Identify Customer Segments

REVIEW | CODE REVIEW | HISTORY

Requires Changes

5 specifications require changes

Great job on the very first submission 👏👏

But still, there are some requirements that your submission does not meet.

I have provided you some hints and references to help you move forward so have a look at those.

Useful References

- Visualizing statistical plots with Seaborn
- Univariate Distribution plot
- sklearn's RobustScaler

Please look at all the tips and requirements that I shared to include in your project. KEEP UP THE GOOD WORK!

Don't get disheartened, you did a great job which deserves a big compliment, think of those changes as a great opportunity to learn more and perfect your skills. 💪💪

I wish you good luck. Looking forward to your next submission.
For any queries, you can ask on Knowledge Portal as well

Stay ! Stay Safe

DON'T FORGET TO RATE MY WORK AS PROJECT REVIEWER! YOUR FEEDBACK IS VERY HELPFUL AND APPRECIATED.

Preprocessing

✓ **All missing values have been re-encoded in a consistent way as NaNs.**

✅ Missing value codes given in feat_info's last column have been used to convert all codes to NaNs.

✅ You have taken care of the 'X' and 'XX' strings in their code

Comments:

- Good job writing a loop to handle the missing value cases. Nice attention to the special encoding of the 'X' and 'XX' cases compared to the others.
- It is necessary to have all the missing values altered in the format that can be comprehended by the panda's tool so that you can later choose to deal with them easily. For e.g. replacing them or dropping rows with missing values etc

rows with missing values etc.

- It's important to take special care when the given dataset has undeclared missing values that may still hinder the data analysis, such as 'X' and 'XX' strings.

Suggestions:

- You can further optimize the code by having only a single for loop with a time complexity of (O(n)). This code takes approximately 1.2 seconds to complete the process of re-encoding the NaN values. It is 1/56 times the total time that students generally come up with.
  To check the execution time of your code, have a look at this post: 🔗 **How to check the execution time of Python script ?**

```
for attribute, miss_index in zip(feat_info['attribute'], feat_info['missing_or_unknown']):
    missing_values = miss_index.strip('[]').split(',')
    missing_values = [int(value) if (value!='X' and value!='XX' and value!='') else value for va
lue in missing_values]
    if missing_values != ['']:
        azdias[attribute] = azdias[attribute].replace(missing_values, np.nan)
```

✓ **Columns with a large amount of missing values have been removed from the analysis. Patterns in missing values have been identified between other columns.**

✅ Columns with a large number of missing values have been removed from the analysis

✅ Patterns in missing values have been identified between other columns

Suggestions:

If a significant number of values are missing from the column, it is worth removing them, but there are still some pros and cons.

Pros:

- Complete removal of data with missing values results in robust and highly accurate model
- Deleting a particular row or a column with no specific information is better since it does not have a high weightage

Cons:

- Loss of information and data
- Works poorly if the percentage of missing values is high (say 30%) compared to the whole dataset

🔗 **5 Ways To Handle Missing Values In Machine Learning Datasets**

✓ **Categorical features have been explored and handled based on if they are binary or multi-level.**

✅ Categorical features have been explored and handled based on if they are binary or multi-level.

Comments:

- Nice job balancing the preprocessing requirements without the issue of dimensionality. Remember that it is essential to avoid diluting the variability captured by components in the PCA step.
- All categorical variables need to have numerical values for further analysis since we would be fitting a Kmeans model to solve the problem, and this particular model only ingests numeric values while training.

Suggestions:

- Always ensure you do not fall into the **curse of dimensionality**.

As the number of features grows, the amount of data we need to accurately distinguish between these features (to give us a prediction) and generalize our model (learned function) grows.

- In most cases, high cardinality makes it difficult for the model to identify such patterns; hence, the model doesn't generalize well to examples outside the training set.

🔗 **Dealing with features that have high cardinality**

🔁 **The data has been split into two parts based on how much data is missing from each row. The subsets have been compared to see if they are qualitatively different from one another.**

❌ The data has been split into two parts based on how much data is missing from each row.

Issues:

- How did you arrive at this threshold?

```
In [ ]: # How much data is missing in each row of the dataset?
        rows_by_missing_s = azdias.T.isna().sum()

In [ ]: rows_by_missing_s.name = "missing_features"
        rows_by_missing_s.index.name = "row_index"

        print("Row Quality:")
        print("all rows:")
        display(rows_by_missing_s.describe())
        print("Rows missing >= 3 features (top 25% of missing):")
        display(rows_by_missing_s.loc[rows_by_missing_s >= 3].describe())

Row Quality:
all rows:
count    891221.000000
mean          5.649894
```

- It is essential to look at the distribution over specific columns for rows with more than a certain number of missing values, with the rows more petite than a certain number of missing values.

