­

**Performance Assessment for D209: Data Mining I  
Task 1: Classification**

Christopher Fischer - 011933891

College of Information Technology, Western Governors University

July 12, 2024

Performance Assessment for D209: Data Mining I – Task 1

This document contains the tasks and outputs required for the “NVM4 TASK 1: CLASSIFICATION ANALYSIS” assessment. All work is original to the author unless otherwise indicated by a citation.

# A - Research Question

To increase the perception of our company in the marketplace as a technology leader, we have designed new service offerings aimed at the tech-savvy consumer. However, we do not yet have a reliable means of identifying which potential customers fit this category.

With this motivation, the data analytics team has been given the budget and scope to perform a study to determine if it is possible to predict which customers are most likely to self-identify as tech-savvy. Armed with a helpful classification model, management can identify tech-savvy customers as they work through the onboarding process and offer our new high-tech services to them.

This effort aims to build a model to classify customers according to their tech-savviness. Management hopes that such a model can be deployed as part of our customer onboarding process so that they can be offered our newest, most exciting service options.

# B – Method Justification

Predicting whether a certain outcome will be true or false is a classic binary classification exercise and one of the most common problems solved with supervised machine learning. This is exactly the type of question this study aims to answer: Is an incoming customer tech-savvy? The team has selected K-Nearest Neighbors (KNN) from several widely used classification algorithms for this study. KNN finds *k* observations with the most similar features for a given observation and then selects the class to which most of those observations belong. This class is then assigned to the observation being modeled. (Bruce, Bruce, & Gedeck, 2020) In this study, the KNN model is expected to return true/false values indicating if an observation is for a customer likely to be skilled with technology.

KNN is a simple technique that measures the distance between feature values in Euclidean space rather than deriving coefficients for an equation. However, KNN does require that predictor variables be numeric. Therefore, our data preparation plan must include appropriate steps to ensure that selected predictor variables will be expressed numerically.

The data science team will develop the code for this study using Python. In its nearly 35 years, Python has become one of the most widely used general-purpose programming languages. Python is backed by a massive community of developers and a vast collection of libraries that extend the core capabilities of the language. (Datacamp, 2022) Another reason for selecting Python is that, while R is an excellent choice for interactive studies, Python is better suited for deployment on production servers as part of a data pipeline. (WGU Information Technology, n.d.)

The following table lists the Python libraries which will support this study.

|  |  |
| --- | --- |
| Module | Purpose |
| Pandas | Provides DataFrame used for data management |
| Missingno | Displays a graphic of missing data used in the cleaning process |
| Matplotlib.pyplot | Tools for visualizing data and results |
| Seaborn | Additional visualization capabilities |
| Sklearn.preprocessing | Implements scaling and one-hot encoding |
| Sklearn.compose | Framework for applying multiple data transformation steps |
| Sklearn.model\_selection | Supports splitting datasets into training and test sets, as well as cross-validation methods |
| Sklearn.neighbors | Implementation of the K-Nearest Neighbors modeling algorithm |
| Sklearn.metrics | Provides several metrics and tools for evaluating model performance |

Table 1 - Python Libraries to be Used

# C – Data Preparation

The following sections outline the team’s plan to clean and prepare the input data for this study.

## C1 – Pre-processing Goal

As mentioned above, KNN requires all predictor variables to be expressed in numerical values. Furthermore, since KNN is a distance-oriented algorithm, predictors with disproportionately large values will tend to skew the results. (Bruce, Bruce, & Gedeck, 2020) Therefore, all numerical values will be standardized to prevent this unintended behavior.

