CSE 805: Machine Learning and Data Mining

Class Project: Phase 3/4: Feature Engineering and Model Building

Dr. Balu Gokaraju

Robert Akinie

Fall 2021

Objective

The purpose of the group project is to give students an opportunity to apply course learning objectives in formulating their own Machine Learning projects, using selected datasets. This report details the feature engineering and model building phases of the overall project.

Analysis

Feature Engineering is an important step in completing any Machine Learning (ML) project. This stage ensures that models built for deployment perform very well. Features are essential for making ML models work, hence this step is necessary for making ML models accurate. Building ML is the main goal for problems that require Machine Learning as solutions.

Problem Description

The data for this problem is populated from TripAdvisor.com. It contains reviews on destinations in 10 categories. Each traveler rating is mapped as Excellent (4), Very Good (3), Average (2), Poor (1), and Terrible (0) and average rating is used against each category per user. Each category represents user feedback on a certain destination.

Feature Engineering Steps

For this project, the remaining sections displays the results from the various feature engineering steps that were taken in preparing the data for the modeling phase. Outliers were removed from the data before the feature engineering steps took place. The Appendix contains the code for each step, including outlier removal, respectively.

Feature Selection:

There are ten attributes in this dataset. For clustering purposes, all the attributes would be considered. Looking at the data, and the problem being addressed, all the features would be kept. This is because the model will create a cluster of groups, based on similarity measure. Attributes with near zero correlation with other attributes may likely be in its own cluster, dependent on the number of predefined clusters. Such attributes would be described as having no relationship with other attributes, and hence users who give high reviews to such attributes are less likely to give similar ratings to the other non-correlated attributes. If all attributes also have very low correlation, it may mean that the data has no cluster structure.

Data Transformation:

In order to easily visualize the model output, PCA would be applied to reduce the dimensionality of the dataset. From doing PCA, this is the explained variance ratio.



It can be seen that the two components contain about 60% of the explained variance. For visualization purposes, this transformed data would be used in the clustering model.

Clustering performance improves when all the data has been transformed to the same scale. The paper associated with this dataset considered only Standardization in its data transformation step. For this step, I will be employing normalization, standardization, and a power transformation on the data independently, to compare the performances of all three for clustering.

For a power transformation, Box-cox will not work here, because the data must be strictly positive. Hence another power transform will be used here, which is the Yeo-Johnson. Standardization is considered here because the initial data has attributes with dissimilar variances. Such a condition will affect the weights given to some attributes in the clustering process. In this situation leaving variances unequal is equivalent to putting more weight on variables with smaller variance, "so clusters will tend to be separated along variables with greater variance." Normalization (MinMaxScaler) is also considered here to rescale all the attribute values to unit length. The figures below show the various descriptive statistics for the independent data transformations.

Table

Description automatically generated

Figure . Descriptive Statistics for Normalization

Table

Description automatically generated with medium confidence

Figure . Descriptive Statistics for StandardizationTable

Description automatically generated with medium confidence

Figure . Descriptive Statistics for Yeo-Johnson

Model Building Steps

The goal of this project is to cluster these user groups into similar classes. The model that was employed in this process is the K-means clustering. Partitioning clustering requires to specify the number of clusters in the dataset. There are various methods to determine the optimal number k, such as the Elbow method, Dunn, and Silhouette analysis.

The Elbow method the elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use. I will be making this analysis using both the Elbow method and the Silhouette analysis.

The following graph shows the elbow graph analysis on each of the data transformation techniques.

Chart

Description automatically generated with medium confidence

Figure . Figure of Elbow method in choosing optimal k

From the Elbow analysis, it is not clear as to what optimal k to choose. There are no defining elbows from each of the data transformed, hence the Silhouette analysis. The Silhouette score measures how similar a point is to its own cluster, as compares to other clusters. The range of the score is between -1 and +1. Scores closer to 1 indicate that the sample is placed in its correct cluster. The best value is 1, and worst is -1. Scores close to 0 indicate overlapping clusters. The following graph displays the Silhouette analysis.

Chart, line chart

Description automatically generated

Figure . Figure of Silhouette Analysis

From the graph, it can be inferred that the highest score occurs for when k = 2. That occurs with the normalized dataset. Hence the most optimal k for this data is k = 2. The scores from the analysis will also show which data transformation contributes to better clustering. Normalized data was chosen, because it obtained the highest silhouette score.

In building the model, the various datasets were split into train/test splits, and the test split was used for the optimal k analysis. The test split was used to predict the cluster that the test data would fall under. Unlike classification and regression models, clustering models have no definite way of checking for performance and accuracy, such as precision and recall, accuracy etc. From revised literature, the quality of a cluster is judged by the Silhouette score, which shows how well the data is clustered for the optimal k. From Figure 5, the highest score is around 0.44. There is also the KMeans score, which is the value that represents the sum of squares of the distances of points from their cluster centroids. The KMeans score is shown below:

Graphical user interface, text, website

Description automatically generated

Figure . KMeans score

Most of the evaluation metrics for clustering require ground truth labels, such as labels one expects our models to output. For this dataset, there are no prior ground truth labels on which such evaluations can be performed.

The following figure shows the clustered data, with its centroids.

