A1:PROPOSAL OF QUESTION

Research Question: *"Which specific features most strongly indicate whether a customer is likely to churn?"*

In the context of a telecommunications company, customer churn represents a significant issue, as retaining existing customers is generally more cost-effective than acquiring new ones. Using the k-Nearest Neighbor (KNN) classification method, the goal is to analyze various customer features—such as monthly charges, bandwidth usage, and customer interaction history—to determine which factors are most indicative of a customer's likelihood to churn. This analysis is critical for developing targeted retention strategies. By identifying key attributes that contribute to churn, the company can focus on those areas to reduce churn rates, optimize customer satisfaction, and ultimately improve profitability. Using KNN will allow us to classify customers based on their similarity to other customers who have churned, providing actionable insights for proactive customer management.

A2:DEFINED GOAL

The primary goal of this analysis is to develop a predictive model that accurately determines the likelihood of a customer churning based on the features available in the dataset. This goal is both reasonable and achievable within the scope of the scenario, as the dataset contains a wide range of predictor variables such as customer demographics, service usage patterns, and billing information. By analyzing these variables, we aim to identify key factors that contribute to customer churn and use this information to build a robust k-Nearest Neighbor (KNN) model. The model should be capable of making reliable predictions on unseen data, enabling the organization to proactively address potential churn risks. Additionally, given the potential imbalance between churned and non-churned customers in the dataset, a secondary goal is to ensure that the model is unbiased. This may involve techniques such as oversampling, undersampling, or the use of synthetic data generation methods to balance the dataset and improve the model's generalizability. The ultimate objective is to create a deployable model that provides actionable insights, allowing the organization to implement targeted interventions that reduce churn rates and enhance customer retention efforts.

B1:EXPLANATION OF CLASSIFICATION METHOD

For this analysis, I have chosen to use the k-Nearest Neighbor (KNN) classification method. KNN is particularly effective in classification tasks due to its simplicity and interpretability. It operates on the principle that similar data points exist in close proximity to each other in the feature space, and it classifies new data points based on the majority label of their nearest neighbors. KNN works by storing all instances of the dataset and, when presented with a new data point, calculates the distance between this new point and all other points in the dataset. The algorithm then selects the 'k' nearest data points (where 'k' is a user-defined number) and assigns the new data point to the class most common among its nearest neighbors. This process effectively makes KNN a "voting system" where each neighbor casts a vote, and the class with the most votes determines the classification of the new data point. In this telecom churn analysis, the KNN model will analyze features such as monthly charges, service usage, customer interaction, and other relevant variables to determine the similarity between customers who have churned and those who have not. By examining the distances between data points in this multi-dimensional space, the model will classify new customers as likely to churn or not based on their proximity to existing churned or non-churned customers.

Expected Outcomes:

The expected outcome is that the KNN model will accurately classify customers into churned or non-churned categories. For instance, if a new customer exhibits similar patterns (e.g., high monthly charges and low service usage) as those customers who have previously churned, the KNN model will likely predict that this new customer is at risk of churning. The model's effectiveness will depend on the selection of the appropriate 'k' value, as too low a value might lead to noise affecting the prediction, while too high a value might dilute the influence of closer neighbors. Additionally, the dataset’s dimensionality and any existing outliers will need to be carefully managed to ensure that the model performs optimally without being skewed by irrelevant or anomalous data points.

Rationale for Choosing KNN:

KNN is an appropriate choice for this task due to its ability to handle multi-class classification problems and its simplicity, which makes it easy to interpret and implement. Its lack of a training phase allows for quick adaptability to new data, making it well-suited for situations where the dataset is constantly evolving. However, considerations must be made for its computational intensity and sensitivity to outliers, which may require pre-processing steps such as normalization or dimensionality reduction to enhance performance. KNN is chosen for its practicality in classifying customer churn, where its strengths in handling classification problems with clear, interpretable outcomes make it a valuable tool in understanding and predicting customer behavior.