Resolution:

- You must create a distribution plot to identify the correct threshold to split the subset effectively.
- Here is one example, this is the most optimal range to divide the subset effectively.

✅ The subsets have been compared to see if they are qualitatively different from one another.

Comments:

- Good visuals and comparisons of the features provided in the notebook.
- These bar charts show us a summary of all the values in a single picture.

✓ **Mixed-type features have been explored, resulting in re-engineered features.**

✅ Two mixed-type features, PRAEGENDE_JUGENDJAHRE and CAMEO_INTL_2015, engineered into two new features.

Comments:

Nice work generating the new features.

- Decade = information about the decade the youth belongs to.
- Movement = information about the prevalent movement
- Wealth = information about the wealth of a family
- Life Stage = stage of the family development.

Why are we doing this operation?

Values in `PRAEGENDE_JUGENDJAHRE` and `CAMEO_INTL_2015` won't be helpful for the algorithm (uninterpretable by the algorithm). That is why we derive these numerical features from using the information in our model effectively.

🔗 **Complete Guide to Feature Engineering: Zero to Hero**

🔁 **A function applying pre-processing operations has been created, so that the same steps can be applied to the general and customer demographics alike.**

✅ A function applying pre-processing operations has been created

❌ same steps can be applied to the general and customer demographics alike

Issues:

- Some issues are highlighted for the notebook sections 1.1.3.**

Resolution:

- Kindly address the feedback provided in the following rubric:

```
- *The data has been split into two parts based on how much data is missing from each row.
The subsets have been compared to see if they are qualitatively different from one another*
```

- Once you address the abovementioned issues, incorporate similar changes in the **clean_data** function.

Suggestions:

- You might want to add a code cell after the **clean_data** function to test it out or to verify that it works on the general demographics data as expected.
- You can use the python's assert method. Here is one sample code:

```
assert azdias_clean_data.shape[0] == azdias_manual_clean.shape[0], "clean_data function is not w
orking properly, rows mismatch"
assert azdias_clean_data.shape[1] == azdias_manual_clean.shape[1], "clean_data function is not w
orking properly, columns mismatch"
print("If this is all you see, you passed the tests")
```

✓ **Dataset includes all original features with appropriate data types and re-engineered features. Features that are not formatted for further analysis have been excluded.**

✅ Dataset includes all original features with appropriate data types and re-engineered features

✅ Features that are not formatted for further analysis have been excluded.

Comments:

- Excellent job at retaining original features with appropriate data types
- You have engineered the features reflecting Wealth, Life Stage, Decade and Movement, which will help increase the variability in the data.
- Features like PRAEGENDE_JUGENDJAHRE and CAMEO_INTL_2015 are worth dropping in their original formulations as they are mixed variable and is uninterpretable by most of the modeling techniques.

Suggestion:

You can also explore other mixed variables like LP_LEBENSPHASE_GROB, PLZ8_BAUMAX, etc., to extract meaningful features to increase the variability further.

Feature Transformation

✓ **Feature scaling has been properly applied to the demographics data. Imputation has been performed to remove remaining missing values.**

✅ Feature scaling has been properly applied to the demographics data.

✅ Imputation has been performed to remove the remaining missing values

✅ Valid justifications are provided for handling the missing values and scaling operation, in the report.

What if you don't handle missing values?

- Certain sklearn estimators do not work if the data contains missing values since their mathematical operations demand the data to have all values in numeric format.
- For instance, if you directly try to fit the Standscaler object without handling missing values, then you will face the following error:

```
ValueError: Input contains NaN, infinity or a value too large for dtype('float64')
```

- So it becomes necessary to handle the missing values before any fitting operation.

What if you don't feature scale the data?

- Suppose a feature's variance is orders of magnitude more than the variance of other features. In that case, that feature might dominate other features in the dataset, which is not something we want to happen.
- This skews the PCA towards high magnitude features when fitted on such data.

✓ **Principal component analysis has been applied to the data to create transformed features. A variability analysis has been performed to justify a decision on the number of features to retain.**

✅ Principal component analysis has been applied to the data to create transformed features

✅ A variability analysis has been performed to justify a decision on the number of features to retain.

Good justification has been provided for choosing 96 components.

The principal component cutoff is justified correctly by observing the turning point or drop-off in variance.

Why are we performing this step?

It is pretty challenging to work with a dataset having hundreds of features. This can lead to an ML model suffering from 🔗 overfitting. So to avoid this type of problem, it is necessary to apply 🔗 Feature Extraction techniques. Advantages of such methods are:

- Accuracy improvements.
- Overfitting risk reduction.
- Speed up in training.
- Improved Data Visualization.
- Increase in explainability of our model.

