D206_Performance_Assessment_revision

May 3, 2021

1 Performance Assessment | D206 Data Cleaning

Ryan L. Buchanan Student ID: 001826691 Masters Data Analytics (12/01/2020) Program Mentor: Dan Estes (385) 432-9281 (MST) rbuch49@wgu.edu

1.1 Part I: Research Question

1.1.1 A. Question or Decision:

Can we determine which individual customers are at high risk of churn? And, can we determine which features are most significant to churn?

1.1.2 B. Required Variables:

The data set is 10,000 customer records of a popular telecommunications company. The dependent variable (target) in question is whether or not each customer has continued or discontinued service within the last month. This column is titled "Churn."
Independent variables or predictors that may lead to identifying a relationship with the dependent variable of "Churn" within the dataset include: 1. Services that each customer signed up for (for example, multiple phone lines, technical support add-ons or streaming media) 2. Customer account information (customers' tenure with the company, payment methods, bandwidth usage, etc.) 3. Customer demographics (gender, marital status, income, etc.).
4. Finally, there are eight independent variables that represent responses customer-perceived importance of company services and features.

The data is both numerical (as in the yearly GB bandwidth usage; customer annual income) and categorical (a "Yes" or "No" for Churn; customer job).

1.2 Part II: Data-Cleaning Plan

1.2.1 C1. Plan to Find Anomalies:

My approach will include: 1. Back up my data and the process I am following as a copy to my machine and, since this is a manageable dataset, to GitHub using command line and gitbash. 2. Read the data set into Python using Pandas' read_csv command. 3. Evaluate the data struture to better understand input data. 4. Naming the dataset as a the variable "churn_df" and subsequent useful slices of the dataframe as "df". 5. Examine potential misspellings, awkward variable naming & missing data. 6. Find outliers that may create or hide statistical significance using histograms. 7. Imputing records missing data with meaningful measures of central tendency (mean, median or mode) or simply remove outliers that are several standard deviations above the mean. *Referenced & paraphrased within above plan (Larose, 2019, p. 29-43).

1.2.2 C2. Justification of Approach:

Though the data seems to be inexplicably missing quite a bit of data (such as the many NAs in customer tenure with the company) from apparently random columns, this approach seems like a good first approach in order to put the data in better working order without needing to involve methods of initial data collection or querying the data-gatherers on reasons for missing information. Also, this the first dataset that I've clean, so I followed the procedures practice in the performance lab as well as tips from StackOverflow and other tutorial resources.

1.2.3 C3. Justification of Tools:

I will use the Python programming language as I have a bit of a background in Python having studied machine learning independently over the last year before beginning this masters program and its ability to perform many things right "out of the box" (Poulson, 2016, section 2). Python provides clean, intuitive and readable syntax that has become ubiquitous across in the data science industry. Also, I find the Jupyter notebooks a convenient way to run code visually, in its attractive single document markdown format, the ability to display results of code and graphic visualizations and provide crystal-clear running documentation for future reference. A thorough installation and importation of Python packages and libraries will provide specially designed code to perfom complex data science tasks rather than personally building them from scratch. This will include: NumPy - to work with arrays Pandas - to load datasets Matplotlib - to plot charts Scikit-learn - for machine learning model classes SciPy - for mathematical problems, specifically linear algebra transformations Seaborn - for high-level interface and atttractive visualizations

A quick, precise example of loading a dataset and creating a variable efficiently is using to call the Pandas library and its subsequent "read_csv" function in order to manipulate our data as a dataframe: import pandas as pd df = pd.read_csv('Data.csv')

1.2.4 C4. Provide the Code:

[ ]:
```
# Install necessary packages
!pip install pandas
!pip install numpy
!pip install scipy
!pip install sklearn
!pip install matplotlib
```

