Shahid Beheshti University

Anita Soroush

98222085

assignment 1 of Machine Learning

Dataset 1: mobile-price-classification

question 1.

At the beginning the raw dataset contains 21 columns (features) and 2000 rows (records).

a short description of each feacher:

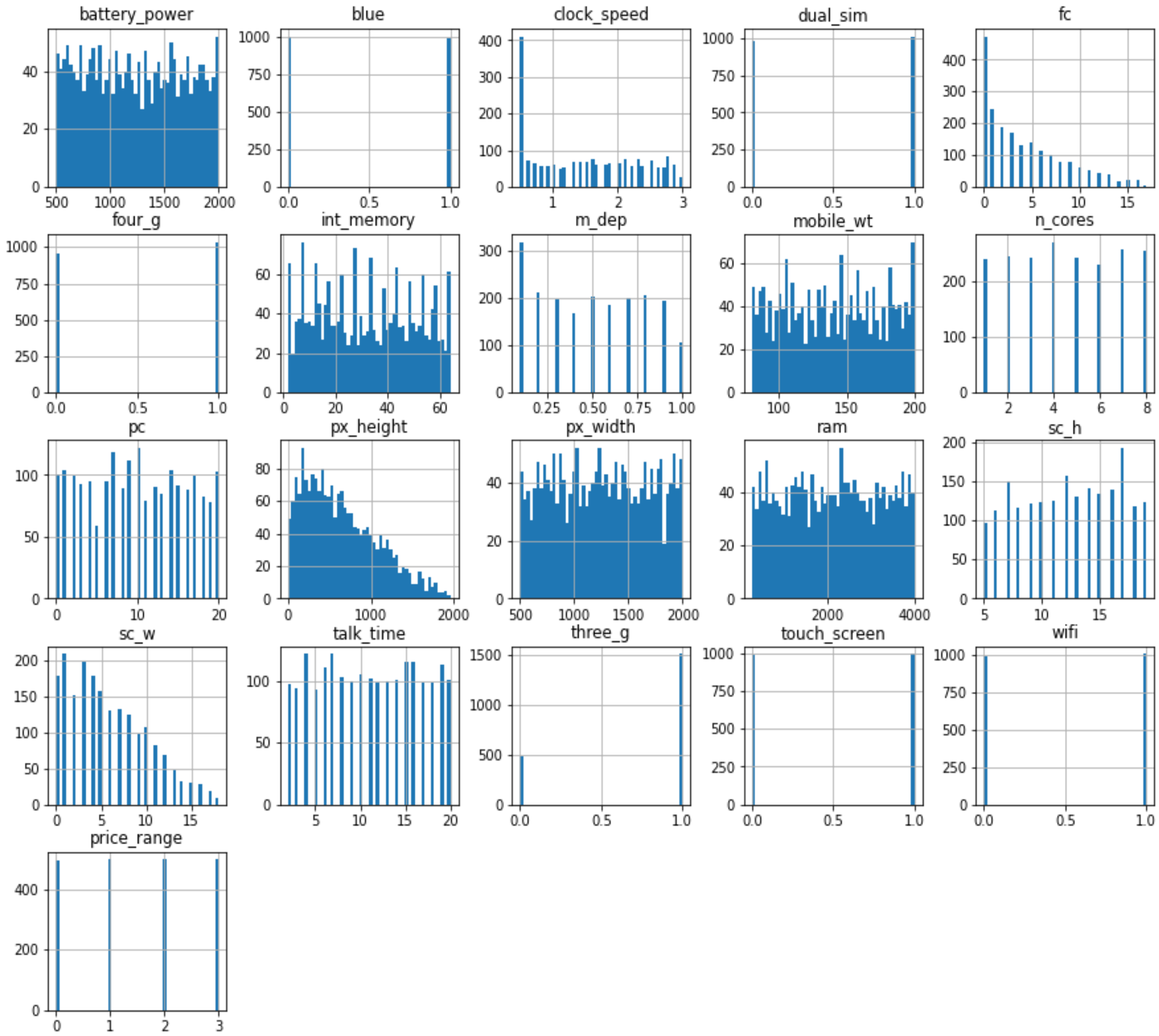
battery_power: **Total energy a battery can store in one time measured in mAh**
blue: **Has bluetooth or not**
clock_speed: **speed at which microprocessor executes instructions**
dual_sim: **Has dual sim support or not**
fc: **Front Camera megapixels**
four_g: **Has 4G or not**
int_memory: **Internal Memory in Gigabytes**
m_dep: **Mobile Depth in cm**
mobile_wt: **Weight of mobile phone**
n_cores: **Number of cores of processor**
pc: **Primary Camera megapixels**
px_height: **Pixel Resolution Height**
px_width: **Pixel Resolution Width**
ram: **Random Access Memory in Megabytes**
sc_h: **Screen Height of mobile in cm**
sc_w: **Screen Width of mobile in cm**

talk_time: **longest time that a single battery charge will last when you are talking**
three_g: **Has 3G or not**
touch_screen: **Has touch screen or not**
wifi: **Has wifi or not**
price_range: **This is the target variable with values of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).**

The dataset **does not involve** any **null** or **duplicated** records.

But what about the **outliers**? (using standard deviation)
An outlier is nothing but the unusually extreme values present in the dataset.
We will see an upper limit and lower limit using 3 standard deviations. Every data point that lies beyond the upper limit and lower limit will be an outlier.
After using this algorithm the number of rows reduced from 2000 to 1988. **So it seems that the raw dataset had 12 outliers.**

question 2.



question 3.

3.1) It seems reasonable if the parameter talk_time has a normal distribution. But does it really have?
To find the answer, I use **Shapiro-Wilk Test** and the result is:

stat=0.947, p=0.000
Probably not Gaussian

3.2) According to the heatmeap shown in the previous question, it seems that there is a positive relation between RAM capacity and price range. But let's check it:
To check this correlation, I use **Pearson's Correlation Coefficient** and the result is:

stat=0.739, p=0.000
Probably dependent

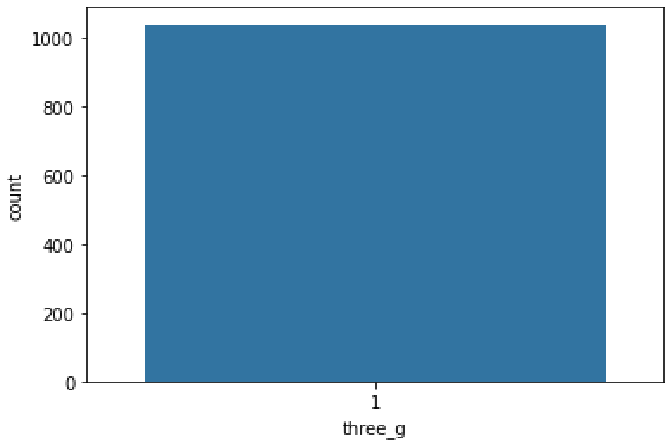
3.3) Is there a significant difference between the cheapest phones and the others when it comes to the mean value of Primary Camera megapixels? In other words do the cheaper phones have significantly weaker primary cameras?
To find the answer, I use **Student's t-test** and the result is:

stat=-1.446, p=0.148
Probably the difference is not significant

3.4) Do manufacturers try to focus on either RAM capacity or the quality of the primary camera? In other words, do these two features have a negative correlation?
To check this correlation, I use **Pearson's Correlation Coefficient** and the result is:

stat=-0.236, p=0.000
Probably dependent

3.5) Before explaining the hypothesis test, I want to mention a point and that is if a device supports 4g Internet, it supports 3g too. The following count plot shows the three_g component of all rows where four_g value is 1:



Moving on to the hypothesis, can we say that the mean price range of phones that support 4g Internet is significantly larger from those that just support 3g? To find the answer, I use **Student's t-test** and the result is:

stat=-0.069, p=0.945

Probably the difference is not significant

StandardScaler : It transforms the data in such a manner that it has mean as 0 and standard deviation as 1. In short, it standardizes the data.

questions **4**, **5** & **8**.

From now on, the data is splitted into 2 parts of training(80%) and testing(20%) and all the confusion matrices which are attached to the report are showing the performance of the model on the **test data**.

First of all, let me briefly explain the idea behind One-vs-One and One-vs-Rest classification. Say we have a classification problem and there are N distinct classes. In this case, we'll have to train a multiple classifier instead of a binary one.

But we can also force python to train a couple of binary models to solve this classification problem. In Scikit Learn we have two options for this, **one-vs-one** and **one-vs-rest** strategies.

I use 2 models for classification:

1) Logistic Regression:

3 important parameters:
1.1) **tol :** ***float, default=1e-4***
Tolerance for stopping criteria.

1.2) **solver :** ***{'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs'***
Algorithm to use in the optimization problem. Default is 'lbfgs'. To choose a solver, you might want to consider the following aspects:

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones;
- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss;
- 'liblinear' is limited to one-versus-rest schemes.

1.3) **multi_class :** ***{'auto', 'ovr', 'multinomial'}, default='auto'***

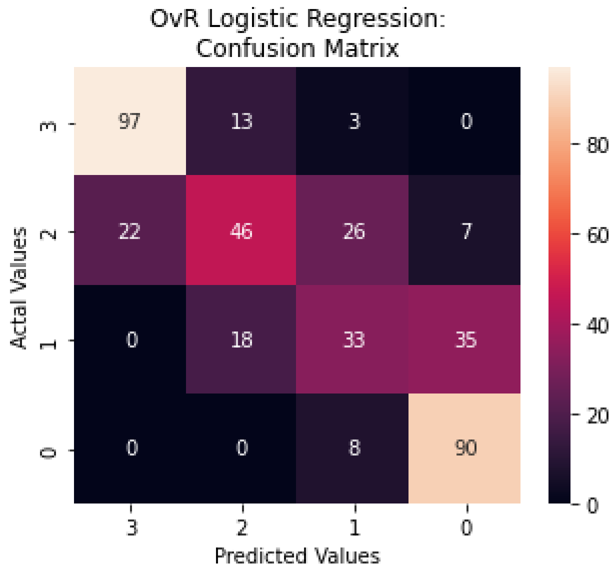
If the option chosen is 'ovr', then a binary problem is fit for each label. For 'multinomial' the loss minimized is the multinomial loss fit across the entire probability distribution, *even when the data is binary*. 'multinomial' is unavailable when solver='liblinear'. 'auto' selects 'ovr' if the data is binary, or if solver='liblinear', and otherwise selects 'multinomial'.

OvO or OvR?

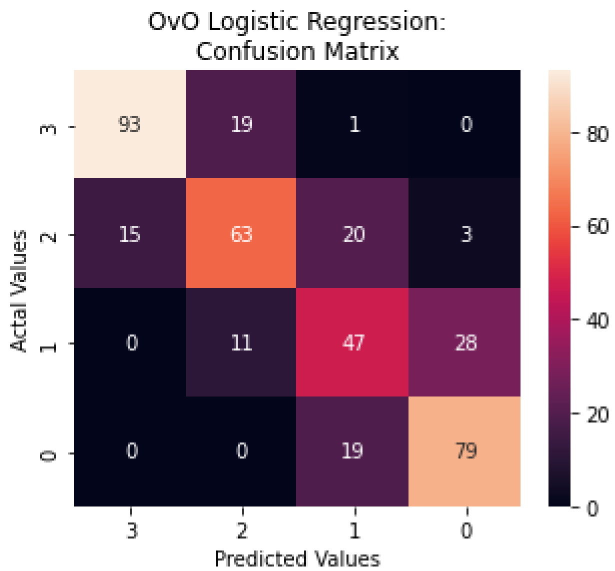
According to the aforementioned **multi_class** parameter, In our case that we have 4 classes of mobile phones price range, in a **default** situation it will go for **multinomial** which is **neither ovo nor ovr**. Instead, the multinomial logistic regression algorithm is an extension to the logistic regression model that involves changing the loss function to cross-entropy loss and predicting probability distribution to a multinomial probability distribution to natively support multi-class classification problems.

After training the model for both strategies, the result is as follows:

Accuracy of OvR Logistic Regression Classifier**:** **<u>0.67</u>**



Accuracy of OvO Logistic Regression Classifier: **<u>0.71</u>**



For this dataset when using Logistic Regression, both strategies worked almost the same, **one-vs-one was slightly better**.

2) SVM:

3 important parameters:

2.1) **kernel:** ***{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'***

Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices.

2.2) **degree:** ***int, default=3***

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

2.3) **decision_function_shape*{'ovo', 'ovr'}, default='ovr'***

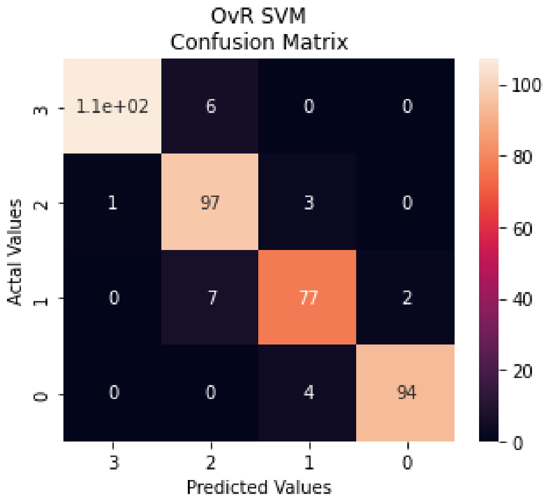
Whether to return a one-vs-rest ('ovr') decision function of shape (n_samples, n_classes) as all other classifiers, or the original one-vs-one ('ovo') decision function of libsvm which has shape (n_samples, n_classes * (n_classes - 1) / 2). However, one-vs-one ('ovo') is always used as a multi-class strategy. The parameter is ignored for binary classification.

OvO or OvR?

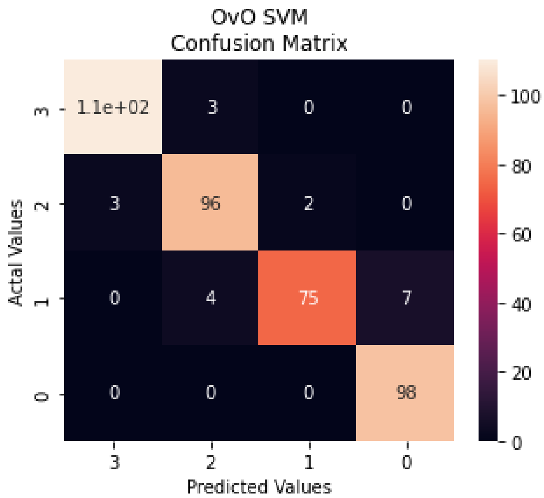
Based on the parameter **decision_function_shape** which was explained beforehand, SVM in scikit-learn package uses one-vs-rest strategy by default.

After training the model for both strategies, the result is as follows:

Accuracy of OvR Classifier: **<u>0.94</u>**



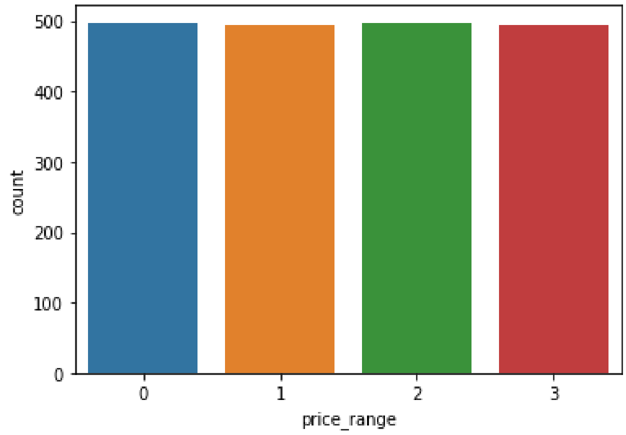
Accuracy of OvO Classifier: **<u>0.95</u>**



SVM came out to be **much more accurate** compared with Logistic Regression. **ovo worked better** than ovr but the difference is negligible.In SVM and Logistic Regression alike, in both strategies of ovo and ovr the first class of price_range (class 0) was detected better than other classes. The reason might be the nature of Logistic Regression which is fundamentally **a binary classifier**; but we should pay attention that the dataset was so small (2000 record) so we **can't** make a general rule that Logistic Regression shows a higher accuracy for the first class in every dataset.

question 6.

The dataset is balanced, since we have 4 classes and each class contains 500 records which is exactly one fourth of the total. this is shown in the following bar chart:



But what if the dataset was not balanced? these are three solutions to combat this problem:

1) Under-sampling:

In this solution we should delete instances from the over-represented class until the dataset is balanced. This deletion can be done in several ways but the one of the easiest and also commonest ways is to delete randomly.

2) Over-sampling:

In this solution we should add instances to the over-represented class until the dataset is balanced. This can be done just by randomly increasing minority class examples by replicating them.

3) More advanced techniques rely on creating new data-points for the minority class to achieve a balanced distribution of classes. SMOTE (Synthetic Minority Oversampling Technique) is one such method.

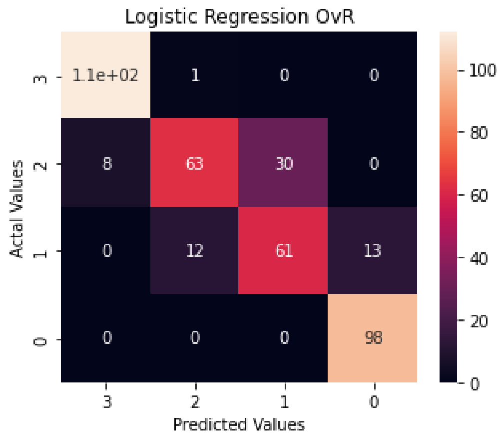
question 7.

In all the following scaling methods, first of all I fitted the scalar on training data and used it to scale both training and test data. After these steps I used the scaled training data to make a Logistic Regression Classifier, and I compared the predictions and the real test data.

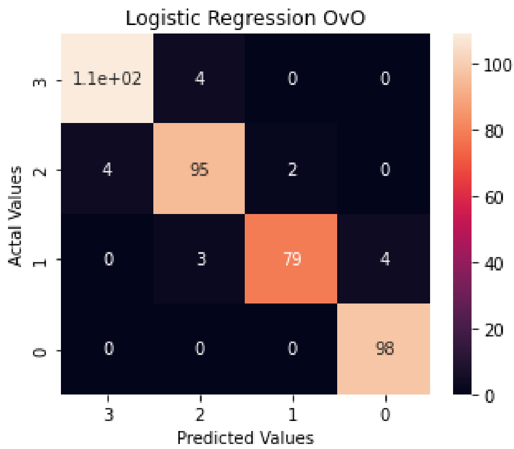
1) Standard Scaler:

It will transform the data such that its distribution will have a mean value 0 and standard deviation of 1.

Accuracy: **0.84**



Accuracy: **0.96**



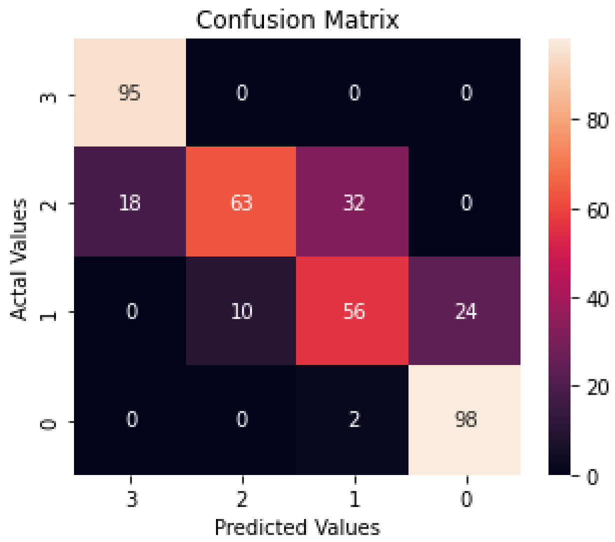
It can be clearly seen that using standard scalar before learning with Logistic Regression in either strategy, **has a very good effect** on the accuracy of the model.

2) Min-Max scalar:

For each value in a feature, MinMaxScaler subtracts the minimum value in the feature and then divides by the range.

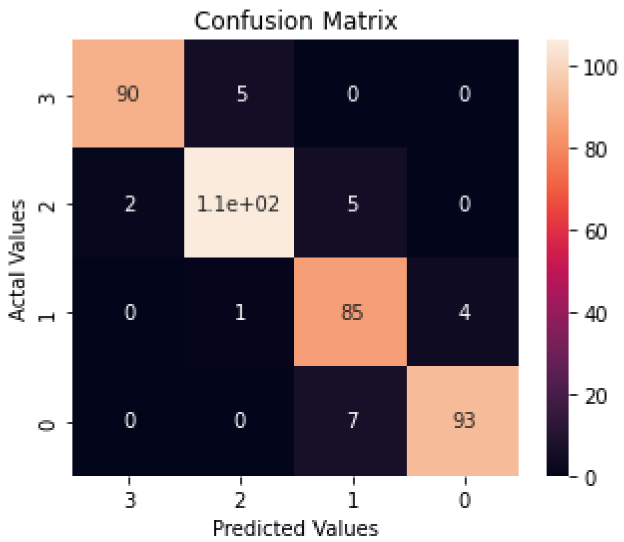
Accuracy: **<u>0.78</u>**

Logistic regression OvR



Accuracy: **<u>0.94</u>**

Logistic Regression OvO

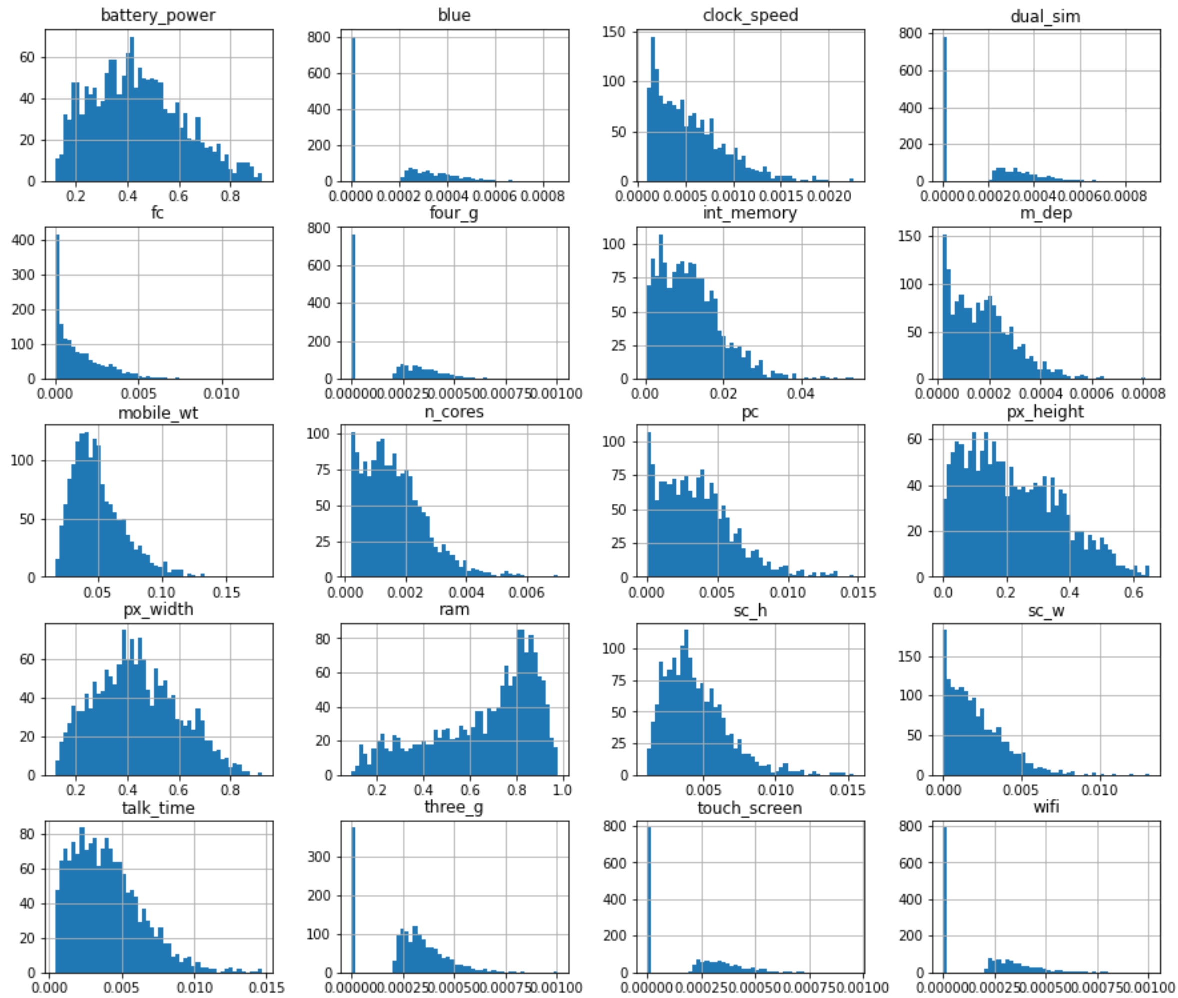


It can be seen that both outcomes have positively changed but **how one-vs-one strategy is impressed is so noteworthy.**

3) Normalizer:

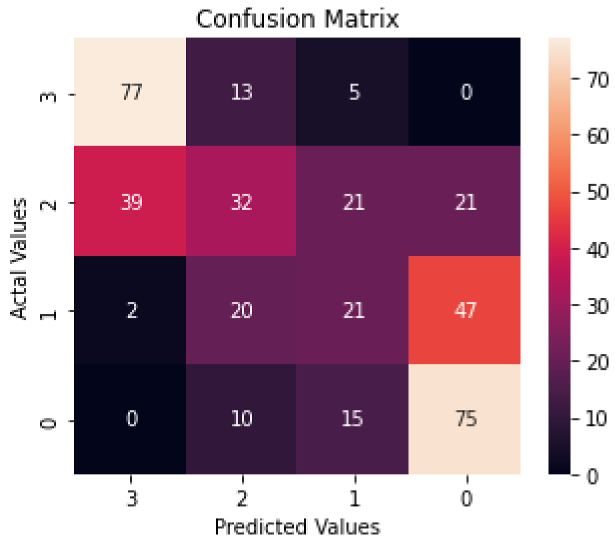
Normalizing samples individually to unit norm. Each sample (i.e. each row of the data matrix) with at least one non zero component is rescaled independently of other samples so that its norm (l1, l2 or inf) equals one.

The effect of normalizing is shown in the following plots:

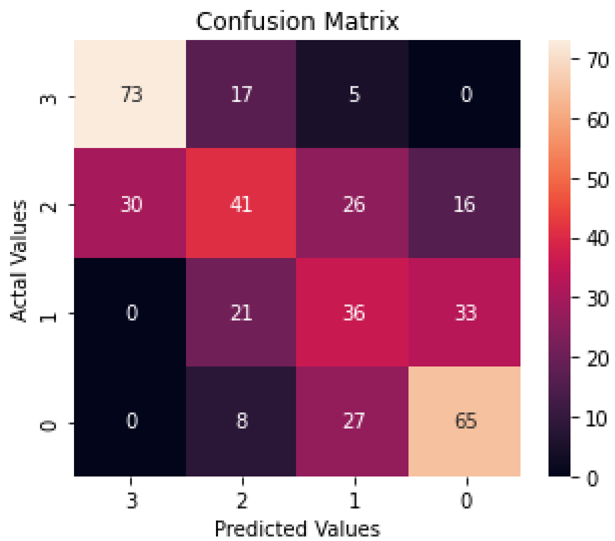


But how does normalizing affect the learning process in Logistic Regression? Check out the matrices below:

Accuracy of OvR Classifier: **<u>0.52</u>**



Accuracy of OvO Classifier: **<u>0.54</u>**



Apparently **it reduces** the accuracy of the model.

question 9.

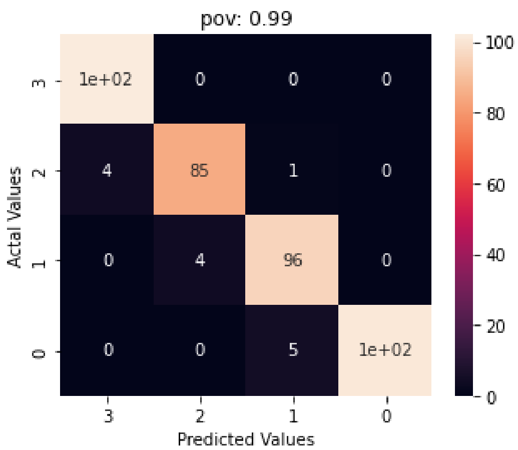
The variance ratio is **the percentage of variance that is attributed by each of the selected components**. Ideally, you would choose the number of components to include in your model by adding the explained variance ratio of each component until you reach a total of around 0.8 or 80% to **avoid overfitting.**

But how to set the amount of pov when using pca?

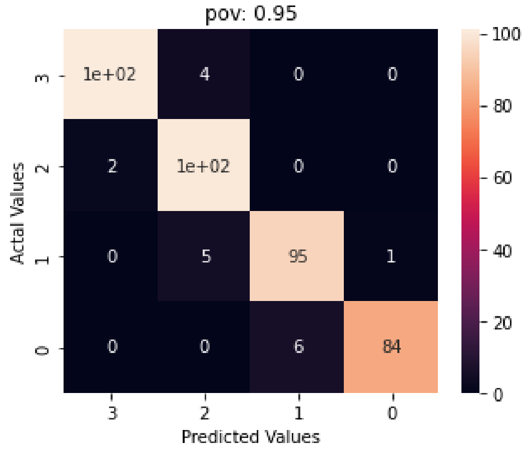
n_components: ***int, float or 'mle', default=None***
If `0 < n_components < 1` and `svd_solver == 'full'`, select the number of components such that the amount of variance that needs to be explained is greater than the percentage specified by n_components.

svd_solver : ***{'auto', 'full', 'arpack', 'randomized'}, default='auto'***

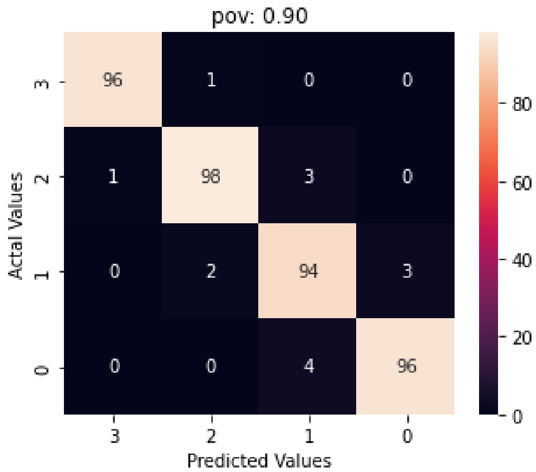
Accuracy of LR OvO Classifier: 0.96
number of principal components: 4



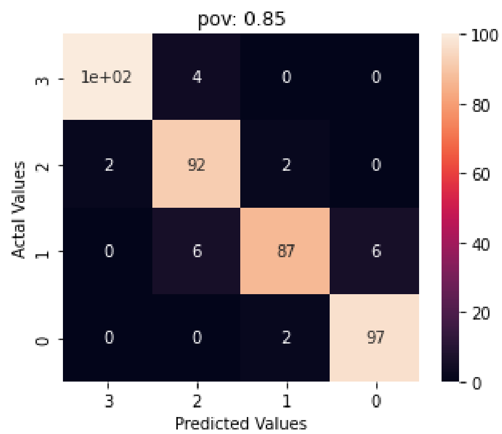
Accuracy of LR OvO Classifier: 0.95
number of principal components: 4



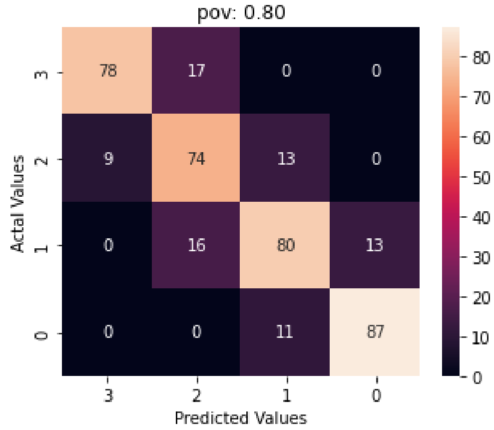
Accuracy of LR OvO Classifier: 0.96
number of principal components: 3



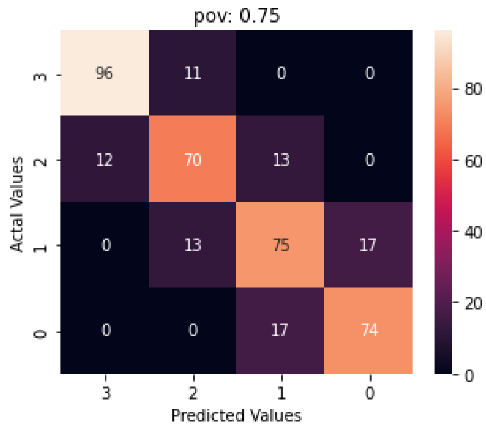
Accuracy of LR OvO Classifier: 0.94
number of principal components: 3



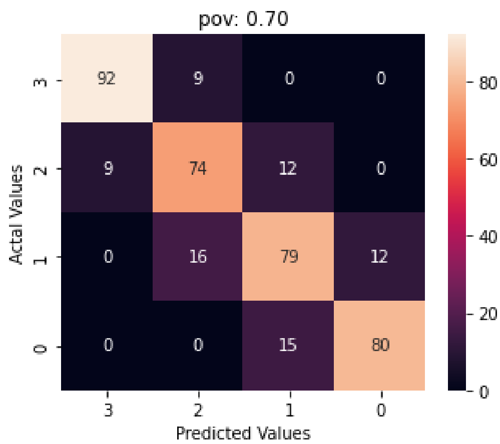
Accuracy of LR OvO Classifier: 0.80
number of principal components: 2



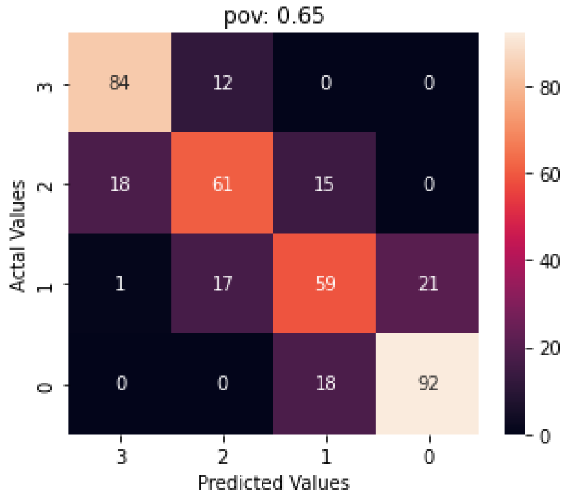
Accuracy of LR OvO Classifier: 0.79
number of principal components: 2



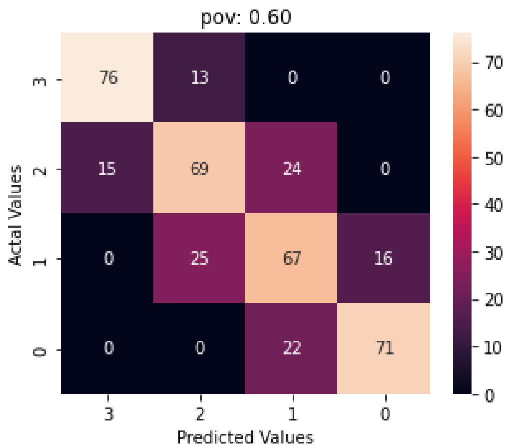
Accuracy of LR OvO Classifier: 0.82
number of principal components: 2



Accuracy of LR OvO Classifier: 0.74
number of principal components: 1



Accuracy of LR OvO Classifier: 0.71
number of principal components: 1

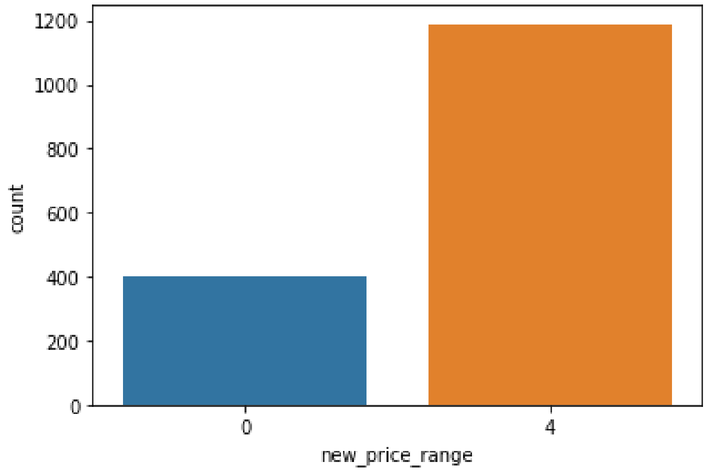


In this dataset, with the reduction of percentage of variance (pov) the accuracy of the model reduced.

Despite making the result even worse, we sometimes need to reduce the number of features (usually using PCA) so that we can **visualize** the dataset in 2 or 3 dimensions. It can help us with choosing more probable methods and more suitable models.

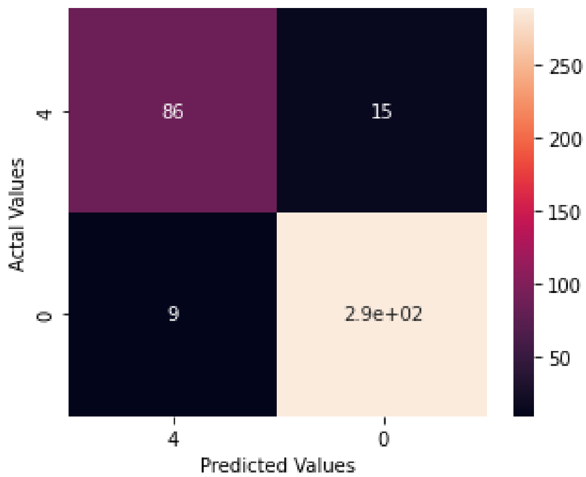
question **10.**

After changing 1's, 2's and 3's into a new label (4), the new dataset is imbalanced, the countplot below clarifies this:

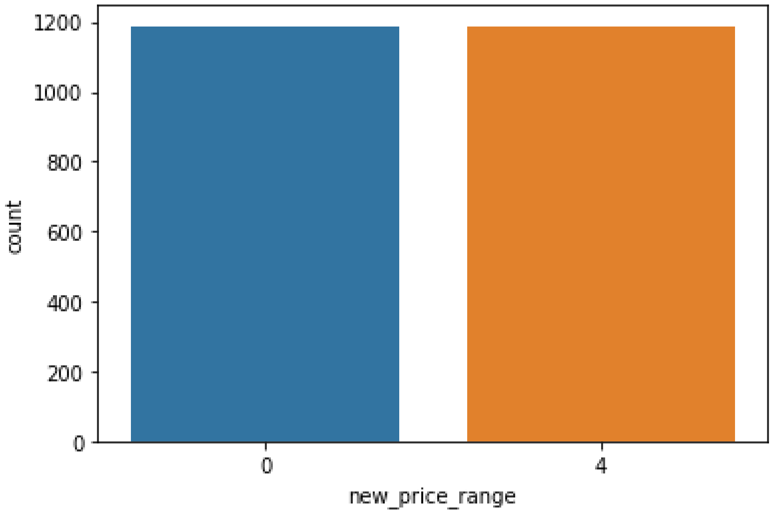


After training the ovo Logistic regression model using this imbalanced data, the result is as follows:

Accuracy: **0.94**

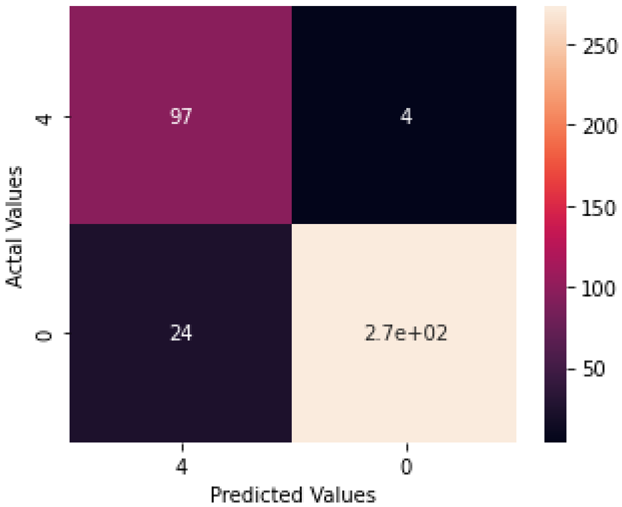


After **oversampling** the **training data** randomly the training part is balanced:



and the training accuracy is shown in the following confusion matrix:

Accuracy: **0.93**



oversampling did not make any specific difference here. maybe because the original dataset is too small (2000 records)

Dataset 2: [apartment-rental-offers-in-germany]

The raw data consists of 49 columns and 268850 rows.

Data Cleaning:

The features are listed below and alongside the percentage of null values for each feature:

regio1	0.000000
serviceCharge	2.569834
heatingType	16.684397
telekomTvOffer	12.132788
telekomHybridUploadSpeed	83.254603
newlyConst	0.000000
balcony	0.000000
picturecount	0.000000
pricetrend	0.681421
telekomUploadSpeed	12.407662
totalRent	15.070485
yearConstructed	21.218151
scoutId	0.000000
noParkSpaces	65.388879
firingTypes	21.188023
hasKitchen	0.000000
geo_bln	0.000000
cellar	0.000000
yearConstructedRange	21.218151
baseRent	0.000000
houseNumber	26.415473
livingSpace	0.000000
geo_krs	0.000000
condition	25.474800
interiorQual	41.906267
petsAllowed	42.615957
street	0.000000
streetPlain	26.413614
lift	0.000000
baseRentRange	0.000000
typeOfFlat	13.618747
geo_plz	0.000000
noRooms	0.000000
thermalChar	39.615399
floor	19.084620
numberOfFloors	36.351869
noRoomsRange	0.000000
garden	0.000000
livingSpaceRange	0.000000
regio2	0.000000
regio3	0.000000
description	7.344988
facilities	19.685326
heatingCosts	68.191185
energyEfficiencyClass	71.066766
lastRefurbish	69.979171
electricityBasePrice	82.575414
electricityKwhPrice	82.575414
date	0.000000

Considering the number of null values and the definition of each feature, I decided to drop some columns at the beginning. The green features are those that I picked to keep and work on.
So the columns are restricted to the following list:

regio1	0.000000
geo_plz	0.000000
heatingType	16.684397
newlyConst	0.000000
yearConstructed	21.218151
cellar	0.000000
livingSpace	0.000000
condition	25.474800
typeOfFlat	13.618747
noRooms	0.000000
garden	0.000000
totalRent	15.070485
hasKitchen	0.000000
lift	0.000000
floor	19.084620

Moving on to **outliers**, these are the numeric features that should be cleaned of outliers: **livingSpace, noRooms, totalRent** and **floor.** (4 columns)
Just like the previous dataset, I used 3 standard deviations, but in 3 different ways.

1) One loop **without chunking:**

number of records before putting outliers aside: 268850
whole run time: **0.1577**
number of records after putting outliers aside: 187513

2) **Chunking up** the dataset into **two parts** and detecting their outliers in two **consecutive** loops:

number of records before putting outliers aside: 268850
whole run time: **0.3365**
number of records after putting outliers aside: 187513

The result is totally reasonable, because we are processing the two parts consecutively and no parallel.

3) **Chunking up** and **parallelism.**

number of records before putting outliers aside: 268850
run time mean: **3.5848**
number of records after putting outliers aside: 187513

It's a little weird but the result got even worse. It may be because of the runtime overhead related to parallelizing.

There is no algorithmic difference between the 3 ways above; So at the end of each of them and dropping the outliers, we've got **187513 records** to keep, out of 268850.

In order to handle the **null values**, I'm gonna fill the NaN's of the columns "heatingType", "condition" and "typeOfFlat" with the fixed expression of "NotAvailable". To achieve this goal, I use 2 different ways.

1) One loop **without partitioning:**

run time : **0.10694**

2) **parallelism** using **Dask:**

Since the columns are independent from each other, I processed them separately at the same time.

run time : **0.00099**

The difference made by dask is impressively significant.

There are still some null records in the feature "yearConstructed" that I'm gonna drop them all.

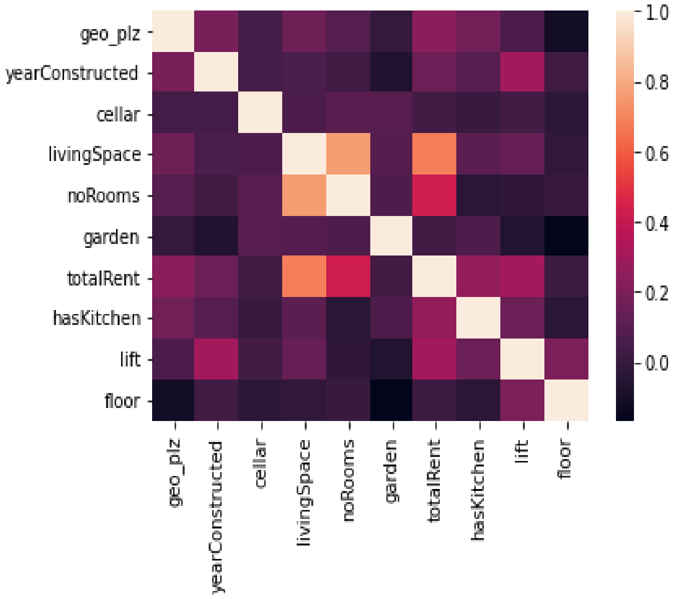
We can also see that 4 records have a "yearConstructed" feature of bigger than 2022, which is **logically impossible**. I drop those rows as well.

After all these cleaning steps the dataset **involves** some duplicated records that I drop altogether.

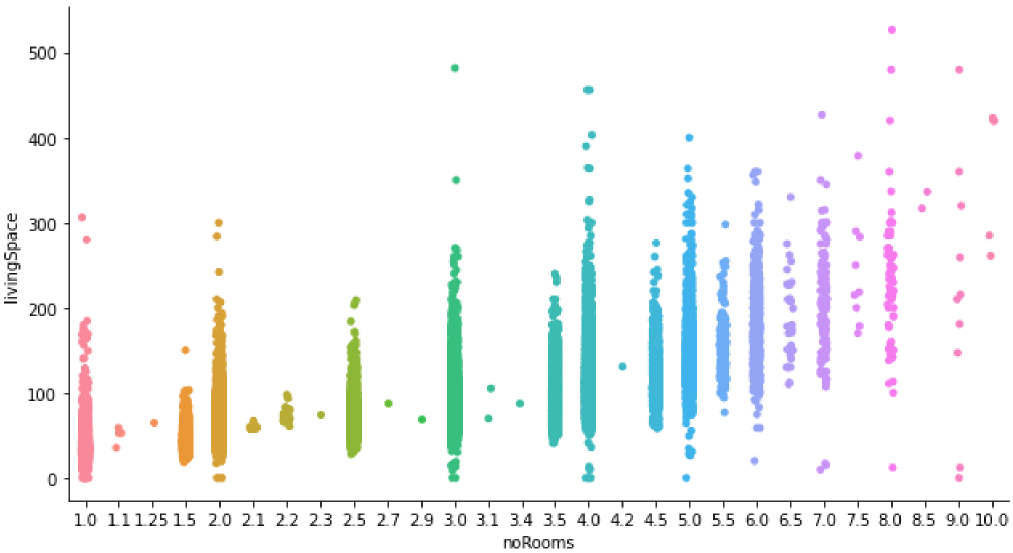
Up to now, we've got 146112 rows and 18 columns of cleaned data.

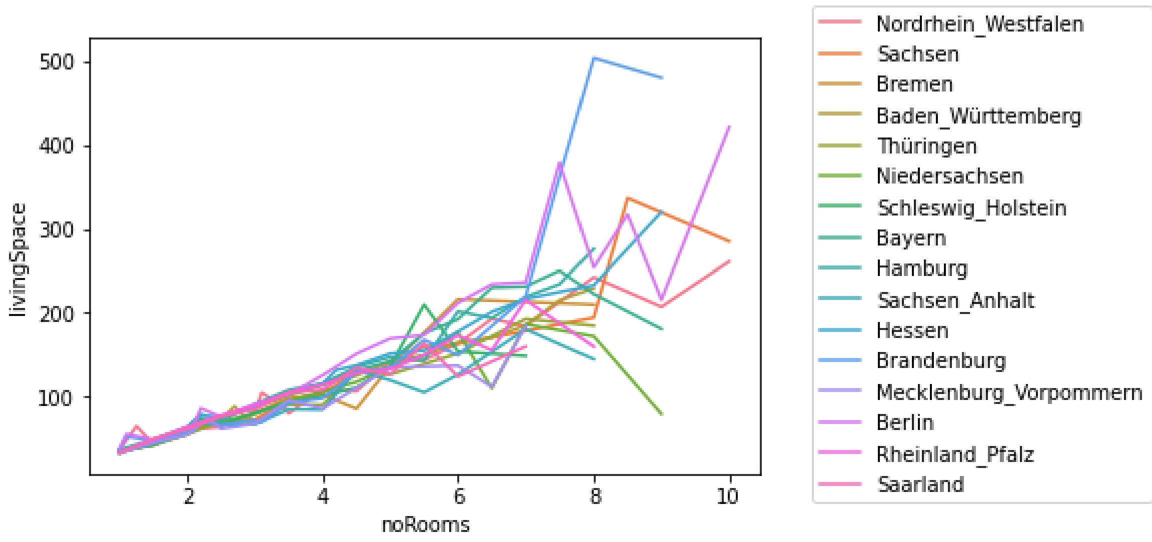
EDA

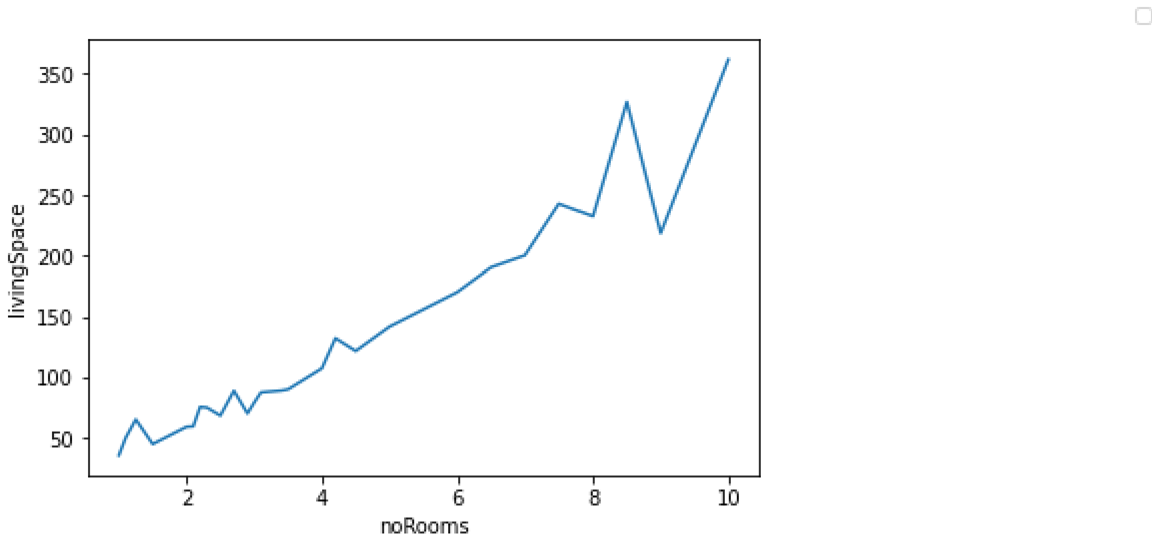
Let’s Start with a correlation heatmap diagram:



The correlation between noRooms and living Space is noticeable. The bigger the living space, the more rooms. The following plots acknowledge this point:

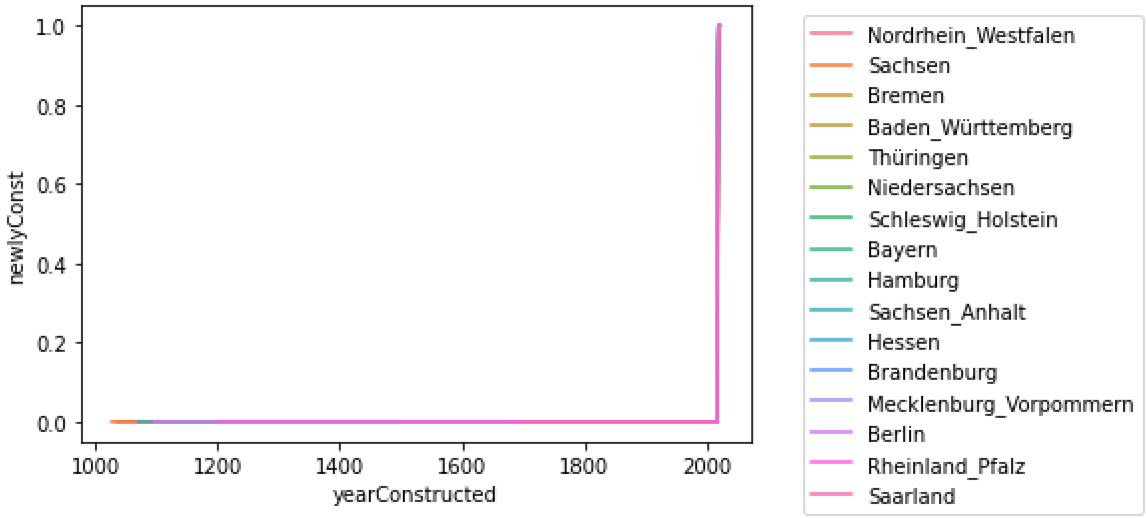




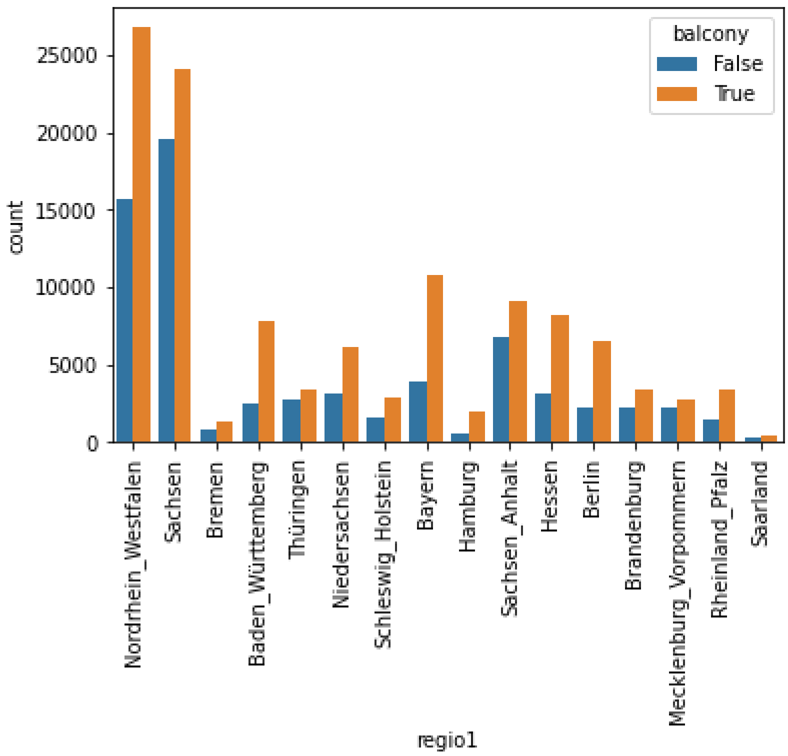


In order to reduce the dependency between the features, I’m gonna drop the noRooms column, because it is represented by livingSpace.

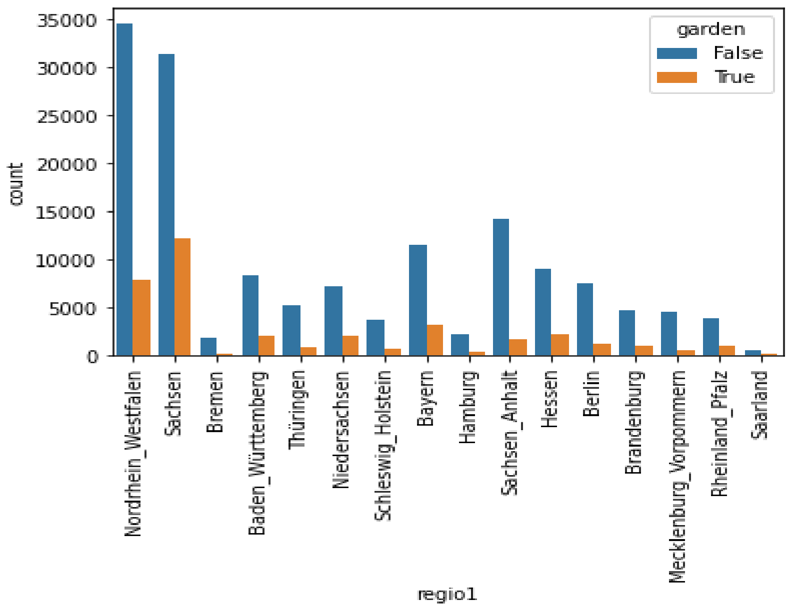
The Line plot below shows that the feature “newlyConst” is 1 only if the construction year is bigger than 2020. So the information of newlyConst can be totally represented by yearConstructed and I’m gonna drop this column as well.



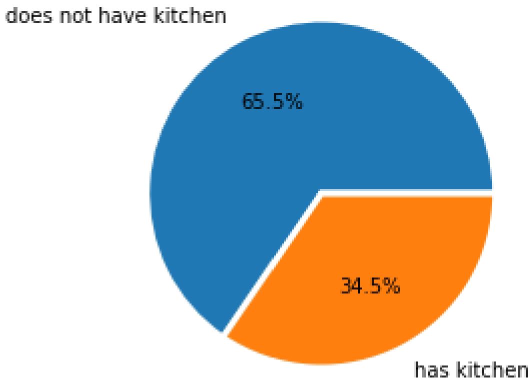
The following barplot shows that in every region the number of apartments that do have balcony is more than the number of those that do not:



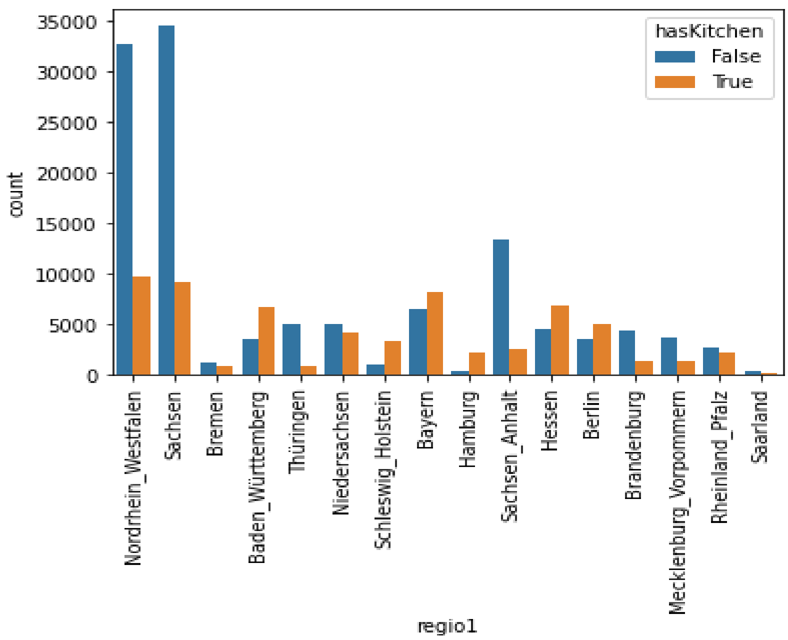
And the apposite is held about garden:



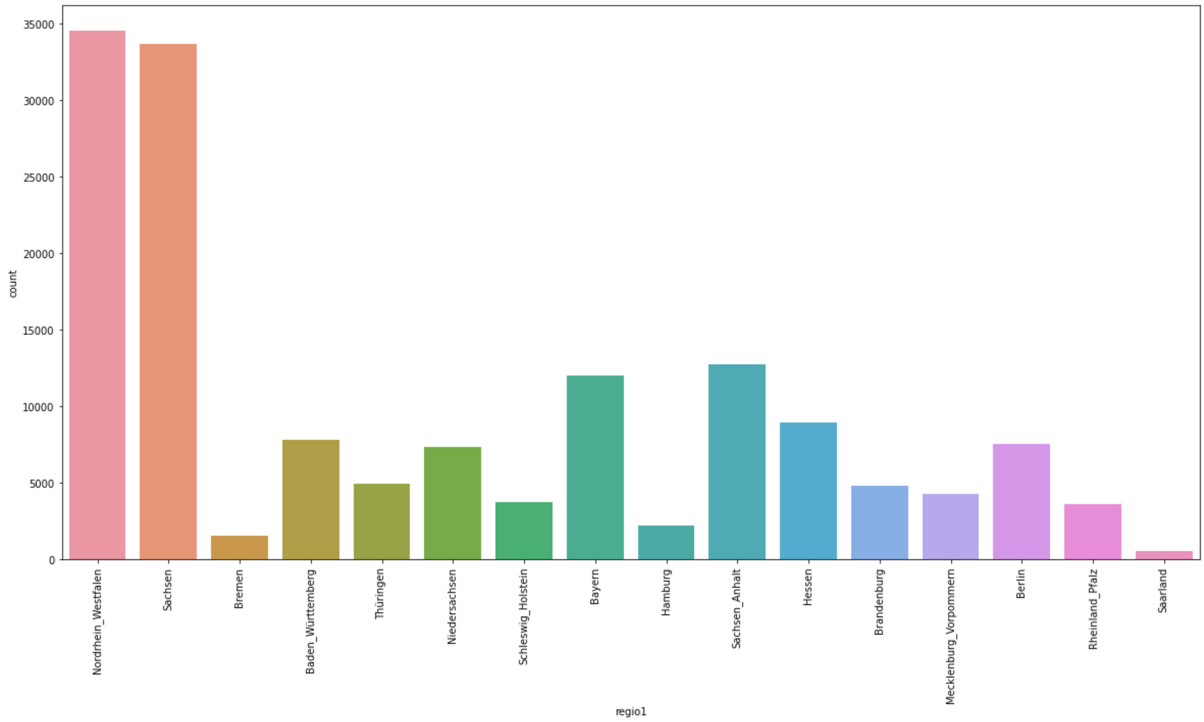
How about the kitchen?
Let's just briefly take a look at the pie chart below:



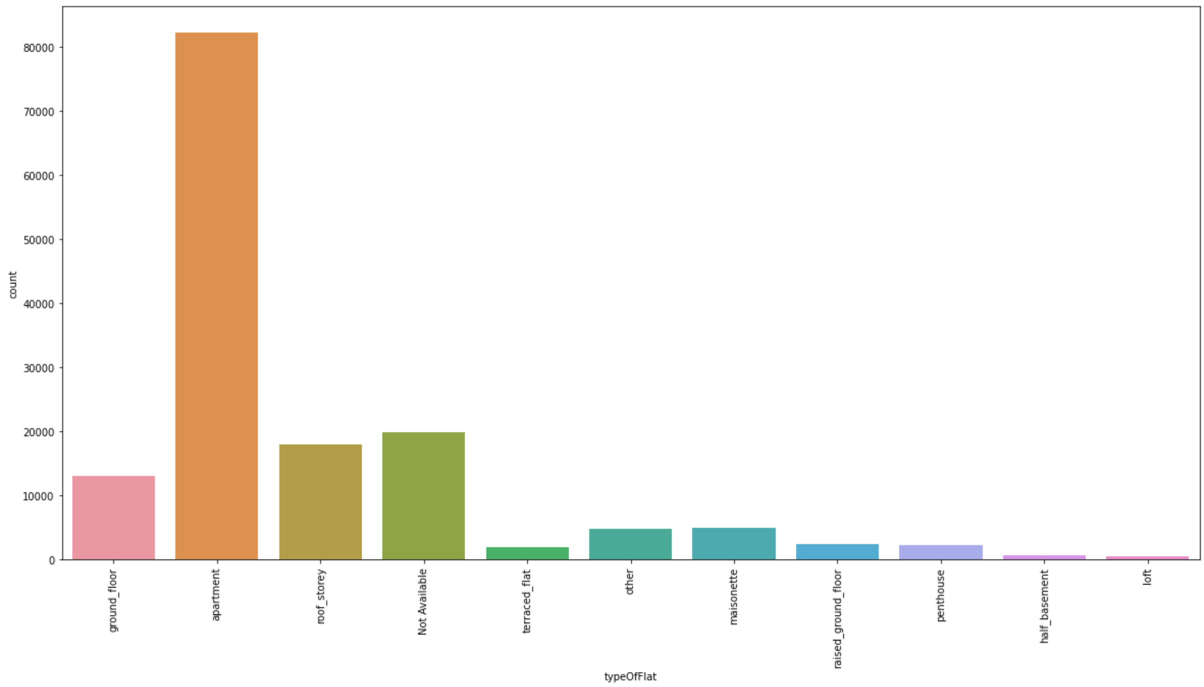
A significantly large proportion of apartments do not have kitchen and based on the following barchart, there is no general rule held for all regions:



The following count plot shows that Nordrhein-Westfalen and Sachsen respectively have the most number of apartment offers among other regions.



And also the apartments make up the biggest part between types of flats:



Model

I used LinearRegression and trained the model using 80% of the data and tested the model on 20% of the data and the result is as follows:

MAE: 173.47531706557996
MSE: 257243.24401621058
R2_score: 0.453644510212342

References:

---

https://analyticsindiamag.com/outlier-detection-using-z-score-a-complete-guide-with-python-codes/

https://www.analyticsvidhya.com/
https://scikit-learn.org/
https://stats.stackexchange.com/questions/31908/what-is-percentage-of-variance-in-pca
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/
https://towardsdatascience.com/hypothesis-testing-in-machine-learning-using-python-a0dc89e169ce
https://michael-fuchs-python.netlify.app/2019/11/13/ovo-and-ovr-classifier/
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html