**Jason Pena**

**WGU University**

D209 PERFORMANCE ASSESSMENT NVM2 TASK 2

PREDICTIVE ANALYSIS

**Part I: Research Question**

1. Describe the purpose of this data mining report by doing the following:
2. Propose one question relevant to a real-world organizational situation that you will answer using one of the following prediction methods:

• decision trees

• random forests

• advanced regression (i.e., lasso or ridge regression)

Which customers are at high risk of churn? And which customer’s features/variables are most significant to churn? This question will be using the classification method: **Decision trees algorithm**

1. 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 goal or objective of this analysis is to be able to predict in advance that a specific customer is likely to churn. "The churn rate, also known as the rate of attrition, is the rate at which customers stop doing business with an entity. It is most expressed as the percentage of service subscribers who discontinue their subscriptions within a given time period (Investopedia, 2020) Stakeholders in the company can benefit by this analysis by understanding more effectively which customers are likely to churn soon because this will provide weight for decisions in marketing improved services to customers with these characteristics and past user experiences.

**Part II: Method Justification**

1. Explain the reasons for your chosen classification method from part A1 by doing the following:
2. Explain how the classification method you chose analyzes the selected data set. Include expected outcomes.

“DTs are ML algorithms that progressively divide data sets into smaller data groups based on a descriptive feature, until they reach sets that are small enough to be described by some label.” (Yse, 2019) Trees are capable of capturing non-linear relationships between features & labels. Also, they do not require feature scaling such as standardization & normalization. “Regression Trees are used when the dependent variable is continuous or quantitative, and Classification Trees are used when the dependent variable is categorical or qualitative.” (Yse, 2019)

A picture containing icon

Description automatically generated

“DTs are composed of nodes, branches and leafs. Each node represents an attribute (or feature), each branch represents a rule (or decision), and each leaf represents an outcome. The depth of a Tree is defined by the number of levels, not including the root node.” (Yse, 2019)

1. Summarize one assumption of the chosen classification method.

“ Below are some of the assumptions we make while using Decision tree:

* In the beginning, the whole training set is considered as the root.
* Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
* Records are distributed recursively on the basis of attribute values.” (Chauhan, 2020)

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

* NumPy
* Pandas
* Seaborn
* Matplot.lib.pylot
* Scikit-learn

“NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.” (Numpy.org, 2021)

“Similar to Numpy, pandas deals primarily with data in 1-D and 2-D arrays; however, pandas handles them differently.“ (Educative.io, 2021)

“In pandas, 1-D arrays are referred to a series. A **series** is created through the pd.Series constructor, which has a lot of optional arguments. The most common argument is data, which specifies the elements of the series.” (Educative.io, 2021)

“A DataFrame is simply a 2-D array. It can be created through the pd.DataFrame constructor, which takes in essentially the same arguments as pd.Series. However, while a series could be constructed from a scalar (representing a single value Series), a DataFrame cannot.” (Educative.io, 2021)

PyLab is a procedural interface to the Matplotlib object-oriented plotting library. Matplotlib is the whole package; matplotlib.pyplot is a module in Matplotlib; and PyLab is a module that gets installed alongside Matplotlib. PyLab is a convenience module that bulk imports matplotlib.pyplot (for plotting) and NumPy (for Mathematics and working with arrays) in a single name space. (tutorialspoints.com, 2021)

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them. (Seaborn.pydata.org, 2021)

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. (tutorialspoints.com, 2021) The tools that will be used in the analysis from sklearn are: Datasets, DecisionTreeRegressor, train\_test\_split, metrics, classification report, preprocessing, mean\_absolute\_error, mean\_squared\_error, RandomForestRegressor, accuracy\_score, GridSearchCV, Kfold, cross\_val\_score, cross\_val\_predict

Part III: Data Preparation

1. Perform data preparation for the chosen data set by doing the following:
2. Describe **one** data preprocessing goal relevant to the classification method from part A1.

One data preprocessing goal relevant to the classification method from part A1 is encoding binary categorical variables into 1/0.

