Data And Information Quality Project Report

- **PROJECT ID**: 21
- **PROJECT NUMBER**: 1
- **ASSIGNED DATASET**: USERS
- **STUDENT**: PASQUALE CASTIGLIONE 10657816
- **ASSIGNED TASK**: CLUSTERING

Contents

1 SETUP CHOICES

1.1 Imputation

1.1.1 Simple Imputation

Because of the nature of the dataset, propagating values from valid cells to cells with missing values, seemed to be the best choice. In fact this method showed very good result with the original data, but applying it to a shuffled version of the dataset showed worse results.

1.1.2 Advanced Imputation

K-Nearest Neighbors was used as the advanced technique to impute missing value as it was able to spot similiarity between tuples and impute values accordingly. Compared to the simple imputation this method proved to be worse with the original dataset but more robust to shuffling.

1.2 Clustering

1.2.1 K-Modes

Because of the categorical nature of the data, *K-Modes*[1] was the first choice. In order to select the best number of clusters, *elbow method* analysis was performed.

1.2.2 K-Means

The second clustering techniques used was *K-Means*, applied using Jaccard as distance measure. *Elbow method* analysis was performed to find out the best number of clusters.

[1] `https://github.com/nicodv/kmodes`

2 PIPELINE IMPLEMENTATION

2.1 Data Load and Dirty Dataset Generation:

First of all the dataset was loaded and, using the provided script, dirty datasets with different completeness levels were generated.

2.2 Data Exploration:

In this phase histograms of the datasets and heatmaps of the datasets with missing values were plotted in order to have a general idea of the data and to take decisions accordingly.

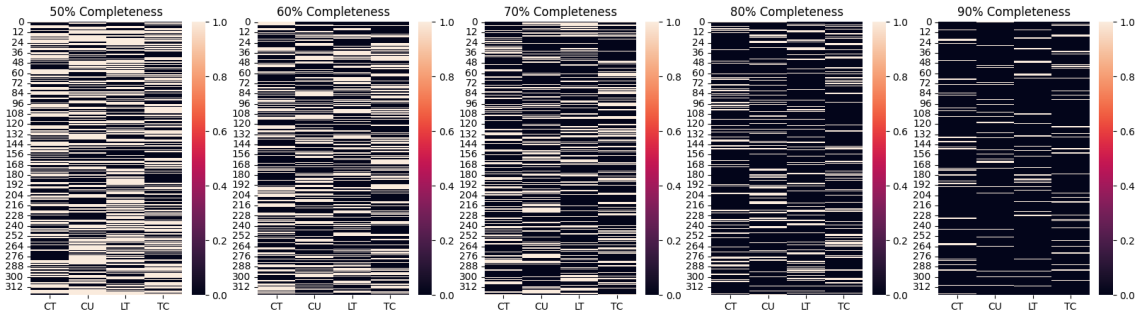

Figure 1: Null Values

2.3 Imputation:

2.3.1 Simple Imputation:

For all the dirty datasets the method `fillna` was used. Firstly with the parameter `method='ffill'` and then with `method='bfill'` in order impute null values in the first row. After the imputation accuracies were computed.

```
users_simple_imp = {}
simple_accuracies = {}
for i, df_dirty in enumerate(users_dirty_all):
    users_dirty = pd.DataFrame(df_dirty, columns=users.columns)
    users_simple_imp[i] = users_dirty.fillna(method='ffill')
    users_simple_imp[i] = users_simple_imp[i].fillna(method='bfill')
    simple_accuracs[i] = accuracy(users, users_simple_imp[i])

simple_accuracies
```

Figure 2: simple imputation

2.3.2 Advanced Imputation:

First of all data was encoded as one hot arrays using the function `get_dummies` from Pandas. Then `KNNImputer` from *sklearn* was used to fit and transform the dirty datasets. After the imputation, data were decoded back to categorical. At the end accuracies were computed.

```
knn_accuracies = {}
users_knn_imp = {}
for i, df_dirty in enumerate(users_dirty_all):
    users_dirty = pd.DataFrame(df_dirty, columns=users.columns)

    users_dirty_one_hot = pd.get_dummies(users_dirty)
    for col in users_dirty.columns:
        users_dirty_one_hot.loc[users_dirty[col].isnull(), users_dirty_one_hot.columns.str.startswith(col)] = np.nan

    knn_imputer = KNNImputer(n_neighbors=5)
    users_knn_imp_one_hot = pd.DataFrame(knn_imputer.fit_transform(users_dirty_one_hot))
    users_knn_imp_one_hot.columns = users_dirty_one_hot.columns

    users_knn_imp[i] = pd.DataFrame()
    for col in users_dirty.columns:
        users_knn_imp[i][col] = users_knn_imp_one_hot.loc[:,
users_knn_imp_one_hot.columns.str.startswith(col)].idxmax(1)
    users_knn_imp[i] = users_knn_imp[i].apply(lambda e: e.str[3:] )
    users_knn_imp[i]
    knn_accuracies[i] = accuracy(users,users_knn_imp[i])

knn_accuracies
```

