|  |
| --- |
| Western Governors University |
| Data Mining 1 |
| D209 Task 1 |

|  |
| --- |
| Allison Casey  7-20-2024 |

**Part I: Research Question**

**A1: PROPOSAL OF QUESTION**

Can customers at risk of churn be identified using the classification method, k-nearest neighbor (KNN)?

**A2: DEFINED GOAL**

The main goal of this analysis is to develop and use a KNN model to help identify customers at risk of churn so that steps could be taken to help prevent the churn of these customers. Along with this overall goal, the analysis should help identify features that cause customer churn.

**Part II: Method Justification**

**B1: EXPLANATION OF CLASSIFICATION METHOD**.

k-NN analyzes the data by comparing an un-classified data point to the nearest or most similar classified data points. The observation is then classified based on the majority classification as those it was compared against. An expected outcome of this would be, to a certain degree of accuracy, that if the nearest k number of data points to an un-classified data point have churned, then this data point will have churned as well and vice versa.

**B2: SUMMARY OF METHOD ASSUMPTION**

One assumption of the k-NN classification is that similar things exist in proximity to each other. If this assumption is not true, then the algorithm will not be useful.

**B3: PACKAGES OR LIBRARIES LIST**

|  |  |
| --- | --- |
| import os,sys | Used to set the directory to work out of |
| import pandas as pd | Used to import the data |
| import numpy as np | Used for array handling and capabilites |
| from sklearn.feature\_selection import SelectKBest | Used to help determine the best features to use for the model |
| from sklearn.model\_selection import train\_test\_split | For splitting the data into train and test sets |
| from sklearn.preprocessing import StandardScaler | To scale the features |
| from sklearn.model\_selection import GridSearchCV | To help get the best hyperparameters |
| from sklearn.neighbors import KNeighborsClassifier | To instantiate the KNN model |
| from sklearn.metrics import confusion\_matrix, roc\_auc\_score | To build the confusion matrix and calculate the ROC AUC score |

**Part III: Data Preparation**

**C1: DATA PREPROCESSING**

One of the pre-processing goals for the data was to check for missing or null values in the data set and treat any if found.

**C2: DATA SET VARIABLES**

|  |  |
| --- | --- |
| **Feature** | **Data Type** |
| Churn | Categorical |
| MonthlyCharge | Numeric |
| Tenure | Numeric |
| Bandwidth\_GB\_Year | Numeric |
| StreamingMovies | Categorical |
| Contract | Categorical |
| Multiple | Categorical |
| InternetService | Categorical |
| DeviceProtection | Categorical |
| OnlineBackup | Categorical |
| PaymentMethod | Categorical |
| Phone | Categorical |

**C3: STEPS FOR ANALYSIS**

1. Detect and treat missing values

*df.info()*

*#Detect and treat NULLS - we can see from the describe call that there is only one column with missing values*

*#InternetService: This column actually doesn't have any null values it just got read in as NULL rather than None indicating the customer doesn't have this service*

*df['InternetService'].fillna('None',inplace=True)*

1. Drop the initial columns that won’t be used (demographic data and survey questions)

*df = df.drop(columns=['Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'TimeZone', 'Area', 'Job', 'Children', 'Age', 'Income', 'Marital', 'Gender', 'Techie', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'])*

1. Review summary statistics for outliers or other concerns

*df.describe()*

*categorical\_columns = ['Churn','Port\_modem', 'Tablet', 'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'Contract', 'InternetService', 'PaymentMethod']*

*for x in categorical\_columns:*

*print(df[x].describe())*

1. One hot encode the categorical variables and rename the columns for clarity

*#One hot encoding*

*categorical\_columns\_yn = ['Churn','Port\_modem', 'Tablet', 'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling']*

*categorical\_columns\_multiple = ['Contract', 'InternetService', 'PaymentMethod']*

*#One hot encode the yes/no columns*

*df = pd.get\_dummies(df, columns=categorical\_columns\_yn, drop\_first=True, dtype = int)*

*#one hot encode the categorical columns with multiple labels so that none get dropped*

*df = pd.get\_dummies(df, columns=categorical\_columns\_multiple, drop\_first=False, dtype = int)*

*#Rename the columns for clarity and to remove spaces*

*df = df.rename(columns = {'Contract\_One year':'Contract\_One\_Year', 'Contract\_Two Year':'Contract\_Two\_Year','InternetService\_Fiber Optic':'InternetService\_Fiber\_Optic', 'InternetService\_Fiber Optic':'InternetService\_Fiber\_Optic', 'PaymentMethod\_Credit Card (automatic)':'PaymentMethod\_CC', 'PaymentMethod\_Electronic Check':'PaymentMethod\_ECheck', 'PaymentMethod\_Mailed Check':'PaymentMethod\_MCheck', 'PaymentMethod\_Bank Transfer(automatic)':'PaymentMethod\_Bank\_Transfer'})*

*df.info()*

1. Use SelectKBest to narrow down the features to try and get the best ones to use for the model

*#Use SelectKBest to further narrow down the features to get the best ones for the model*

*X = df.drop('Churn\_Yes', axis=1)*

*y = df["Churn\_Yes"]*

*feature\_names = X.columns*

*# Initialize the class and call fit\_transform*

*skbest = SelectKBest(k='all') # k= features*

*X\_new = skbest.fit\_transform(X, y)*

*X\_new.shape*

*### Finding P-values to select statistically significant features*

*p\_values = pd.DataFrame({'Feature': X.columns, 'p\_value':skbest.pvalues\_}).sort\_values('p\_value')*

