can inform targeted interventions to reduce churn rates.

Factors like the length of tenure, monthly charges, and the type of services subscribed (e.g., online security, device protection) could significantly impact the likelihood of churn. Additionally, customer satisfaction metrics from survey responses about service quality may also play a critical role. Using KNN, we can cluster similar customers and observe the common factors among those who decide to churn, yielding actionable insights for reducing future churn rates.

B2. Summarize one assumption of the chosen classification method.

The biggest assumption of the K-Nearest Neighbors (KNN) classification method is the relevance of the distance metric. The performance of KNN heavily relies on the metric used to calculate distances between data points. Euclidean distance is the most common choice, but it may not be the best choice for all types of data. The effectiveness of KNN assumes that the chosen metric accurately reflects the meaningful similarities between data points. For example, Euclidean distance can be inappropriate for high-dimensional spaces due to the "curse of dimensionality".

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

To perform this data analysis, I will develop a comprehension Python script which utilizes the following python libraries and tools:

1. The Pandas library. Pandas is a Python library used for data manipulation and analysis that provides data structures and operations for manipulating numerical tables and time series. The most prominent feature of this package is the `pandas.DataFrame` constructor which allows for the storage and manipulation of data with rows and columns, similar to a spreadsheet or SQL table.

2. The NumPy library. Numpy is a Python library that adds support for multi-dimensional arrays and matrices and high-level mathematical functions like Fourier transforms and matrix multiplications to operate on these arrays.

3. The MatPlotLib library. MatPlotLib is a visualization library in Python. It is a base library that provides a lot of flexibility but can be complex when attempting to create advanced visualizations.

4. The Seaborn library. Seaborn is another visualization library in Python built on top of MatPlotLib that provides a high-level interface for creating graphs such as bar charts, line charts, histograms, scatter plots, etc and simplifies many aspects of visualization.

5. SciPy's statsmodels API. Statsmodels is a Python module that provides classes and functions to estimate many different statistical models and conduct statistical tests and statistical data exploration. It includes various tools for modeling statistics, including linear regression, time series analysis, etc.

    a. variance_inflation_factor. The variance_inflation_factor function within the statsmodels module provides multicollinearity diagnosis functionality. The VIF, or Variance Inflation Factor, is a measure of multicollinearity in regression models. Multicollinearity occurs when two or more predictor variables in a model are highly correlated, which can make the model's estimates very sensitive to changes. VIF quantifies how much the variance of an estimated regression coefficient increases if your predictors are correlated.

    b. mosaic. The mosaic function within the statsmodels module is used to visualize the relationship between two categorical variables in a mosaic plot. Each rectangle in the mosaic plot represents a combination of categories and the size of the rectangle is proportional to the frequency of that combination occurring within the dataset.

6. The Scikit-Learn/sklearn library. The sklearn library offers a broad array of machine learning algorithms including classification, regression, clustering, and dimensionality reduction. Additionally, it provides utilities for model fitting, data preprocessing, model selection, and evaluation, making it extremely versatile. One of the core strengths of sklearn is its consistent and intuitive API and integration with other libraries in the Python data science stack, such as NumPy and SciPy. While primarily designed for small to medium data sets, sklearn can handle much larger data sets by integrating it with other tools in the Python ecosystem. Specifically, the following functions will be used from the sklearn library:

    a. train_test_split. The train_test_split function splits the dataset into a training set and a testing set which is essential for training and validating machine learning models without overfitting. The parameter test_size takes either a float or an integer and specifies the size of the test set. Alternatively, one can use the parameter train_size to specify the size of the training set. Lastly the parameter random_state allows for setting the model seed and helps maintain reproducibility.

    b. preprocessing. The preprocessing function includes scaling, transforming, and encoding features, allowing for normalized and standardized data to be used in a machine learning model.

    c. SelectKBest. The SelectKBest function allows for feature selection according to the highest k scores.

    d. f_classif. The f_classif function computes the ANOVA F-value for a provided sample.

    e. KNeighborsClassifier. The KNeighborsClassifier function provides a type of classification model that predicts the class of data points based on the k-nearest neighbors to a query. scikit learn also includes RadiusNeighborsClassifier which implements learning based on the number of neighbors within a fixed radius r of each training point. However, we will be using only the KNeighborsClassifier as it is the most commonly used technique and the data is likely to be uniformly sampled.

