D209 – Assessment NVM2 TASK 2: PREDICTIVE ANALYSIS

Part I: Research Question

A. Describe the purpose of this data mining report by doing the following:

1. Propose **one** question relevant to a real-world organizational situation that you will answer using **one** of the following classification methods:
   - decision trees
   - random forests
   - advanced regression (i.e., lasso or ridge regression)

Predicting which customers are more likely to churn. Which features provided in the churn dataset are more relevant to the analysis? I will be using DECISION TREE ALGORITHM.

2. 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.

The company's stakeholder will have access to know which features are more determinant in predict the churn and with that information they will be able to create/modify marketing campaigns to make sure more customers are retained.

Part II: Method Justification

B. Explain the reasons for your chosen classification method from part A1 by doing the following:

1. Explain how the prediction method you chose analyzes the selected data set. Include expected outcomes.

I chose decision trees to analyze my data set. Decision tree is a supervised machine learning algorithm and can be used to solve regression and classification problems. The algorithm works looking at the entire training data set (root node), dividing it in two or more sub-nodes. From here the algorithm divides the sub-nodes in smaller sub-nodes (decision node) until they can't do it anymore. At this point, this sub-node is called a leaf. Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification of the example[1].

Expected Outcomes: The Expected Value is **the average outcome if the decision was made multiple times**[2].

2. Summarize **one** assumption of the chosen prediction method.

The Decision Tree algorithm assumes all independent variables interact each other, which it is generally not the case every time[3].

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

I have been working with Python since D206 and will continue learning this language. I have been working with the free version of Pycharm (Community). The packages and libraries are:

-) Panda: package to read our dataset, present some statistics of the features, clean and modify data
-) Numpy: performs a wide variety of mathematical operations on arrays
-) Matplotlib: important package in any data science project. It helps with data visualization.
-) Scikit-learn: packages that will split, test, train and make predictions on our dataset
-) Seaborn: this package will be used for more detailed graphs and matrices

Part III: Data Preparation

C. Perform data preparation for the chosen data set by doing the following:

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

As explained in D208, mathematical models cannot be successfully accomplished with categorical data so before attempting a model we will have to create dummy variables to convert categorical into numerical data. For categorical data with only Yes and No, I will be modifying the feature to 1 and 0.

2. 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 continuous or categorical.

The initial dataset is my entire dataset presented in “churn_raw_data.csv”. Variables that have “int64” or “float64” are continuous and variables with “object” are categorical. The only exceptions are the variables listed below from 30 to 37. These are the customer survey variables and they are not continuous, they are discrete ordinal variables.

```
# Loading the Original Churn Dataset
churn_df2 = pd.read_csv('/Users/bia/Desktop/churn_raw_data.csv')

#Variables Types
churn_df2.info()
```

Picture 1: Dataset Info

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

1-) Read the dataset using Panda (read_csv) and naming it “churn_df2

```
# Loading the Original Churn Dataset
churn_df2 = pd.read_csv('/Users/bia/Desktop/churn_raw_data.csv')
```

2-) Analyze basic stats of the dataset using “.describe()”

```
#Basic Stats
churn_desc = churn_df2.describe()
print(churn_desc)
```

```
memory usage: 4.0+ MB
None
        Unnamed: 0    CaseOrder  ...         item7         item8
count  10000.00000  10000.00000  ...  10000.000000  10000.000000
mean    5000.50000   5000.50000  ...      3.509500      3.495600
std     2886.89568   2886.89568  ...      1.028502      1.028633
min        1.00000      1.00000  ...      1.000000      1.000000
25%     2500.75000   2500.75000  ...      3.000000      3.000000
50%     5000.50000   5000.50000  ...      4.000000      3.000000
75%     7500.25000   7500.25000  ...      4.000000      4.000000
max    10000.00000  10000.00000  ...      7.000000      8.000000
```

