Group #6 Assignment 2

Mahmut Osmanovic Isac Paulsson Sebastian Tuura
Mohamed Al Khaled

January 2, 2025

Contents

1 Introduction

The report covers the two tasks. The first task is a classification task, whilst the other a deals regression. The classification task is a binary classification problem. The objective is to classify individuals as having an income above a set threshold or below it. The regression task focuses on predicting the median monetary value of houses in virtue of properties related to the houses. The houses are located in California, the price is the median price of a block of houses. Note that the models are trained on the same data as to enable as fair model comparisons.

The classification is done in virtue of a set of attributes about them. The set income threshold is below or equal to $50K$ or above it. The parameters about the individuals include information such gender, years of education, marital status and hours worked a week. The target column in the regression task is *median-house-hold income*. House attributes include, but is not limited to, *ocean proximity*, *bedroom count* and *housing median age*.

2 Task 1: Classification

2.1 Data Pre-processing and Feature Selection

The dataset consists of the attributes that are specified in table 1.

Attribute	**Type**
`age`	Number (integer)
`workclass`	String
`fnlwgt`	Number (integer)
`education`	String
`educational-num`	Number (integer)
`marital-status`	String
`occupation`	String
`relationship`	String
`gender`	String
`capital-gain`	Number (integer)
`capital-loss`	Number (integer)
`hours-per-week`	Number (integer)
`native-country`	String
`income`	String

Table 1: Attributes and Their Types

First we identified plenty of rows that contained missing values. Since there is an abundance of data points (48842 rows), we decided to simply drop rows with missing data rather than impute the values. There are many rows, trying to find patterns within the missing data and correctly impute the missing values should not change the final result dramatically (since data is not scarce). Missing data was identified with question marks *?*. Afterwards, we colored the target column with blue and red for the two classes. All numeric data was *min-max* normalized. This is because some modeling techniques that were utilized (such as K-Nearest-Neighbors) require normalized that in order to achieve reliable results. For others, it is highly recommend, for example, using Logistic Regression. In Logistic Regression, normalization of data can aid in convergence efficiency. The Gender attribute, since it is binary (*male* or *female*) was numerically encoded using a rule engine (*males = 1* and *females = 0*). Other multi-class nominal columns were one-hot encoded.

Figure 1 highlights some of the insights in our initial data exploration phase. For example, the largest employers of each subcategory was the private sectors. It was

the most frequent employer for both high and low income earners.

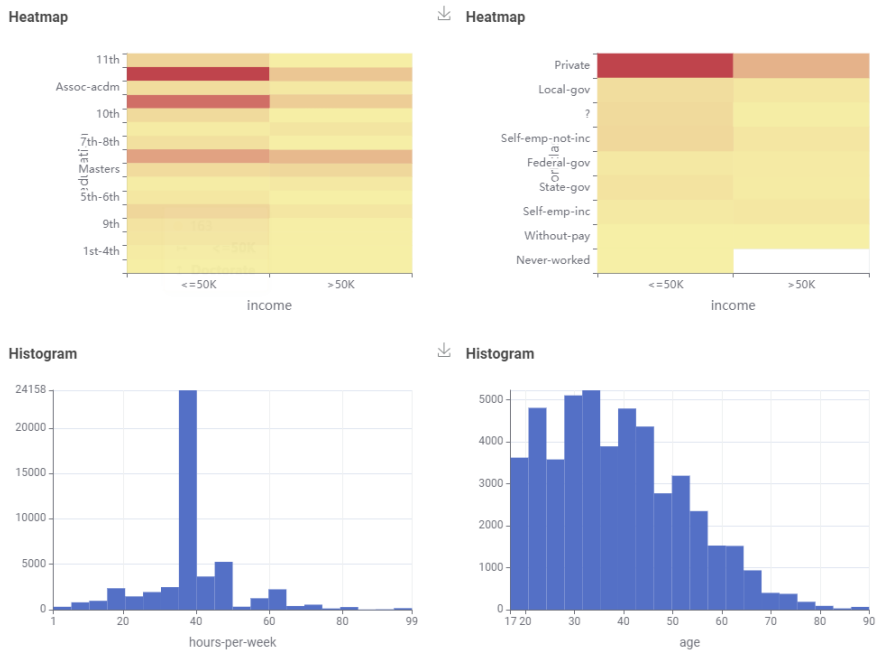

Figure 1: Data Visualization

One can also notice that most individuals work usual *40* hours work weeks, with some outliers working much more than that and others less. Although age is positively skewed, that is to be expected. Humans work less in older ages. There was a correlation between highly educated people and high income.