1. Identify the initial data set variables that you will use to perform the classification question from part A1. And classify each variable as continuous or categorical.

Out of the 50 variables in the initial data set, 34 variables have been identified to be used in the classification analysis:

**Continuous** variables in the data set:

* Age (Age of customer as reported in sign-up information)
* Bandwidth\_GB\_Year (Data usage in gigabytes)
* Children (Number of children)
* Contacts (Number of times customer contacted technical support)
* Email (Number of emails sent out to customer)
* Income (Annual income of customer)
* MonthlyCharge (The amount charged to the customer monthly)
* Outage\_sec\_perweek (Average number of seconds per week of system outages in the customer’s neighborhood)
* Tenure (Number of months the customer has stayed with the provider)
* Yearly\_equip\_failure (The number of times customer’s equipment failed and had to be reset/replaced in the past year)

**Categorical** variables in the data set:

* Churn (If the customer is likely to leave)
* Contract (The contract term of the customer (month-to-month, one year, two year))
* Device Protection (Whether the customer has device protection add-on (yes, no))
* Gender (Customer self-identification as male, female, or nonbinary)
* InternetService (Customer’s internet service provider (DSL, fiber optic, None))
* Multiple (Whether the customer has multiple lines (yes, no))
* OnlineBackup (Whether the customer has an online backup add-on (yes, no))
* OnlineSecurity (Whether the customer has an online security add-on (yes, no))
* PaperlessBilling (Whether the customer has paperless billing (yes, no))
* Phone (Whether the customer has a phone service (yes, no))
* Port\_modem (Whether the customer has a portable modem (yes, no))
* StreamingMovies (Whether the customer has streaming movies (yes, no))
* StreamingTV (Whether the customer has streaming TV (yes, no))
* Tablet (Whether the customer owns a tablet such as iPad, Surface, etc. (yes, no))
* Techie (Whether the customer considers themselves technically inclined (based on customer questionnaire when they signed up for services) (yes, no))
* TechSupport (Whether the customer has a technical support add-on (yes, no))

**Discrete** variables in the data set:

* Item1: Timely response
* Item2: Timely fixes
* Item3: Timely replacements
* Item4: Reliability
* Item5: Options
* Item6: Respectful response
* Item7: Courteous exchange
* Item8: Evidence of active listening

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

The data preparation goals and data manipulations will include:

1. Import the dataset into Python.
2. Evaluate the data structure to gain a better understanding of the variables and data types.
3. Provide a name to identify my dataset. The naming convention I chose for my dataset is: Churn\_df
4. Data manipulations made to the data set will be named: df
5. Check for any misleading variable names and rename them.
6. Check for any missing data that could skew the model.
7. Missing data will be inputted with measures of central tendency.
8. Create visualizations to identify any outliers that could affect the model.
9. Summaries of univariate and bivariate statistics to search for any flags
10. Removes less meaningful categorical values from the dataset to provide a fully numerical dataframe to continue with the analysis.
11. Extract and use prepped dataset for the Decision Tree Model.

Annotated Code with explanation of each step:

# Standard Data Science Imports

import numpy as np

import pandas as pd

from pandas import Series, DataFrame

Here I am importing NumPy as pd and pandas as pd. I am also importing Series, and DataFrame

“NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.” (Numpy.org, 2021)

“Similar to Numpy, pandas deals primarily with data in 1-D and 2-D arrays; however, pandas handles them differently.“ (Educative.io, 2021)

“In pandas, 1-D arrays are referred to a series. A **series** is created through the pd.Series constructor, which has a lot of optional arguments. The most common argument is data, which specifies the elements of the series.” (Educative.io, 2021)

“A DataFrame is simply a 2-D array. It can be created through the pd.DataFrame constructor, which takes in essentially the same arguments as pd.Series. However, while a series could be constructed from a scalar (representing a single value Series), a DataFrame cannot.” (Educative.io, 2021)

# Visualization libraries

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

Next I will be importing the Visualization libraries.