Picture 2: Basic Stats of the dataset

3-) I dropped some features that would not help to predict the churn rate and also I replaced some non descriptive labels in Customer Survey columns (8 last columns).

```
#Dropping Columns Job, Timezone, Education and customer_ID
churn_df = churn_df2.drop(columns=['Job', 'Timezone', 'Customer_id',
'Education'])
#Dropping the 1st column "unnamed"
churn_df2 = churn_df2.iloc[:,1:]

#Dropping more non important columns to predict churn
churn_df2 = churn_df2.drop(columns=['CaseOrder', 'Interaction', 'City',
'State', 'County', 'Zip', 'Lat', 'Lng', 'Population'])

#Renaming the last 8 survey columns for a more descriptive value
churn_df2.rename(columns = {'item1':'CS Responses', 'item2':'CS Fixes',
'item3':'CS Replacements',
                            'item4':'CS Reliability', 'item5':'CS Options',
'item6':'CS Respectfulness',
                            'item7':'CS Courteous', 'item8':'CS
Listening'},inplace=True)
```

4-) Check for missing data.

```
#Finding missing values in my dataset
churn_df2.isnull().any(axis=1)
null_values = churn_df2.isna().any()
print(null_values)
#How many rows of data are we missing?
data_null_sum = churn_df2.isnull().sum()
print(data_null_sum)
```

Picture 3: Columns with "TRUE" have missing data

5-) The data presented missing values so for numerical features I replaced missing values for the median and for categorical for the mode.

```
#Filling the missing data with the median of each variable
#We saw that the columns Children, Age, Income, Tenure and Bandwidth_GB_Year
have missing values
na_cols = churn_df2.isna().any()
na_cols = na_cols[na_cols == True].reset_index()
na_cols = na_cols["index"].tolist()
for col in churn_df2.columns[1:]:
    if col in na_cols:
        if churn_df2[col].dtype != 'object':
            churn_df2[col] =
churn_df2[col].fillna(churn_df2[col].median()).round(0)
```

```
#Phone and Techie Columns are categorical with missing values as well
print(churn_df2['Phone'].unique())
print(churn_df2['Techie'].unique())
#Since We have "YES" "NO" and "NAN" we will need to replace the nan values
for something
df_stats_phone = churn_df2['Phone'].describe()
print(df_stats_phone)

df_stats_phone = churn_df2['Techie'].describe()
print(df_stats_phone)

#Since these are categorical columns, I am going to replace the "NAN" values
for whatever shows more
#Phone --> "YES"
#Techie --> "NO"

churn_df2 = churn_df2.fillna(churn_df2.mode().iloc[0])

# #Making sure all values were replaced by the median (num) and mode (cat),
so we check against missing data again
missing_data_clean = churn_df2.isna().any()
print(missing_data_clean)
```

```
Name: Techie, dtype: object
Area                   False
Children               False
Age                    False
Employment             False
Income                 False
Marital                False
Gender                 False
Churn                  False
Outage_sec_perweek     False
Email                  False
Contacts               False
Yearly_equip_failure   False
Techie                 False
Contract               False
Port_modem             False
Tablet                 False
InternetService        False
Phone                  False
Multiple               False
OnlineSecurity         False
OnlineBackup           False
DeviceProtection       False
TechSupport            False
StreamingTV            False
StreamingMovies        False
PaperlessBilling       False
PaymentMethod          False
Tenure                 False
MonthlyCharge          False
Bandwidth_GB_Year      False
CS Responses           False
CS Fixes               False
CS Replacements        False
CS Reliability         False
CS Options             False
CS Respectfulness      False
CS Courteous           False
CS Listening           False
dtype: bool
```

Picture 4: Missing data successfully replaced

6-) Extract the clean data set:

```
# #Extracting the clean dataset
churn_df2.to_csv('churn_clean.csv')
churn_df2= pd.read_csv('churn_clean.csv')
```

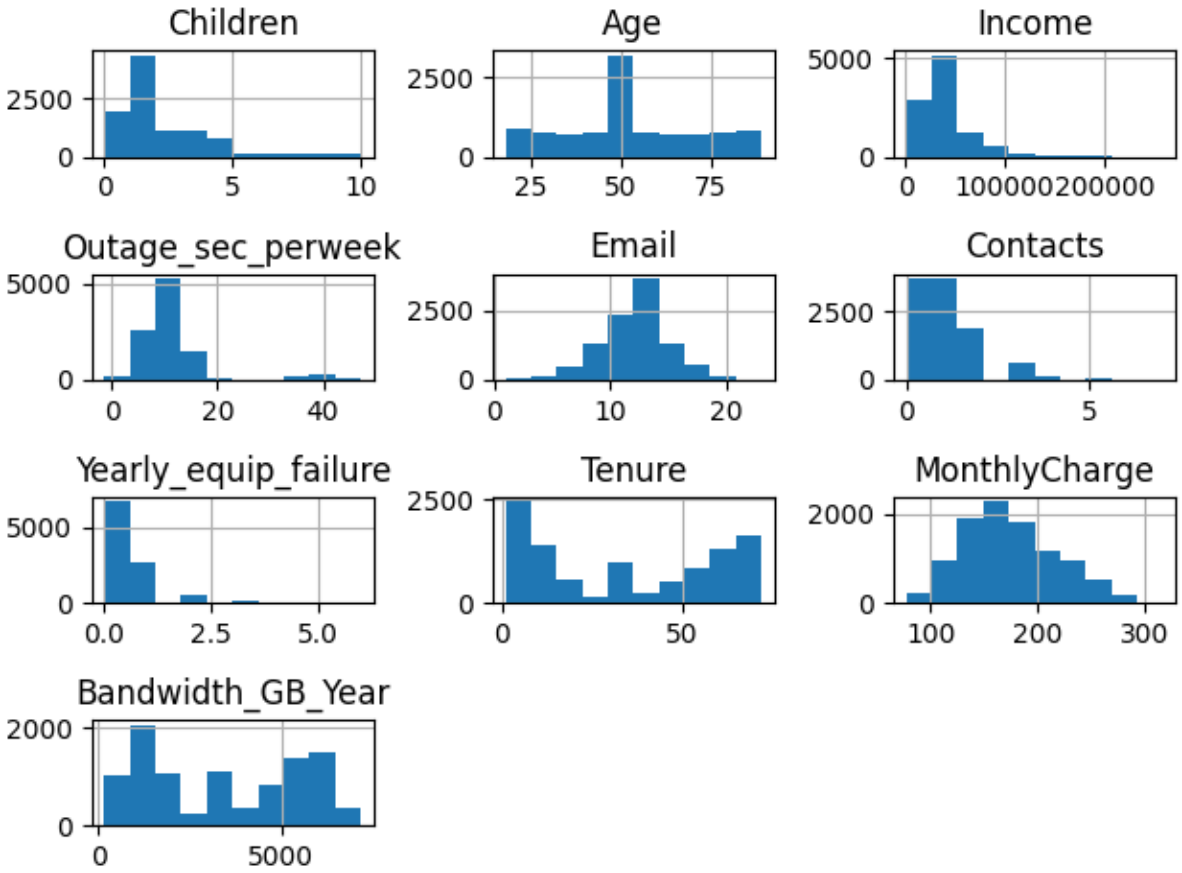
7-) Examine numerical data for outliers and plot a histogram

```
# Checking for outliers in the continuous variables
numerical_churn_df2 = churn_df2[['Tenure','MonthlyCharge',
'Bandwidth_GB_Year']]

# Checking for outliers at 25%, 50%, 75%, 90%, 95% and 99%
```

```
print(numerical_churn_df2.describe(percentiles=[.25, .5, .75, .90, .95,
.99]))

#Graphs of Numerical Features
churn_df2[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
'Contacts', 'Yearly_equip_failure', 'Tenure',
           'MonthlyCharge', 'Bandwidth_GB_Year']].hist()
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
plt.show()
```

	Tenure	MonthlyCharge	Bandwidth_GB_Year
count	10000.000000	10000.000000	10000.000000
mean	34.640500	174.076305	3397.122700
std	25.188194	43.335473	2072.726654
min	1.000000	77.505230	156.000000
25%	9.000000	141.071078	1312.000000
50%	36.000000	169.915400	3382.000000
75%	60.000000	203.777441	5466.000000
90%	67.000000	238.683060	6094.100000
95%	70.000000	253.824616	6343.050000
99%	72.000000	275.859482	6717.010000
max	72.000000	315.878600	7159.000000