    f. GridSearchCV. The GridSearchCV function allows for parameter tuning, the aim of which is to find the best parameters for a model, improving the model's accuracy and performance. One of the attributes of all GridSearchCV objects is best_params_, which contains

    g. confusion_matrix. The confusion_matrix function computes a confusion matrix to evaluate the accuracy of a classification. The output of this function is a two-dimensional array where each row represents an actual index and each column represents a predicted index. The indices of the confusion matrix can be specified using the labels argument.

    h. roc_auc_score. The roc_auc_score function computes the area under the receiver operating characteristic curve (ROC AUC) from prediction scores. An AUC score of 1 indicates a perfect model; an AUC of 0.5 is equivalent to random guessing, and an AUC score of 0 suggests a perfectly incorrect model.

    i. roc_curve. The roc_curve function computes the receiver operating characteristic. The function returns three values: fpr, tpr, and thresholds.

    j. classification_report. The classification_report function returns a summary of the precision, recall, F1 score for each class in the form of a dictionary.

C1. Describe one data preprocessing goal relevant to the classification method from part A1.

My goal with regards to cleaning the sample data is to create a uniform DataFrame to which the K-Nearest Neighbors (KNN) classification method can be applied and from which useful business insights can be drawn. This is similar to my data cleaning goal in D208, with the exception of the classification method used.

The provided dataset is cleaned yet contains several data anomalies which should be corrected:

- Zip codes are provided in int28 format instead of as a string and as a result have lost their leading zeroes. These rows will be converted to strings.
- Time zone categories are redundant. For example, separate time zones exist for EST: New York, Detroit, and others. These categories will be reduced to the standard US time zones. The following mappings will be used:
    - America/New_York will be mapped to EST.
    - America/Detroit will be mapped to EST.
    - America/Indiana/Indianapolis will be mapped to EST.
    - America/Kentucky/Louisville will be mapped to EST.
    - America/Indiana/Vincennes will be mapped to EST.
    - America/Indiana/Tell_City will be mapped to EST.
    - America/Indiana/Petersburg will be mapped to EST.
    - America/Indiana/Knox will be mapped to EST.
    - America/Indiana/Winamac will be mapped to EST.
    - America/Indiana/Marengo will be mapped to EST.
    - America/Toronto will be mapped to EST.
    - America/Chicago will be mapped to CST.
    - America/Menominee will be mapped to CST.
    - America/North_Dakota/New_Salem will be mapped to CST.
    - America/Denver will be mapped to MST.
    - America/Phoenix will be mapped to MST.
    - America/Boise will be mapped to MST.
    - America/Los_Angeles will be mapped to PST.
    - America/Anchorage will be mapped to AKST.
    - America/Nome will be mapped to AKST.
    - America/Sitka will be mapped to AKST.
    - America/Juneau will be mapped to AKST.
    - Pacific/Honolulu will be mapped to HAST.
    - America/Puerto_Rico will be mapped to AST.
    - America/Ojinaga will be mapped to MST.

For nominal categorical data, one hot encoding will be used as it is the most widespread approach. This approach involves creating a new column for each category, which contains a binary encoding of 0 or 1 to denote whether a particular row belongs to this category. This can be achieved using the get_dummies() method within the pandas library.

C2. Identify the initial data set variables that you will use to perform the analysis for the classification question from part A1 and classify each variable as numeric or categorical.

The following variables will be used to perform my data analysis for my classification question, “Which customer factors contribute most to a customer's decision to churn?”:

- **Population**: Population within a mile radius of the customer, based on census data. (Numeric)
- **Area**: Classification of the customer's area (rural, urban, suburban). (Categorical)
- **TimeZone**: Time zone of the customer's residence. (Categorical)
- **Children**: Number of children in the customer’s household as reported during sign-up. (Numeric)
- **Age**: Age of the customer as reported during sign-up. (Numeric)
- **Income**: Annual income of the customer as reported at the time of sign-up. (Numeric)
- **Marital**: Marital status of the customer. (Categorical)
- **Gender**: Gender of the customer as they self-identify. (Categorical)
- **Churn:** Whether the customer discontinued service within the last month (yes, no) (Categorical)