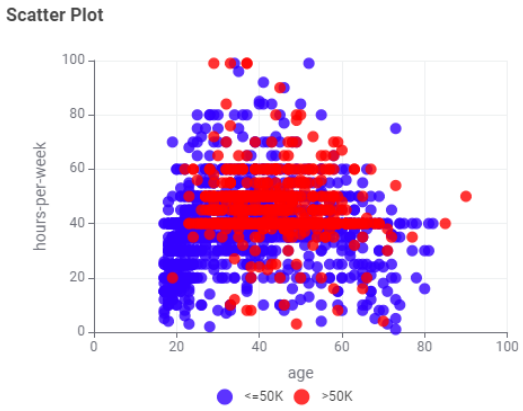

Figure 2: Highlights the relationship between hours worked, age and income.

Figure 3 highlights the relationship between age and hours worked in a scatter plot. As expected, very old and fairly young individuals don't work as many hours as middle aged people. Old people often have physical restrictions whilst many young people are undertaking higher education. Young people rarely ever make above $50K$. Those which make above $50K$ rarely work less than 40 hours a week.

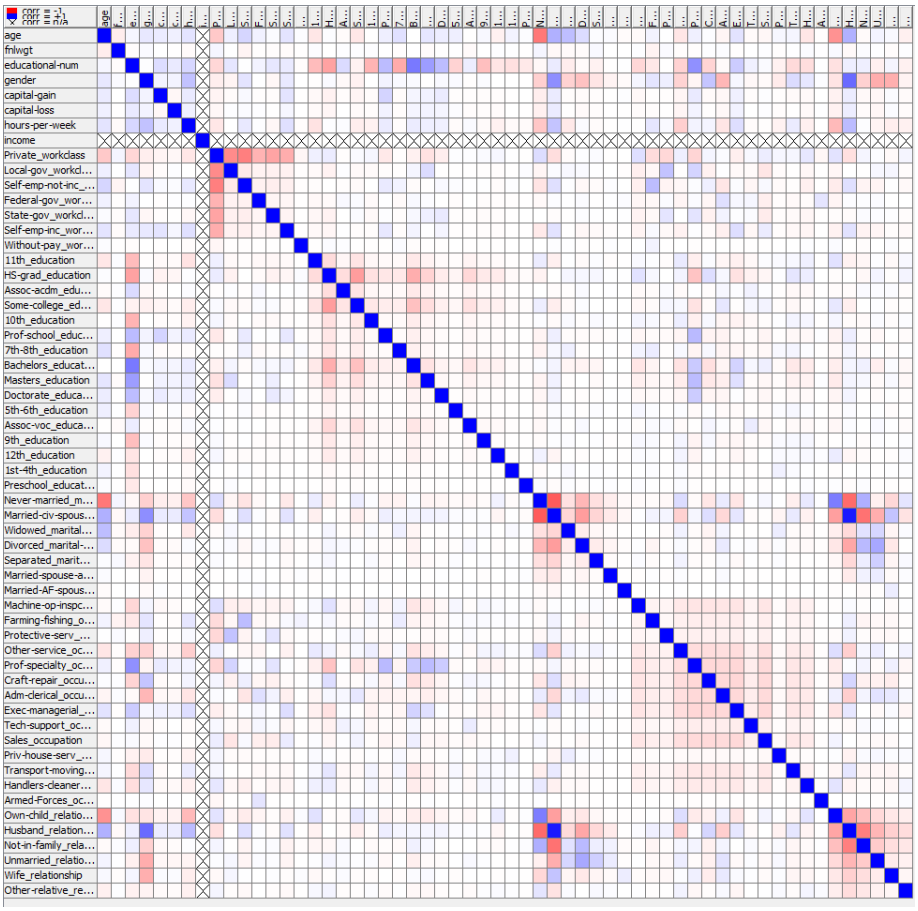

Figure 3: Highlights the Linear Correlation between Non-Target Attributes

Even though there are some significant linear relationships between some of the attributes, the same attributes have almost no correlation with a multitude of others. Meaning that even if there is some shared information carried within the various attributes, there are many of newly introduced relationships. Hence, we kept all attributes, and if anything, made sure to transform them.

2.2 Machine Learning Classifiers

Three models were used to classify the data points as either being above the set income bracket or below it. The utilized three classification models were: Logistic Regression, Decision Trees and Random Forest. There are several reasons as to why those specific classification models were chosen. Logistic Regression was chosen due to the structure of the problem itself. Logistic Regression ought to be used in binary classification problems, which is precisely what this task is (individuals either belong below or above the set income threshold). Although, it does require the conversion of the non-numeric data-types to numeric ones. It operates on numerical matrices for computations. KNN was chosen in virtue of our curiosity. Are we able to get good performance out of it? It also requires all attributes to be numerical since its use of distance metrics. The choice of the Random Forest Classifier was due to its ability to deal with complex relationships between attributes. The income is rarely based on a single attribute.

2.3 Model Comparison

The X-Partitioner node, together with the X-Aggregator was used to implement 10-fold cross validation. Cross validation does not just generate one partition of data into test and train sets. Rather, for 10 folds, it splits the data into 10 as equal parts as possible, trains a model on 9 out of the 10 and subsequently tests on the hold out set. This process is repeated 10 times, so that each part has been used as a test set once. Its results are more generalizable. If one just splits a dataset into two parts, one may by chance end up with a partitioning which is either over or under-optimistic. Cross-validation averages the results such that the final outcome achieves better generalization.

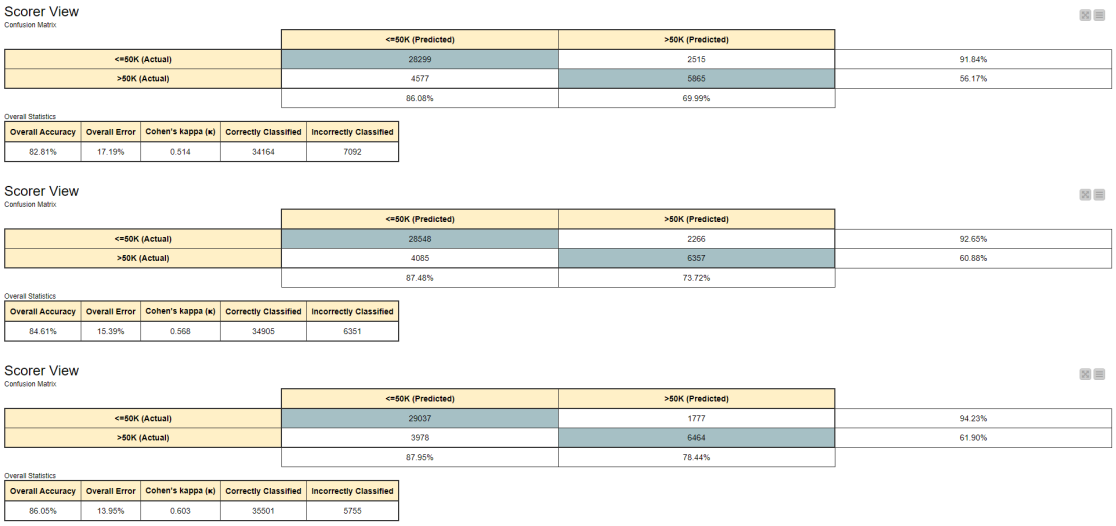

Figure 4: TOP = KNN, MID = Logistic Regression, BOT = Random Forest

All models performed well, their accuracy varying between about 82% and 86%. Likewise all handled the positive class fairly well (a salary less than or equal to $50K$), with a classification success rate over 90% in all models. However, in the same manner, all struggled with high-income earner, with a classification success rate of about 60%. This is very likely due to the class imbalance. About $\frac{3}{4}$ of the data points belonged to the positive class. In future tasks, we should attempt to either over-sample the minority class (using for example SMOTE) or just randomly drop data points from the majority class. However, it might also be the case that predicting high income earners is difficult, and might require additional attributes that aids in explaining the target variable variability.

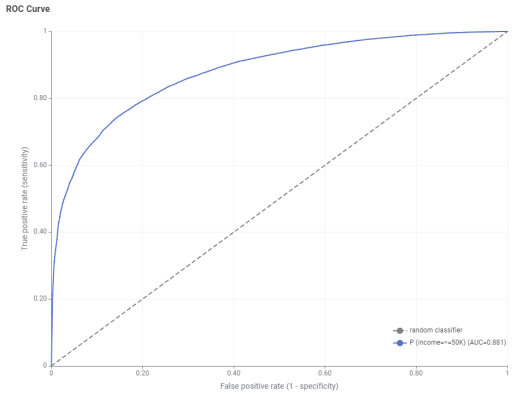

Figure 5: ROC Curve KNN

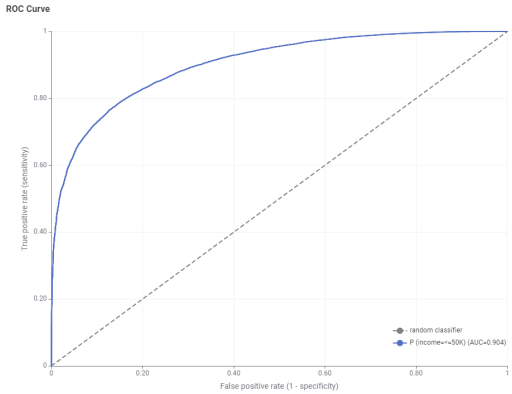

Figure 6: ROC Curve LR

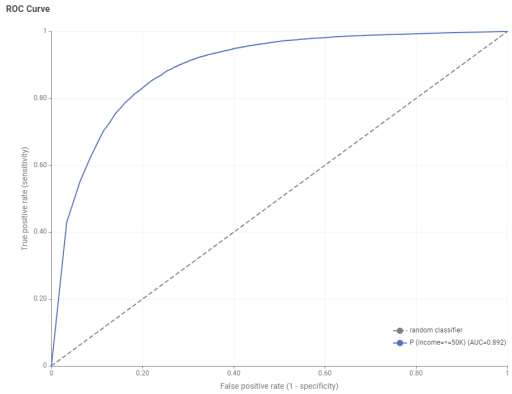

Figure 7: ROC Curve RF

Notice that Random Forest outperforms Logistic Regression in all numeric scores in figure 4, however, the AUC in the ROC plot is ever so slightly larger, how come? The ROC curve is typically plotted using the predicted probabilities rather than just the raw overall class predictions. The probabilities generates by Logistic Regression are every so slightly more better than the ones generated by the Random Forest Classifier. The probabilities in turn are generated by varying the threshold that is set for which values below it are counted as belonging to one of the two binary classes and if above, then belonging to the other. The probabilities are than generated in virtue of a specific classification at that specific threshold. A larger AUC area indicates that the model is in general better at maintaining a good balance between the true-positive-rate and false-positive-rate. However, since the AUC score is very comparable between LR and RF, and given that RF dominates in all other numeric scores, it is to be preferred.

Table 2 summarizes the numerical evaluations that have been elaborated for.

Table 2: Model Performance Comparison

Model	**Accuracy (%)**	**AUC**
Logistic Regression (LR)	84.61	0.904
k-Nearest Neighbors (kNN)	82.81	0.881
Random Forest (RF)	86.05	0.892

2.4 Parameter Optimization

For logistic regression, the chosen solver was *Stochastic average gradient* since it performs better at larger dataset than the alternative. There are several ways in which one can chose the neighbor amount in KNN. One solution is to sample a region of natural integers and inspect the results. Other methods include practical rules of thumb. One such is to use the square root of the amounts of observations as the neighbor count. In our case, that amounted to $\sqrt{37131} \approx 192.69$. However, to aid the classifier, we choose an odd number of neighbors, 191. In Random Forest, we chose a large amount of models, since that equates to more decision trees voting on the available classes. According to the Law of Large numbers, this should increase model performance. The final prediction becomes less and less dependent on any individual decision trees prediction. The noise, the errors in prediction are expected to cancel out. However, we are limited by the computational time it takes to run the RF classifier on our hardware.

3 Task 2: Regression

3.1 Data Pre-processing and Feature Selection

The housing dataset contains data regarding houses in California with the following features: **coordinates, median household age, total number of rooms, population, households, median income, median house value & ocean proximity**. Only one row contained a missing value so it was simply removed. An additional 903 or 4.4% of the total of 20639 rows where also removed, netting us a total of 19736 rows to work with. The named 903 rows where removed using DBScan and the removed rows where labled as "Noise". The DBScan was configured with the following parameters ϵ: 2000 & minimum points: 18

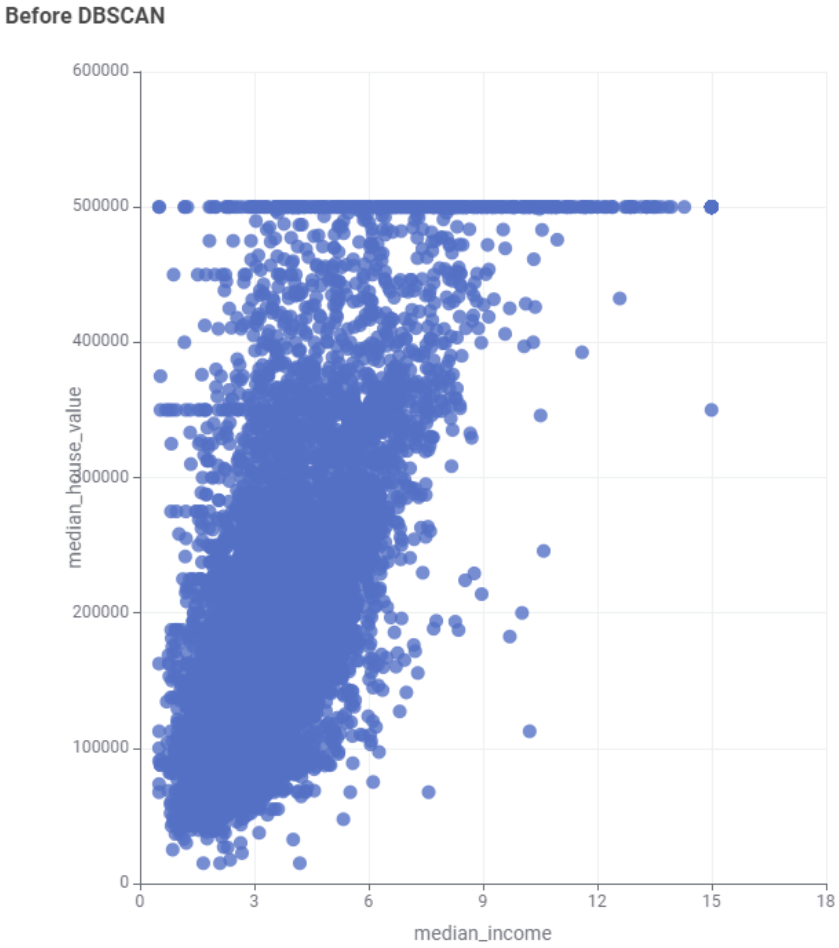

Figure 8: Before DBSCAN

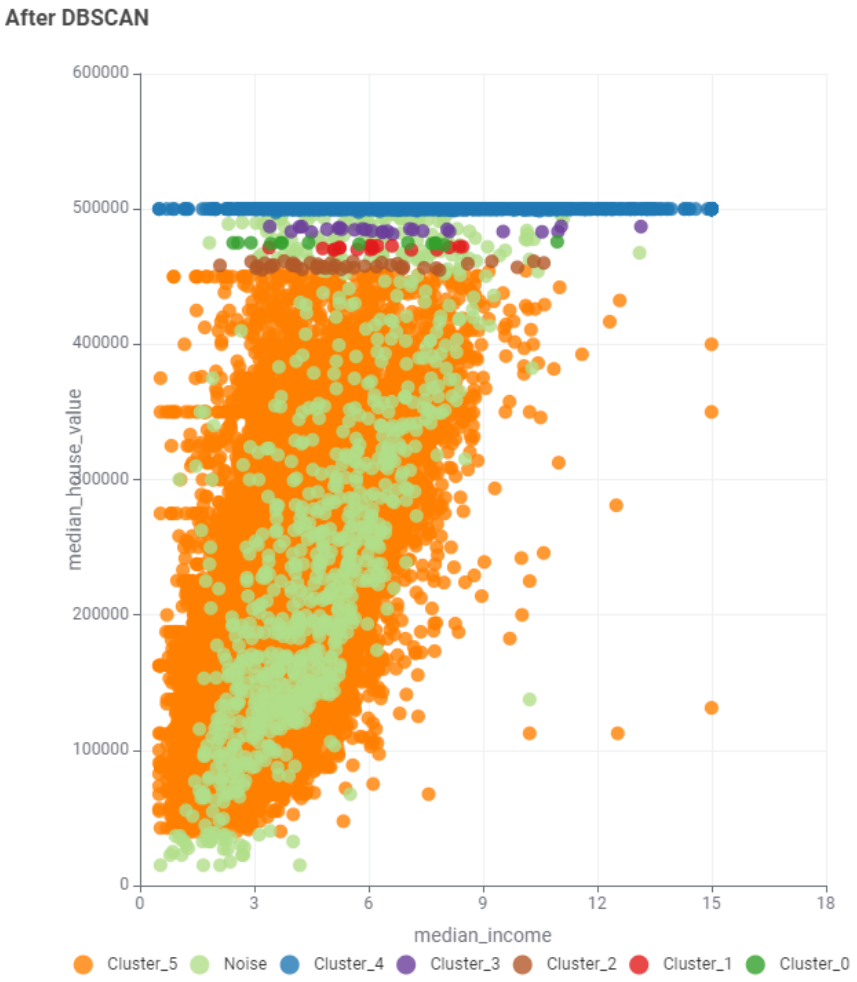

Figure 9: After DBSCAN Colored

Due to the multidimensionality of the graph it looks like the points are overlapping, however viewing it from a different angle it might be more apparent.

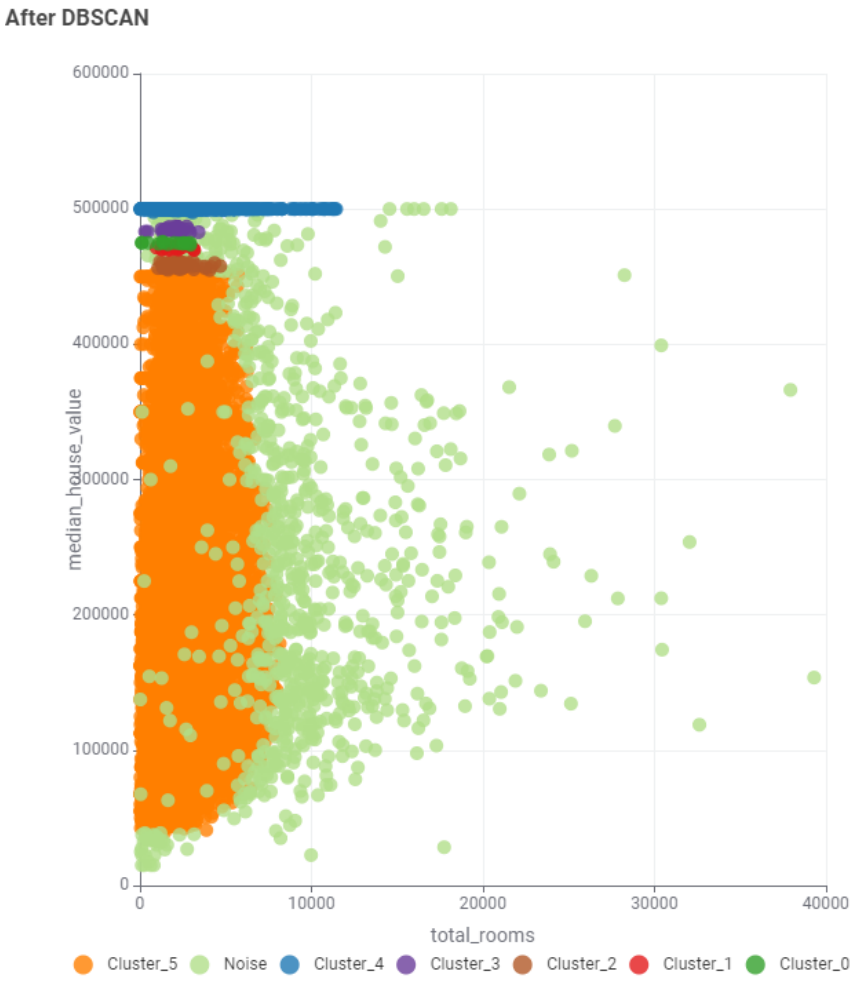

Figure 10: After DBSCAN Colored, Different Angle

Feature selection was done using linear correlation, see the matrix below. Values closer to 1 or -1 represent stronger correlations, while those closer to 0 indicate little connection between the variables. As you will see for median house value none of the values are particularly close to 0, for this reason all parameters where included into the models. Since the regression models only handle numerical values the **ocean proximity** parameter was encoded using one hot encoding.