B2:SUMMARY OF METHOD ASSUMPTION

One fundamental assumption of the k-Nearest Neighbor (KNN) classification method is the local smoothness assumption. This assumption posits that data points that are close to each other in the feature space are more likely to belong to the same class. In other words, if two customers in our dataset exhibit similar characteristics—such as similar monthly charges, service usage patterns, or customer interaction histories—they are assumed to have a similar likelihood of churning.

KNN relies on distance measures like Euclidean distance, Manhattan distance, or cosine similarity to quantify how close data points are to each other in the multi-dimensional feature space. The underlying assumption is that the closer two points are, the more related they are in terms of their class labels. For instance, in the context of customer churn, if a new customer has very similar attributes to several customers who have previously churned, KNN assumes that this new customer is also likely to churn. This assumption is critical to the functioning of KNN, as the accuracy of the model's predictions depends on the idea that proximity in the feature space correlates with similarity in class labels. To validate this assumption, one might use dimensionality reduction techniques like Principal Component Analysis (PCA) or visualization tools such as scatter plots to examine how data points cluster in the feature space. If the data exhibits clear clusters, this would support the local smoothness assumption and indicate that KNN is a suitable method for the classification task. The local smoothness assumption in KNN is that proximity implies similarity in class labels, making it essential for the model to correctly classify new data points based on the distance from existing, labeled data points.

B3:PACKAGES OR LIBRARIES LIST

For this analysis, I have selected several Python packages, each of which plays a crucial role in supporting different aspects of the data analysis and model development process. Here's a list of the chosen packages along with the justification for their use:

1. **NumPy and Pandas: Data Analysis and Preparation**
   - **NumPy**: This package provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It is essential for performing efficient numerical computations, which are foundational for data manipulation tasks such as calculating distances between data points in KNN.
   - **Pandas**: Pandas is the go-to library for data manipulation and analysis in Python. It offers powerful data structures like DataFrames, which make it easy to clean, transform, and explore data. For this analysis, Pandas is used to manage the dataset, handle missing values, and prepare the data for modeling by selecting relevant features and splitting the data into training and testing sets.
2. **Matplotlib and Seaborn: Data Visualization**
   - **Matplotlib**: Matplotlib is a widely-used plotting library that enables the creation of static, interactive, and animated visualizations in Python. It is particularly useful for generating basic plots like histograms, scatter plots, and line charts, which help in understanding the distribution and relationships of variables in the dataset.
   - **Seaborn**: Built on top of Matplotlib, Seaborn provides a high-level interface for drawing attractive and informative statistical graphics. It simplifies the process of creating complex visualizations like heatmaps, pair plots, and box plots, which are invaluable for identifying patterns, trends, and potential outliers in the data. These visualizations support the exploration of relationships between features and the target variable (churn) before applying the KNN model.
3. **Scikit-learn: Machine Learning and Predictive Modeling**
   - **Scikit-learn**: This is a robust and versatile machine learning library in Python, which offers a wide range of tools for model development, including classification, regression, clustering, and more. For this analysis, Scikit-learn provides the implementation of the K-Nearest Neighbor (KNN) algorithm. It also includes utilities for data preprocessing (e.g., normalization, splitting datasets), model

evaluation (e.g., cross-validation, confusion matrices), and hyperparameter tuning, all of which are critical for building and refining the predictive model.

These libraries are chosen because they complement each other in covering all the necessary steps for a comprehensive data analysis and model-building process. NumPy and Pandas facilitate efficient data handling and preparation, Matplotlib and Seaborn enable insightful data visualization, and Scikit-learn provides the tools for implementing and evaluating the KNN model. Together, they create a seamless workflow from data exploration to the final predictive analysis, making Python a powerful environment for this churn analysis task.