Picture 5: Percentile Analysis of Num Features



Picture 6: Histograms

8-) Change all categorical data either replacing Yes and No for 1 and 0 or creating dummy variables for features with 3 or more levels.

```
#Changing Variables from NO/YES to 0/1
churn_df2.Churn.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Phone.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.PaperlessBilling.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Techie.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Port_modem.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Tablet.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Multiple.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.OnlineSecurity.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.OnlineBackup.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.DeviceProtection.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.TechSupport.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.StreamingTV.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.StreamingMovies.replace({"Yes":1, "No":0}, inplace = True)

#Creating a dummy variable for some of the categorical variables with 3 or
more levels
```

```
dummy1 = pd.get_dummies(churn_df2[['Marital', 'Contract', 'PaymentMethod',
'Gender', 'InternetService', 'Area',
                                   'Employment']])
#Adding the results to the master dataframe
churn_df2 = pd.concat([churn_df2, dummy1], axis=1)

#We have created dummies for the below variables, so we can drop them
churn_df2 = churn_df2.drop(['Marital', 'Contract', 'PaymentMethod', 'Gender',
'InternetService', 'Area','Employment'], 1)
```

9-) Extract the desired dataset (prepared_churn_data.csv) to start modelling using KNN method.

```
#Extract the "Prepared"" dataset
#Dropping the 1st column
churn_df2 = churn_df2.iloc[:,1:]
churn_df2.to_csv('prepared_churn_data.csv')
churn_df2 = pd.read_csv('prepared_churn_data.csv')
df = churn_df2.columns
print('The dataset columns are ', df)
```

4. Provide a copy of the cleaned data set. This dataset will be uploaded when the report, code and Panopto video recording are all ready.

Part IV: Analysis

D. Perform the data analysis and report on the results by doing the following:

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

```
#Decision Tree Algorithm
#Y variable is the target: Churn
y = churn_df2.Churn.values
#Removing Churn from remaining data
X = churn_df2.drop('Churn', axis = 1)

SEED = 1
#Train - Test - Split: 70%-30%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3,
random_state = SEED)

#Decision Tree model
# decision_tree = DecisionTreeRegressor(max_depth = 8, min_samples_leaf =
0.1, random_state = SEED)
decision_tree = DecisionTreeClassifier(max_depth = 8, min_samples_leaf = 0.1,
random_state = SEED)

#Fit dataframe to Decision Tree Regressor model
decision_tree.fit(X_train, y_train)

#Calculate y_pred
y_pred = decision_tree.predict(X_test)
y_train_pred = decision_tree.predict(X_train)

#Compute test set MSE
```

```
mse_decision_tree = MSE(y_test, y_pred)
print('Initial MSE score is: ', mse_decision_tree)

#Calculate Root Mean Squared Error (RMSE)
rmse_decision_tree = mse_decision_tree**(1/2)

#Print initial RMSE
print('Initial RMSE score: {:.3f}'.format(rmse_decision_tree))

cm_train = confusion_matrix(y_train, y_train_pred)
cm_test = confusion_matrix(y_test, y_pred)

print('Confusion Matrix Train is ',cm_train)
print('Confusion Matrix Test is ',cm_test)
```

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

Our initial RMSE score for the decision tree classifier model is 0.420. The Initial MSE is 0.176 and below is displayed the confusion matrix. The accuracy score for this initial model is 0.824.

```
      dtype='object')
Initial MSE score is:  0.176
Initial RMSE score: 0.420
Confusion Matrix Train is  [[4882  294]
 [ 848  976]]
Confusion Matrix Test is  [[2042  132]
 [ 396  430]]
Accuracy score of Decision Tree model: 0.824
```

