Part 1

A1.
Using K-Nearest Neighbor (KNN) classification technique, what are the significant variables in customer churn that telecommunications can observe to create or change retention policies?

A2.
A goal of the data analysis is to identify customers who will churn and the most significant predictors contributing to the churn. As a result, hopefully, telecommunication companies can evaluate retention policies and make necessary changes.

Part 2

B1.
The KNN classification technique predicts customer churn by analyzing a discrete number, K, which is the closest labeled data points. The model then conducts a majority vote to predict a label. For instance, if the method evaluates at K=8, the model will look for 8 labeled data points, and conducts a majority vote to predict the unlabeled data point. For the selected data set, I anticipate the KNN technique will observe K labeled data points in the customer churn data, identifying the customer churn status. The machine learning model learns from and stores the data for future predictions. Using these labeled data points, the model then predicts the unlabeled data points, which will provide predictions for customer churn.

B2.
One assumption of the KNN method is that similar labeled points imply a similar outcome. KNN determines the label of a data point by considering its K nearest neighbors, which is usually based on the Euclidean distance. The assumption is that if a majority of the K nearest neighbors share a particular label, the model predicts that the new data point will also belong to that label. Meaning, data points that are close to each other in the feature space are expected to display similar behaviors.

B3.
In the process of conducting the analysis, Python was the chosen programming language, and the following packages and libraries were utilized to support various stages of the analysis. The choice of each package and/or library is justified based on its role in enhancing the overall analytical workflow.

- Pandas was used to import and manipulate the dataset. It facilitated the creation of bins for analysis in the data cleaning process, which is a crucial step in understanding the data distribution and determining the appropriate cleaning strategies. During the classification analysis, pandas was used for one-hot encoding, which is necessary in converting categorical variables into a format suitable for analysis.
- NumPy was employed for array creation and mean calculation which contributed to the preparation of data and for analysis.
- Missingno was used to visualize and identify missing data patterns.
- Seaborn was used to generate an informative heatmap to visualize the correlation between missingness in variables.

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University

- Matplotlib.pyplot was used to generate all visualizations such as bar charts and heatmaps. These visualizations displayed the distribution of missing values to make it easier to make informed decisions during the analysis.
- Fancyimpute / KNN and sklearn. impute / SimpleImputer was used to impute missing data for numerical and non-numerical variables.
- Sklearn.preprocessing / StandardScaler was used for standardizing data to ensure consistency in the scale of numerical features.
- Sklearn.model_selection / train_test_split was used for dividing the dataset into training and testing sets to facilitate model evaluation.
- Sklearn.model_selection / RandomizedSearchCV was to identify the optimal number of neighbors in the K-nearest neighbors classification method to enhance the model's predictive capabilities.
- Sklearn.metrics / roc_auc_score, classification_report, confusion_matrix, and accuracy_score metric tools were used to evaluate the performance of the classification model. These libraries provided insights into the AUC score, classification report, confusion matrix, and overall accuracy.
- Sklearn.feature_selection / SelectKBest, f_classif was used for feature selection to identify the most relevant features for enhanced model construction.

In summary, the chosen packages and libraries contributed to comprehensive and effective data cleaning, transformation, analysis, feature selection, model construction, and statistical calculations.

In text citations:
("The classification challenge", n.d.)
("Preprocessing data", n.d.)

Part 3

C1.
It is important to acknowledge that this method uses distance explicitly when making predictions. Since, distance can vary from feature to feature, features with larger scales can have a larger impact on the predictions, potentially skewing the results and introducing bias. For example, the variable MontlyCharge ranges from 79 to 291 while the variable Bandwidth_GB_Year from 155 to 7159. Without normalization, Bandwidth_GB_Year might have a larger influence on the distance calculations. To mitigate this issue and ensure that all features contribute equally to the distance calculations, normalization is necessary. This preprocessing technique will contribute to a more balance influence of each feature on model predictions.

C2.

Variable	Classification
City	Categorical
State	Categorical
County	Categorical
Zip	Numerical
Lat, Lng	Numerical

Population	Numerical
Area	Categorical
TimeZone	Categorical
Job	Categorical
Children	Numerical
Age	Numerical
Income	Numerical
Marital	Categorical
Gender	Categorical
Churn	Categorical
Outage_sec_perweek	Numerical
Email	Numerical
Contacts	Numerical
Yearly_equip_failure	Numerical
Techie	Categorical
Contract	Categorical
Port_modem	Categorical
Tablet	Categorical
InternetService	Categorical
Phone	Categorical
Multiple	Categorical
OnlineSecurity	Categorical
OnlineBackup	Categorical
DeviceProtection	Categorical
TechSupport	Categorical
StreamingTV	Categorical
StreamingMovies	Categorical
PaperlessBilling	Categorical
PaymentMethod	Categorical
Tenure	Numerical
MonthlyCharge	Numerical
Bandwidth_GB_Year	Numerical
Item1	Numerical
Item2	Numerical
Item3	Numerical
Item4	Numerical
Item5	Numerical
Item6	Numerical
Item7	Numerical
Item8	Numerical

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University

C3.
The steps used to prepare the data for analysis involved data cleaning and preprocessing.

Data Cleaning Plan:

Step 1: Exploratory Data Analysis

1. Loaded CSV file into Jupyter notebook using pandas library and converted the CSV file into a Data Frame.
2. Used the methods, ‘describe()’, ‘info()’, and ‘dtypes()’ to observe the summary statistics, data types, and nullity, the data types of the data frame.
3. Identified if there was duplicated information using the ‘duplicated()’ method.
4. Visualized missing data across the entire data frame using missingno and matplotlib libraries, using the ‘bar()’ and ‘show()’ methods.
5. Created a correlation matrix using the ‘corr()’ method, and utilized the correlation matrix to create a heatmap using the seaborn library to understand the correlation between variables missing values.
6. Determined missingness type(s) if applicable.

```
import pandas as pd
import numpy as np
import missingno as msno
churn_raw_data = pd.read_csv ('/Users/jasminemoniquecooper/Downloads/churn_raw_data.csv')
pd.set_option('display.max_columns', None)
churn_raw_data.head(10)

churn_raw_data.dtypes
churn_raw_data.describe()

churn_duplicates = churn_raw_data.duplicated()
churn_raw_data[churn_duplicates]

churn_raw_data.isna().sum()
#overall summary of missing data in the data frame

#visualize missingness
column_order = churn_raw_data.isnull().sum().sort_values().index
msno.bar(churn_raw_data[column_order])
plt.show()

import seaborn as sns
import matplotlib.pyplot as plt

# Generate a correlation matrix
correlation_matrix = churn_raw_data.corr()
```

```
# Set the figure size to make the heatmap larger
plt.figure(figsize=(12, 10))

# Create a heat map using seaborn with customizations
heatmap = sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5,
annot_kws={"size": 8})

# Increase the font size of the color bar (optional)
cbar = heatmap.collections[0].colorbar
cbar.ax.tick_params(labelsize=12)

# Increase the font size of the annotations
for text in heatmap.texts:
   text.set_size(8)

# Display the heat map
plt.show()
```

Step 2: Impute Missing Data & Create Clean Data file

1. Upon completion of investigation, the variables below had missing values:
    a. Children- 7505 (numerical)
    b. Income- 7510 (numerical)
    c. Techie- 7523 (non-numerical)
    d. Age- 7525 (numerical)
    e. Phone- 8974 (non-numerical)
    f. Bandwidth GB Year- 8979 (numerical)
    g. Tech Support- 9009 (non-numerical)
    h. Tenure- 9069 (numerical)
2. By analyzing the summary statistics, I noticed that the population had a minimum value of zero. Since it is highly unlikely population is zero, I replaced all population records with values equal to zero with NaN values utilizing the numpy library and the 'nan' and 'isnan' methods. This allows Python to recognize missing values properly prior to imputation. The missing Population values were imputed using the K-Nearest Neighbor technique.
3. The numerical values appear to be missing at random because the missing values could probably be predicted by other observed values in the data set. For example, missing number of children values could be relative to age and/or marital status (e.g., if a customer is an unmarried young adult, they are likely to be within the age range 18-30, and not list the number of children that they have since it might not be applicable). To treat numerical missing values, I decided to use the K-Nearest Neighbor imputation technique because missing values can likely be predicted based on similar observed variables and values.
4. The non-numerical values also appear to be missing at random because the missing values could also probably be predicted by other observed values in the data set. For example, missing Techie values could also be relative to age (e.g., if a customer is between ages 50-90, it is unlikely that they would be considered a Techie). Likewise, missing values for Phone and Tech Support could likely be related to age due to the fast advancement of technology overtime. To impute non-numerical variables' values, I decided to use the Simple Imputation mode technique, and used frequency to fill in missing data. Furthermore, to validate this imputation technique, I analyzed

