DIQ Project Report

Project ID: 49

Project Number: 2

Dataset: *house.csv*

Student: Felipe Azank dos Santos (10919711)

Task: Classification

1 - Setup Choices

1.1 - Machine Learning Models

The two selected machine learning models are: K Nearest Neighbors (KNN) and Support-Vector Machine (SVM).

This choice was made once that KNN is known for not handling outliers well, since its a non-parametric model, which leads to problems in extrapolating assumptions for very different data (does not create a function about the data). The SVM, on the other hand, is a parametric one, which is good for extrapolating assumptions, but not good enough to not show changes in the model. Studying them may help understand clearly the impact in each type of model. Tree based Models were not chosen since they usually handle well (specially ensemble models) outliers.

1.2 - Chosen ML performance evaluation metrics

In order to evaluate the process, we will use accuracy, precision. The last one should be added to make us able to detect any classification bias in the dataset. However, given that the problem consists in a multi-labeled classification problem, the precision results should be considerably low in comparison to the accuracy.

1.3 - Outlier detection techniques selected

In order to try increasing the model accuracy back to its original process, it's applied two outlier detection algorithms in order to correct the dataset:

- Local Outlier Factor (LOF)
- Connectivity-Based Outlier Factor (COF) Algorithm

1.3.1 - Local Outlier Factor (LOF)

Algorithm that stipulates a measurement on how isolated a given point is from its neighbors. This is a good measure because the outliers detected are located outside the

min-max values from each column. The application is based in the code developed by the professor during the exercices section

1.3.2 - Connectivity-Based Outlier Factor (COF) Algorithm

This technique is an improvement of the LOF algorithm, in which a similar measurement of isolation is calculated and then the values are sorted and ranked to detect the outliers in the data. The goal of using this algorithm is to compare if it does a better job than its simpler version. Its implementation can be found in the *pyod* library.

2 - Pipeline Implementation

The detailed procedures can be seen in the notebook. In summary, the procedures taken are a simple implementation of a common pipeline for machine learning models:

1. Data and simple python libraries importation
2. Basic Analysis of data types, missing value and data distribution
3. Outlier imputation from the library given
4. Applying train_test split in the data
5. Feature Engineering process: applying One Hot Encoder in categorical variables and Standardizing numerical values.

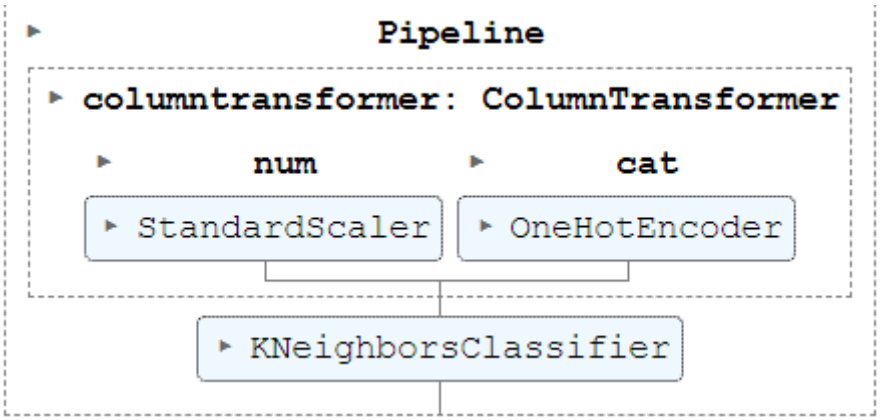

Fig: model pipeline KNN

6. Fitting the model to the dataset in order to get the evaluation metrics from before the cleaning process
7. Outlier Detection implementation of both algorithms
8. Outlier Handling: changing the values identified in the columns by the mean of the column
9. Fitting of the ML models with the new data cleared
10. Comparison of results

All steps above are well defined and explained in the notebook, along with the implementation code.

3 - Results

	Data_Frame	KNN_accuracy	KNN_precision_	SVM_accuracy	SVM_precision
0	10% outliers	0.854305	0.146259	0.741722	0.281336
1	10% outliers_LOF_Fixed	0.867550	0.145556	0.768212	0.298421
2	10% outliers_COF_Fixed	0.854305	0.145270	0.761589	0.297055
3	20% outliers	0.854305	0.187785	0.788079	0.338294
4	20% outliers_LOF_Fixed	0.880795	0.313333	0.801325	0.333073
5	20% outliers_COF_Fixed	0.867550	0.145556	0.807947	0.350033
6	30% outliers	0.867550	0.313063	0.834437	0.305556
7	30% outliers_LOF_Fixed	0.887417	0.314318	0.841060	0.309036
8	30% outliers_COF_Fixed	0.874172	0.145695	0.827815	0.315263
9	40% outliers	0.874172	0.230856	0.761589	0.323907
10	40% outliers_LOF_Fixed	0.834437	0.229021	0.781457	0.326417
11	40% outliers_COF_Fixed	0.880795	0.258634	0.768212	0.324977
12	50% outliers	0.854305	0.145270	0.834437	0.388773
13	50% outliers_LOF_Fixed	0.867550	0.201952	0.841060	0.402564
14	50% outliers_COF_Fixed	0.874172	0.145695	0.834437	0.370447

Fig: results from the models

After evaluating the results, it is possible to see that, on a general scale, the accuracy of the models increased and the precision in some cases increased drastically. However, there are cases where the opposite situation happens, leading to believe that some outlier values were not substituted properly.

One phenomenon that could explain the difficulty to collect the clear outliers is the *Dimensionality Curse*, process in which, given the high dimensionality of the dataset (multiple columns), the computing of the distance between points begins to grow out of control, becoming extremely hard to select what is a big difference, and what is indeed an outlier.

Two ways to dealing with that can be brought to discussion for further improvement of the outlier detection models:

- 1- Apply Principal Components Analysis (PCA) and try reducing the dimensions: this way the dimensionality curse can become weaker, however the outlier measures will be mixed up with other columns

- 2- Apply the outlier detection procedures column-wise: so that we identify the index of each problem given

In conclusion, it's possible to see that the outlier handling was indeed helpful in changing the precision metric and had an effect on the accuracy of the model (However not big enough).