Data Science - Project Report
Group: DoData

- Dataset Description:

Dataset: Nomao dataset
URL: https://archive.ics.uci.edu/dataset/227/nomao

- Existing analysis:

The articles/papers which have already analyzed this dataset are listed below:

- Laurent Candillier, Vincent Lemaire (2013), Active learning in the real-world design and analysis of the Nomao challenge.

- Problem statement:

- Challenges faced:

  There were no data cleaning/preparation steps applied to the dataset in the paper discussed in the existing analysis section. The challenges we faced while preparing the dataset are described below:

  (a) Dataset loading:

  The dataset was stored as a .data file and did not have column names in the data frame when read as a CSV file. Thus, column names were added manually to have a complete data frame.

  (b) Handling missing values:

  When the data set was processed, it consisted of missing values in large numbers. The percentage of missing values varied for each column ranging from 0% to 97.4%. Thus, we had to come up with an appropriate percentage threshold for deciding whether to completely remove the column from the dataset or whether to impute values for a specific column. The choice of this threshold was purely based on the amount of data we would retain after removing columns having a percentage of missing values more than the threshold value so that we don't lose a lot of information and neither have imputed columns with values imputed-based on very little information. After trying a series of percentage values for the threshold, we decided on 60% as the threshold solely depending on the tradeoff described above. This ensures the data quality by reducing amount of noise and irrelevant infromation. We removed all the columns having more than 60% missing values and imputed missing values in other columns using KNN imputer. The number of neighbors (k) was decided after parameter tuning.

(c) Encoding:

The dataset only consisted of nominal categorical variables and not ordinal variables. So, one-hot encoding (ohe) was applied to the twenty-nine columns.

(d) Outlier handling:

For this preprocessing step, we first need to scale the values of encoded and imputed data. The challenge we faced here was whether to apply normalization or standard scaling. Since our dataset did not follow a normal distribution, we decided to apply standard scaling. In order to detect and handle outliers, first, we need to choose which method to apply for outlier detection among many available methods such as z-score, IQR method, LOF, etc. The challenge we faced while deciding the appropriate method was the distribution of the columns in the dataset. Methods such as z-score with mean assume data to be normally distributed. When we applied *normaltest()* to our dataset, we found that our dataset was not normally distributed. In this case, we went for a modified z-score, i.e. z-score with median and having an absolute threshold of 5, i.e. if the absolute z-score value of an attribute for a specific row is greater than 5, then we identify that row as an outlier. To retain critical information for our data the threshold of 5 was chosen. For the treatment of outliers, they were removed since we want our ML/DL models to be robust, and working with outliers can affect them significantly.

(e) Dimensionality reduction (PCA):

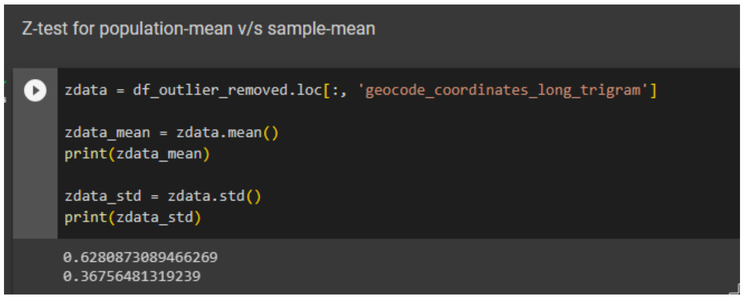
Before applying PCA, we first split the dataset into the training set and a testing set to avoid any leakage between the two. The main issue with applying the PCA algorithm was deciding the number of principal components to use. To resolve this issue we tried to retain as many components that could cumulatively explain about 90% variance of original data.

- Hypothesis:

  Performed a z-test for the population mean for the attribute ‘`geocode_coordinates_long_trigram`’

  Null hypothesis: Mean is equal to 0.6
  Alternative hypothesis: Mean is greater than 0.6

  Using a one-sided critical region with level of significance(alpha) = 0.05 .