the clean data to ensure that it fit the well-considered estimate, and the clean data portrayed that the group that are not techie's lies between ages 50-90.

```
churn_raw_data.Population[churn_raw_data.Population == 0].count()

#determine if missing income values are related to employment

# Select the rows where income is missing
missing_income = churn_raw_data[churn_raw_data['Income'].isnull()]

# Group the missing income data by employment status
grouped = missing_income.groupby(['Employment'])

# Count the number of missing income values for each employment status
missing_count = grouped.size()

# Calculate the percentage of missing income values for each employment status
percentage_missing = missing_count / churn_raw_data.groupby(['Employment']).size() * 100

# Display the results
print(percentage_missing)
#determine if missing income values are related to employment and age

# Select the rows where income is missing
missing_income = churn_raw_data[churn_raw_data['Income'].isnull()]

# Group the missing income data by employment status
grouped = missing_income.groupby(['Employment', pd.cut(missing_income['Age'], bins=[0, 18, 30, 50,
np.inf])])

# Count the number of missing income values for each employment status
missing_count = grouped.size()

# Calculate the percentage of missing income values for each employment status
percentage_missing = missing_count / churn_raw_data.groupby(['Employment',
pd.cut(churn_raw_data['Age'], bins=[0, 18, 30, 50, np.inf])]).size() * 100

# Display the results
print(percentage_missing)

#part time employment between ages 0-18 is missing data the most

import pandas as pd

# Select the rows where the 'Techie' column is null
missing_techie = churn_raw_data[churn_raw_data['Techie'].isnull()]
```

```
# Define age group bins
age_bins = [18, 30, 50, float('inf')]

# Group the missing 'Techie' data by age
grouped = missing_techie.groupby(pd.cut(missing_techie['Age'], bins=age_bins))

# Count the number of missing 'Techie' values for each age group
missing_count = grouped.size()

# Calculate the percentage of missing 'Techie' values for each age group
total_count_by_age_group = churn_raw_data.groupby(pd.cut(churn_raw_data['Age'],
bins=age_bins)).size()
percentage_missing = (missing_count / total_count_by_age_group) * 100

# Display the results
print(percentage_missing)

import pandas as pd

# Select the rows where 'number of children' is missing
missing_children = churn_raw_data[churn_raw_data['Children'].isnull()]

# Define age group bins
age_bins = [18, 30, 50, float('inf')]  # Adjust the bins as needed

# Group the missing 'number of children' data by marital status and age
grouped = missing_children.groupby(['Marital', pd.cut(missing_children['Age'], bins=age_bins)])

# Count the number of missing 'number of children' values for each marital status and age group
missing_count = grouped.size()

# Calculate the percentage of missing 'number of children' values for each marital status and age group
total_count_by_group = churn_raw_data.groupby(['Marital', pd.cut(churn_raw_data['Age'],
bins=age_bins)]).size()
percentage_missing = (missing_count / total_count_by_group) * 100

# Display the results
print(percentage_missing)

import numpy as np
churn_raw_data.loc[churn_raw_data.Population == 0, 'Population'] = np.nan
churn_raw_data.Population[np.isnan(churn_raw_data.Population)]

pip install fancyimpute
from fancyimpute import KNN

columns_to_impute = ['Children', 'Income', 'Age', 'Bandwidth_GB_Year', 'Population', 'Tenure']
```

```
# Create an instance of the KNN imputer
knn_imputer = KNN()

# Get the column indices of the columns to impute
columns_to_impute_indices = [churn_raw_data.columns.get_loc(col) for col in columns_to_impute]

# Perform imputation on the selected columns
imputed_values = knn_imputer.fit_transform(churn_raw_data.iloc[:, columns_to_impute_indices])

# Assign the imputed values back to the original dataset
churn_raw_data.iloc[:, columns_to_impute_indices] = imputed_values

# Verify if the imputations are in the original dataset
print(churn_raw_data.head())

from sklearn.impute import SimpleImputer

column_to_impute_two = ['Techie', 'Phone', 'TechSupport']

# Create the SimpleImputer object with strategy='most_frequent'
imputer = SimpleImputer(strategy='most_frequent')

# Fit and transform the categorical variables using the imputer
churn_raw_data[column_to_impute_two] =
imputer.fit_transform(churn_raw_data[column_to_impute_two])

import pandas as pd
# Define age group bins
age_bins = [18, 30, 50, float('inf')]

# Group the data by age
grouped = churn_raw_data.groupby(pd.cut(churn_raw_data['Age'], bins=age_bins))

# Count the number of 'Techie' values for each age group
tech_count = grouped['Techie'].value_counts().unstack(fill_value=0)

# Calculate the percentage of 'Techie' values for each age group
tech_percentage = (tech_count / tech_count.sum(axis=1).values[:, None]) * 100

# Display the results
print(tech_percentage)

#data clean
import matplotlib.pyplot as plt
column_order = churn_raw_data.isnull().sum().sort_values().index
msno.bar(churn_raw_data[column_order])
```