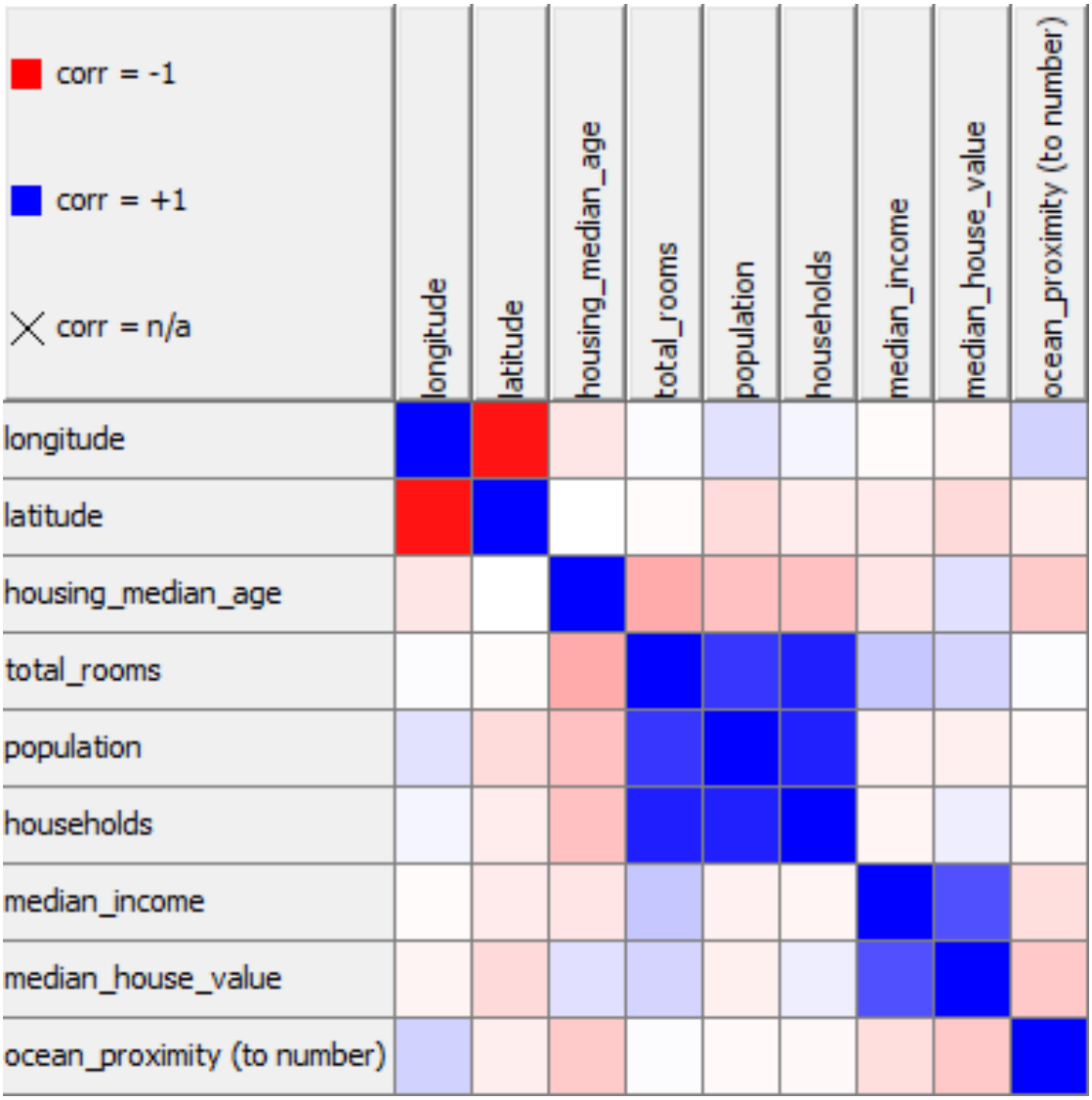

Figure 11: Linear correlation

3.2 Machine Learning Regressors

A total of three different regressors where tested: **Linear, Random Forest & Gradient Boosted Trees Regression**.

When dealing with large datasets the performance of regression models depends on their scalability, computational efficiency & ability to handle high-dimensional

data. Linear regression performs quite well with larger data sets in terms of speed and scalability because of its computational complexity is quite low. Where as Random Forest Regressors can handle large datasets better than a single decision tree due to parallelization during tree construction. However it's computational cost grows significantly with data size as it constructs multiple trees and stores them in memory. Gradient Boosted Trees are specifically optimized for performance on large datasets especially modern implementations the downside being that it requires turning of hyper parameters.

3.3 Model Comparison

With 10 fold cross-validation and without any optimizations done, the regressors scored the following:

1 Linear Regression
   - R^2: 64,4%
   - *MSE*: 0.02

2 Random Forest Regression
   - R^2: 79,2%
   - *MSE*: 0.012

3 Gradient Boosted Trees
   - R^2: 79,7%
   - *MSE*: 0.012

3.4 Parameter Optimization