- **Outage_sec_perweek**: Average number of seconds per week of system outages experienced by the customer. (Numeric)
- **Email**: Number of emails sent to the customer in the last year. (Numeric)
- **Contacts**: Number of times the customer contacted technical support. (Numeric)
- **Yearly_equip_failure**: Number of times the customer's equipment failed and needed replacement or reset in the past year. (Numeric)
- **Techie**: Indicates whether the customer considers themselves technically inclined. (Categorical)
- **Contract**: The type of service contract the customer has (month-to-month, one year, two years). (Categorical)
- **Port_modem**: Whether the customer uses a portable modem. (Categorical)
- **Tablet**: Whether the customer owns a tablet device. (Categorical)
- **InternetService**: Type of internet service the customer has (DSL, fiber optic, none). (Categorical)
- **Phone**: Whether the customer has a phone service. (Categorical)
- **Multiple**: Whether the customer has multiple lines. (Categorical)
- **OnlineSecurity**: Whether the customer has an online security service. (Categorical)
- **OnlineBackup**: Whether the customer uses an online backup service. (Categorical)
- **DeviceProtection**: Whether the customer uses a device protection service. (Categorical)
- **TechSupport**: Whether the customer has technical support service. (Categorical)
- **StreamingTV**: Whether the customer uses streaming TV service. (Categorical)
- **StreamingMovies**: Whether the customer uses streaming movies service. (Categorical)
- **PaperlessBilling**: Whether the customer has opted for paperless billing. (Categorical)
- **PaymentMethod**: Method by which the customer makes payments (e.g., electronic check, mailed check, bank transfer, credit card). (Categorical)
- **Tenure**: Number of months the customer has been with the service provider. (Numeric)
- **MonthlyCharge**: Average monthly charge billed to the customer. (Numeric)
- **Bandwidth_GB_Year**: Average amount of data in GB used by the customer per year. (Numeric)
- **Item1**: Timely response - This survey item evaluates how quickly the company responds to customer inquiries and requests, which can significantly influence customer satisfaction and perception of the company's service efficiency. (Numeric)
- **Item2**: Timely fixes - This item measures the promptness of the company in addressing and resolving technical or service-related issues reported by customers. (Numeric)
- **Item3**: Timely replacements - This question assesses how swiftly the company manages to replace faulty or inadequate equipment or services that do not meet the customer's expectations or needs. (Numeric)
- **Item4**: Reliability - This survey question gauges the consistency and dependability of the company's services, reflecting how often customers face issues or disruptions. (Numeric)
- **Item5**: Options - This item looks at the variety and flexibility of service options available to customers, allowing them to choose services that best fit their needs and preferences. (Numeric)
- **Item6**: Respectful response - This question assesses the respectfulness and professionalism of the company's responses to customer interactions, which can affect the customer's overall service experience. (Numeric)

- **Item7**: Courteous exchange - Similar to respectful responses, this survey item evaluates the courtesy extended by the company during customer interactions, including politeness and positive communication. (Numeric)
- **Item8**: Evidence of active listening - This item measures the extent to which customers feel that the company listens to and understands their concerns or requests, which is crucial for effective service and support. (Numeric)

C3. Explain each of the steps used to prepare the data for the analysis. Identify the code segment for each step.

My goal with regards to cleaning the sample data is to create a uniform DataFrame to which the K-Nearest Neighbors (KNN) classification method can be applied and from which useful business insights can be drawn. To achieve this, the following data transformations will be performed:

1. Select only the columns that are relevant to the research question.
2. Standardize timezones using a mapping function.
3. Convert columns intended to represent boolean values to booleans with “Yes” being mapped to True and “No” being mapped to False.
4. Convert nominal variables that include repeating values to categories.
5. Use one-hot encoding to create dummy columns, with the first category of each categorical variable being dropped to avoid the dummy variable trap.

The annotated code below achieves these objectives:

1. **Select only the columns that are relevant to the research question.**

Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.graphics.mosaicplot import mosaic
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import preprocessing
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve,
classification_report
from sklearn.neighbors import KNeighborsRegressor

filename = "churn_clean.csv"
df = pd.read_csv(filename, keep_default_na=False, na_values=["NA"])
```

```
# C3: Data Cleaning

# Remove unused columns
df = df[[
    "Population", "Area", "TimeZone", "Children", "Age", "Income", "Marital", "Gender",
"Churn",
    "Outage_sec_perweek", "Email", "Contacts", "Yearly_equip_failure", "Techie",
    "Contract", "Port_modem", "Tablet", "InternetService", "Phone", "Multiple",
    "OnlineSecurity", "OnlineBackup", "DeviceProtection", "TechSupport", "StreamingTV",
    "StreamingMovies", "PaperlessBilling", "PaymentMethod", "Tenure", "MonthlyCharge",
    "Bandwidth_GB_Year", "Item1", "Item2", "Item3", "Item4", "Item5", "Item6", "Item7",
"Item8"
]]
```

2. **Standardize time zones using a mapping function.**

Code