C1:DATA PREPROCESSING

In the context of preparing data for the k-Nearest Neighbor (KNN) algorithm, one critical preprocessing step involves handling categorical variables by transforming them into a numerical format that the model can interpret. KNN relies on calculating distances between data points in a multi-dimensional feature space, which requires all features to be numeric. As categorical variables represent discrete categories rather than continuous values, they must be converted into a numerical representation to be used effectively in the model. In this analysis, the `internet_service` column was originally a categorical variable with multiple distinct categories (DSL, Fiber Optic, None). Instead of using one-hot encoding, I applied a simplified approach by transforming the column into a binary variable. Both `Fiber Optic` and `DSL` were assigned a value of `1`, indicating the customer uses an internet service, while `None(N/A)` was assigned a value of `0`, indicating no internet service. This transformation consolidates the categories into a single binary column, which reduces the complexity of the model and allows KNN to treat the presence or absence of internet service uniformly across both types. This step is crucial because KNN uses distance measures (such as Euclidean distance) to determine the similarity between data points. Without converting categorical data into a numerical format, the model would not be able to calculate distances correctly, leading to errors in classification. By transforming the `internet_service` variable into a binary format, we simplify the data while retaining its core meaning, ensuring the KNN algorithm can accurately interpret it alongside other features such as `monthly_charge` or `tech_support`.

C2:DATA SET VARIABLES

Independent variables:

1. **techie**: categorical (1 for Yes, 0 for No)
2. **port_modem**: categorical (1 for Yes, 0 for No)
3. **tablet**: categorical (1 for Yes, 0 for No)
4. **phone**: categorical (1 for Yes, 0 for No)
5. **multiple_lines**: categorical (1 for Yes, 0 for No)
6. **online_security**: categorical (1 for Yes, 0 for No)
7. **online_backup**: categorical (1 for Yes, 0 for No)
8. **device_protection**: categorical (1 for Yes, 0 for No)

9. **tech_support**: categorical (1 for Yes, 0 for No)
10. **streaming_tv**: categorical (1 for Yes, 0 for No)
11. **streaming_movies**: categorical (1 for Yes, 0 for No)
12. **paperless_billing**: categorical (1 for Yes, 0 for No)
13. **internet_service**: categorical (1 for Fiber Optic/DSL, 0 for other)

Dependant Variable

1. **Churn:** categorical (1 for Yes, 0 for No)

While these variables were categorical (representing choices like Yes/No or types of services), they have been encoded as binary numeric values to make them compatible with the K-Nearest Neighbor (KNN) algorithm. KNN operates based on numerical data, so this transformation allows the algorithm to compute distances and classify customers effectively.

C3:STEPS FOR ANALYSIS

The following outlines the steps taken to clean and prepare the data for the KNN classification analysis, including the relevant code segments used for each step:

1. Loading the Data

Code Segment:

```
# import data from csv

df = pd.read_csv('/content/drive/MyDrive/D209/churn_clean.csv',
index_col=0)

df.head()
```

Explanation: The dataset is imported from a CSV file using the Pandas library. This is the first step to read the data into a DataFrame for further manipulation and analysis.

2. Handling Missing Values

Code Segment:

```
# Fixing 'None' values in the 'InternetService' column

for col in ['InternetService']:

    df[col].fillna('N/A', inplace=True)

    print(df[col].value_counts())
```

Explanation: While there were no explicit null values in the `internet_service` column, the value "None" is misinterpreted as a NULL value. To handle this, the code replaces "None" with "N/A". This ensures that the missing value is handled appropriately during analysis and modeling.