✓ **Weights on at least three principal components are used to make inferences on correlations between original features of the data. General meanings are ascribed to principal components where applicable.**

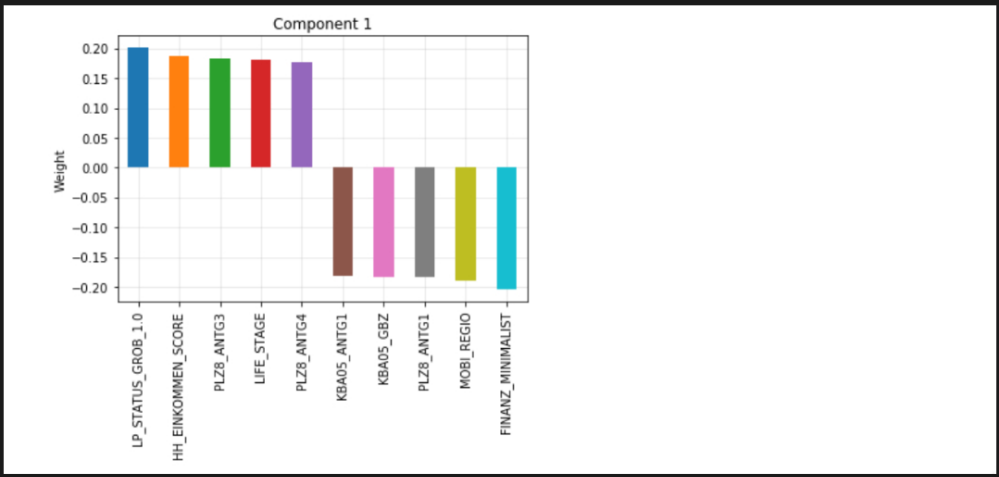
✅ You should map each weight to their corresponding feature name, then sort the features according to weight.

Comments:

- Good job! all the features have been mapped to correct weights.

Suggestions:

- Instead of analyzing the numerical weights, you can visualize the most prominent features as shown below:



- (hint: use pandas plot() method)

✅ You should investigate and interpret feature associations from the first three principal components

Comments:

- The interpretation of the printed-out component coefficients is well done!
- This part is tricky since a lot of the coefficients have 'better' meanings associated with smaller values.

🔗 Interpretation of the Principal Components

Clustering

🔁 **Multiple cluster counts have been tested on the general demographics data, and the average point-centroid distances have been reported. A decision on the number of clusters to use is made and justified.**

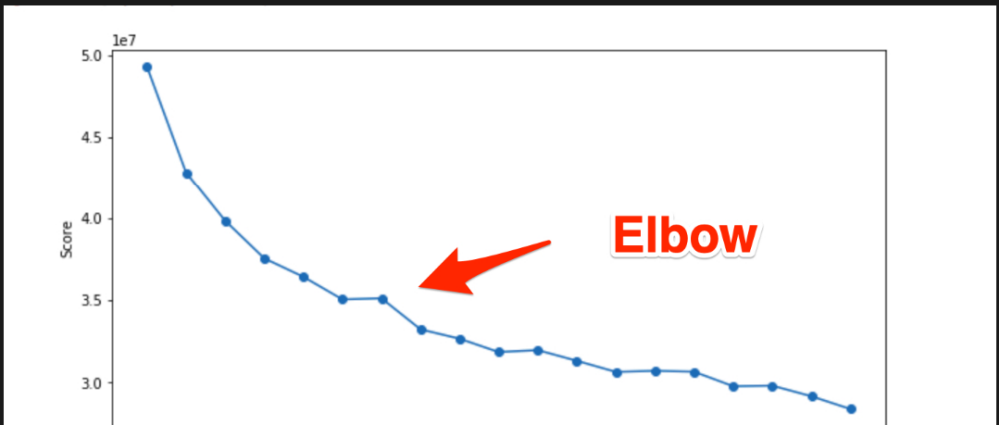
❌ Multiple cluster counts have been tested on the general demographics data

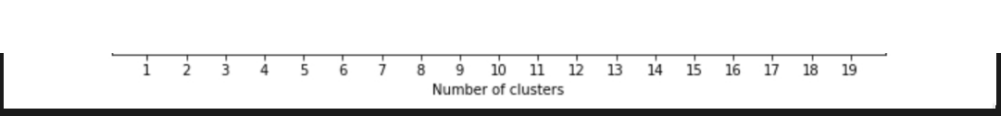
Issues:

- Due to the short-range defined for the "for" loop, the visualization does not arrive at any turning point/elbow.

Resolution:

- You should increase the range of K if you struggle to find the elbow.
- Try at least from K=2. Create one scree plot like this. Using this range, you might come up with an optimal number of centroids.





- If KMeans take a longer time to fit, then you can go for 'minibatchkmeans' to significantly reduce the fitting time.
- But do not go beyond 30.

Cleaning, feature transformation, dimensionality reduction, and clustering models are applied properly to the customer demographics data.