```
# Mapping of locations to time zones
time_zone_map = {
    "America/New_York": "EST",
    "America/Detroit": "EST",
    "America/Indiana/Indianapolis": "EST",
    "America/Kentucky/Louisville": "EST",
    "America/Indiana/Vincennes": "EST",
    "America/Indiana/Tell_City": "EST",
    "America/Indiana/Petersburg": "EST",
    "America/Indiana/Knox": "EST",
    "America/Indiana/Winamac": "EST",
    "America/Indiana/Marengo": "EST",
    "America/Toronto": "EST",
    "America/Chicago": "CST",
    "America/Menominee": "CST",
    "America/North_Dakota/New_Salem": "CST",
    "America/Denver": "MST",
    "America/Phoenix": "MST",
    "America/Boise": "MST",
    "America/Los_Angeles": "PST",
    "America/Anchorage": "AKST",
    "America/Nome": "AKST",
    "America/Sitka": "AKST",
    "America/Juneau": "AKST",
    "Pacific/Honolulu": "HAST",
    "America/Puerto_Rico": "AST",
    "America/Ojinaga": "MST",
}

# Replace the TimeZone column with the mapped values
df["TimeZone"] = df["TimeZone"].map(time_zone_map)
```

3. **Convert columns intended to represent boolean values to booleans with “Yes” being mapped to True and “No” being mapped to False.**

Code

```
# Convert boolean columns to actual boolean types
boolean_columns = [
    "Churn",
    "Techie",
    "Port_modem",
    "Tablet",
    "Phone",
    "Multiple",
    "OnlineSecurity",
    "OnlineBackup",
    "DeviceProtection",
    "TechSupport",
    "StreamingTV",
    "StreamingMovies",
    "PaperlessBilling",
]
for column in boolean_columns:
    df[column] = df[column].map({"Yes": True, "No": False})
```

4. **Convert nominal variables that include repeating values to categories.**

Code

```
# Convert remaining categorical data to category dtype
nominal_columns = [
    "Area",
    "TimeZone",
    "Marital",
    "Gender",
    "Contract",
    "InternetService",
    "PaymentMethod",
]
for column in nominal_columns:
    df[column] = df[column].astype("category")
```

5. **Use one-hot encoding to create dummy columns, with the first category of each categorical variable being dropped to avoid the dummy variable trap.**

Code

```
# Get dummy columns
df = pd.get_dummies(df, columns = nominal_columns, drop_first = True)
```

C4. Provide a copy of the cleaned data set.

See churn_encoded.csv.

D1. Split the data into training and test data sets and provide the file(s).

To split the data into training and test data sets, we will use the `train_test_split` function found within the `sklearn.model_selection` library. This step is crucial in order to avoid overfitting, which may cause the training and testing errors to be vastly different.

Next, each of the four data sets is saved as a csv file for evaluation.

Code

```
df_X = df.drop(["Churn"], axis=1).copy()
df_y = df["Churn"].copy()

X = df_X.assign(const=1)
y = df_y

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =
1111)

# Save the training and testing datasets to CSV files for reference
X_train.to_csv("churn_Xtrain.csv", index = False)
X_test.to_csv("churn_Xtest.csv", index = False)
y_train.to_csv("churn_ytrain.csv", index = False)
y_test.to_csv("churn_ytest.csv", index = False)
```

See attached churn_Xtrain.csv, churn_Xtest.csv, churn_ytrain.csv, and churn_ytest.csv.

D2. Describe the analysis technique you used to appropriately analyze the data. Include screenshots of the intermediate calculations you performed.

To analyze the data, I will be using the sci-kit learn APIs GridSearchCV and KNeighborsClassifier.

Firstly, in order to perform a K-Nearest Neighbors classification, one must determine an ideal value for k.

Code