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University

```
plt.show()

cleaned_data_two = churn_raw_data
new_name_two = 'churn_cleaned_data'
new_cleaned_data_two = cleaned_data_two.copy()
new_cleaned_data_two.name = new_name_two
new_cleaned_data_two.to_csv('new_cleaned_data_two.csv', index=False)
churn_raw_data.to_csv(r'/Users/jasminemoniquecooper/Downloads/new_cleaned_data_two.csv')
```

Step 3: One hot encoding

1. Drop the unnecessary columns from the clean data frame to prevent creating additional binary variables not needed.
2. Convert all categorical variables to numerical variables using one hot encoding. This process involves creating separate binary variables for each category in a categorical variable. The drop_first argument was used to prevent multicollinearity. For example, the Phone variable, where only Phone_Yes is created, and the absence of the phone service is represented by Phone_Yes = 0 (or Phone_No = 1).
3. Separate binary variables created for Area and PaymentMethod were renamed for proper Python syntax and for ease of use in code.
4. Transform the Churn variable by mapping the ‘yes’ or ‘no’ options to numerical variables.

```
# Drop columns not needed for analysis
columns_to_drop = ['Customer_id', 'Interaction', 'CaseOrder']
churn_clean_data_one = churn_clean_data.drop(columns_to_drop, axis=1)

#one hot encoding
columns=['City', 'State', 'County', 'Timezone', 'Job', 'Education','Gender', 'Marital','Employment','Area',
'PaymentMethod', 'Techie', 'Contract', 'Port_modem', 'Tablet',
'InternetService','Phone','Multiple','OnlineSecurity', 'OnlineBackup','DeviceProtection',
'TechSupport','StreamingTV', 'StreamingMovies','PaperlessBilling']
churn_clean_encoded = pd.get_dummies(churn_clean_data_one, columns=columns, drop_first = True)

#Map the "Churn" variable to numeric values

churn_clean_encoded["Churn"] = (churn_clean_encoded["Churn"] == 'Yes').astype(int)

# Rename columns for proper Python syntax
column_name_mapping = {
   "PaymentMethod_Bank Transfer(automatic)": "Bank_Transfer",
   "PaymentMethod_Credit Card (automatic)": "Credit_Card",
   "PaymentMethod_Electronic Check": "Electronic_Check",
   "PaymentMethod_Mailed Check": "Mailed_Check",
   "Area_Suburban": "Suburban",
   "Area_Urban": "Urban",
   "Area_Rural": "Rural"
}
```

```
churn_clean_encoded.rename(columns=column_name_mapping, inplace=True)

churn_clean_encoded.head(10)
```

Step 4: Normalize the data

1. Normalize the data to bring features to a comparable scale. Mathematically, the code will be utilizing standardization, by subtracting the mean and dividing by the standard deviation.
2. Ensure the data set has been normalized and features are on a standardized scale by ensuring variance is approximately equal to one.

```
from sklearn.preprocessing import StandardScaler

#standardize the data
scaler = StandardScaler()
churn_clean_scale = scaler.fit_transform(churn_clean_encoded)

#check standardization to make sure variance is 1

import numpy as np
variance_values = np.var(churn_clean_scale, axis=0)

for i, column in enumerate(churn_clean_encoded.columns):
   print(f"Column: {column}, Variance: {variance_values[i]}")
```

Code:
See code/script attached and below:

Please see Churn Clean Data.ipynb attached for this code.