3. Renaming Columns

Code Segment:

```
# Rename columns to be more Python-friendly

col_head = {

    'Customer_id': 'customer_id',

    'Interaction': 'interaction',

    'UID': 'uid',

    'City': 'city',

    'State': 'state',

    'County': 'county',

    'Zip': 'zip',

    'Lat': 'lat',

    'Lng': 'lng',

    'Population': 'population',

    'Area': 'area',

    'TimeZone': 'timezone',

    'Job': 'job',

    'Children': 'children',

    'Age': 'age',

    'Education': 'education',
    'Employment': 'employment_type',

    'Income': 'income',

    'Marital': 'marital_type',

    'Gender': 'gender',

    'Churn': 'churn',

    'Outage_sec_perweek': 'outage_sec_perweek',

    'Email': 'email',

    'Contacts': 'contacts',

    'Yearly_equip_failure': 'yearly_equip_failure',

    'Techie': 'techie',

    'Contract': 'contract',

    'Port_modem': 'port_modem',

    'Tablet': 'tablet',

    'InternetService': 'internet_service',

    'Phone': 'phone',

    'Multiple': 'multiple_lines',

    'OnlineSecurity': 'online_security',

    'OnlineBackup': 'online_backup',

    'DeviceProtection': 'device_protection',

    'TechSupport': 'tech_support',

    'StreamingTV': 'streaming_tv',

    'StreamingMovies': 'streaming_movies',

    'PaperlessBilling': 'paperless_billing',
    'PaymentMethod': 'payment_method',

    'Tenure': 'tenure',

    'MonthlyCharge': 'monthly_charge',

    'Bandwidth_GB_Year': 'bandwidth_gb_year',

    'Item1': 'response_timeliness',

    'Item2': 'fix_timeliness',

    'Item3': 'replacement_timeliness',

    'Item4': 'service_reliability',

    'Item5': 'service_options',

    'Item6': 'respectful_communication',

    'Item7': 'courteous_interaction',

    'Item8': 'active_listening_skill'

}

df.rename(columns=col_head, inplace=True)
```

Explanation: This step standardizes column names by converting them to lowercase and more Python-friendly formats. This simplifies the process of referencing columns during data cleaning and analysis.

4. Dropping Irrelevant Columns

Code Segment:

```
# Dropping columns not relevant to the analysis

df = df.drop(columns=['customer_id', 'interaction', 'uid', 'city',
'state', 'county', 'timezone',

                      'job', 'zip', 'marital_type', 'lat', 'lng',
'population', 'area', 'children',
                      'age', 'income', 'gender', 'outage_sec_perweek',
'email', 'contacts',

                      'yearly_equip_failure', 'contract',
'payment_method', 'tenure',

                      'monthly_charge', 'bandwidth_gb_year',
'response_timeliness',

                      'fix_timeliness', 'replacement_timeliness',
'service_reliability',

                      'service_options', 'respectful_communication',
'courteous_interaction',

                      'active_listening_skill'])
```

Explanation: Unnecessary or irrelevant columns that do not contribute to the prediction of churn are dropped from the dataset. This step reduces the dimensionality of the data, making the analysis more focused and improving model performance by eliminating noise.

5. Converting Yes/No Columns to Binary Values

Code Segment:

```
# Converting yes/no columns to 1/0

yes_no_col = ['churn', 'techie', 'port_modem', 'tablet', 'phone',
'multiple_lines',

              'online_security', 'online_backup', 'device_protection',
'tech_support',

              'streaming_tv', 'streaming_movies', 'paperless_billing']

df[yes_no_col] = df[yes_no_col].replace({'Yes': 1, 'No': 0})
```

Explanation: All Yes/No categorical columns, such as `churn`, `techie`, and others, are converted into binary values (1 for Yes, 0 for No). This transformation is necessary because KNN requires numeric input for calculating distances, and converting categorical data into numeric format ensures the algorithm can process it.

6. Handling the `internet_service` Column:

Code Segment:

```
# Convert internet_service column into 1/0

df['internet_service'] = df['internet_service'].replace({'Fiber
Optic': 1, 'DSL': 1, 'N/A': 0})
```

Explanation: The `internet_service` column originally had multiple categories (`Fiber Optic`, `DSL`, `None`), which have now been consolidated into a single binary variable. Both `Fiber Optic` and `DSL` are represented as `1`, and `N/A` (formerly “None”) is represented as `0`. This simplifies the data and reduces the dimensionality of the feature space.