```
# Determine the ideal value for k
parameters = {"n_neighbors": range(1, 50)}
gridsearch = GridSearchCV(KNeighborsRegressor(), parameters)
gridsearch.fit(X_train, y_train)

print(gridsearch.best_params_)
print(gridsearch.best_score_)
```

Result

```
{"n_neighbors": 28}
0.1687078480616308
```

The code shown above uses GridSearchCV to try all integer values of k from 1 to 49 as possible candidates for the best parameter.

GridSearchCV will perform a thorough cross-validated grid-search over the specified parameter range. This means it will train a KNeighborsRegressor for each value of k from 1 to 49 and assess its performance using cross-validation, thus evaluating the model multiple times across different splits of the data.

GridSearchCV will fit the KNeighborsRegressor model multiple times to different subsets of the data according to the cross-validation strategy (which defaults to 5-fold cross-validation if not otherwise specified).

The code above outputs the following:

- **gridsearch.best_params_**: This attribute of GridsearchCV will display the parameter (k value) that resulted in the best average performance over the cross-validation subsets.
- **gridsearch.best_score_**: This shows the best average score achieved with the best_params_. This score is the average of the scores obtained on the validation sets during the cross-validation. The default scoring metric for KNeighborsRegressor is the R^2 score, which measures the proportion of variance in the dependent variable that is predictable from the independent variables.

The k-value of 28 resulted in the best average performance over the cross-validation subsets. Next, I performed KNN using a value of k = 28.

Code

```
knn = KNeighborsClassifier(n_neighbors = 28)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
```

The next step of my data analysis involves creating a confusion matrix to identify the false positives and false negative values created by the KNN model. Additionally, I will print out the percentage of these values.

Code

```
# Create a confusion matrix to compare the test (actual) values to the predicted values
cm = confusion_matrix(y_test, y_pred)

# Create a DataFrame from the matrix for readability
df_cm = pd.DataFrame(cm,
                     index = [label for label in ["Predicted No Churn", "Predicted
Churn"]],
                     columns = [label for label in ["Actual No Churn", "Actual Churn"]])

print(df_cm)

# Extract FP and FN
total = cm.sum()
fp = cm[0][1]
fp_percent = fp / total * 100
fn = cm[1][0]
fn_percent = fn / total * 100

print("False Positives (FP):", fp, f"({fp_percent}%)")
```

```
print("False Negatives (FN):", fn, f"({fn_percent}%)")
```

Result

```
                    Actual No Churn  Actual Churn
Predicted No Churn             1346           127
Predicted Churn                 408           119
False Positives (FP): 127 (6.35%)
False Negatives (FN): 408 (20.4%)
```

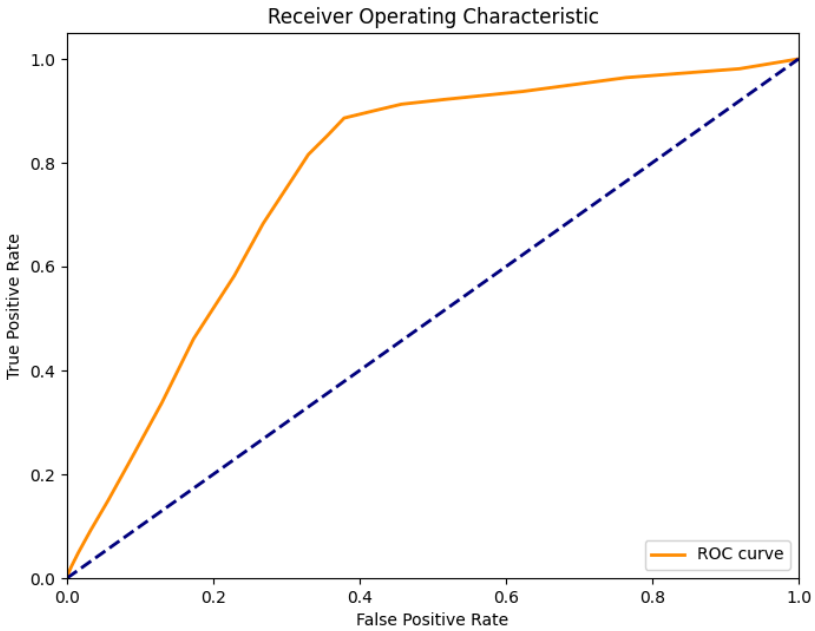
Lastly, I will plot the ROC curve, calculate the area under the curve, and generate a classification report.

Code

```
# Plot the ROC curve
y_pred_prob = knn.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color="darkorange", lw=2, label="ROC curve")
plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic")
plt.legend(loc="lower right")
plt.show()

# Calculate the area under the curve
auc = roc_auc_score(y_test, y_pred_prob)
print(f"Area under curve: {auc}")

# Generate a classification report
print(classification_report(y_test, y_pred))
```

Result

```
Area under curve: 0.7704867243527067
              precision    recall  f1-score   support