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them. (Seaborn.pydata.org, 2021)

#Scikit-learn

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

# Scikit-learn

import sklearn

from sklearn import datasets

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from sklearn.metrics import classification\_report

from sklearn import preprocessing

from sklearn.metrics import mean\_absolute\_error as MAE

from sklearn.metrics import mean\_squared\_error as MSE

from sklearn.metrics import mean\_squared\_error as MSE

# Import model, splitting method & metrics from sklearn

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import cross\_val\_predict

Then I will now import the Scikit-learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. (tutorialspoints.com, 2021)

# Loading the data set into Pandas dataframe

churn\_df = pd.read\_csv(r'C:\Users\Hydraconix\Desktop\DATA\churn\_clean.csv')

# Examining fist five records of dataset

churn\_df.head()

# Viewing DataFrame descriptive information

churn\_df.info

# Getting an overview of descriptive stats

churn\_df.describe()

# Getting data types of features

churn\_df.dtypes

# Checking for null values

churn\_df.isnull()

# Renaming the last 8 Survey Columns for better description of variables

churn\_df.rename(columns = {'Item1' : 'TimelyResponse',

'Item2' : 'Fixes' ,

'Item3' : 'Replacements' ,

'Item4' : 'Reliability' ,

'Item5' : 'Options' ,

'Item6' : 'Respectfulness' ,

'Item7' : 'Courteous' ,

'Item8' : 'Listening'},

inplace=True)

# Converting ordinal categorical data into numeric variables

churn\_df['DummyInternetService'] = churn\_df.InternetService.map({'None' : 0, 'DSL' : 1, 'Fiber Optic' : 2})

churn\_df['DummyContract'] = churn\_df.Contract.map({'Month-to-month' : 0, 'One year' : 1, 'Two Year' : 2})

churn\_df['DummyGender'] = churn\_df.Gender.map({'Nonbinary' : 0, 'Male' : 1, 'Female' : 2})

# Histograms of continuous variables

churn\_df[['Age', 'Bandwidth\_GB\_Year', 'Children', 'Contacts', 'Email', 'Income', 'MonthlyCharge', 'Outage\_sec\_perweek', 'Tenure', 'Yearly\_equip\_failure', 'DummyInternetService', 'DummyContract', 'DummyGender']].hist()

plt.savefig('churn\_pyplot.jpg')

plt.tight\_layout()

Graphical user interface, diagram, application

Description automatically generated

The visualizations of central tendency have revealed normal distribution for: Outage\_sec\_perweek, Email, and MonthlyCharge. When analyzing the visualization for Bandwidth\_GB\_Year and Tenure, the histograms displays bimodal distributions.

# A scatterplot to get an idea of correlations between potentially related variables

sns.scatterplot(x=churn\_df['MonthlyCharge'], y=churn\_df['Churn'], color='green')

plt.show()

A picture containing text

Description automatically generated

# A scatterplot to get an idea of correlations between potentially related variables

sns.scatterplot(x=churn\_df['Outage\_sec\_perweek'], y=churn\_df['Churn'], color='green')

plt.show()

A picture containing chart

Description automatically generated

# A scatterplot to get an idea of correlations between potentially related variables

sns.scatterplot(x=churn\_df['Tenure'], y=churn\_df['Churn'], color='green')

plt.show()

Chart

Description automatically generated with low confidence

# I will now set the plot style to ggplot

plt.style.use('ggplot')

# Countplots of categorical variables

plt.figure()

sns.countplot(x='DeviceProtection', hue='Churn', data=churn\_df, palette='RdBu')

plt.xticks([0,1], ['No', 'Yes'])

plt.show()

Chart, bar chart

Description automatically generated

plt.figure()

sns.countplot(x='Multiple', hue='Churn', data=churn\_df, palette ='RdBu')

plt.xticks([0,1],['No','Yes'])

plt.show()

Chart, bar chart

Description automatically generated

plt.figure()

sns.countplot(x='Techie', hue='Churn', data=churn\_df, palette ='RdBu')

plt.xticks([0,1],['No','Yes'])

plt.show()