C4.
Cleaned data set name - add excel and csv files

Please see 'new_cleaned_data_two.csv' file attached to submission

In text citations:
("Completeness", n.d.)
("Importing flat files using pandas", n.d.)
("Is data missing at random?", n.d.)
("Mean, median, & mode imputations", n.d.)
("Imputing using facncyimpute", n.d.)
("Measures of center", n.d.)
("Measures of spread", n.d.)
("GeeksforGeeks," n.d.)
("Centering and scaling", n.d.)
("Preprocessing data", n.d.)
("Area under the ROC curve", n.d.)

Part 4

D1.
Please see the following files attached to the submission: X_train.csv, X_test.csv, y_train.csv, y_test.csv

D2.
The analysis included hyperparameter tuning, model performance metrics, and feature selection. To begin my analysis, I utilized RandomizedSearchCV to identify the optimal value for n_neighbors. This method aims to test a fixed number of hyperparameter settings from specified probability distributions. As a result, the search provides the optimal number of n_neighbors and the corresponding cross validate score, reflecting the model's best performance during the cross-validation process. In my model, the output was n_neighbors equal = 18, an accuracy score of 73.5%, and an AUC of 62%. Since these outputs are relatively low, I conducted feature selection utilizing the optimal n_neighbors = 18 to improve my model.

After hyperparameter tuning and feature selection, I conducted an analysis that encompassed the accuracy score, precision, recall, and F1-score as displayed in the classification report, along with the AUC. Observations from the classification report are as follows:

- For class 0 (customer did not churn):
    - Precision: 91% of the predicted positive instances were true positives.
    - Recall: 92% of the actual positive instances were predicted correctly.
    - F1-score: 92%, which indicates that there is a good balance between precision and recall in the model for class 0. This suggests that there is a relatively low number of false positives and false negatives, which implies that the model has good performance for class 0.
    - Support: 1456 data points that represent the number of actual instances of class 0 in the dataset.
- For class 1 (customer did churn)
    - Precision: 78% of the predicted positive instances were true positives.
    - Recall: 76% of the actual positive instances were predicted correctly.
    - F1-score: 77%, which also indicates that there is a good balance between precision and recall in the model for class 1. In comparison to class 0, the lower F1-score highlights a challenge in correctly identifying instances of churn and could be an area for improvement.
    - Support: 544 data points that represent the number of actual instances of class 1 in the dataset.

Additionally, the model correctly classified 88% of all instances, indicating the model with selected features performs well.

Given the robust classification metrics observed for both classes, I examined the AUC to obtain a comprehensive understanding of the model's overall performance. As a result, the AUC metric, calculated with selected features, yielded a value of 93%. This reinforces the hypothesis that the model, with the selected features, excels in its performance and demonstrates proficiency in accurately classifying potential outcomes.

Also, it appears that the most significant variables are Tenure, MonthlyCharge, Bandwidth_GB_Year, Contract_One year, Contract_Two year, Multiple_yes, StreamingTV_yes, and StreamingMovies_Yes.

Overall, the refined model demonstrates excellent performance for both classes, with a clear indication of significant features contributing to the model's predictive accuracy.

D3.