       False       0.77      0.91      0.83      1473
        True       0.48      0.23      0.31       527

    accuracy                           0.73      2000
   macro avg       0.63      0.57      0.57      2000
weighted avg       0.69      0.73      0.70      2000
```



D3. Provide the code used to perform the classification analysis from part D2.

See main.py.

E1. Explain the accuracy and the area under the curve (AUC) of your classification model.

The accuracy of my KNN model is 0.73 which means that 73% of all the predictions made by my model are correct. Accuracy is a straightforward measure of overall performance in correctly identifying both classes.

The area under the curve of my KNN model is 0.77. The value of the area under the curve ranges from 0 to 1, where a value of 0.5 suggests no discriminative ability (i.e. equivalent to random chance) and a value of 1 indicates a perfect discrimination. A value of 0 would indicate a perfectly inaccurate model.

Therefore, the AUC of 0.77 indicates a good ability to predict a customer's churn decision, although it's not excellent.

Accuracy can be misleading in the presence of imbalanced classes. For example, in a dataset where 90% of the data are from the negative class, a model can achieve 90% accuracy by merely predicting negative every time, which doesn't truly reflect the model's predictive power. AUC is not affected by the class imbalance as it measures the model's ability to rank predictions correctly rather than its ability to achieve high accuracy. It assesses how well the model can differentiate between classes, regardless of the actual decision threshold.

E2. Discuss the results and implications of your classification analysis.

This KNN model demonstrates a decent level of overall accuracy and a good area under the curve (AUC), however, investigating its precision, recall, and f1-score for True Class reveals potential issues:

- The precision for the True Class is 0.48, indicating that less than half (0.5) of the instances predicted as True are actually True. This low precision could be problematic if the cost of false positives is high. As my research paper aims to predict if a customer will churn, the cost of false positives involves predicting that a

customer will churn when in fact, they will not, which is not costly. However, an organization may offer discounts to customers it predicts will soon churn to increase tenure, and therefore false positives could cut into its bottom line.

- The recall for the True Class is 0.23, meaning the model identifies only 23% of all actual True cases. This suggests a high number of False negatives. This is more costly for an organization as the failure to predict a customer's decision to churn could lead to lost revenue.
- The score of F-1 for the True Class further confirms that the model is not very effective at predicting the True Class.

E3. Discuss one limitation of your data analysis.

The most significant limitation of my data analysis is the imbalance in class distribution: 1473 False values and 527 True values. Imbalanced data can lead the model to develop a bias toward the majority class, in this case, False, leading to poorer performance on the minority class, as reflected in the low recall and F1 score for the True class.

E4. Recommend a course of action for the real-world organizational situation from part A1 based on your results and implications discussed in part E2.

Based on the results, implications, and limitations previously discussed, a real world organization should focus on resampling the dataset to address the class imbalance. Doing this can address the data imbalance issues described in parts E2 and E3, allowing for an increase in the precision, recall, and F-1 score of the True Class.

Once the data has been resampled, more effective data analytics can be performed on the dataset to yield insights used to make business decisions.

G. Web Sources

"Hyperparameter tuning using GridSearchCV and KerasClassifier", GeeksforGeeks.org – https://www.geeksforgeeks.org/hyperparameter-tuning-using-gridsearchcv-and-kerasclassifier/

"K Nearest Neighbors with Python | ML", GeeksforGeeks.org – https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/

sklearn SelectKBest Documentation – https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

sklearn f_classif Documentation – https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif

KNeighborsClassifier – https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

GridSearchCV Documentation – https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Scikit Nearest Neighbors Guide – https://scikit-learn.org/stable/modules/neighbors.html#classification

Scikit Tuning Hyperparameters Guide – https://scikit-learn.org/stable/modules/grid_search.html#grid-search

confusion_matrix Documentation –

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

roc_auc_score Documentation –
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

roc_curve Documentation –
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html

classification_repot Documentation –
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

Model Validation in Python, DataCamp –
https://campus.datacamp.com/courses/model-validation-in-python/validation-basics?ex=1

The k-Nearest Neighbors (kNN) Algorithm in Python –
https://realpython.com/knn-python/

H. Works Consulted

Classification: K Nearest neighbours. (n.d.). http://www.cs.ucc.ie/~dgb/courses/tai/notes/handout4.pdf