The Gradient Boosted Trees regressesion model was chosen for the optimization due to the simple fact that it performed the best without any optimizations though not with a large margin. An extention was installed called **KNIME Optimization extension** that included two nodes: **Parameter Optimization Loop Start & End** Connecting these to the model, the start being connected to the X-Partitioner node which was used for the 10 fold cross-validation & the end node being connected to after the numeric scorerer. The start node let's us define flow variables with start, stop & step size variables that we can then feed into the **GBT** model. The stop node will let us choose a variable we want to maximize & will list the optimal

values for the flow variables created earlier.

The following flow variables where created:

- learning_rate
  - start value: 0,1
  - stop value: 0,25
  - step value: 0,05
- number_of_models
  - start value: 100
  - stop value: 500
  - step value: 50

The reason for the given start & stop values are that it takes quite long time to perform the loops. With the cross-validation the loop ran a total of 350 times, 10 times with the cross-validation times 35 with the start & stop nodes. The stop node granted the following "optimal" parameters:

- learning_rate: 0.25
- number_of_models: 450

With R^2: 85.9% & MSE: 0.009. An improvement by 6,2% for the R^2 value & an improvement by 25% for the MSE value. Below you can see the final optimization loop.

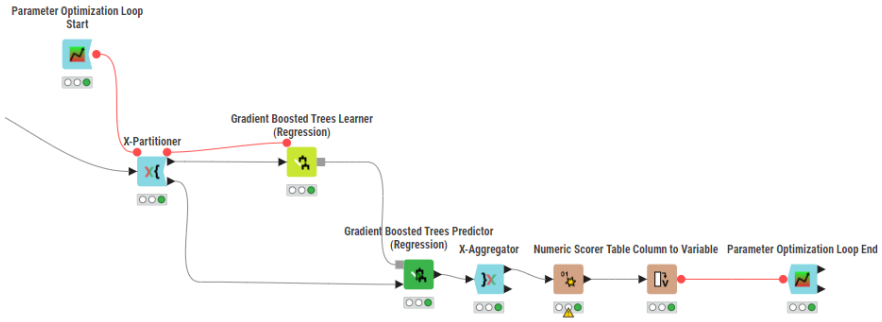

Figure 12: Optimization Loop