Code for one hot encoding:

[2]:
```
# Drop columns not needed for analysis
columns_to_drop = ['Customer_id', 'Interaction', 'CaseOrder']
churn_clean_data_one = churn_clean_data.drop(columns_to_drop, axis=1)

#one hot encoding
columns=['City', 'State', 'County', 'Timezone', 'Job', 'Education','Gender', 'Marital','Employment','Area', 'PaymentMethod', '
churn_clean_encoded = pd.get_dummies(churn_clean_data_one, columns=columns, drop_first = True)

#Map the "Churn" variable to numeric values

churn_clean_encoded["Churn"] = (churn_clean_encoded["Churn"] == 'Yes').astype(int)

# Rename columns for proper Python syntax
column_name_mapping = {
    "PaymentMethod_Bank Transfer(automatic)": "Bank_Transfer",
    "PaymentMethod_Credit Card (automatic)": "Credit_Card",
    "PaymentMethod_Electronic Check": "Electronic_Check",
    "PaymentMethod_Mailed Check": "Mailed_Check",
    "Area_Suburban": "Suburban",
    "Area_Urban": "Urban",
    "Area_Rural": "Rural"
}

churn_clean_encoded.rename(columns=column_name_mapping, inplace=True)

churn_clean_encoded.head(10)
```

Code for Standardizing data:

```
from sklearn.preprocessing import StandardScaler

#standardize the data
scaler = StandardScaler()
churn_clean_scale = scaler.fit_transform(churn_clean_encoded)
churn_clean_scale_df = pd.DataFrame(churn_clean_scale, columns=churn_clean_encoded.columns)
```

```
# Standardize the training set
X_train_scaled = scaler.fit_transform(X_train)
# Use the same scaler to standardize the testing set
X_test_scaled = scaler.transform(X_test)
```

Code for validating data includes a check to ensure that the variance of the standardized data is approximately 1:

```
from sklearn.preprocessing import StandardScaler

#standardize the data
scaler = StandardScaler()
churn_clean_scale = scaler.fit_transform(churn_clean_encoded)
churn_clean_scale_df = pd.DataFrame(churn_clean_scale, columns=churn_clean_encoded.columns)
```

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University

Code to identify best k value identified by RandomizedSearchCV() along with the training accuracy score:

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.neighbors import KNeighborsClassifier
import numpy as np

param_dist = {'n_neighbors': np.arange(1, 50)}
knn = KNeighborsClassifier()
knn_random = RandomizedSearchCV(knn, param_distributions=param_dist, n_iter=10, cv=5)
knn_random.fit(X_train_scaled, y_train)
```


```
best_params = knn_random.best_params_
best_score = knn_random.best_score_
print("Best Parameters:", best_params)
print("Best Score:", best_score)
```

```
Best Parameters: {'n_neighbors': 18}
Best Score: 0.735
```

Code to create KNN model without feature selection (test / train data split, model fitting, model predictions, model evaluation):

[5]:
```
from sklearn.model_selection import train_test_split
# Split data into test and training data sets
X = churn_clean_scale_df
y = churn_clean_encoded['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=8, stratify=y)
```

[4]:
```
#fit the KNN model using the randomized search result of k=18

from sklearn.metrics import roc_auc_score

knn = KNeighborsClassifier(n_neighbors = 18)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("The accuracy of the KNN model:")
print(knn.score(X_test, y_test))

y_pred_proba = knn.predict_proba(X_test)[:, 1]
print("The AUC for the KNN model:")
print(roc_auc_score(y_test, y_pred_proba))
```

```
The accuracy of the KNN model:
0.735
The AUC for the KNN model:
0.6153170324733668
```

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University

Code to create KNN model with feature selection (test / train data split, model fitting, model predictions, model evaluation):

```
#use feature selection to improve the model based on the optimal number of n_neighbors
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=8


knn_all_features = KNeighborsClassifier(n_neighbors=18)
knn_all_features.fit(X_train, y_train)
y_pred_all_features = knn_all_features.predict(X_test)
accuracy_all_features = accuracy_score(y_test, y_pred_all_features)
print(f'Accuracy with all features: {accuracy_all_features}')


k_best = SelectKBest(f_classif, k=10)
X_train_selected = k_best.fit_transform(X_train, y_train)
X_test_selected = k_best.transform(X_test)


selected_feature_indices = k_best.get_support(indices=True)
print(f'Selected feature indices: {selected_feature_indices}')


selected_feature_names = X.columns[selected_feature_indices]
print(f'Selected feature names: {selected_feature_names}')


knn_selected_features = KNeighborsClassifier(n_neighbors=18)
knn_selected_features.fit(X_train_selected, y_train)
y_pred_selected_features = knn_selected_features.predict(X_test_selected)
accuracy_selected_features = accuracy_score(y_test, y_pred_selected_features)
print(f'Accuracy with selected features: {accuracy_selected_features}')
```