Each step in the data cleaning and preprocessing process is essential for preparing the dataset for KNN classification. Handling missing values, transforming categorical variables, and splitting the data into training and test sets are critical for ensuring that the model can learn from the data effectively and make accurate predictions.

C4:CLEANED DATA SET

See Attached.

D1:SPLITTING THE DATA

See 4 attached files.

D2:OUTPUT AND INTERMEDIATE CALCULATIONS

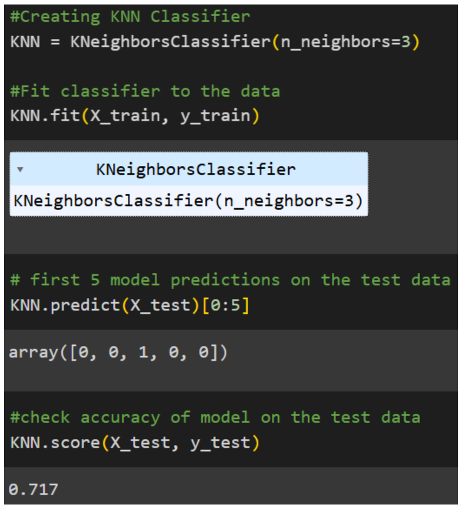
K-Nearest Neighbor (KNN), a simple yet powerful machine learning algorithm commonly used for classification tasks. The KNN algorithm classifies a new data point based on its proximity to other data points in the dataset. Here’s an overview of how KNN was applied to analyze the customer churn data:

1. Basic Concept of KNN:
    - KNN works on the assumption that data points that are close to each other in the feature space are likely to belong to the same class. In this case, the classes are whether a customer churned (`churn = 1`) or did not churn (`churn = 0`).
    - KNN does not make any assumptions about the underlying data distribution, making it a non-parametric method. This flexibility is useful when working with complex datasets like customer churn, where features may not follow a normal distribution.
2. Distance Calculation:
    - The KNN algorithm calculates the distance between the data points based on the features selected. In this analysis, features such as `techie`, `online_security`, `phone`, and others were used.

- The most commonly used distance measure in KNN is Euclidean distance, which calculates the straight-line distance between two points in multi-dimensional space.

3. Choosing the Value of 'k':
    - The value of 'k' (the number of neighbors considered) plays a crucial role in the performance of the KNN model. If k is too small, the model may become overly sensitive to noise in the data (i.e., outliers). If k is too large, the model may smooth out important patterns, leading to reduced accuracy.
    - In this analysis, the optimal value of 'k' was determined using GridSearchCV, which tested different values of 'k' and selected the one that provided the best performance on the validation data.
4. Training and Predictions:
    - The KNN model was trained using the training data (80% of the dataset). For each test data point, the algorithm finds the 'k' nearest points in the training set and assigns the class based on a majority vote among these neighbors.
    - If the majority of the neighbors are customers who churned, the model predicts that the test data point (customer) will churn, and vice versa.
5. Model Evaluation:
    - The KNN model's performance was evaluated using several metrics, including accuracy, confusion matrix, classification report, and ROC-AUC score.
    - These metrics help assess the model's ability to correctly classify churned and non-churned customers. The ROC-AUC score was particularly useful as it measures the model's ability to distinguish between the two classes, providing a balanced view of model performance beyond simple accuracy.
6. Advantages of KNN for This Analysis:
    - Interpretability: KNN is easy to understand and interpret, especially in business contexts like customer churn, where decision-makers need to understand the reasons behind the model's predictions.
    - Flexibility: Since KNN is non-parametric, it can handle a wide variety of data types and distributions, making it suitable for the diverse range of customer attributes in the dataset.