Chart, bar chart

Description automatically generated

plt.figure()

sns.countplot(x='TechSupport', hue='Churn', data=churn\_df, palette ='RdBu')

plt.xticks([0,1],['No','Yes'])

plt.show()

Chart, bar chart

Description automatically generated

# A scatter matrix of the discrete variables for high level overview of potential relationships & distributions

churn\_discrete = churn\_df[['Churn', 'TimelyResponse', 'Fixes', 'Replacements', 'Reliability', 'Options', 'Respectfulness', 'Courteous', 'Listening']]

pd.plotting.scatter\_matrix(churn\_discrete, figsize = [30, 30])

Chart

Description automatically generated

# An individual scatterplot for viewing relationship of key financial feature against target variable

sns.scatterplot(x = churn\_df['TimelyResponse'], y = churn\_df['Churn'], color='red')

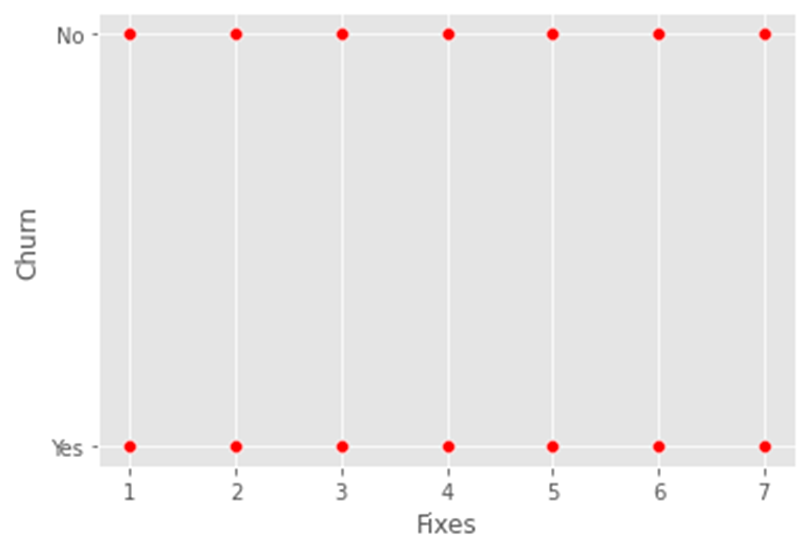
plt.show()

Chart

Description automatically generated

sns.scatterplot(x = churn\_df['Fixes'], y = churn\_df['Churn'], color='red')

plt.show()



sns.scatterplot(x = churn\_df['Replacements'], y = churn\_df['Churn'], color='red')

plt.show()

Chart

Description automatically generated

# Converting binary categorical variables to numeric variables

churn\_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn\_df['Churn']]

churn\_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn\_df['Techie']]

churn\_df['DummyPort\_modem'] = [1 if v == 'Yes' else 0 for v in churn\_df['Port\_modem']]

churn\_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn\_df['Tablet']]

churn\_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn\_df['Phone']]

churn\_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in churn\_df['Multiple']]

churn\_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in churn\_df['OnlineSecurity']]

churn\_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in churn\_df['OnlineBackup']]

churn\_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in churn\_df['DeviceProtection']]

churn\_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in churn\_df['TechSupport']]

churn\_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in churn\_df['StreamingTV']]

churn\_df['DummyStreamingMovies'] = [1 if v == 'Yes' else 0 for v in churn\_df['StreamingMovies']]

churn\_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in churn\_df['PaperlessBilling']]

#Drop original categorical features from dataframe for further analysis

churn\_df = churn\_df.drop(columns=['Churn', 'Contract', 'DeviceProtection', 'Gender', 'InternetService', 'Multiple' , 'OnlineBackup', 'OnlineSecurity', 'PaperlessBilling', 'Phone', 'Port\_modem', 'StreamingMovies', 'StreamingTV', 'Tablet', 'Techie', 'TechSupport'])

#Remove the other less meaningful categorical variables from dataset to provide fully numerical dataframe for further analysis

churn\_df = churn\_df.drop(columns=['Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Marital', 'PaymentMethod'])

# Provide a copy of the prepared data set

churn\_df.to\_csv(r'C:\Users\Hydraconix\Desktop\'churn\_prepared\_dt.csv')

Part IV: Analysis

1. Perform the data analysis and report on the results by doing the following:
2. Split the data into training and test data sets and provide the file(s).