Picture 7: Initial MSE and RMSE Scores, Confusion Matrix and Accuracy

To check if I am able to improve the accuracy, I am now going to create a pipeline and normalize the data. A pipeline is a way to automate the machine learning workflow by allowing preprocessing of the data and instantiation of the estimator to occur in a single piece of code [5].

```
#Create pipeline object
#Normalize Data
steps = [('scaler', StandardScaler()),('decision_tree',
DecisionTreeClassifier())]

#Split Dataframe
pipeline = Pipeline(steps)

#Scale dataframe with pipeline object
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled =
train_test_split(X, y, test_size = .3, random_state = SEED)
decision_tree_scaled = pipeline.fit(X_train_scaled, y_train_scaled)
#Predict from scaled dataframe
y_pred_scaled = pipeline.predict(X_test_scaled)
```

```
#Print new accuracy
print('New accuracy score of scaled Decision Tree model:
{:0.3f}'.format(accuracy_score(y_test_scaled, y_pred_scaled)))

#Classification Report after scaling
print(classification_report(y_test_scaled, y_pred_scaled))

#Calculation of the Confusion Matrix
confusion_matrix = confusion_matrix(y_test_scaled, y_pred_scaled)
print('Confusion Matrix Scaled Model is: ',confusion_matrix)

#Calculating AUC Score
pred_prob = decision_tree.predict_proba(X_test)

#ROC Curve
fpr1, tpr1, thresh1 = roc_curve(y_test, pred_prob[:,1], pos_label=1)

#ROC curve for tpr = fpr
random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)

#AUC Score
auc_score = roc_auc_score(y_test, pred_prob[:,1])

print('AUC Score is ',auc_score)
```

```
Initial MSE score is:  0.176
Initial RMSE score: 0.420
Confusion Matrix Train is  [[4882  294]
 [ 848  976]]
Confusion Matrix Test is  [[2042  132]
 [ 396  430]]
Accuracy score of Decision Tree model: 0.824
New accuracy score of scaled Decision Tree model: 0.831
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      2174
           1       0.69      0.69      0.69       826

    accuracy                           0.83      3000
   macro avg       0.79      0.79      0.79      3000
weighted avg       0.83      0.83      0.83      3000

Confusion Matrix Scaled Model is:  [[1921  253]
 [ 254  572]]
AUC Score is  0.893334109250642
```

Picture 8: Scaled Model Results

We can see that our accuracy increased from 0.824 to 0.831 just by scaling our model. The AUC result is 0.8933

The Area Under the Curve (AUC) is **the measure of the ability of a classifier to distinguish between classes** and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes [6].

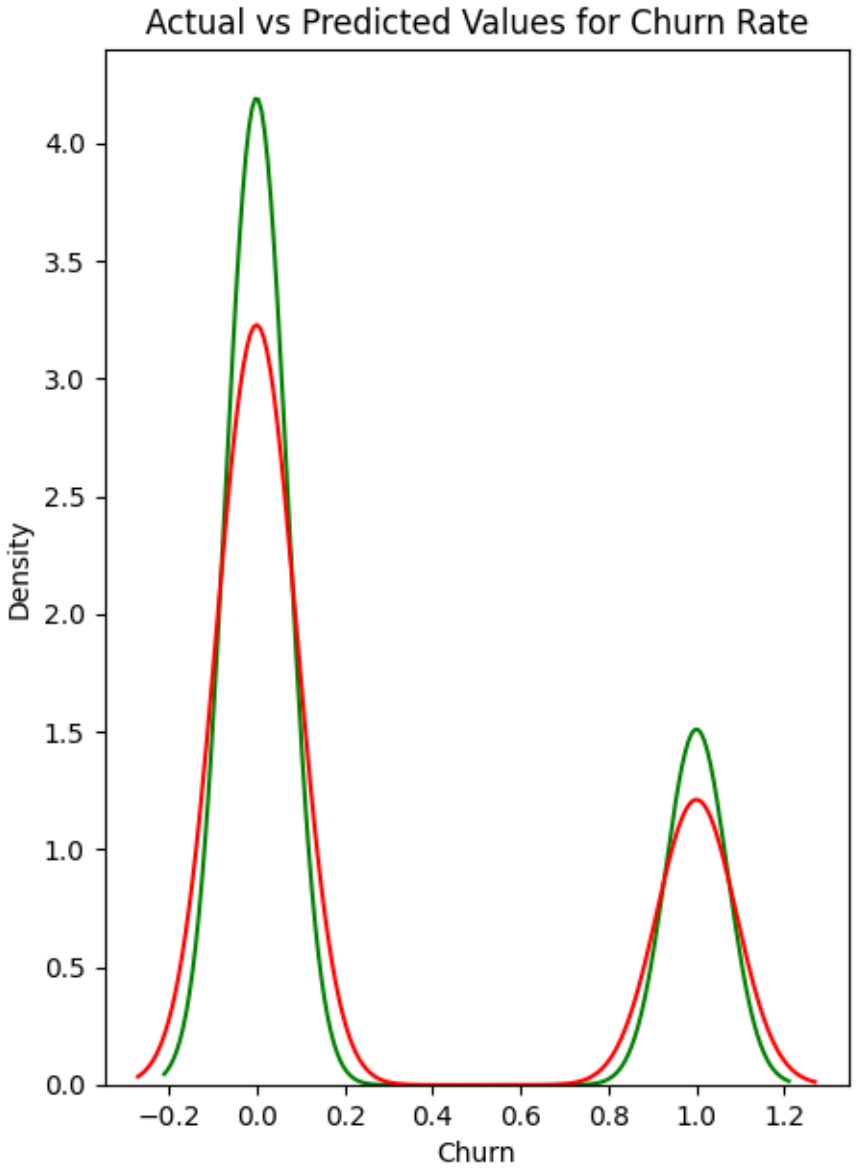
AUC is the area under the ROC. AUC Score is 0.8933 (shown below) and it represents a good accuracy. The scaled model MSE and RMSE are respectively, 0.1686 and 0.411.

```
      dtype='object')
Initial MSE score is:  0.176
Initial RMSE score: 0.420
Confusion Matrix Train is  [[4882  294]
 [ 848  976]]
Confusion Matrix Test is  [[2042  132]
 [ 396  430]]
Accuracy score of Decision Tree model: 0.824
MSE score is:  0.16866666666666666
RMSE score: 0.411
New accuracy score of scaled Decision Tree model: 0.831
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      2174
           1       0.69      0.69      0.69       826

    accuracy                           0.83      3000
   macro avg       0.79      0.79      0.79      3000
weighted avg       0.83      0.83      0.83      3000

Confusion Matrix Scaled Model is:  [[1923  251]
 [ 255  571]]
AUC Score is  0.893334109250642

Process finished with exit code 0
```

Picture 9: Scaled Model Results

```
#Plot Prediction and Actual Values
plt.figure(figsize=(5, 7))
ax = sns.distplot(churn_df2['Churn'], hist=False, color="g", label="Actual
Values")
sns.distplot(y_pred_scaled, hist=False, color="r", label="Predicted Values",
ax=ax)
plt.title('Actual vs Predicted Values for Churn Rate')
plt.show()
```

In the following plot we can see the actual churn values in green and the predicted in red.



Picture 10: Actual vs Predicted Churn Values

3. Provide the code used to perform the classification analysis from part D2.
Code is along the documentation.

Part V: Data Summary and Implications

E. Summarize your data analysis by doing the following:

1. Explain the accuracy and the mean squared error (MSE) of your prediction model.

As shown in Picture 9, my scaled model accuracy is 0.83 and MSE is 0.1686

2. Discuss the results and implications of your prediction analysis.

The AUC score is a good metric to measure binary classification. Our AUC score of 0.893 means that our model has a good accuracy.

In order to try to improve our accuracy, I will implement the **cross validation** method and find the optimal value of "max_depth" and "criterion".

k-Fold Cross Validation: Cross-validation is when the dataset is randomly split up into 'k' groups [7]. One group is the test and the rest is the training set. The model is trained and a score is achieved. This same process is repeated for every distinct group (since my cross validation is 5, we will do this same step 5 times, every time using a different group as the test set).

```
#k-Fold Cross Validation Method to find the optimal number for max_depth and
criterion
param_grid = {'criterion':['gini','entropy'],'max_depth': np.arange(1, 50)}
dtc = DecisionTreeClassifier()
dtc_cv = GridSearchCV(dtc, param_grid, cv=5)
#Fit Model
dtc_cv.fit(X_train, y_train)
#Print the optimal number of n_neighbors
print('Best parameters for this Decision Tree Classifier:
{}'.format(dtc_cv.best_params_))
```

```
Confusion Matrix Scaled Model is:  [[1925  249]
 [ 251  575]]
AUC Score is  0.893334109250642
Best parameters for this Decision Tree Classifier: {'criterion': 'gini', 'max_depth': 6}

Process finished with exit code 0
```

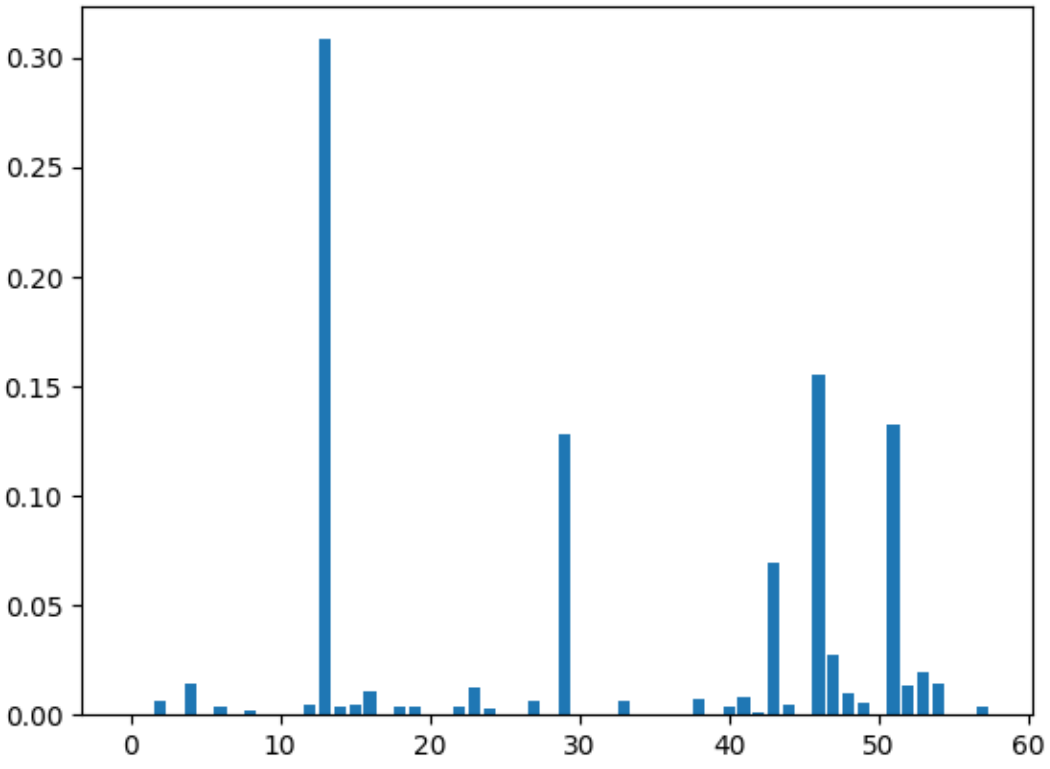
Picture 11: k-Fold Cross Validation Parameters

3. Discuss **one** limitation of your data analysis.

Overfitting is very common in decision tree algorithm. If the algorithm is over fitted, the tree would be extremely sensitive making the model to have a high variance. And therefore, this model would not generalize well on unseen data [4].

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

```
# define dataset
X, y = make_classification(n_samples=1000, n_features=58, n_informative=5,
n_redundant=5, random_state=1)
# define the model
model = DecisionTreeClassifier()
# fit the model
model.fit(X, y)
# get importance
importance = model.feature_importances_
# summarize feature importance
for i,item in enumerate(importance):
    print('{0:s}: {1:.2f}'.format(churn_df2.columns[i], item))
# plot feature importance
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()
```



Picture 12: Feature Importance Decision Tree

The MSE of 0.17 suggests that we have a high accuracy model, and with that in hands, stakeholders will be able to predict when a customer might churn. According to the plot above, we have 5 features that influence churn the most: Multiple, CS Respectfulness, Area_Suburban, Gender_Prefer Not To Answer, PaymentMethod_Mailed Check. The course of action would be to not sign up customers with multiple lines, be respectful when contacted by any customer,

and set up customers with auto payment instead of mailed check for example. All the features coefficients are in the two pictures below.

```
Unnamed: 0: 0.00
Children: 0.00
Age: 0.01
Income: 0.00
Churn: 0.01
Outage_sec_perweek: 0.00
Email: 0.00
Contacts: 0.00
Yearly_equip_failure: 0.00
Techie: 0.00
Port_modem: 0.00
Tablet: 0.00
Phone: 0.00
Multiple: 0.31
OnlineSecurity: 0.00
OnlineBackup: 0.00
DeviceProtection: 0.01
TechSupport: 0.00
StreamingTV: 0.00
StreamingMovies: 0.00
PaperlessBilling: 0.00
Tenure: 0.00
MonthlyCharge: 0.00
Bandwidth_GB_Year: 0.01
CS Responses: 0.00
CS Fixes: 0.00
CS Replacements: 0.00
CS Reliability: 0.01
CS Options: 0.00
CS Respectfulness: 0.13
CS Courteous: 0.00
CS Listening: 0.00
Marital_Divorced: 0.00
Marital_Married: 0.01
Marital_Never Married: 0.00
```

Picture 13: Feature Importance Part I

```
Marital_Never Married: 0.00
Marital_Separated: 0.00
Marital_Widowed: 0.00
Contract_Month-to-month: 0.00
Contract_One year: 0.01
Contract_Two Year: 0.00
PaymentMethod_Bank Transfer(automatic): 0.00
PaymentMethod_Credit Card (automatic): 0.01
PaymentMethod_Electronic Check: 0.00
PaymentMethod_Mailed Check: 0.07
Gender_Female: 0.00
Gender_Male: 0.00
Gender_Prefer not to answer: 0.16
InternetService_DSL: 0.03
InternetService_Fiber Optic: 0.01
InternetService_None: 0.01
Area_Rural: 0.00
Area_Suburban: 0.13
Area_Urban: 0.01
Employment_Full Time: 0.02
Employment_Part Time: 0.01
Employment_Retired: 0.00
Employment_Student: 0.00
Employment_Unemployed: 0.00

Process finished with exit code 0
```

Picture 14: Feature Importance Part II

Part VI: Demonstration

F. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

The video recorded can be found here:
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=f565dbca-db0a-4a96-a0bd-adb601774bc0

G. Record the web sources used to acquire data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

H. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

[1] (2020, Jan) CHAUHAN, Nagesh Singh Decision Tree Algorithm, Explained
https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

[2] (2019, Jun 3rd) Teaching Guide: Decision Trees https://www.aqa.org.uk/resources/business/as-and-a-level/business-7131-7132/teach/teaching-guide-decision-trees

[3] BHALLA, Deepanshu Decision Tree in R: Step by Step Guide https://www.listendata.com/2015/04/decision-tree-in-r.html

[4] (2020, March 4th) KANA, Michel How to Find Decision Tree Depth via Cross-Validation https://towardsdatascience.com/how-to-find-decision-tree-depth-via-cross-validation-2bf143f0f3d6

[5] (2021, Jan 25th) BOYLES, Jennifer Beginner's Guide to k-Nearest Neighbors & Pipeline in Classification https://medium.com/analytics-vidhya/beginners-guide-to-k-nearest-neighbors-pipelines-in-classification-704b87f534e2

[6] (2020, June 16th) BHANDARI, Aniruddha AUC-ROC Curve in Machine Learning Clearly Explained https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

[7] (2018, Sept 26th) ALLIBHAI, Eijaz Building a k-Nearest-Neighbors (k-NN) Model with Scikit-learn https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a