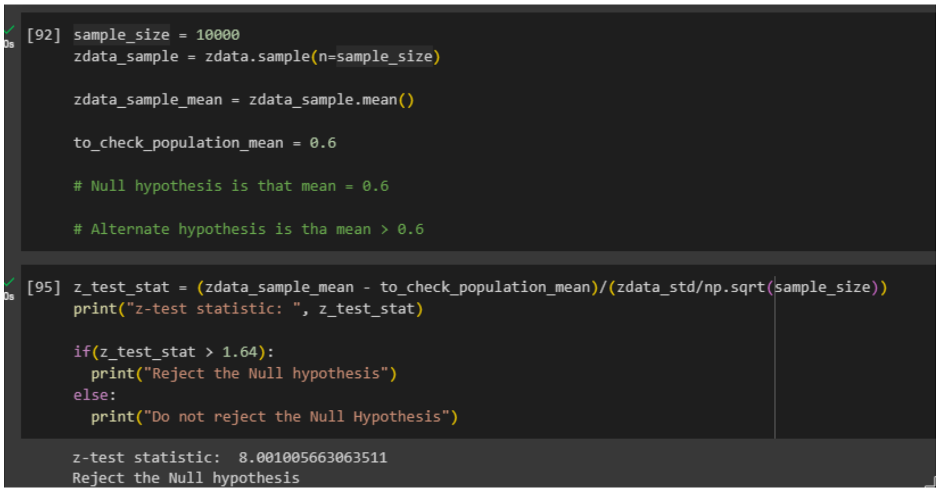

Z-test for population-mean v/s sample-mean

```
zdata = df_outlier_removed.loc[:, 'geocode_coordinates_long_trigram']

zdata_mean = zdata.mean()
print(zdata_mean)

zdata_std = zdata.std()
print(zdata_std)
```

```
0.6280873089466269
0.36756481319239
```



```
[92] sample_size = 10000
     zdata_sample = zdata.sample(n=sample_size)

     zdata_sample_mean = zdata_sample.mean()

     to_check_population_mean = 0.6

     # Null hypothesis is that mean = 0.6

     # Alternate hypothesis is tha mean > 0.6
```

```
[95] z_test_stat = (zdata_sample_mean - to_check_population_mean)/(zdata_std/np.sqrt(sample_size))
     print("z-test statistic: ", z_test_stat)

     if(z_test_stat > 1.64):
       print("Reject the Null hypothesis")
     else:
       print("Do not reject the Null Hypothesis")
```

```
z-test statistic:  8.001005663063511
Reject the Null hypothesis
```

- Inferences in the future:

The inferences and features we wish to learn in the future using ML/DL models are:

(i) Feature importance: We wish to find the importance or relevance of features with respect to labels using models such as decision trees, etc.

(ii) Similarity Index: The objective of the whole dataset is to identify whether two places or spots are the same. If they are the same, then they are labeled as +1 else, they are labeled as -1. The problem with this labeling is that it does not provide any metric about how similar or dissimilar the two places are. Suppose there is a threshold value, say 0.5, and the label is assigned on some metric measure if the metric's value for two places is greater than the threshold value i.e. 0.5, then assign the label as +1 else, assign the label as -1. For instance, suppose the places A and B are a lot closer to each other as compared to another pair; say, C and D are not that close to each other, but let's say they are close enough to be assigned the same label as A and B, i.e., +1.
Thus, we feel a need for a similarity index that ranges from 0 to 1, which signifies the level of similarity between two places