```
Accuracy with all features: 0.728
Selected feature indices: [   0    1   14   15   16 8441 8442 8448 8453 8454]
Selected feature names: Index(['Unnamed: 0.1', 'Unnamed: 0', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'Contract_One year', 'Contract_Two Year',
       'Multiple_Yes', 'StreamingTV_Yes', 'StreamingMovies_Yes'],
      dtype='object')
Accuracy with selected features: 0.8795
```

```
y_pred_proba_selected = knn_selected_features.predict_proba(X_test_selected)[:, 1]
print("The AUC for the KNN model with selected features:")
print(roc_auc_score(y_test, y_pred_proba_selected))
```

```
The AUC for the KNN model with selected features:
0.9349730577327084
```

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University

Code for classification analysis (confusion matrix and classification report):

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_selected_features))
```

```
              precision    recall  f1-score   support

           0       0.90      0.93      0.92      1456
           1       0.81      0.73      0.77       544

    accuracy                           0.88      2000
   macro avg       0.86      0.83      0.84      2000
weighted avg       0.88      0.88      0.88      2000
```

```
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test, y_pred_selected_features)
print(cm1)
```

```
[[1361   95]
 [ 146  398]]
```

In text citations:
(“Measuring Model performance”, n.d.)
(“Hyperparameter tuning”, n.d.)
(“How good is your model?”, n.d.)
(“Classification metrics”, n.d.)
(“SelectKBest”, 2023)

Part 5

E1.
To assess the performance and accuracy of the classification model with feature selection, I utilized the accuracy and the area under the curve (AUC). The AUC quantifies the performance of predictive models with a number between 0 and 1 that conveys how well the model can classify the likelihood of belonging to a class or category, providing a valuable measure of predictive performance. An AUC closer to 1 indicates that the model can perform better at distinguishing between instances. It conveys that the model’s predictions are well calibrated and effectively stabilizes class imbalance. In this context, equipped with selected features, I observed an AUC of approximately 93%, which suggests that the model excels at predicting customer churn. Additionally, accuracy, which is the overall ability of the

model to correctly predict the correct classification, was approximately 88%, which indicates robust performance.

E2.

The results of my classification analysis indicate the model's proficiency in classifying outcomes. However, when considering the potential impact of imbalances in the test sets, there was a higher proportion of data leaning towards customers who did not churn. This imbalance might have contributed to the model's predictions, since the KNN classification technique was used. Despite standardizing the data, the use of the KNN method introduces challenges because it relies on information from its nearest neighbors to make predictions. Since the data favors customers not churning in the actual dataset, this imbalance, even after standardization, may have impacted the predictions as reflected in the classification report (majority true negatives) and confusion matrix analysis. Examining the mean of the individual predicted probabilities for both the non-selected features model and the selected features models displays low probabilities. Although the selected features lead to an improved model, there remains a subtle challenge in the classification's accuracy and the model's confidence, despite the high accuracy score. The implications of this classification analysis suggest that the model without selected features performs better than random chance. However, with the inclusion of chosen features, there is a heightened chance of accurately classifying customer churn. This presents a valuable opportunity to provide insights to telecommunication companies.

E3.

As briefly discussed, one limitation in my data analysis stems from a skewness in the distribution of the actual data within the test set. Currently, there are more instances of customer who did not churn. This skewness raises an intriguing question: what if the test set were slightly skewed towards customer churn? Exploring this scenario could provide insights into how predictions would be inexpertly differently. This limitation is important in the context of KNN classification model. The KNN algorithm relies on the notion of proximity to nearest neighbors for making predictions. When faced with skewed distribution in the test set, where there's an overrepresentation of one class (customers who did not churn), the model may be inclined to favor predictions aligned with the majority class. This can introduce biases and impact the overall interpretability of the model's predictions. In summary, addressing and acknowledging this limitation becomes crucial for ensuring robustness and reliability of the model.

E4.

Given that the most impactful variables identified include Tenure, MonthlyCharge, Bandwidth_GB_Year, Contract_One year, Contract_Two year, Multiple_yes, StreamingTV_yes, and StreamingMovies_Yes, there is a valuable opportunity for telecommunications companies to leverage this information for strategic decision-making, potentially leading to policy adjustments and enhanced customer retention strategies. For instance, it can be observed that the type of contract a customer holds in relation to customer churn is important. The data suggests that yearly contracts influence customer behavior. The telecommunications companies might consider discontinuing the month-to-month contract option, which could contribute to reducing customer churn. Additionally, an analysis of how tenure and monthly charges vary across different contract types can present valuable insights. By understanding the combination of these dynamics, the companies can tailor their policies for customers with specific contract types, optimizing pricing strategies, or customizing service offerings based on the observed trends. In conclusion, the significant variables provide telecommunications with a variety of strategic possibilities.

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University

In text citations:
(“Measuring model performance”, n.d.)
(“Classification metrics”, n.d.)
(“The classification challenge”, n.d.)

Part 5

H.
Citations for code:

DataCamp. (n.d.). Completeness [Video file]. Retrieved from https://campus.datacamp.com/courses/cleaning-data-in-python/advanced-data-problems-3?ex=8

DataCamp. (n.d.). Importing flat files using pandas [Video file]. Retrieved from https://campus.datacamp.com/courses/introduction-to-importing-data-in-python/introduction-and-flat-files-1?ex=15

DataCamp. (n.d.). Mean, median, & mode imputations [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/imputation-techniques?ex=1

DataCamp. (n.d.). Imputing using fancyimpute [Video file]. Retrieved from https://campus.datacamp.com/courses/dealing-with-missing-data-in-python/advanced-imputation-techniques?ex=1

DataCamp. (n.d.). Selecting features for better model performance [Video file]. Retrieved from https://campus.datacamp.com/courses/dimensionality-reduction-in-python/feature-selection-ii-selecting-for-model-accuracy?ex=1

Lekhana_Ganji. (2023, April 18). Machine Learning - One Hot Encoding of Datasets in Python. GeeksforGeeks. https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/

DataCamp. (n.d.). Working with model objects [Video file]. Retrieved from https://campus.datacamp.com/courses/introduction-to-regression-with-statsmodels-in-python/predictions-and-model-objects-2?ex=5

GeeksforGeeks. (n.d.). How to Rename Columns in Pandas DataFrame. Retrieved from https://www.geeksforgeeks.org/how-to-rename-columns-in-pandas-dataframe/

DataCamp. (n.d.). The classification challenge [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/classification?ex=6

DataCamp. (n.d.). Preprocessing data [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/preprocessing-and-pipelines?ex=1

DataCamp. (n.d.). Centering and scaling [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/preprocessing-and-pipelines?ex=9

DataCamp. (n.d.). How good is your model? [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/fine-tuning-your-model?ex=1

DataCamp. (n.d.). Area under the ROC curve [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/fine-tuning-your-model?ex=7

DataCamp. (n.d.). Hyperparameter tuning [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/fine-tuning-your-model?ex=9

scikit-learn. (2023). SelectKBest. scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

DataCamp. (n.d.). Classification metrics [Video file]. Retrieved from https://campus.datacamp.com/courses/model-validation-in-python/validation-basics?ex=9

DataCamp. (n.d.). Measuring model performance [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/classification?ex=9

I.
Citations for content:

DataCamp. (n.d.). The classification challenge [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/classification?ex=6

scikit-learn. (2023). SelectKBest. scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

DataCamp. (n.d.). How good is your model? [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/fine-tuning-your-model?ex=1

DataCamp. (n.d.). Area under the ROC curve [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/fine-tuning-your-model?ex=7

DataCamp. (n.d.). Hyperparameter tuning [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/fine-tuning-your-model?ex=9

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University

DataCamp. (n.d.). Classification metrics [Video file]. Retrieved from https://campus.datacamp.com/courses/model-validation-in-python/validation-basics?ex=9

DataCamp. (n.d.). Measuring model performance [Video file]. Retrieved from https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/classification?ex=9

Jasmine Cooper
D209
Classification Analysis
Task 1
December 1, 2023
Western Governor's University