[ ]:
```
# Standard imports
import numpy as np
import pandas as pd
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[ ]:
```
# Load data set into Pandas dataframe
churn_df = pd.read_csv('churn_raw_data.csv')
```

[ ]:
```
# Display Churn dataframe
churn_df
```

[ ]:
```
# List of Dataframe Columns
df = churn_df.columns
print(df)
```

[ ]:
```
# Remove redundant "Unnamed" column at beginning & display first five records
df = churn_df.drop(churn_df.columns[0], axis = 1)
df.head()
```

[ ]:
```
# Rename last 8 survey columns for better description of variables
df.rename(columns = {'item1':'Responses',
                    'item2':'Fixes',
                     'item3':'Replacements',
                     'item4':'Reliability',
                     'item5':'Options',
                     'item6':'Respectfulness',
                     'item7':'Courteous',
                     'item8':'Listening'},
          inplace=True)
```

[ ]:
```
# Find number of records and columns of dataset
df.shape
```

[ ]:
```
# Describe Churn dataset statistics
df.describe()
```

[ ]:
```
# Remove less meaningful variables from statistics description
df_stats = df.drop(columns=['CaseOrder', 'Zip', 'Lat', 'Lng'])
df_stats.describe()
```

[ ]:
```
# Calculate Churn Rate
df.Churn.value_counts() / len(df)
```

[ ]:
```
# Review data types (numerical => "int64" & "float64"; & categorical =>␣
 ↪"object") in data set
df.dtypes
```

[ ]:
```
# Re-validate column data types and missing values
df.columns.to_series().groupby(df.dtypes).groups
```

[ ]:
```
# Find missing values
df.isnull()
```

[ ]:
```
# Access only rows from dataframe containing missing values
df.isnull().any(axis=1)
```

[ ]:
```
# Woah, lots of empty fields!  Immediately noticeable as "True" in columns of␣
 ↪"Children", "Age", "Income", "Techie", "Phone", "Tenure"
```

```
# Display the specific columns with NAs
df.isna().any()
```

[ ]:
```
# Confirm missing observations numbers
data_nulls = df.isnull().sum()
print(data_nulls)
```

[ ]:
```
# Store rows with missing values in a new variable
rows_with_missing_values = df.isnull().any(axis=1)
df[rows_with_missing_values]
```

[ ]:
```
# Examine columns for misspellings in categorical variables using unique()␣
 ↪method
df['Employment'].unique()
```

[ ]:
```
df['Area'].unique()
```

[ ]:
```
df['Timezone'].unique()
```

[ ]:
```
# Well then, how many unique jobs are there and will this variable help us out␣
 ↪much?
len(df['Job'].unique())
```

[ ]:
```
df['Children'].unique()
```

[ ]:
```
df['Age'].unique()
```

[ ]:
```
# Examine age range
age_range = df['Age'].unique()
print(sorted(age_range))
```

[ ]:
```
df['Education'].unique()
```

[ ]:
```
df['Employment'].unique()
```

[ ]:
```
df['Marital'].unique()
```

[ ]:
```
df['Gender'].unique()
```

[ ]:
```
df['Contract'].unique()
```

[ ]:
```
df['PaymentMethod'].unique()
```

[ ]:
```
# Display any duplicate rows in the dataframe.
data_duplicates = df.loc[df.duplicated()]
print(data_duplicates)
```

[ ]:
```
# Identify the standard deviation of every numeric column in the dataset
data_std = df_stats.std()
print(data_std)
```

[ ]:
```
data_nulls = df_stats.isnull().sum()
print(data_nulls)
```

```
[ ]: # Impute missing fields for variables Children, Age, Income, Tenure and␣
      ↪Bandwidth_GB_Year with median or mean
     df_stats['Children'] = df['Children'].fillna(df['Children'].median())
     df_stats['Age'] = df['Age'].fillna(df['Age'].median())
     df_stats['Income'] = df['Income'].fillna(df['Income'].median())
     df_stats['Tenure'] = df['Tenure'].fillna(df['Tenure'].median())
     df_stats['Bandwidth_GB_Year'] = df['Bandwidth_GB_Year'].
      ↪fillna(df['Bandwidth_GB_Year'].median())
```

```
[ ]: data_nulls = df_stats.isnull().sum()
     print(data_nulls)
```

1.3 Anomaly Detection & Data Visualization

```
[ ]: # Create histograms of important variables
     df_stats[['Children', 'Age', 'Income', 'Tenure', 'MonthlyCharge',␣
      ↪'Bandwidth_GB_Year']].hist()
     plt.savefig('churn_pyplot.jpg')
     plt.tight_layout()
     # plt.close()
```