Chart, scatter chart

Description automatically generated

Figure . Figure of clustered data.

Appendix

# Imports

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from pandas.plotting import scatter\_matrix

from sklearn.decomposition import PCA

from sklearn.preprocessing import PowerTransformer

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.cluster import KMeans

from sklearn import metrics

# Loading of data

data = pd.read\_csv("tripadvisor\_review.csv")

data.head(10)

'''

Data shows a number of outliers.

Outliers in this dataset will be defined as values greater/less than 3 standard deviations from the mean.

The following code will count such observations.

'''

lower, upper = np.mean(data) - 3\*np.std(data), np.mean(data) + 3\*np.std(data)

mask = (data.drop(columns = "User ID") < lower) | (data.drop(columns = "User ID") > upper)

outlier = mask.sum(axis = 0)

print(outlier)

#'''Removal of outliers'''

data[mask] = np.nan

data.dropna(inplace = True)

data.describe()

#Phase 3

data = data.drop(columns = 'User ID')

#PCA

pca = PCA(n\_components=2)

data1 = pca.fit\_transform(data)

print(pca.explained\_variance\_ratio\_)

#Normalized, Jeo-Johnson and Standardized respectively

N, YJ, S = data1, data1, data1

#Data transformations

scaler = MinMaxScaler(feature\_range = (0, 1))

rescaled\_N = scaler.fit\_transform(N)

rescale\_N = pd.DataFrame(rescaled\_N, columns = ['Component 1', 'Component 2'])

rescale\_N.describe()

scaler2 = StandardScaler().fit(S)

rescaled\_S = scaler2.transform(S)

rescale\_S = pd.DataFrame(rescaled\_S, columns = ['Component 1', 'Component 2'])

rescale\_S.describe()

scaler3 = PowerTransformer(method = 'yeo-johnson')

rescaled\_YJ = scaler3.fit\_transform(YJ)

rescale\_YJ = pd.DataFrame(rescaled\_YJ, columns = ['Component 1', 'Component 2'])

rescale\_YJ.describe()

#Train Test Split

X\_train\_S, X\_validation\_S = train\_test\_split(rescale\_S, test\_size=0.20, random\_state=1, shuffle=True)

X\_train\_YJ, X\_validation\_YJ = train\_test\_split(rescale\_YJ, test\_size=0.20, random\_state=1, shuffle=True)

X\_train\_N, X\_validation\_N = train\_test\_split(rescale\_N, test\_size=0.20, random\_state=1, shuffle=True)

#Section for Elbow method

distortion = []

distortion2 = []

distortion3 = []

K = range(1,10)

for k in K:

kmeanModel = KMeans(n\_clusters=k)

kmeanModel2 = KMeans(n\_clusters=k)

kmeanModel3 = KMeans(n\_clusters=k)

kmeanModel.fit(X\_train\_S), kmeanModel2.fit(X\_train\_YJ), kmeanModel3.fit(X\_train\_N)

distortion.append(kmeanModel.inertia\_)

distortion2.append(kmeanModel2.inertia\_)

distortion3.append(kmeanModel3.inertia\_)

plt.figure(figsize=(16,8))

plt.plot(K, distortion, label = 'Standardized')

plt.plot(K, distortion2, '-', label = 'Power Transform, YJ')

plt.plot(K, distortion3, '--', label = 'Normalized')

plt.xlabel('k')

plt.ylabel('Distortion')

plt.legend()

plt.title('The Elbow Method showing the optimal k')

plt.show()

# determine k using silhouette method

distortions = []

distortions2 = []

distortions3 = []

K = range(2,10)

for k in K:

kmeans\_S = KMeans(n\_clusters=k, init='k-means++').fit(X\_train\_S)

kmeans\_YJ = KMeans(n\_clusters=k, init='k-means++').fit(X\_train\_YJ)

kmeans\_N = KMeans(n\_clusters=k, init='k-means++').fit(X\_train\_N)

labels = kmeans\_S.labels\_

labels2 = kmeans\_YJ.labels\_

labels3 = kmeans\_N.labels\_

distortions.append(metrics.silhouette\_score(X\_train\_S, labels, metric = 'euclidean'))

distortions2.append(metrics.silhouette\_score(X\_train\_YJ, labels, metric = 'euclidean'))

distortions3.append(metrics.silhouette\_score(X\_train\_N, labels, metric = 'euclidean'))

# Plot the elbow

plt.plot(K, distortions, label = 'Standardized')

plt.plot(K, distortions2, '-.', label = 'Power Transform, YJ')

plt.plot(K, distortions3, '--', label = 'Normalized')

plt.xlabel('k')

plt.ylabel('Distortion')

plt.title('Silhouette score')

plt.legend()

plt.show()

kmeans = KMeans(n\_clusters=2, random\_state=1, init='k-means++').fit(X\_train\_S)

pred = kmeans.predict(X\_train\_N)

from collections import Counter

Counter(kmeans.labels\_)

plt.scatter(X\_train\_N.iloc[:, 0], X\_train\_N.iloc[:, 1], c=pred, s=50, cmap='summer')

centers = kmeans.cluster\_centers\_

plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);

metrics.silhouette\_score(X\_train\_N, labels, metric = 'euclidean')