# List features for analysis

features = (list(churn\_df.columns[:-1]))

print('Features for analysis include: \n', features)

# Re-read fully numerical prepared dataset

churn\_df = pd.read\_csv(r'C:\Users\Hydraconix\Desktop\'churn\_prepared\_dt.csv')

# Set predictor features & target variable

X = churn\_df.drop('DummyChurn', axis=1).values

y = churn\_df['DummyChurn'].values

# Create training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state =1)

# Instantiate Decision Tree Regressor model

dt = DecisionTreeRegressor(max\_depth = 8, min\_samples\_leaf = 0.1, random\_state = 1)

# Fit dataframe to Decision Tree Regressor model

dt.fit(X\_train, y\_train)

Output:

DecisionTreeRegressor(max\_depth=8, min\_samples\_leaf=0.1, random\_state=1)

# Predict Outcomes from test set

y\_pred = dt.predict(X\_test)

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

The analysis technique I used to appropriately analyze the data is calculating the Root Mean Square Error (RMSE) to see how far the data points are from the regression line. This will provide how accurate our model has been.

1. Provide the code used to perform the prediction analysis from part D2.

# Compute test set MSE

mse\_dt = MSE(y\_test, y\_pred)

# Compute test set RMSE

rmse\_dt = mse\_dt\*\*(1/2)

# Print initial RMSE

print('Initial RMSE score Decision Tree model: [:.3f]'.format(rmse\_dt))

Output:

Initial RMSE score Decision Tree Regressor model: 0.343

Part V: Data Summary and Implications

1. Summarize your data analysis by doing the following:
2. Explain the accuracy and the mean squared error (MSE) of your prediction model.

# Compute the coefficient of determination (R-squared)

scores = cross\_val\_score(dt, X, y, scoring='r2')

# Print R-squared value

print('Cross validation R-squared values: ', scores)

Output:

Cross validation R-squared values: [0.29953671 0.29785279 0.44989404 0.0949159 0.07501985]

# Print Mean Squared Error

print('With a manual calculation, the Mean Squared Error: {:.3f} '.format(sum(abs(y\_test - y\_pred)\*\*2)/len(y\_pred)))

# Or

print('Using scikit-lean, the Mean Squared Error: {:.3f}'.format(MSE(y\_test, y\_pred)))

Output:

With a manual calculation, the Mean Squared Error: 0.118

Using scikit-lean, the Mean Squared Error: 0.118

# Calculate & print the Root Mean Squared Error

RMSE = MSE(y\_test, y\_pred)\*\*(1/2)

# Print the Root Mean Squared Error

print('Root Mean Squared Error: {:.3f} '.format(RMSE))

Output:

Root Mean Squared Error: 0.343

# Get parameters of Decision Tree Regression model

dt.get\_params()

Output:

{'ccp\_alpha': 0.0,

'criterion': 'mse',

'max\_depth': 8,

'max\_features': None,

'max\_leaf\_nodes': None,

'min\_impurity\_decrease': 0.0,

'min\_impurity\_split': None,

'min\_samples\_leaf': 0.1,

'min\_samples\_split': 2,

'min\_weight\_fraction\_leaf': 0.0,

'presort': 'deprecated',

'random\_state': 1,

'splitter': 'best'}

# Define grid of hyperparameters

params\_dt = {'max\_depth': [4, 6, 8],

'min\_samples\_leaf': [0.1, 0.2],

'max\_features': ['log2', 'sqrt']}

# Re-intantiate Decision Tree Regressor for cross validation

dt = DecisionTreeRegressor()