Figure 3: advanced imputation

2.4 Clustering:

2.4.1 K-Modes:

First of all elbow analysis was performed to choose the number of clusters, after choosing it the model was fitted . In this phase *cluster's centroids* were printed and a column with the assigned cluster was added to the dataframe

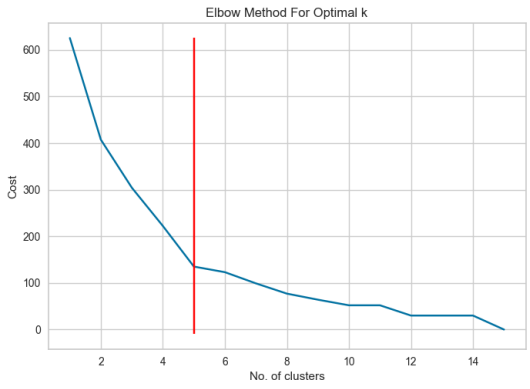

Figure 4: Elbow Method for the original dataset

```
[['CT_range_1' 'CU_range_1' 'LT_range_1' 'holiday']
 ['CT_range_4' 'CU_range_1' 'LT_range_4' 'sport']
 ['CT_range_2' 'CU_range_1' 'LT_range_2' 'holiday']
 ['CT_range_4' 'CU_range_1' 'LT_range_1' 'game']
 ['CT_range_1' 'CU_range_4' 'LT_range_4' 'ECommerce']]
```

Figure 5: K-Modes Centroids

2.4.2 K-Means:

First of all the distance matrix was computed using Jaccard as distance between tuples. Then the usual elbow plot was plotted in order to choose the best number of clusters. At the end a column with the assigned cluster was added to the dataframe. To asses the quality of the clustering silhuette score was computed using the function `silhuette_score` from sklearn.

Simple Imputed Dataset

```
simple_score = {}
for i, df in enumerate(users_simple_imp.values()):
    dist_matrix = np.asarray( [[jaccard_dist(a,b) for _,b in df.iterrows()] for _,a in df.iterrows()])
    model = KMeans()
    visualizer = KElbowVisualizer(model, k=(2,16))
    visualizer.fit(dist_matrix)
    k = visualizer.elbow_value_
    kmeans_clust = KMeans(k).fit(dist_matrix)
    simple_score[percentages[i]] = [round(silhouette_score(dist_matrix, kmeans_clust.labels_, metric='precomputed'),2), k]
    df['Cluster'] = kmeans_clust.labels_

simple_score
```

Figure 6: K-Means on simple imputed datasets

3 RESULTS

3.1 Imputation

To assess the quality of the imputation, accuracy was computed as the number of rows that were imputed as the original over all the rows.

```
def accuracy(df, df_imputed):
    accuracy = {}
    for col in df.columns:
        equal = np.where(df[col] == df_imputed[col], True, False)
        accuracy[col] = round(equal[equal == True].sum() / len(equal), 2)
    accuracy['avg'] = round(sum(accuracy.values())/len(accuracy),2)
    return accuracy
```

Figure 7: Function that compute the accuarcy

3.1.1 Simple Imputation Accuracy

	CT	CU	LT	TC	avg
50%	0.89	0.90	0.93	1.00	0.93
60%	0.91	0.94	0.92	1.00	0.94
70%	0.93	0.94	0.96	1.00	0.96
80%	0.94	0.97	0.96	1.00	0.97
90%	0.98	0.99	0.99	1.00	0.99

Table 1: Simple Imputation

	CT	CU	LT	TC	avg
50%	0.67	0.83	0.64	0.66	0.70
60%	0.69	0.89	0.74	0.72	0.76
70%	0.80	0.90	0.80	0.78	0.82
80%	0.84	0.92	0.87	0.85	0.87
90%	0.93	0.97	0.95	0.93	0.94

Table 2: Simple Imputation on Shuffled Dataset

3.1.2 Advanced Imputation Accuracy

	CT	CU	LT	TC	avg
50%	0.81	0.89	0.76	0.79	0.81
60%	0.84	0.93	0.85	0.87	0.87
70%	0.93	0.95	0.90	0.92	0.92
80%	0.92	0.97	0.94	0.95	0.94
90%	0.97	1.00	0.97	0.98	0.98

Table 3: KNN Imputation

	CT	CU	LT	TC	avg
50%	0.75	0.92	0.75	0.78	0.80
60%	0.87	0.93	0.84	0.88	0.88
70%	0.88	0.95	0.91	0.92	0.92
80%	0.94	0.97	0.94	0.95	0.95
90%	0.98	0.99	0.98	0.99	0.99

Table 4: KNN Imputation on Shuffled Dataset

3.2 Clustering

3.2.1 K-Means

To asses the quality of k-means clustering *silhuette score* was computed.
Original Dataset: 0.72, 5 clusters

	score	**clusters**
50%	0.25	6
60%	0.31	5
70%	0.32	6
80%	0.44	5
90%	0.58	5

Table 5: Silhouette score for simple imputed datasets

	score	**clusters**
50%	0.58	6
60%	0.61	5
70%	0.64	5
80%	0.68	5
90%	0.72	5

Table 6: Silhouette score for advanced imputed datasets