KNN was chosen as the analysis technique due to its effectiveness in classifying customers based on their similarity to other customers in the dataset. By leveraging distance-based classification, KNN provides a straightforward yet powerful method for predicting whether a customer is likely to churn, aiding in proactive retention strategies.



```
#Creating KNN Classifier
KNN = KNeighborsClassifier(n_neighbors=3)

#Fit classifier to the data
KNN.fit(X_train, y_train)
```

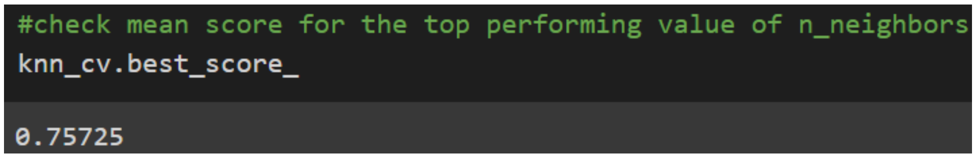
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

```
# first 5 model predictions on the test data
KNN.predict(X_test)[0:5]
```

```
array([0, 0, 1, 0, 0])
```

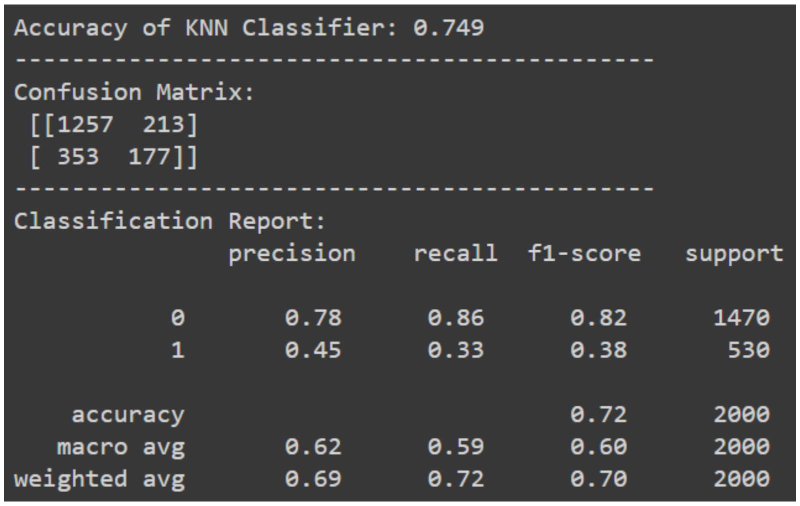
```
#check accuracy of model on the test data
KNN.score(X_test, y_test)
```

```
0.717
```



```
#check mean score for the top performing value of n_neighbors
knn_cv.best_score_
```

```
0.75725
```



```
Accuracy of KNN Classifier: 0.749
--------------------------------------------
Confusion Matrix:
 [[1257  213]
 [ 353  177]]
--------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.86      0.82      1470
           1       0.45      0.33      0.38       530

    accuracy                           0.72      2000
   macro avg       0.62      0.59      0.60      2000
weighted avg       0.69      0.72      0.70      2000
```