# Instantiate GridSearch cross validation

dt\_cv = GridSearchCV(estimator=dt,

param\_grid=params\_dt,

scoring='neg\_mean\_squared\_error',

cv=5,

verbose=1,

n\_jobs=-1)

# Fit model to

dt\_cv.fit(X\_train, y\_train)

Output:

Fitting 5 folds for each of 12 candidates, totalling 60 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 26 tasks | elapsed: 3.4s

[Parallel(n\_jobs=-1)]: Done 60 out of 60 | elapsed: 3.5s finished

GridSearchCV(cv=5, estimator=DecisionTreeRegressor(), n\_jobs=-1,

param\_grid={'max\_depth': [4, 6, 8],

'max\_features': ['log2', 'sqrt'],

'min\_samples\_leaf': [0.1, 0.2]},

scoring='neg\_mean\_squared\_error', verbose=1)

# Print best parameters

print('Best parameters for this Decision Tree Regressor model: {}'.format(dt\_cv.best\_params\_))

Output:

Best parameters for this Decision Tree Regressor model: {'max\_depth': 8, 'max\_features': 'log2', 'min\_samples\_leaf': 0.1}

# # Generate model best score

print('Best score for this Decision Tree Regressor model: {:.3f}'.format(dt\_cv.best\_score\_))

Output:

Best score for this Decision Tree Regressor model: -0.137

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

With a low Mean Squared Error = 0.118 we have a decent model with a high accuracy of prediction.

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

"Overfitting is one of the major problem for every model in machine learning. If model is overfitted it will poorly generalize to new samples. To avoid decision tree from overfitting we remove the branches that make use of features having low importance. This method is called as Pruning or post-pruning. Also, we need to be careful with parameter tuning as learned trees can manifest biases if certain classes dominate (Yadav, 2019).

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

It is vital that stakeholders within the company understand the predictor variables used in the analysis created a relatively low mean squared error with results showing 0.118. The recommended course of action would be to analyze the features that are appear in common with those leaving the company and attempt to reduce the probability of this happening in the future. Offering to improve services such as replacements or increasing online backup can improve the customer’s experiences.

Part VI: Demonstration

1. Video Link: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=7bcba444-d864-496e-bc59-ae2f00668ff3> s
2. Record the web resources used to acquire the data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

Dennis, T. (2019, July 25). Confusion Matrix Visualization. Medium. <https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea>

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

*An introduction to seaborn*. - seaborn 0.11.2 documentation. (n.d.). Retrieved November 20, 2021, from <https://seaborn.pydata.org/introduction.html>

Frankenfield, J. (2021, May 19). Churn rate. Investopedia. Retrieved October 24, 2021, from <https://www.investopedia.com/terms/c/churnrate.asp>

Kawerk, E. (n.d.). *Decision tree for classification: Python*. campus.datacamp.com. Retrieved February 1, 2022, from <https://campus.datacamp.com/courses/machine-learning-with-tree-based-models-in-python/classification-and-regression-trees?ex=1>

Matplotlib - PyLab module. (n.d.). Retrieved November 20, 2021, from <https://www.tutorialspoint.com/matplotlib/matplotlib_pylab_module.htm>

Scikit learn - introduction. (n.d.). Retrieved November 20, 2021, from <https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.html>

*What is NumPy?*.- NumPy v1.21 Manual. (n.d.). Retrieved November 20, 2021, from <https://numpy.org/doc/stable/user/whatisnumpy.html>

*What is Pandas in python? Educative.io. (n.d.). Retrieved November 20, 2021, from* [*https://www.educative.io/edpresso/what-is-pandas-in-python*](https://www.educative.io/edpresso/what-is-pandas-in-python)

Yadav, P. (2019, September 23). Decision tree in machine learning. Medium. Retrieved February 1, 2022, from <https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96>

Yse, D. L. (2019, April 30). The Complete Guide to Decision Trees. Medium. Retrieved January 31, 2022, from <https://towardsdatascience.com/the-complete-guide-to-decision-trees-28a4e3c7be14>