```
[ ]: # Some odd distributions here, let's see some box plots for outliers
     # Create a boxplot of user duration, payment & usage variables
     df_stats.boxplot(['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year'])
     plt.savefig('churn_boxplots.jpg')
```

```
[ ]: # Let's see monthly charge separately
     df_stats.boxplot(['MonthlyCharge'])
     # Definitely outliers but not sure how that effects PCA down the line
```

```
[ ]: # Let's see a Seaborn boxplot fee & bandwidth
     sns.boxplot('MonthlyCharge', data = df_stats)
     plt.show()
     # Definitely outliers but not sure what to do with them
```

1.4 Extract Clean dataset to 'churn_clean.csv'

```
[ ]: # Extract Clean dataset
     df_stats.to_csv('churn_clean.csv')
```

```
[ ]: # Reload cleaned data & remove all variable except user services payment info␣
      ↪and survey data
     churn_user = pd.read_csv('churn_clean.csv')
```

```
[ ]: # Slice off all but last eleven service realted variables
     data = churn_user.loc[:, 'Tenure':'Listening']
     data.head()
```

1.5 PCA (D206 Performance Lab)

[ ]:
```
# Import Scikit Learn PCA application
from sklearn.decomposition import PCA
```

[ ]:
```
# Normalize the data
churn_normalized = (data - data.mean()) / data.std()
```

[ ]:
```
# Select number of components to extract
pca = PCA(n_components = data.shape[1])
```

[ ]:
```
# Create a list of PCA names
churn_numeric = data[['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',␣
 ↪'Responses',
                       'Fixes', 'Replacements', 'Reliability', 'Options',
                       'Respectfulness', 'Courteous', 'Listening']]
pcs_names = []
for i, col in enumerate(churn_numeric.columns):
    pcs_names.append('PC' + str(i + 1))
print(pcs_names)
```

[ ]:
```
# Call PCA application & convert the dataset of 11 variables into a dataset of␣
 ↪11 components
pca.fit(churn_normalized)
churn_pca = pd.DataFrame(pca.transform(churn_normalized),
                        columns = pcs_names)
```

[ ]:
```
# For a scree plot import matplotlib & seaborn libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
# Run the scree plot
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance')
plt.show();
```

[ ]:
```
# Extract the eigenvalues
cov_matrix = np.dot(churn_normalized.T, churn_normalized) / data.shape[0]
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for␣
 ↪eigenvector in pca.components_]
```

[ ]:
```
# Plot the eigenvalues
plt.plot(eigenvalues)
plt.xlabel('Number of Components')
plt.ylabel('Eigenvalue')
plt.show();
```

[ ]:
```
# Select the fewest components
for pc, var in zip(pcs_names, np.cumsum(pca.explained_variance_ratio_)):
    print(pc, var)
```

[ ]:
```
# Above, we see that 86% of variance is explained by 7 components
# Create a rotation
rotation = pd.DataFrame(pca.components_.T, columns = pcs_names, index =␣
 ↪churn_numeric.columns)
print(rotation)
```

[ ]:
```
# Output loadings for components
loadings = pd.DataFrame(pca.components_.T,
                        columns = pcs_names,
                        index = data.columns)
loadings
```

[ ]:
```
# Finally, extract reduced dataset & print 3 components
churn_reduced = churn_pca.iloc[ : , 0:3]
print(churn_reduced)
```

1.6 Part III: Data Cleaning

1.6.1 D. Data Cleaning Summary

D1.Cleaning Findings: There was much missing data with meaningful variable fields including Children, Age, Income, Tenure and Bandwidth_GB_Year. Given mean and variance of these variables, it seemed reasonable to impute missing values with median values. Many categorical (such as whether or not the customer was "Techie") & non-numeric (columns for customer ID numbers & related customer transaction IDs) data were not included in analysis given they seemed less meaningful to interpretation and decision-making. The anomalies discovered were not significant & were mitigated as follows.

D2.Justification of Mitigation Methods: Mitigated missing values with imputation using median values. MonthlyCharge variable shows outliers so left alone. This does not seem significant to this analysis.

D3.Summary of Outcomes: Cleaned dataset to leave remaining variables describing customer tenure, monthly service charge, yearly bandwidth usage & responses to survey. It seems pretty straightforward at this point.

D4.Mitigation Code: (see code above and Panopto recording)

D5.Clean Data: (see attached file 'churn_clean.csv')

D6.Limitations: Limitations given the telecom company data set are that the data are not coming from a warehouse. In this scenario, it is as though I initiated and gathered the data. So, I am not able to reach out to the staff that organized & gathered this information to ask them why certain NAs are there, why are fields such as age or yearly bandwidth usage missing information that might be relevant to answering questions about customer retention or churn. In a real world project, you would be able to go down to the department where these folks worked and fill in the empty fields or discover why fields are left blank.

D7. The accurate factual data for missing fields that might be recoverable given the ability to access the staff in the data warehouse, such as company tenure with the company, may give a slighty different overall picture to the analysis. While imputation may provide a path to move forward and give decision-makers reasonable answers, there really is no reason for these data to be missing. The limitations here could be remedied by instituting stricter data acquisition procedures, follow-ups & feedback.

1.6.2 E. PCA Application

E1. Principal Components: The principal components, and what I determined to be "most important", in this dataset include survey responses to:

```
<li>Timely Responses</li>
<li>Timely Fixes</li>
<li>Timely Replacements</li>
<li>Respectful Response</li>
```

E2. Criteria Used: Intuition about customer service suggests that feedback from user survey might offer the most important components when analyzing churn rate. Also, since survey results were the easiest to select as numeric predictors of whether or not a user would leave the company I included the 8 responses as variables for the PCA. And, of course, users' tenure with the company as well as monthly charge & yearly GB use are seem like significant numeric data points for analysis. I used a scree plot & extracted the eigenvalues for visualization of where the "elbow was bending". The elbow bent at about 3 but kept an eigenvalue above 1 until the tenth component. Then, the fewest number of components were selected based on the 86% explanation at 7 components using the Numpy cumsum method. A rotation & loadings were created which suggested the "most important" features of the dataset.

E3. Benefits: The loadings suggest the variables involved in timely action with regard to customer satisfaction (Responses, Fixes, Replacements & Respectfulness) should be given greater emphasis and hopefully help reduce the churn rate from the large number of 27% & "increase the retention period of customers" by targeting more resources in the direction prompt customer service (Ahmad, 2019, p. 1). Again, this seems an intuitive result but now decision-makers in the company of reasonable verification of what might have been a "hunch".

1.7 Part IV: Supporting Documents

1.7.1 F. Video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=fd1f0c02-19fc-487f-a797-ad1400f442cf

1.7.2 G. Sources for Third-Party Code

Cmdline. (2018, March 20). How To Change Column Names and Row Indexes in Pandas? Python and R Tips. https://cmdlinetips.com/2018/03/how-to-change-column-names-and-row-indexes-in-pandas/

Larose, C. D. & Larose, D. T. (2019). Data Science: Using Python and R. John Wiley & Sons, Inc.

Poulson, B. (2016). Data Science Foundations: Data Mining LinkedIn Learning. https://www.linkedin.com/learning/data-science-foundations-data-mining/anomaly-detection-in-python?u=2045532

Sree. (2020, October 26). Predict Customer Churn in Python. Towards Data Science. https://towardsdatascience.com/predict-customer-churn-in-python-e8cd6d3aaa7

VanderPlas, J. (2017). Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, Inc.

1.7.3 H. Sources

Ahmad, A. K., Jafar, A & Aljoumaa, K. (2019, March 20). Customer churn prediction in telecom using machine learning in big data platform. Journal of Big Data. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0191-6

Altexsoft. (2019, March 27). Customer Churn Prediction Using Machine Learning: Main Approaches and Models. Altexsoft. https://www.altexsoft.com/blog/business/customer-churn-prediction-for-subscription-businesses-using-machine-learning-main-approaches-and-models/

Frohbose, F. (2020, November 24). Machine Learning Case Study: Telco Customer Churn Prediction. Towards Data Science. https://towardsdatascience.com/machine-learning-case-study-telco-customer-churn-prediction-bc4be03c9e1d

Mountain, A. (2014, August 11). Data Cleaning. Better Evaluation. https://www.betterevaluation.org/en/evaluation-options/data_cleaning

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('D206_Performance_Assessment_revision.ipynb')
```

[ ]:
```
```