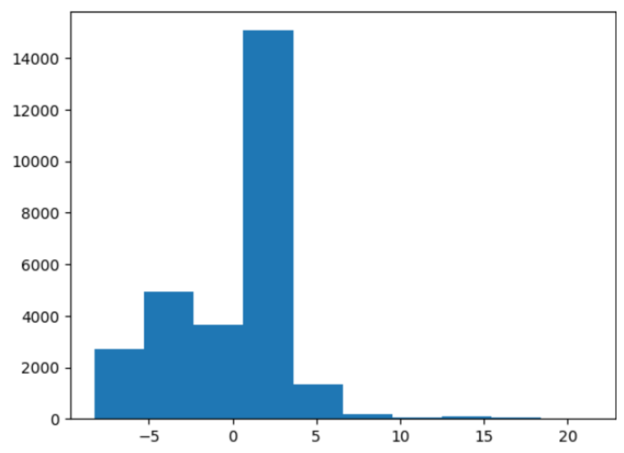
Our problem statement is unique due to the fact that we apply preprocessing not already applied in the existing analysis and the inferences we extract from the dataset have not already been discussed in any of the existing analysis papers.

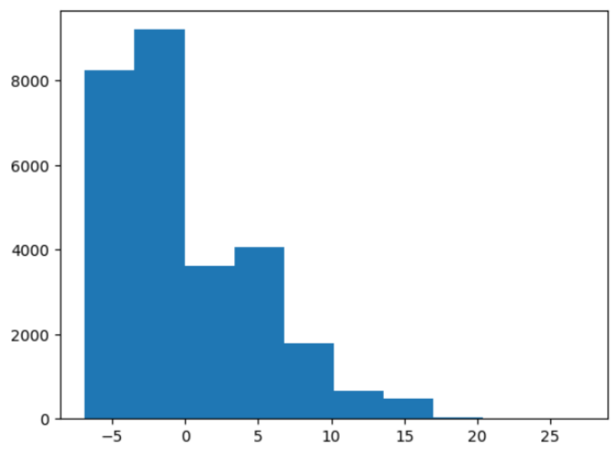
- Appendix:

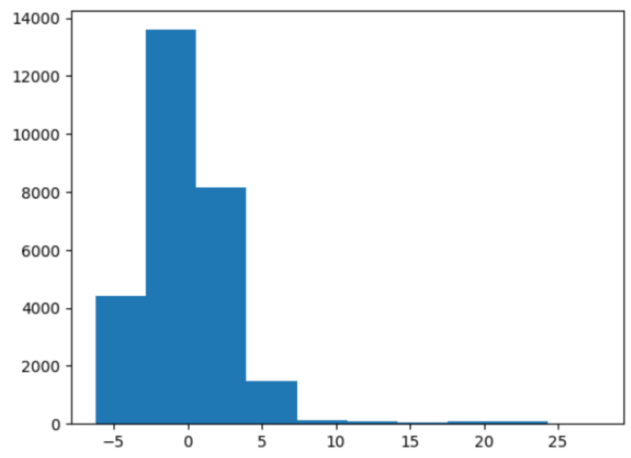
- Existing analysis description:

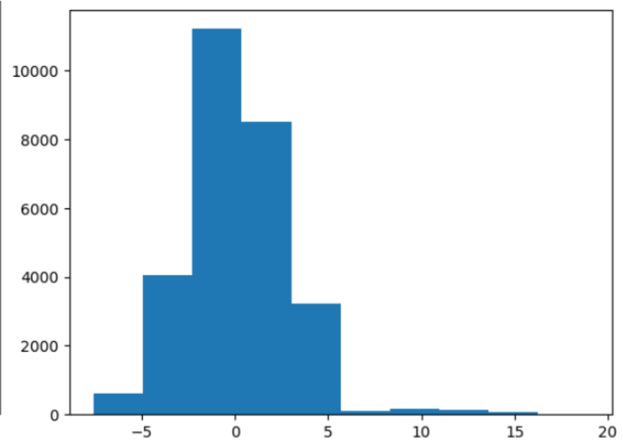
In this paper, the author has used the dataset to create the model for data deduplication process. In this paper, the author talks about the nomao search engine which is used to search for the places and rank the result based on users' likes and dislikes. This has multiple steps involved into it like: extraction and structuralization of local content, query understanding and information retrieval, results ranking, personalization and recommendations, etc. During the first step of content extraction, data comes from multiple sources from the web and might refer to the same location so in order to identify which location is the same or different and aggregate accordingly we have to do a data deduplication process. And for this deduplication process, we are creating a machine learning classifier using active learning that will automate this task. For the creation of the model, we will need the dataset which will capture the distribution of the dataset but we know that it is a real world dataset and real world dataset has some inherent issues associated with it such as :- scalability of the proposed active learning method, representativeness of the training dataset, and practicability of the labeling process. The Nomao dataset has millions of examples and our first task would be to find the most relevant examples and send it for annotation. In this paper, the author has randomly selected 1,00,000 data points and created an Unlabeled Dataset(UD), 29,104 examples to create an initial seed training dataset which is labeled by hand(ID), and 1,985 examples to create a test dataset(TD) which is disjoint from training dataset. As these datasets are real world and created using experimentation they will not the similar distribution and have bias in it which can be detected using a ml algorithm called MODL in which we assign positive class to a training dataset and negative class to the testing dataset and train a ml classification model in it. If our model gives good performance in classifying this new variable created then we can say that both dataset follow different distributions. Then on this training dataset ID we apply boosting of stumps machine learning algorithm and measure the performance on test dataset(TD) we will then notice the performance of the model. Then using an active learning algorithm called marg we select additional 917 data points, label them and retrain the model using margin boosting algorithm and once again check the performance of the model. Then we repeat the above task and select additional 964 and 995 data points using wmarg and wmarg5 respectively, label it, and retrain the model on these data points using wmargin and wmargin5 algorithm and check the improvement in the performance of the model on the test data(TD). We will see the error rate dropping from 6% to 5% and finally coming down to 2.33% and we came to the conclusion that model performances could be improved using better active learning methods.

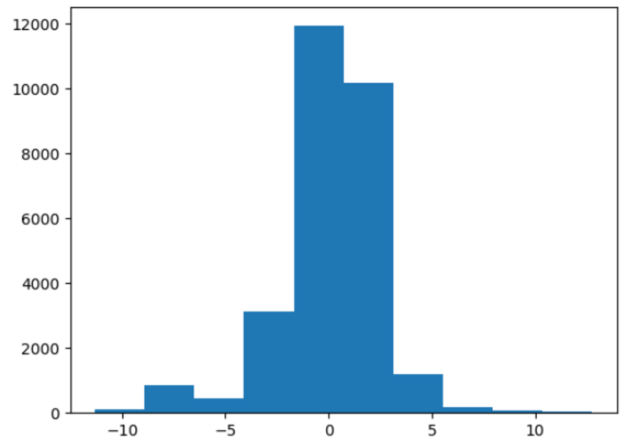
- Data distribution of PCA components:

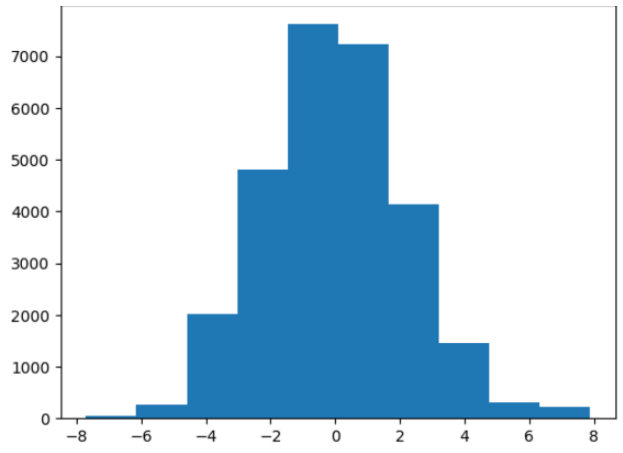












- Code and relevant information:

Project details