D3:CODE EXECUTION

```
# Splitting data into train and test data (D1)
X = df[['techie', 'port_modem', 'tablet', 'phone', 'multiple_lines',
    'online_security', 'online_backup', 'device_protection', 'tech_support',
    'streaming_tv', 'streaming_movies', 'paperless_billing',
    'internet_service']].assign(const=1)

y = df['churn']

# Split the dataset into training and testing sets (80% train, 20% test)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
#Creating KNN Classifier
KNN = KNeighborsClassifier(n_neighbors=3)

#Fit classifier to the data
KNN.fit(X_train, y_train)
# first 5 model predictions on the test data
KNN.predict(X_test)[0:5]
#check accuracy of model on the test data
KNN.score(X_test, y_test)
from sklearn.model_selection import GridSearchCV

# Determine what is the best number of neighbors to use
param_grid = {'n_neighbors' : np.arange(1, 25)}

# Instantiate the KNeighborsClassifier object
knn = KNeighborsClassifier()

# Instantiate the GridSearchCV object, searching across the provided parameter grid and 5 fold
cross validation
knn_cv = GridSearchCV(knn, param_grid, cv=5)

# Fit to training data
knn_cv.fit(X_train, y_train)

# Find best parameter from GridSearchCV
knn_cv.best_params_
#check mean score for the top performing value of n_neighbors
knn_cv.best_score_
y_pred = KNN.predict(X_test)
# Create a KNN classifier with k=20
knn = KNeighborsClassifier(n_neighbors=20)

# Train the classifier on the training set
knn.fit(X_train, y_train)

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


# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy of KNN Classifier: {accuracy}")
print('-' * 45)
```

```
#confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix: \n', conf_matrix)
print('-' * 45)

#classification report
print('Classification Report:\n', classification_report(y_test, y_pred))
```

E1:ACCURACY AND AUC

1. Accuracy: The accuracy of the KNN classifier is reported as 0.749, meaning that the model correctly predicted the class (churned or not churned) for 74.9% of the customers in the test set. Accuracy measures the proportion of total correct predictions (both true positives and true negatives) out of all predictions made. However, while accuracy is an important metric, it doesn't always give a complete picture—especially in imbalanced datasets. In this case, where the churned class (1) is much smaller than the non-churned class (0), the accuracy can be misleading. The confusion matrix provides more insight by showing that:

- The model correctly classified 1257 non-churned customers and 177 churned customers.
- It incorrectly classified 213 non-churned customers as churned (false positives) and 353 churned customers as non-churned (false negatives).

The relatively lower precision and recall for the churn class (1) reflect that the model struggles with correctly identifying churned customers, which can be a key concern for the business.

2. Area Under the Curve (AUC): The AUC (Area Under the ROC Curve) is 0.725, which means the model has a 72.5% chance of correctly distinguishing between a randomly chosen churned and a non-churned customer. The AUC value represents the model's ability to differentiate between the positive class (churned customers) and the negative class (non-churned customers). An AUC of 0.5 would indicate random guessing, while an AUC of 1 indicates perfect classification. An AUC of 0.725 indicates that the model has a reasonable ability to separate churned customers from non-churned ones, but it still leaves room for improvement, especially when focusing on reducing the number of false negatives (customers predicted as non-churned but who actually churn).

E2:RESULTS AND IMPLICATIONS

The KNN classification model yielded an accuracy of 74.9%, a confusion matrix with relatively higher false negatives, and an AUC score of 0.725, indicating moderate classification performance. While these results provide a reasonable starting point, they also highlight several important implications for the business, especially in the context of customer churn.

1. Accuracy and Class Imbalance:

The model’s accuracy of 75.2% seems promising on the surface, but a deeper look into the confusion matrix reveals some concerns:

- The model performed well in identifying non-churned customers (correctly classifying 1,255 out of 1,470), but it struggled with correctly predicting churned customers. It only correctly classified 177 churned customers out of 530.
- A high number of 353 false negatives (churned customers predicted as non-churned) is particularly concerning. These are the customers the business risks losing, as they are mistakenly classified as not at risk of churn.

Given that the dataset is imbalanced, with more non-churned customers than churned customers, the accuracy metric might not fully capture the model’s performance on the minority class (churned customers). The lower recall for the churn class indicates that the model is missing a significant portion of customers who will churn, which is critical for retention strategies.

2. AUC Score and Model Discrimination:

The AUC score of 0.725 suggests that the model has a moderate ability to distinguish between churned and non-churned customers. This means that, in terms of ranking customers by churn risk, the model performs better than random guessing, but there is room for improvement. An AUC closer to 1 would imply stronger discriminatory power, which is especially important for identifying the customers who are most at risk of churn.