## C2 – Selected Predictors

| Variable Name | Data Type |
| --- | --- |
| Age | Numeric |
| Income | Numeric |
| Children | Numeric |
| Population | Numeric |
| State | Categorical |
| Area | Categorical |
| Marital | Categorical |
| Gender | Categorical |
| Port\_modem | Categorical |
| Tablet | Categorical |
| Phone | Categorical |
| Multiple | Categorical |
| OnlineSecurity | Categorical |
| OnlineBackup | Categorical |
| DeviceProtection | Categorical |
| TechSupport | Categorical |
| StreamingTV | Categorical |
| StreamingMovies | Categorical |

Table 2 - Study Predictor Variables

## C3 – Data Preparation Steps and Results

# Check missing data

msno.matrix(df[X\_full], fontsize = 12, labels=True)

plt.title('Missing Data Matrix')

plt.show()

A black and white grid

Description automatically generated with medium confidence

Figure 1 - Missing Data Plot

No variables exhibit missing data.

#Check values in yes/no variables

print(df[yes\_no\_variables][~df[yes\_no\_variables].isin(['Yes','No'])].count())

A screen shot of a computer

Description automatically generated

Boolean variables all contain valid values.

#Detect potential outliers

df\_z = (df[numerical\_variables] - df[numerical\_variables].mean())/df[numerical\_variables].std(ddof=0)

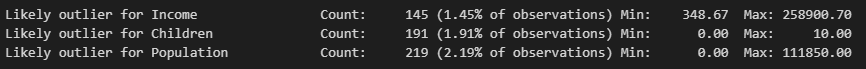
outlier\_cols = df\_z.loc[: , (df\_z > 3.0).any()].columns

for col in outlier\_cols :

    cnt = len(df\_z[df\_z[col]>3])

    min, max = df[col].min(), df[col].max()

    print('Likely outlier for {0:<20}\t Count: {1:7d} ({2:5.2%} of observations)\tMin: {3:>9.2f}\tMax: {4:>9.2f}'.format(col,cnt,cnt/10000,min,max))



# Deeper look at Population

sns.boxplot(data=df, x='Population')

plt.show()

print(df['Population'].value\_counts())

A graph of a number of people

Description automatically generated

Figure 2 - Boxplot of Population

While a small percentage of values exceed the norms for the variables *Income* and *Children*, none is definitively invalid. Therefore, these values will be retained in the study. There are several observations with an invalid value (zero) for *Population*. These will be removed from consideration.

# Drop rows with Population <= 0

df = df[df.Population > 0]

print(df['Population'].count())

9903

# Change "Never Married" to "NeverMarried" to prevent one-hot encoding problems later

df['Marital'] = df['Marital'].replace('\s+', '',regex=True)

print(df['Marital'].value\_counts())  
Marital

Divorced 2067

Widowed 2013

Separated 1990

NeverMarried 1935

Married 1898

Name: count, dtype: int64

# Reexpress yes/no columns as numbers

yesno\_dict = {'No': 0, 'Yes': 1}

for col in yes\_no\_variables:

    df[col] = df[col].map(yesno\_dict)

print(df[yes\_no\_variables].info())  
Index: 9903 entries, 0 to 9999

Data columns (total 11 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Techie 9903 non-null int64

1 Port\_modem 9903 non-null int64

2 Tablet 9903 non-null int64

3 Phone 9903 non-null int64

4 Multiple 9903 non-null int64

5 OnlineSecurity 9903 non-null int64

6 OnlineBackup 9903 non-null int64

7 DeviceProtection 9903 non-null int64

8 TechSupport 9903 non-null int64

9 StreamingTV 9903 non-null int64

10 StreamingMovies 9903 non-null int64

# Rexpress State with frequency encoding

df['State'] = df['State'].map(df['State'].value\_counts().to\_dict())

print(df['State'].info())

Series name: State

Non-Null Count Dtype

-------------- -----

9903 non-null int64

dtypes: int64(1)

# Rexpress Area with simple ordinal encoding

area\_map = {'Rural': 1, 'Suburban': 2, 'Urban' : 3}

df['Area'] = df['Area'].map(area\_map)

print(df['Area'].info())

Series name: Area

Non-Null Count Dtype

-------------- -----

9903 non-null int64

dtypes: int64(1)

# Scaling and One-hot encoding

preprocessor = make\_column\_transformer(

 (StandardScaler(), numerical\_variables),

 (OneHotEncoder(), onehot\_variables),

  remainder='passthrough',

  verbose\_feature\_names\_out=False

)

X = pd.DataFrame(preprocessor.fit\_transform(df[X\_vars]), columns=preprocessor.get\_feature\_names\_out())

y = df[y\_vars]

## C4 – Prepared Data File

This file contains the cleaned and prepared data used in subsequent modeling activities.



