renmoney

Decision Science Entry Test

Presentation of Results

Seghe Momodu

Friday, July 9th , 2021

Background.

As part of the recruitment exercise for the Renmoney Decision Science team, CRM data, detailing the behaviour of customers for a US-based company was provided. The task was to optimally cluster the customers, from a statistics point of view, in such a manner that the results would be used for the next step business actions.

The Data

- The CRM data has 900 rows and 615 columns. The datatypes are both continuous in float, integer types and categorical in object datatype formats.
- A first observation on exploring the dataset is the large number of missing values and constant zero values that fill the rows of the dataset.
- To handle the data, the dataset was split into three (3) dataframes according to the datatypes, in the ratio {float64: 334, int64: 271, object: 10}. The float datatypes had over 60% of the columns filled with NaN values, with the rest of the columns dominated by NaN and zero values.
- The columns with missing values across the three dataframes were dropped rather than imputed because imputation would skew causing a distortion that may bias the feature selection and machine learning algorithms that would be used on the data.
- Dropping the missing value columns greatly reduced the number of columns from 334 to 4 and 10 to 2 for the float and object dataframes. The int dataframes had no missing values but was very sparse, owing to the large number of columns filled with zero values. The dataset is so noisy!

** Visualizations for this exercise can be found in the accompanying Google Colab Notebook for this exercise.

Feature Engineering.

Reconstructing the dataset

- The int dataframe had 271 columns.
- The observation was that a large portion of the columns, over 70% were splits from One-Hot Encoding of the categorical columns.
- In order to reduce the number of columns, the dataset was manually reconstructed the dataset to recover the previously categorical columns.
- The regular expressions (RegEx) was employed to decouple the dataframe and compile it by matching the one-hot encoded category columns to their main column.
- On recovery of the categorical columns, the dataframe was successfully compressed to 60 columns – 25 integer columns and 35 categorical (object) columns.
- The dataframe was again split into two (2) dataframes – int and object dataframes. The object dataframe was further processed by chopping off the underscore-separated-prefixes from the rows. The data was cleaned and incomplete words from the striping were fixed.
- The categorical column values were Label Encoded. Label encoding was the obvious choice to increase the vertical (in-column) variability of the dataframe values, thus somewhat decreasing the sparsity of the data.

** Visualizations for this exercise can be found in the accompanying Google Colab Notebook for this exercise.

Feature Selection.

Feature Selection technique – Variance Threshold

- Although the dataset was reduced from 271 to 60 columns by reconstructing the dataset, the dataset was still very sparse. Some of the columns were filled with constant zero values, some columns were binarily-filled with values of 0 or 1 and a few columns had actual values but were still plagued by the zero-constant.

- To manage this occurrence, a feature selection technique to help identify and select the features that have the most discriminative power the set of all the features with the direct effect of increasing the performance of the Clustering algorithm.

- A feature selection technique known as **Variance Threshold** was employed to help identify and select the features from the array of low variance, constant - zero - variance and sparsity.

- Variance threshold is an unsupervised feature selection technique that removes features that do not met a set threshold. This technique was used to remove the zero-constant features at threshold = 0 and the low-variance features with a threshold set at

- The dataframes were concatenated together to a final dataset of 11 columns.

- Summary statistics after feature selection revealed that the dataset was riddled with outliers across the 11 columns. The data was scaled by fitting it with the Robust Scaler, a scaler highly robust to outliers.

** Visualizations for this exercise can be found in the accompanying Google Colab Notebook for this exercise.

Model Development.

Finding the Optimal number of clusters.

- The Silhouette Analysis technique was used to find the optimal number of clusters for the clustering algorithm. The Silhouette method was the most ideal because it gives a measure of how close each point in one cluster is to points in the neighboring clusters and thus providing a way to assess parameters like number of clusters visually.

- The evaluation metric used is the Silhouette score. Our silhouette score is **0.664** with the number of clusters, **k = 6**. This was chosen as the optimal number of clusters, because there was a better balance between a more stable degree of visual separation between the clusters and a silhouette score close to 0.7.

- The centroids of the KMeans co-ordinates and the labels are computed for the data. The centroids are concatenated to a list of the used features and inverse transformed. This inverse transformed data aids the interpretability of the clusters and their attributes and is used for the next step business actions, although, at this time that is outside the scope of this task.

** Visualizations for this exercise can be found in the accompanying Google Colab Notebook for this exercise.

Model Development, contd.

Dimensionality Reduction with Principal Component Analysis.

- As earlier established, the dataset greatly suffers from the curse of dimensionality. The technique, Principal Component Analysis (PCA) shall be employed to extract a lower dimension feature set that would be fed into the KMeans algorithm by projecting the dataset from an otherwise high dimension of not-so-relevant features. The direct effect of this technique is a compression of the dataset, capturing as much information/variance as possible.

- The explained variance ratio which is the amount of variance explained by each of the principal components is **{PC1: 0.91780554, PC2: 0.04950848}**. Well, I think the principal components worked hard!

- Looking at the Principal Components plots for KMeans, it is observed that the Principal Components also returned the number of clusters **as k = 6**.

***An assumption made about the date features is that since the date values were in numeric format, the values are assumed to be the number of days from the present day. For example, an observation with a RegisteredUserCreateDate of 2256 days (6 years approx.) could be interpreted to be the number of days since the user registration profile was created.*

** Visualizations for this exercise can be found in the accompanying Google Colab Notebook for this exercise.

Conclusion.

Key Takeaways.

- The NaN values were not imputed. This is because information (business relevance) about the features was unavailable. Imputing the features would skew the results. Only features with no missing values were used.

- Following a series of feature engineering and feature selection techniques, the dataset was reduced from 615 columns to 11 columns. While there was a high degree of in-column sparsity, evidenced by a high proportion of zero-filled columns, the dataset was left as is instead of dropping the columns. There was a concern about losing the information contained in the data.

- KMeans is greatly affected by outliers. In other to manage the outliers, the data was scaled with the Robust Scaler. The robust scaler acts to remove the median and scale the data according to the inter quantile range (IQR).

- KMeans was unable to figure out the clusters correctly. This is because KMeans tries to minimize the within-cluster variation, it gives more weight to bigger clusters than smaller ones. As seen in the plot, the data points in smaller clusters may be left away from the centroid in order to focus more on the larger cluster.

- KMeans typically assumes spherical shapes for clusters (with the radius equal to the distance between the centroid and the furthest data point) and doesn’t work well when clusters are in shapes other than spherical shapes. As seen in the KMeans plot in the notebook.

** Visualizations for this exercise can be found in the accompanying Google Colab Notebook for this exercise.