3. Implications for the Business:

- Missed Opportunities for Retention: The high false negative rate (353 customers who churned but were predicted not to) means that the company could miss out on opportunities to retain customers at risk of leaving. From a business perspective, these are lost customers who might have been saved with targeted retention strategies.
- False Positives: The model also produced 213 false positives (non-churned customers predicted as churned), which could result in misallocated resources. While focusing on at-risk customers is important, targeting customers who are unlikely to churn could lead to inefficient use of marketing or customer service efforts.

4. Potential Model Improvements:

To improve the model’s effectiveness, especially for identifying churned customers, several steps can be taken:

- Handling Imbalanced Data: Implementing techniques such as oversampling the minority class (churned customers) or using SMOTE (Synthetic Minority Over-sampling Technique) could help balance the dataset and improve the model’s ability to correctly classify churned customers.
- Alternative Classification Models: Exploring more sophisticated classification algorithms such as Random Forest, Logistic Regression, or XGBoost may offer better performance

for predicting churn, especially in handling complex decision boundaries and imbalanced datasets.

- Feature Engineering: Introducing new features or improving existing ones (e.g., creating interaction terms or derived metrics from customer behavior data) may enhance the model's predictive power.

5. Actionable Insights:

Despite its limitations, the model still provides useful insights. By identifying key features associated with churn, such as the presence of tech support or online security, the business can focus on improving services and customer experience in those areas to reduce churn rates. Additionally, the business can use the model as part of a broader customer retention strategy by combining it with other predictive analytics and customer behavior data.

E3:LIMITATION

One significant limitation of the data analysis is the class imbalance between churned and non-churned customers. In the dataset, the number of customers who did not churn (majority class) far exceeds the number of those who churned (minority class). This imbalance biases the k-Nearest Neighbor (KNN) model toward predicting the majority class, resulting in a higher number of false negatives, where churned customers are misclassified as non-churned. Consequently, the model struggles to detect customers at risk of churning, which is a key business objective. Metrics such as accuracy, which was 75.2%, can be misleading in this context, as it reflects the model's ability to predict non-churned customers correctly while underperforming in identifying churned customers, with a recall of only 34%. Additionally, while the AUC score of 0.725 indicates moderate performance, it could be improved with more balanced training data. To address this limitation, techniques like oversampling the minority class, undersampling the majority class, or applying synthetic methods such as SMOTE can be employed to create a more balanced dataset. These approaches would enhance the model's ability to accurately identify churned customers, improving the effectiveness of customer churn predictions.

E4:COURSE OF ACTION

Based on the results and implications discussed in part E2, I recommend the organization take a two-pronged approach to address customer churn:

1. Improve Predictive Model Performance:

Given the model's struggles with identifying churned customers (high false negatives), the organization should prioritize improving the classification model by addressing the class imbalance in the data. Techniques like oversampling the churned customers, undersampling the non-churned customers, or using SMOTE (Synthetic Minority Over-sampling Technique) should be applied to create a more balanced dataset. Additionally, exploring more sophisticated classification models, such as Random Forest or XGBoost, could yield better results in identifying at-risk customers. By improving the model's recall for churned customers, the organization will be better equipped to target interventions toward those who are most likely to leave, minimizing churn rates.

2. Implement Targeted Retention Strategies:

Based on the model’s current ability to predict churn, the organization can immediately begin implementing targeted retention strategies for the customers identified as at-risk. Features that were influential in predicting churn, such as customers without tech support or online security, provide actionable insights. The organization should focus on improving customer service in these areas, offering personalized retention strategies, discounts, or service upgrades to reduce churn. By proactively engaging customers who are identified as likely to churn, the organization can improve retention and reduce the costs associated with acquiring new customers.

Enhancing the predictive model and implementing targeted retention strategies based on the model's insights will help the organization effectively address customer churn and improve long-term customer loyalty.