Andreas Sebastian Dunn
Western Governors University, D209: Data Mining
November 2, 2021

D209, Task 1

Part 1

Using the k-nearest neighbor algorithm, can we discover how certain X variables influence customer churn in the telecom industry? In this paper I will discuss the use case for k-nearest neighbors as a way to predict customer churn, a binary categorical variable.

Our goal is to use the telecom data and run a k-nearest neighbors algorithm and some testing on the algorithm to see how strong our predictor variables are and whether they can predict customer churn.

Part 2

Using sklearn for Python we can use KNN to model how our independent variables may predict customer churn. KNN works by looking at the "closest" neighboring data points and assigns a label to a new data point depending on how close that new data point is to existing data. It draws a decision boundary around the modeled data, which are called clusters, and any new data point is labeled depending on where it falls relative to those clusters.

For the purposes of this model we will split our data set into two groups, the training group and the test group, in order to see how well our model works. The expected outcome from the model is a prediction on whether a customer will churn or not given how well our model can predict new Y values from new X values.

Using the KNN algorithm we can use some of our data in a training set, fit the data to the KNN model, and get an output. Then we input the test set into the model and get our predicted value for Y, in this case, the categorical variable of churn.

KNN assumes that data points will naturally cluster together and therefore we can use the proximity to these clutters as a way to predict a new value of y. It also assumes that the underlying distribution of the data is unknown, as opposed to regression which assumes it is normally distributed.

In this paper I will be using python and the available libraries that come loaded in the anaconda package: scikit learn, numpy, pandas, etc.

Part 3

The most import processioning goal in machine learning is making sure your data set is clean and not missing any values, which I have done shown in the code attached to this paper. After we make sure our values are all clean and not missing any data we can move onto modeling. We then perform the KNN and find the ideal amount of k. Lower values of k are associated with over fitting and higher values of k with under fitting. Higher k means the machine must use more computation power and the model is considered less complex than lower k. We can use a model complexity curve to show how accurate our training set is versus the test set and find an ideal amount of k.

We will be using 3 independent variables to determine customer churn. Those features are MonthlyCharge, Bandwidth_GB_Year, and Outage_sec_perweek which are all continuous variables.

My steps include importing the proper libraries into the python console and loading the .csv file into a pandas dataframe. Then I will save our features and target to their respective variables, encode our target variable to a dummy, call the appropriate sklearn methods to split and train/test the data and interpret the results.

The data set is provided.

Part 4

We start with our already clean and labeled dataset and import that into our pandas dataframe. In order for KNN to work, we need to split our data into training and test sets. The reason being is that we need our model to have data to train with and then we need to test that training with the rest of our data. Usually for KNN the split is 80-20, 80% for training and 20% for testing.

After we have tested our data, we can now use the KNN model and predict a new value of Y from a new value of X and attach the appropriate label to it. The code is provided with this paper.

Part 5

As is shown in the python code, our R^2 is 0.818 which is a pretty good score to explain how well our model can predict on new data. However we need to go further than R^2 because it may be biased by the way we split our training and test sets. To measure our model's performance, we have to find a good value of k. We enumerate over several values of k to find an ideal number in python. As can be shown in the model complexity curve, our ideal number of k appears to be around 7-8.

The implications are as follows: given a new X value, our model can, with 85% accuracy, classify a Y value. In this case it is whether a customer will churn or not. Our output is a new column of Y with 0 for No to churn and 1 for Yes to churn. Next we examine the AUC which is represented by the

ROC score. The ROC score shows a value of 0.86 which is very good. ROC is a measurement of how well our model can distinguish between classes so that it can apply the correct label. The higher the score is to 1 the better the performance of our model. The AUC-ROC curve is way to measure the performance of our classification model: it tells use how well our model can correctly classify new information.

Some limitations of KNN are that it is not the most complex classification or prediction algorithm. In order for it to behave well with new data it needs to be trained a lot which can be computationally expensive. It also does not work on anything other than continuous variables however in our case a simple recode of the target variable is a simple solution.

With our training and test sets complete and showing a good accuracy level and ROC score, it is clear that our classification algorithm should work well at predicting customer churn with an 85% accuracy. My recommendation based out this exercise is for the industry to focus on narrowing down the reasons why higher costs, usage, and outages are a good measurement of customer churn. It may be that customers who decide to leave for a different ISP are clustered together due to higher costs or usage levels.

All sources and code are from course videos and lectures or official docs:
https://scikit-learn.org/stable/
https://pandas.pydata.org/docs/
https://numpy.org/doc/