# D – Analysis

The following sections outline the steps taken to build the model desired for this study.

## D1 – Train/Test Split

One of the dangers analysts face in building machine learning models is that the algorithms can easily go too far in fitting the training data. In other words, rather than learning a general model for accomplishing the desired prediction, the model, in essence, memorizes the training data. The analyst should partition the available data into training and testing sets to avoid this. Doing so provides significant unseen data to test the model's performance built on the training data. (Massaron & Boschetti, 2016) Accordingly, the team will split the provided dataset into 80% training and 20% testing.

# Split data into train and test sets with 80/20 split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5470, stratify=y)

X\_train.to\_csv('churn\_X\_train.csv', index=False)

X\_test.to\_csv('churn\_X\_test.csv', index=False)

y\_train.to\_csv('churn\_y\_train.csv', index=False)

y\_test.to\_csv('churn\_y\_test.csv', index=False)

These attachments contain the partitioned predictor and target data.



## D2/D3 – Modeling Technique and Code

As mentioned in [B – Method Justification](#_B_–_Method), the team has selected K Nearest Neighbors (KNN) as the machine learning algorithm for this study. While KNN is an excellent algorithm, finding the best parameters to configure the algorithm requires further consideration. Failing to do so will limit the algorithm’s ability to produce the model with the best performance. (Bruce, Bruce, & Gedeck, 2020) Additionally, given the relatively small dataset (*n*=10,000), a cross-validation method must be employed to ensure that our model is as accurate on unseen data (test) as on seen data (training). (Alpaydin, 2014)

The *GridSearchCV* method will provide both hyperparameter tuning and cross-validation for our study. The only hyperparameter we will tune is *n\_neighbors*, which controls the value of *k* in K Nearest Neighbors. For cross-validation, we will select a typical 5-fold arrangement. With 5-fold cross-validation, the dataset will be split five different ways, with each iteration containing 20% of the data for testing and 80% for training. In this way, every observation in the dataset will be used for training four times and testing one time. Last, the model which produced the optimal scoring metric will be labeled “best”. For this study, the *f1 score* will be the metric used. *F1* attempts to reflect a blend of both precision and recall. It is a good choice for cases where false positives and negatives are considered equally problematic. (Agrawal, 2024)

A pink and black text with yellow squares

Description automatically generated with medium confidence

Figure 3 - Five-fold Cross-validation (source: DataCamp)

Here is the code and results for creating and fitting the KNN model.

# Set-up hyperparameter values

neighbors = list(range(1, 31))

parameters = dict(n\_neighbors = neighbors)

# Set-up base model

knn = KNeighborsClassifier()

# Set-up cross validation

clf = GridSearchCV(knn, parameters, cv=5, scoring='f1', verbose=1, return\_train\_score=False, n\_jobs=-1)

# Fit the model

clf.fit(X\_train,y\_train.values.ravel())

clf\_best = clf.best\_estimator\_

# Print results

print('Best score : ' , clf.best\_score\_)

print('Best params : ' , clf.best\_params\_)

print(clf\_best.get\_params())

Fitting 5 folds for each of 30 candidates, totalling 150 fits

Best score : 0.14306419595615444

Best params : {'n\_neighbors': 1}

{'algorithm': 'auto', 'leaf\_size': 30, 'metric': 'minkowski', 'metric\_params': None, 'n\_jobs': None, 'n\_neighbors': 1, 'p': 2, 'weights': 'uniform'}

# E – Data Summary and Implications

Unfortunately, the results of this study are disappointing. Here are some performance metrics achieved by the “best” model.

