Andreas Sebastian Dunn

Western Governors University, D209: Data Mining

November 7, 2021

D209, Task 2

Part 1

In this paper I will use ridge regression to explore the customer churn data from the telecom industry in an attempt to show the relationship between customer demographics and churn. Ridge regression is similar to linear regression with one key difference; ridge regression uses a hyper parameter called lambda to 'penalize' large coefficients. Large beta (coefficients) can lead to linear regression becoming too sensitive to new inputs and therefore using lambda to control beta can help in tuning a regression model for optimal performance.

The goal of this analysis is to measure how the features of MonthlyCharge, Bandwidth_GB_Year, and Outage_sec_perweek impact customer churn. All our features are continuous variables and our target is a categorical variable that will be recoded into a dummy variable in order for us to estimate it using ridge regression.

Part 2

Using sklearn for Python we can use ridge regression to model how our independent variables may predict customer churn. Multiple linear regression is an algorithm designed to predict Y based on the slope of our features variables by using the OLS method. The OLS method seeks to minimize the sum of squared residuals in order to fit a trend line, that trend line is then used as the basis for predicting a new target.

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 regression algorithm we can use some of our data in a training set, fit the data to the 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.

Ridge regression assumes that the underlying distribution is normally distributed, there is a linear relationship between X and Y, and that features are independent of one another. Linear regression through OLS and ridge regression are essentially the same except that ridge regression has an additional hyper parameter called alpha, that is used to "regularize" the coefficients so that the model does not get skewed by big coefficients. Big coefficients can over fit the regression line so penalizing a model with alpha is desirable. We can standardize all our features so that they are all on the same scale by increasing our level of alpha, also known as lambda, which will decrease the weight of big coefficients.

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 regression is making sure our data is clean and does not have any missing data. We can check this using built in python methods and is shown in the code provided with this paper. Then we split our data into a training set and test set to ensure that our model is accurate when it tests new values of X on predicting Y. We also need to make sure our target variable is numeric and not categorical so we create a dummy variable out of our response.

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 us to evaluate our model 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. For ridge regression I have split the data 70-30; 70% of the data is used to train the model and 30% is used to test that training.

[1]:
```
import pandas as pd
import numpy as np
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt

df = pd.read_csv(r'C:\Users\dre2\Desktop\WGU\D208\churn_clean.csv')
```

[2]:
```
df['churn_dummy'] = [1 if v == 'Yes' else 0 for v in df['Churn']]

df = df[['Outage_sec_perweek', 'MonthlyCharge', 'Bandwidth_GB_Year', 'churn_dummy']]
```

[3]:
```
target_column = df['churn_dummy']
predictors = list(set(list(df.columns))-set(target_column))
df[predictors] = df[predictors]/df[predictors].max()
```

[4]:
```
X = df.iloc[:, 1:-1].values
y = df.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=40)
```

[5]:
```
rr = Ridge(alpha=0.01)
rr.fit(X_train, y_train)
pred_train_rr= rr.predict(X_train)

pred_test_rr= rr.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_rr)))
print(r2_score(y_test, pred_test_rr))
```

```
0.35103871730610103
0.3637303509289348
```

Part 5

After fitting the model and testing it we have our results: our R^2 is 0.36 suggesting that our model's independent variables only explain 36% of the variance in our target variable; our model is not very good at consistently predicting a new Y value. In other words, our R^2 isn't strongly correlating the relationship between X and Y. The mean-squared error is another metric of model performance that measures how much our errors deviate from each other. They are squared in order to become all positive numbers and then an average is taken of those to get an understanding of how our data is spread around the regression line. Our MSE is 0.35 which suggests that our trend line fits the data very closely.

What these metrics mean in practicality is that our 3 features, while having data that is very close to the trend line as represented by MSE, do not have a strong linear relationship between X and Y. Our regression line is fitted well but that line isn't very reliable due to the low r-squared.

The main limitation of this analysis is lack of relevant data points: if we had more data to use we could possibly increase r-squared and show a stronger relationship and therefore have stronger predictive power. Another limitation is that OLS works great with linear relationship and perhaps the relationship between our target and features isn't linear and therefore our model performance is weak.

Based on the results of this model I would suggest that the telecom company gather more feature variables in order to increase r-squared so that some stronger predictions can be made. For this analysis, our features do not explain enough variation in the target variable suggesting more features need to be selected or better features need to be selected. While our MSE is low suggesting that our errors are close to the trend line, the trend line itself isn't a reliable source of demonstrating a strong linear relationship.

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/