*p\_values[p\_values['p\_value'] < .05]*

*features\_to\_keep = p\_values['Feature'][p\_values['p\_value'] < .05]*

*# Print the name of the selected features*

*print(features\_to\_keep)*

*df = df.drop(columns = ['PaymentMethod\_Bank\_Transfer','Population','Outage\_sec\_perweek','Email','Contacts','Tablet\_Yes','Yearly\_equip\_failure','OnlineSecurity\_Yes','PaperlessBilling\_Yes','TechSupport\_Yes','Port\_modem\_Yes','PaymentMethod\_CC','PaymentMethod\_MCheck'])*

*df.info()*

1. Scale the numeric values for knn

*Initialize StandardScaler*

*scaler = StandardScaler()*

*# Apply StandardScaler to numeric columns*

*features = ['Tenure','MonthlyCharge', 'Bandwidth\_GB\_Year']*

*for x in features:*

*df[x] = scaler.fit\_transform(df[x].to\_numpy().reshape(-1,1))*

1. Export the prepared data

*df.to\_csv('PREPARED\_churn\_clean\_data.csv')*

See D209\_Task1.ipynb for the full code

**C4: CLEANED DATA SET**

PREPARED\_churn\_clean\_data.csv

**Part IV: Analysis**

**D1: SPLITTING THE DATA**

Data split and with a test size of .2:

*# Set up the data as X and y and split the data for training and testing*

*X = df.drop('Churn\_Yes', axis=1)* *# Set up the data as X and y and split the data for training and testing*

*X = df.drop('Churn\_Yes', axis=1)*

*y = df['Churn\_Yes']*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1, stratify=y)*

*# Export training and test datasets*

*X\_train.to\_csv('X\_train.csv')*

*X\_test.to\_csv('X\_test.csv')*

*y\_train.to\_csv('y\_train.csv')*

*y\_test.to\_csv('y\_test.csv')*

See the following files for the exported split data:

X\_train.csv

X\_test.csv

y\_train.csv

y\_test.csv

see D209\_Task1.ipynb for full code

**D2: OUTPUT AND INTERMEDIATE CALCULATIONS**

The analysis technique that was used on the data was KNN which is used to predict the label of a data point based on the k closest neighbors. GridSearchCV was used to calculate the best number of neighbors to use with a range of 1 to 30. This way multiple values will get set and the value for k(n-neighbors) with the best performance can be determined and used in the final model.

As per the previous step the data was split and then a parameter grid was set up to determine the best number of neighbors with GridSearchCV with a range of 1 to 30:

**A screen shot of a computer code

Description automatically generated**

Using this parameter setting the model was built and fit to the training data:

A screenshot of a computer program

Description automatically generated

Then the model was used to predict with the test data so that a confusion matrix could be generated, it could be tested for accuracy, and get the AUC score to validate whether the model is useful:

A screenshot of a computer program

Description automatically generated

**D3: CODE EXECUTION**

D209\_Task1.ipynb

**Part V: Data Summary and Implications**

**E1: ACCURACY AND AUC**

The accuracy of the model is the correct classifications divided by the total samples. This was calculated for both the overall model performance using the test data as well as on the training data.

* Model Accuracy: .886 or 88.6%
* Training Accuracy: .90125 or 90.125%

The AUC score is the area under an ROC curve with a value from 0 to 1 where a higher score indicates that the model is better at distinguishing between positive and negative classifications. So, in this case whether the model is effective at distinguishing between a customer churning and not churning. An AUC score of 0 would indicate the model is 100% wrong, .5 would be the equivalent of random guesses, and 1 would indicate that the model is 100% correct.

* AUC score: 0.94

**E2: RESULTS AND IMPLICATIONS**

This classification model can predict customer churn with approximately 88% accuracy and the AUC score of .94 indicates that the model does a significantly better job at distinguishing whether a customer will churn than randomly guessing. As a result, the model can be used to predict churn with some statistical certainty.

**E3: LIMITATION**

One limitation of the analysis would be the data set itself. In an ideal situation the data would be evenly distributed for churn, but as it is the data is skewed in favor of non-churn customers with there being less data for customers that churned. This could result in the model classifying churn incorrectly.

**E4:** **COURSE OF ACTION**

Based on the AUC and accuracy scores the company can use this model to help predict customers who may be at risk of churn. However, it would also be advisable to try to continue to improve the model and data. Gathering more data and having more data to test and train the model against could help make it better. The model could likely also be improved by doing more hyper-parameter tuning on parameters that weren’t tuned in this model. It would also be smart to use the model to dig deeper into exploring which features contribute to churn as well as potentially building more models that consider the demographic data that was left out of this model.

**Part VI: Demonstration**

**F: PANOPTO RECORDING**

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=42f05532-79f2-48df-9cd7-b1b80015b7fa>

**Sources**

Elleh, Festus. “Task 1 Classification.” Data Mining 1 . 2024.

“Kneighborsclassifier.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. Accessed 20 July 2024.

“Roc\_auc\_score.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.metrics.roc\_auc\_score.html. Accessed 20 July 2024.

“What Is the K-Nearest Neighbors Algorithm?” *IBM*, 4 Oct. 2021, www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20(KNN,used%20in%20machine%20learning%20today.