A black and white screen with white text

Description automatically generated

Figure 4 - Model Performance Metrics

This shows that the model accurately identified true positives and negatives 71% of the time. Even more alarming are the exceedingly low precision and recall scores. These show that the model did an abysmal job of avoiding false positives and negatives.

Looking at the ROC curve, which plots the true positive rate against the false positive rate, the story does not get any more encouraging. The curve demonstrates that the model is no better at predicting the outcome than random guessing.

A graph of a positive rate

Description automatically generated with medium confidence

Figure 5 - ROC AUC Curve

One limitation of this study is that no feature elimination was undertaken. If additional exploratory data analysis were performed, the set of features used to train the model could have been reduced to exclude those features with limited correlation to the target variable.

As the results of this model are no better than random coin flips, the model will not provide the guidance hoped for. Should the business stakeholders wish to continue this effort, steps should be taken to gather additional demographic and behavioral data pertaining to current and potential customers. Then, this study could be revised to incorporate the additional features and implement one or more feature elimination techniques. Combined, these steps should provide a better outcome than this study achieved.

# F – Recorded Code Review

A recording of the code review presentation was uploaded with this submission. For quick reference, that video may be found here: [Panopto Recording](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=dfc98f20-a6bd-433f-a430-b1af00ed5bf3)

This Python notebook contains all the code used for this study.



# G – Third-Party Code References

Buser, A. (2023, 04 24). *A Beginner’s Guide to using sklearn make\_column\_transformer*. Retrieved from Medium: https://medium.com/@buser.andre/a-beginners-guide-to-using-sklearn-make-column-transformer-4ac9a7f78fb3

Eiler, J. (2017, 01 16). *Remap values in pandas column with a dict, preserve NaNs*. Retrieved from Stackoverflow: https://stackoverflow.com/questions/20250771/remap-values-in-pandas-column-with-a-dict-preserve-nans

MacPhee-Cobb, L. (2016, 03 20). *A column-vector y was passed when a 1d array was expected*. Retrieved from StackOverflow: https://stackoverflow.com/questions/34165731/a-column-vector-y-was-passed-when-a-1d-array-was-expected

Neural Ninja. (2023, 06 12). *Frequency Encoding: Counting Categories for Representation*. Retrieved from Let's Data Science: https://letsdatascience.com/frequency-encoding/

Piepenbreier, N. (2022, 02 23). *One-Hot Encoding in Scikit-Learn with OneHotEncoder*. Retrieved from Datagy: https://datagy.io/sklearn-one-hot-encode/

Ray, S. (2020, 10 21). *Show all colums of a pandas-dataframe in ".describe()"*. Retrieved from Stackoverflow: https://stackoverflow.com/questions/64455605/show-all-colums-of-a-pandas-dataframe-in-describe

# H – Referenced Works

Agrawal, S. (2024, 06 05). *Metrics to Evaluate your Classification Model to take the Right Decisions*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/

Alpaydin, E. (2014). *Introduction to Machine Learning.* MIT Press. Retrieved 07 12, 2024, from https://ebookcentral.proquest.com/lib/westerngovernors-ebooks/detail.action?docID=3339851

Bruce, P., Bruce, A., & Gedeck, P. (2020). *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python.* O'Reilly Media, Incorporated. Retrieved 07 12, 2024, from https://ebookcentral.proquest.com/lib/westerngovernors-ebooks/detail.action?docID=6173908

Datacamp. (2022). *Python vs R for Data Science: Which Should You Learn?* Retrieved 03 22, 2024, from Datacamp Blog: https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference

Massaron, L., & Boschetti, A. (2016). *Regression Analysis with Python : Discover Everything You Need to Know About the Art of Regression Analysis with Python, and Change How You View Data.* Packt Publishing. Retrieved 07 12, 2024, from https://eds.p.ebscohost.com/eds/ebookviewer/ebook?sid=cd9d41a8-0c80-4baf-bad5-57a9c7381229%40redis&ppid=pp\_29&vid=0&format=EB

WGU Information Technology. (n.d.). *R or Python*. Retrieved 03 22, 2024, from https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html