✅ Feature wrangling, selection and engineering using clean_data() function.

Comments:

- Awesome! the clean_data function is applied successfully without any issue.
- You might receive a different set of features after cleaning the customer data. Have a look at this post to know the workaround.

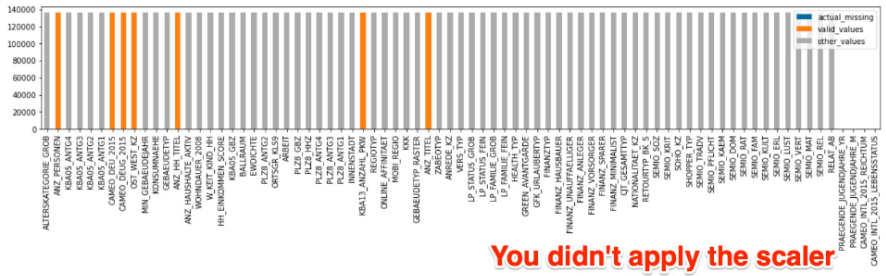
✅ Data transformation using already fitted Imputer object

❌ Data transformation using already fitted Scaler object.

✅ Feature extraction using already fitted PCA object

✅ Data segmentation using already fitted KMeans model.

5 rows × 213 columns



You didn't apply the scaler transformation.

```
In [ ]: customers_pca = pca.transform(customers_main)
        customer_predict = k_means_6.predict(customers_pca)

        del customers_main
```

Step 3.3: Compare Customer Data to Demographics Data

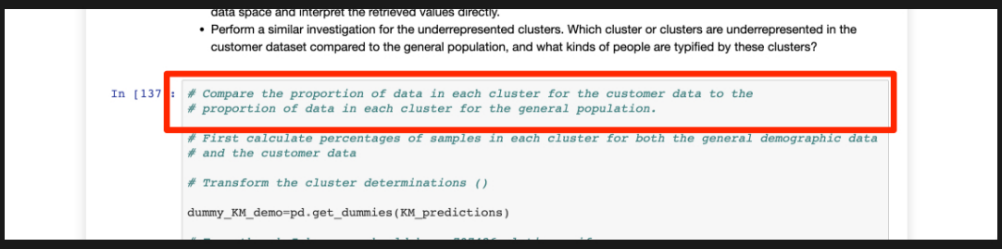
- You might face an error **'ValueError: operands could not be broadcast together with shapes'.** To resolve this, have a look at this post.

A comparison is made between the general population and customers to identify segments of the population that are central to the sales company's base as well as those that are not.

❌ Compute the proportion of data points in each cluster and visualize, for the general population and the customer data

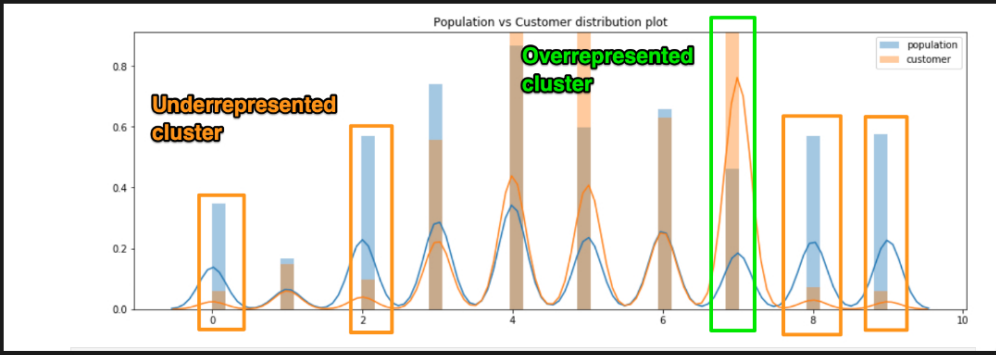
Issues:

- You have not performed the comparison of the data proportion in each cluster for the customer data to the proportion of data in each cluster for the general population



Resolution:

- Use a bar plot for this task. Here is an example:



- If you are struggling to create one, look at this article.

❌ Identified overrepresented and underrepresented clusters

❌ Inferred the kind of people being represented by each cluster type.

Issues:

- Your procedure to get to the final interpretation is wrong.

Resolution:

- Do the following task:
    1. You need to mention the cluster number which is overrepresented and underrepresented.
    2. Once you identified, you need to use the fitted objects to `inverse transform` the data of the identified clusters. This way, you will have the unscaled version of the data
    3. You need to describe the type of people being reflected by analyzing the inverse_transformed data (hint: use data dictionary).
- We focus on identifying the segments in the general demographic data where the company can focus on its sales.
- If you still struggle with the correct inference, reach out to a mentor at the knowledge hub for proper guidance. It is difficult to fully explain the methodology here.

RESUBMIT PROJECT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH