Machine learning

Predicting house prices

November 2019

Introduction

The goal of this assignment is to predict the price of a house based on given set of features. It is an attempt to a Kaggle competition titled "House Prices: Advanced Regression Techniques"[1]. Unlike the previous assignments, this is a regression task. That means, that the target variable is not discrete (class label), but numeric (continuous value) and instead of recognizing patterns we recognize trends. Figure 1 presents flowchart of the process of doing this project.

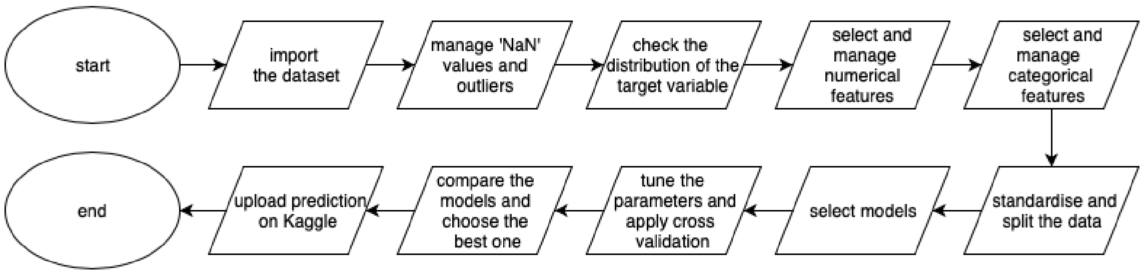

start
import the dataset
manage 'NaN' values and outliers
check the distribution of the target variable
select and manage numerical features
select and manage categorical features
standardise and split the data
select models
tune the parameters and apply cross validation
compare the models and choose the best one
upload prediction on Kaggle
end

Figure 1: Flowchart

Dataset

The dataset used in this project is the Ames Housing dataset compiled by Dean De Cock for use in data science education[2]. It describes the sale of individual residential property in Ames, Iowa from 2006 to 2010 and originally came directly from the Assessor's Office. The dataset prepared for this task contains 2930 observations and 79 explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values. The variables describe almost every feature of a house, starting with the basic ones, like total living area, neighbourhood or year in which the house was built, to less relevant ones, like quality of a fireplace, area of a swimming pool or number of bathrooms in a basement. Since not every house has a swimming pool or a basement, this information is represented by null values. Moreover, in the dataset there are also unusual sale cases indicated by the feature 'SaleCondition'. If its value is different than "normal", the price the house was sold might not be representative. There are also some outliers in the dataset - big houses sold for a relatively low price.

Managing 'NaN' values

After having understood the dataset, the first thing that has to be done it is to check how many missing values there are. In order to do that, by using the function 'missing_values_function', all the missing values per each feature are counted and returned. Figure 2 shows the total number and the percentage of missing values per each feature. As we can see from figure 2, there are many features with a lot of missing values. However, not for all of them it means that the data have been wrongly recorded but that some houses don't have those facilities. In particular:

- NaN value for **PoolQC**, means that the house does not have the Swimming pool.
- NaN value for **Alley**, means that there is no alley access.
- NaN value for **MiscFeature**, means that there are no miscellaneous feature.
- NaN value for **Fence**, means that there is no fence.
- The same is registered also for other twelves features.

However, there are three other features that instead have "real" missing values, meaning that something went wrong during the data recording. These features are:

[1]`https://www.kaggle.com/c/house-prices-advanced-regression-techniques`, access December 2019
[2]`http://jse.amstat.org/v19n3/decock.pdf`, access December 2019

- **LotFrontage**.
- **GarageYrBlt**.
- **MasVnrArea**.

These three features are also summarized in figure 3.

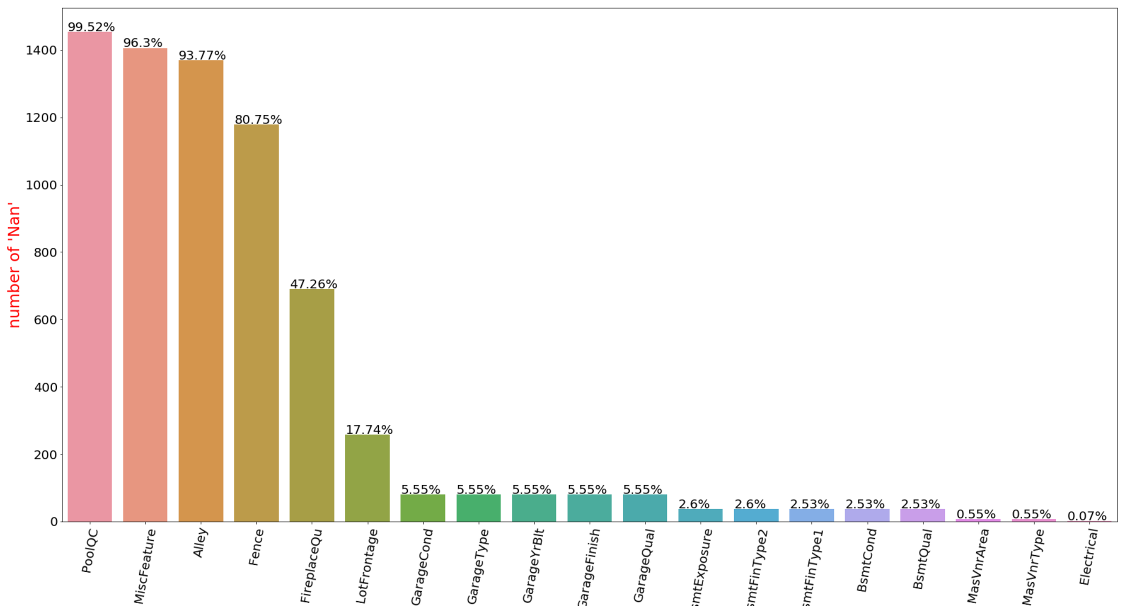


Figure 2: Histogram of all NaN values

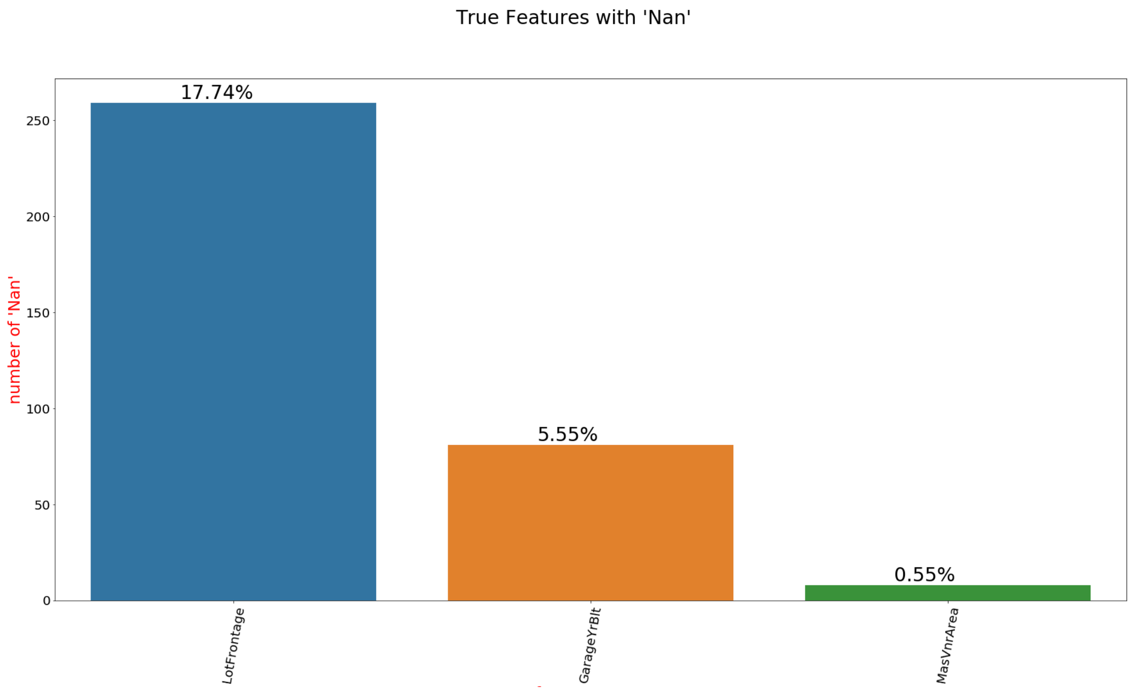


Figure 3: Histogram true NaN

As can be seen, LotFrontage has roughly 18% of missing values, while GarageYrBlt has 5.5% and MasVnrArea less then 1%. Of course, before performing any kind of analysis, fixing NaN values is necessary. For that reason, the ”fix_nan_function” function is defined. For all the features that belong to the first group (houses that do not have some characteristics) the missing values are substituted with ’NO’. For the features belonging to the second group (LotFrontage, GarageYrBlt and MasVnrArea), every sample (house) is checked distinctly. Each missing value is substituted by the mean value based on the belonging neighborhood. For example, house 24 is missing ’LotFrontage’. Because of this its neighborhood is checked and and it appears to be ’Sawyer’. Then, the average value of ’LotFrontage’ in the ’Sawyer’ neighborhood is computed and this value is inserted to fill the ’NaN’ value.

Managing outliers

The second step, regarding the data cleaning part, is to detect and delete outliers that could mislead the predictive function. Two such values were detected by comparing the target feature ’SalePrice’ and the second most relevant feature ’GrLivArea’. The two outliers are reported in red in the scatter plot on Figure 4.

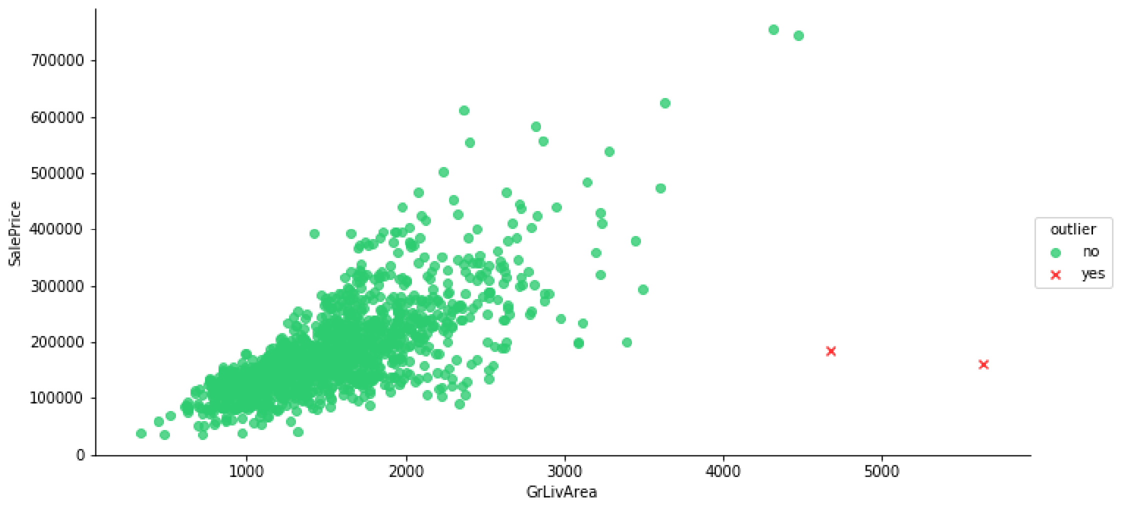


Figure 4: Scatter plot for Outliers

As can be seen, the two houses represented in red are true outliers because they don’t follow the general positive trend between the two features of the remaining houses. For that reason these two outliers are deleted from the dataset. The ’SalePrice’ and ’GrLivArea’ scatter plot after outliers elimination can be seen on Figure 5.

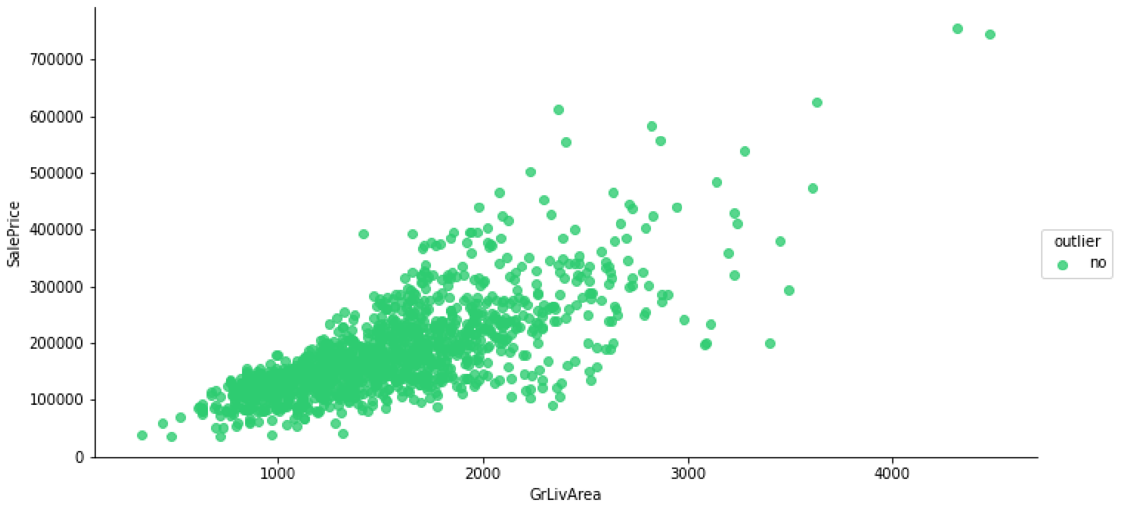


Figure 5: 'SalePrice' and 'GrLivArea' scatter plot after outliers elimination

Checking the distribtion of 'SalePrice'

Now that missing values and ouliers have been managed, it is necessary take a look to the target feature ('SalePrice'). The first thing to do is to compute the Skewness, which measures the symmetry of a distribution, and the Kurtosis which measures how heavy are the tails compared to a normal distribution. 'SalePrice' has:

- **Skewness:** 1.88.
- **Kurtosis:** 6.523.

The Figure 6 shows the distribution of 'SalePrice' and the qqplot, used to check if it follows a theoretical normal distribution.

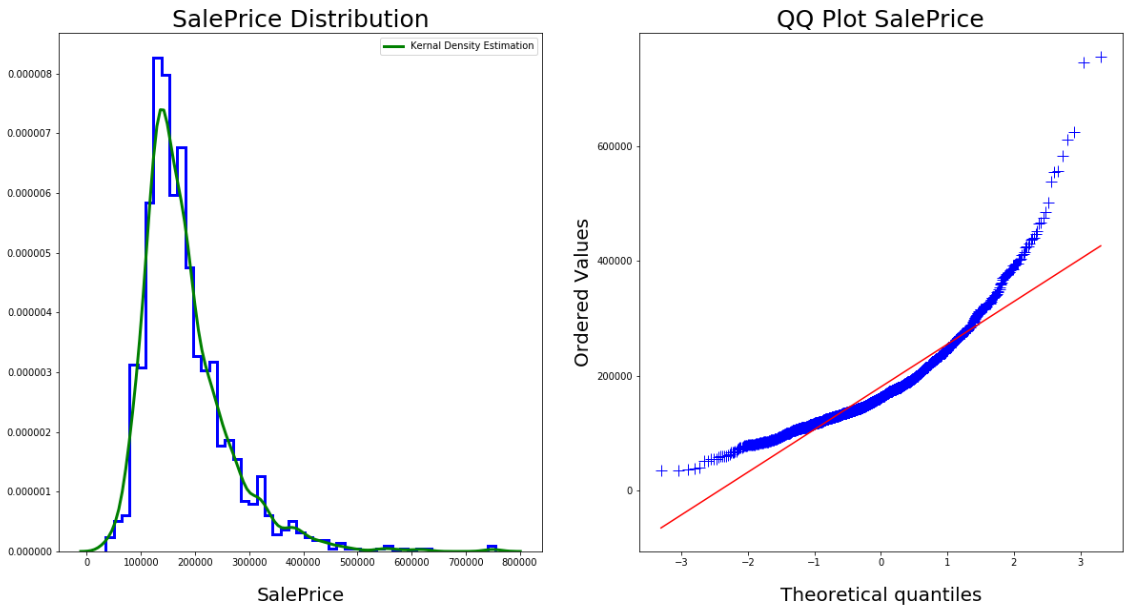


Figure 6: SalePrice density and qqplot

As can be seen from the left picture, the 'SalePrice' feature does not follow a normal distribution and this is confirmed by the qqplot on the right because the data distribution does not lie on the red line (representing the theoretical normal distribution). For that reason, different transformations are applied on the target feature in order to try to reach a normal distribution. The reason for that is, that normal distribution has many properties that can be helpful during the prediction. The Figure 7 shows for different transformations, in particular:

- **Natural Logarithm**.
- **Square Root**.
- **Cubic Root**.
- **Square**.

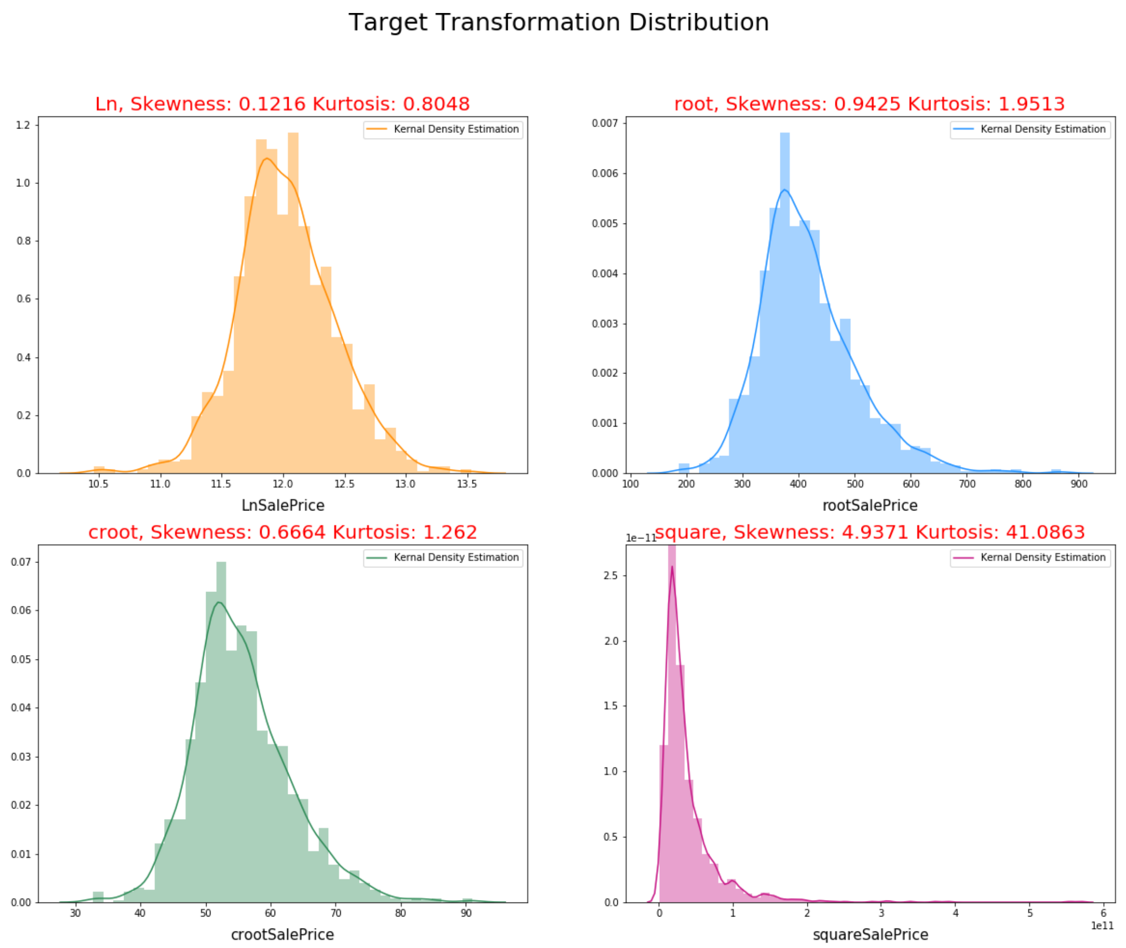


Figure 7: SalePrice transformations

The Figure shows that the best transformation is the Natural Logarithm. In fact, after 'ln' transforamtion it is possible seeing a great improvement regarding the Skewness and Kurtosis:

- **Skewness:** 0.1226.
- **Kurtosis:** 0.8048.

Bacause of this, the ln transormation is applied. Figure 8 shows the density plot and qqplot after this transformation.

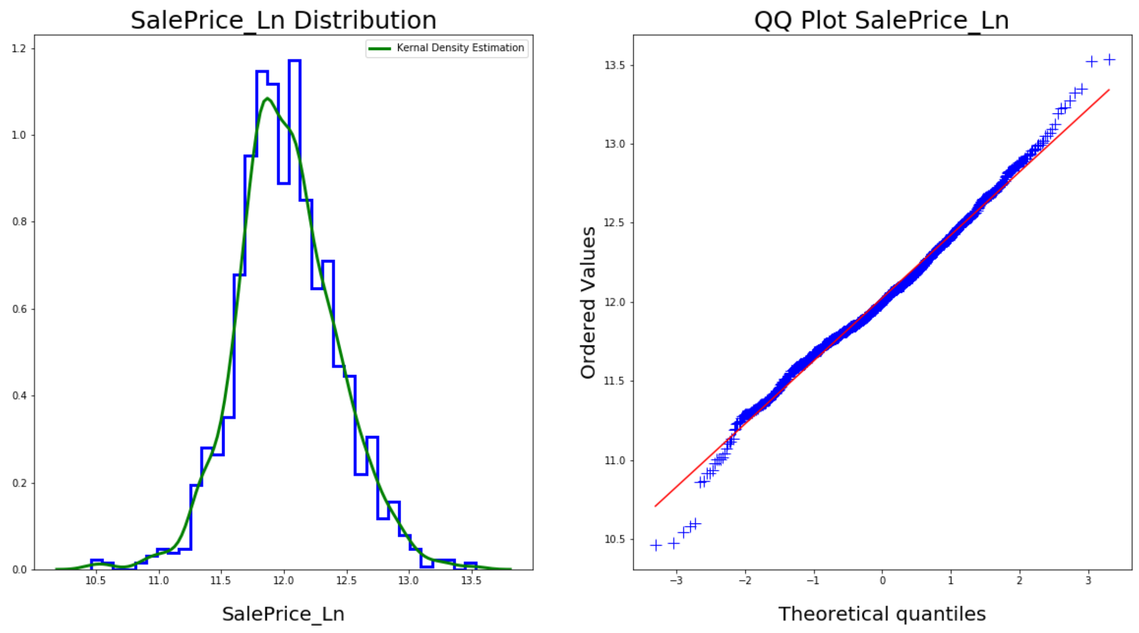


Figure 8: SalePrice 'ln' transformations

Numerical Features Selection and Cleaning

There are 39 features. All of them cannot be used to build up a good predictive model, because a too complex model is not able to generalize. Because of that the most significant features have to be selected. A good approach is to measure the significance of a variable by computing the Pearson correlation coefficient. It can be expressed by the formula:

$$r_{xy} = \sum_{i=1}^{n} \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

In this project, only the features that have a Pearson correlation coefficient greater than 0.5 and a p-value lower than 5% are considered as significant. By using 'Pearson_numerical' function the Pearson correlation coefficient and the R^2, necessary to interpret the correlation coefficients, are computed. After that, the following eleven features are selected:

- **OverallQual:** Pearson = 0.796, p-value $\approx$ 0.
- **YearBuilt:** Pearson = 0.524, p-value $\approx$ 0.
- **YearRemodAdd:** Pearson = 0.508, p-value $\approx$ 0.
- **TotalBsmtSF:** Pearson = 0.651, p-value $\approx$ 0.
- **1stFlrSF:** Pearson = 0.632, p-value $\approx$ 0.
- **GrLivArea:** Pearson = 0.735, p-value $\approx$ 0.
- **FullBath:** Pearson = 0.562, p-value $\approx$ 0.
- **TotRmsAbvGrd:** Pearson = 0.538, p-value $\approx$ 0.
- **GarageYrBlt:** Pearson = 0.500, p-value $\approx$ 0.
- **GarageCars:** Pearson = 0.641, p-value $\approx$ 0.
- **GarageArea:** Pearson = 0.629, p-value $\approx$ 0.

The Figure 9 shows the linear regression plot for all the selected numerical features.

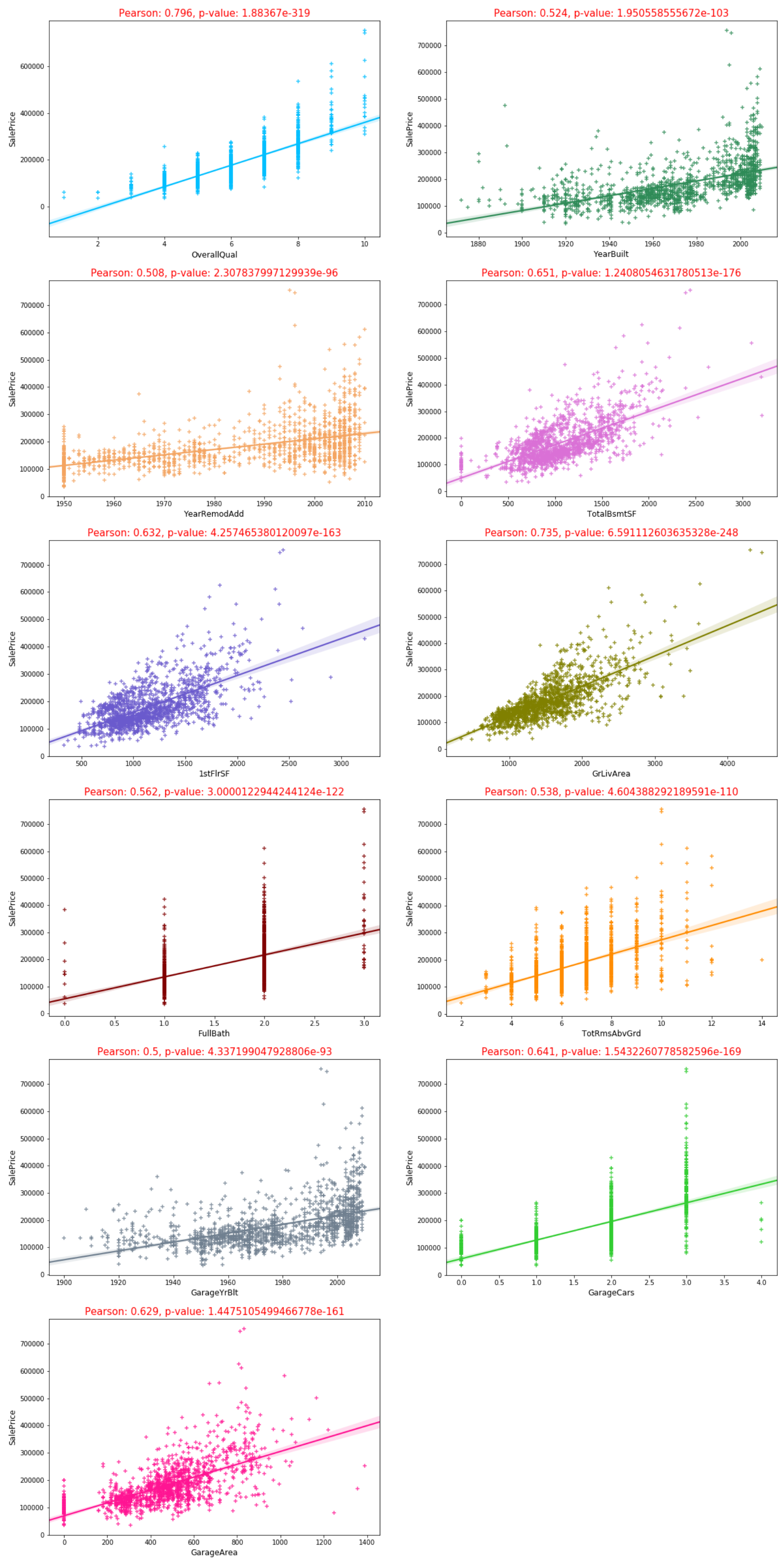
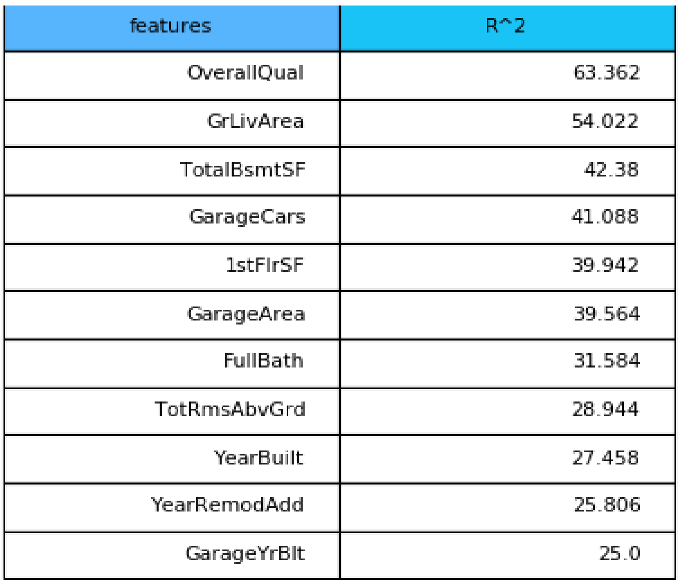


Figure 9: Regression Plot

The Figure 10 shows the R^2 computed per each of the chosen eleven features. The R^2 value explains how much of the variance of the SalePrice can be explained by the independent variable. For example, GrLivArea has a $R^2 = 0.54$ meaning that the 54% of the variance of the SalePrice is explained by this variable. Moreover, all these features are reliable because they have a p-value close to zero. In fact, the p-value (error) says that it is certain that a specific feature has an influence on the 'SalePrice' and the R^2 says how strong this influence is.



features	R^2
OverallQual	63.362
GrLivArea	54.022
TotalBsmtSF	42.38
GarageCars	41.088
1stFlrSF	39.942
GarageArea	39.564
FullBath	31.584
TotRmsAbvGrd	28.944
YearBuilt	27.458
YearRemodAdd	25.806
GarageYrBlt	25.0

Figure 10: R^2 summary table

In order to avoid mislead predictions and to reduce the complexity of the models, it is necessary to look for multicollinearity. For this purpose, the confusion matrix (Figure 11) is returned. It can helps to see which features are strongly related with each other.

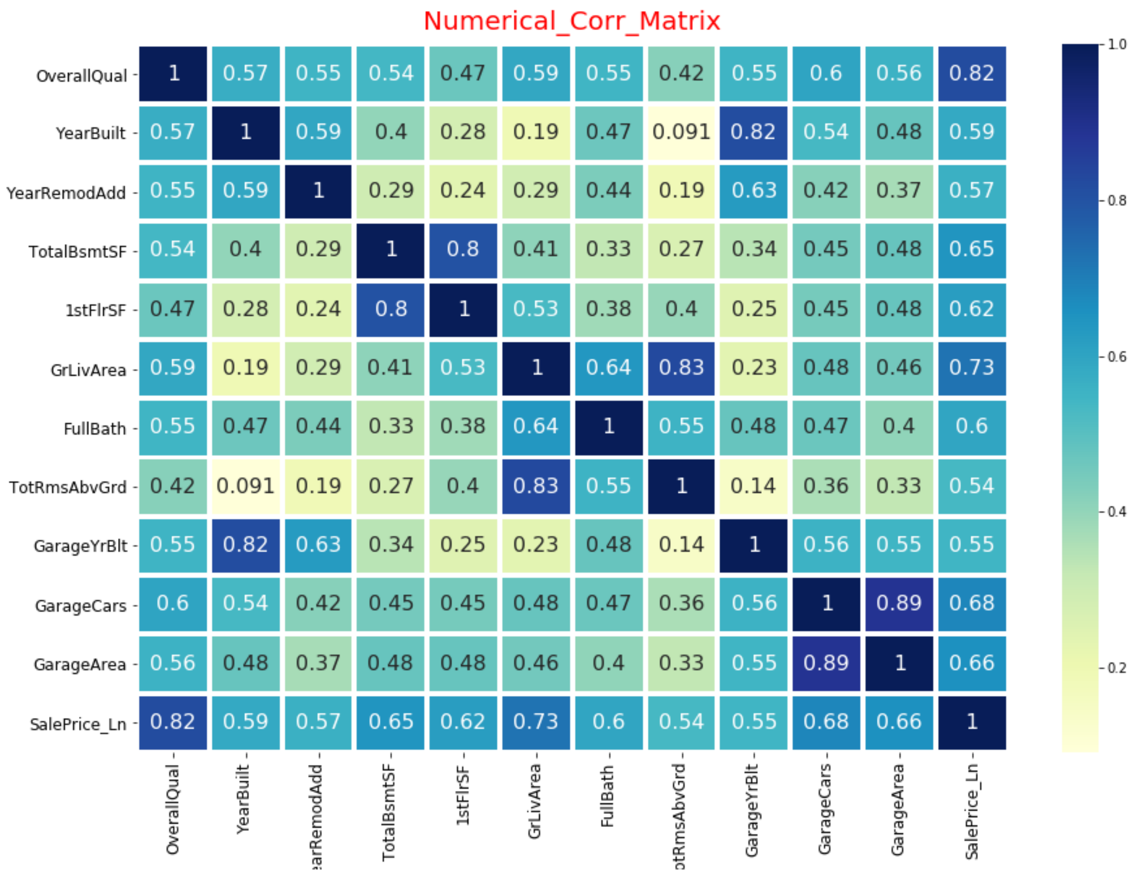


Figure 11: Numerical Correlation Matrix

As can be seen from the confusion matrix, there are 8 features that are strongly related with each other. It means that these features combined don't add significantly more information regarding the 'SalePrice'. Hence, only half of them can be chosen. Only the features that have an higher correlation with the target feature are selected. More precisely:

- '1stFlrSF' is strongly correlated (0.8) with 'TotalBsmtSF'. For that reason '1stFlrSF' is deleted, because it has a lower correlation with the 'SalePrice' feature.
- 'GarageYrBlt' is strongly correlated (0.82) with 'YearBuilt'. For that reason 'GarageYrBlt' is deleted, because it has a lower correlation with the 'SalePrice' feature.
- 'TotRmsAbvGrd' is strongly correlated (0.83) with 'GrLivArea'. For that reason 'TotRmsAbvGrd' is deleted, because it has a lower correlation with the 'SalePrice' feature.
- 'GarageCars' is strongly correlated (0.83) with 'GarageArea'. For that reason 'GarageAres' is deleted, because it has a lower correlation with the 'SalePrice' feature.

Numerical Features Transformation

The next step is to check their skewness and the kurtosis in order to understand if they also follow a normal distribution. If it is not the case, some transformations are applied with the scope to find which ones better reduce skeweness and kurtosis. Skewness between [-1, 1] and kurtosis between [-2,2] represent optimal results. After trying different transformations it is possible to discover that the following four features benefit of a Natural Logarithm transformation:

- 'GrLivArea',
- 'TotalBsmtSF',

- 'FullBath',
- 'GarageCars'.

Categorical Features Cleaning and Selection

Categorical features are not numbers which means, that they cannot be used to compute correlation coefficients in order to evaluate the relation degree with the target feature. For that reason a different approach has been adopted. In order to find which categorical features result to be highly significant the ANOVA (Analysis of Variance) has been computed. There are many approaches that can be performed in order to carry out the Analysis of Variance. The chosen approach is based on grouping the continuous variable (SalePrice) using the categorical variable, measuring the variance in each group and comparing it to the overall variance of the continuous variable. If the variance after grouping decreases drastically, it means that the categorical variable can explain most of the variance of the continuous variable and so the two variables likely have a strong association. If the variables have no correlation, then the variance in the groups is expected to be similar to the original variance. After running the Analysis of Variance (section 1.5.1 in the code), the result shows that the following five features are really relevant regarding the target variance explanation:

- 'MSZoning',
- 'Neighborhood',
- 'ExterQual',
- 'BsmtQual',
- 'KitchenQual'.

Label Encoding for Categorical Features

There are several approaches that can be used based. For this specific task it has been decided to assign to every class of every feature the most appropriate value. In order to find the most appropriate values the box-plots have been created. As can be seen from figures 12 - 16, it is possible to distinguish different encoding per each of the five features.

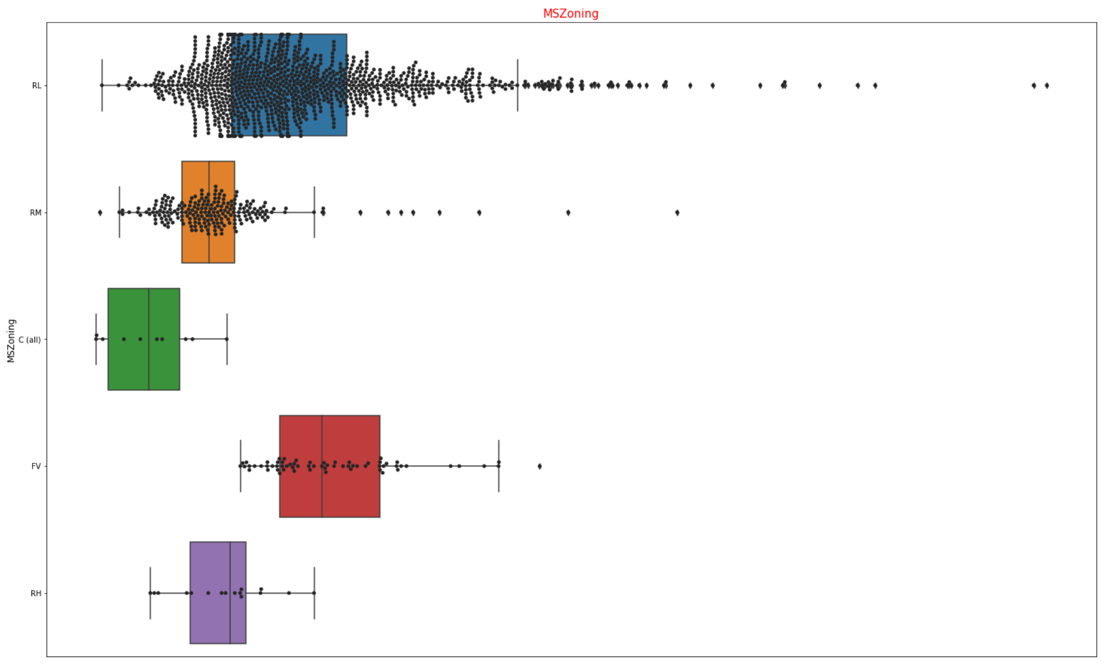


Figure 12: Box plot for 'MSZoning'

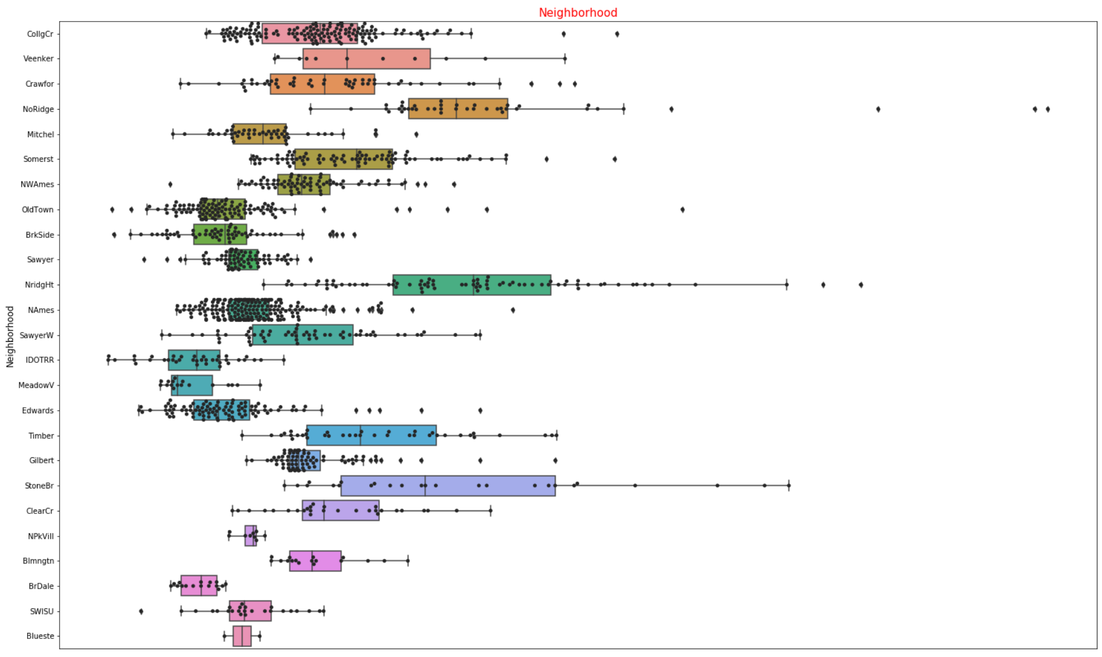


Figure 13: Box plot for 'Neighborhood'

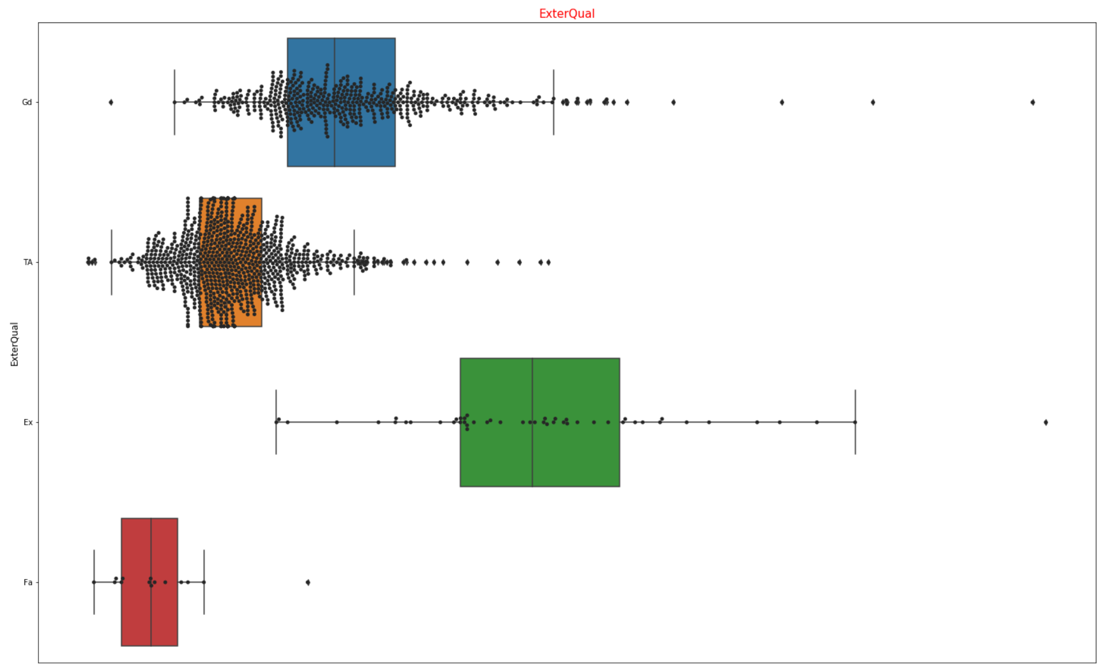


Figure 14: Box plot for 'ExterQual'

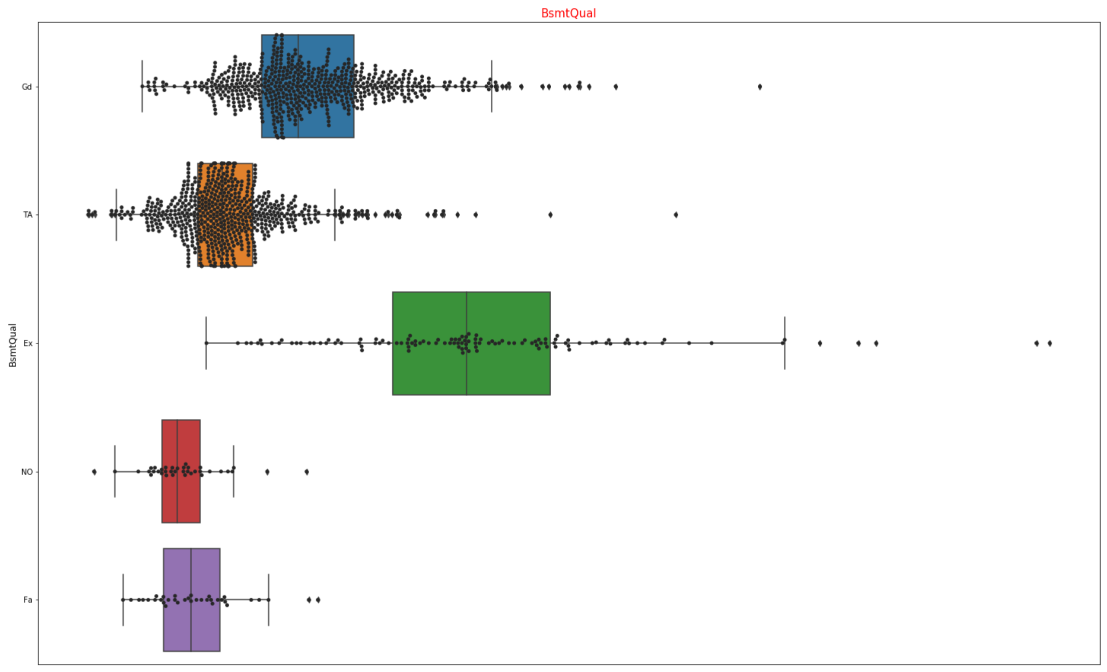


Figure 15: Box plot for 'BsmtQual'

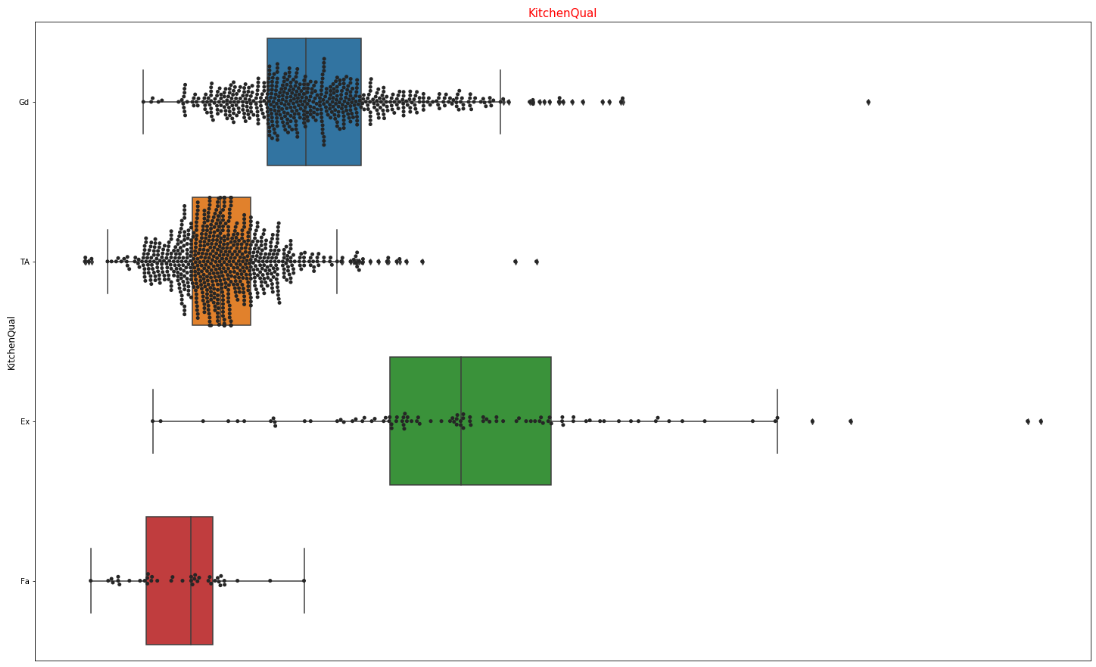


Figure 16: Box plot for 'KitchenQual'

In particular, for 'MSZoning' feature, four different classes can be distinguished and the following encoding has been applied:

- 'C (all)': 1,
- 'RM': 2,
- 'RH': 2,
- 'RL': 3,
- 'FV': 4.

'Neightborhood' looks more complicated because there are many classes and many of them seem to be the very similar. In this case it has been decided to group all the features in 3 classes, where:

- 'Timber', 'ClearCr', 'NWAmes', 'Somerst', 'Crawfor', 'Veenker', 'CollgCr','Blmngtn', 'Gilbert': 2,
- 'StoneBr', 'NridgHt','NoRidge': 3,
- for the remained neighborhoods: 1.

For 'ExterQual' the following encoding has been applied:

- 'Ex': 4,
- 'Gd': 3,
- 'TA': 2,
- 'Fa': 1.

For 'BsmtQual':

- 'Ex': 4,
- 'Gd': 3,

- 'TA': 2,
- 'Fa': 1,
- 'NO': 1.

Lastly, for 'KitchenQual':

- 'Ex': 4,
- 'Gd': 3,
- 'TA': 2,
- 'Fa': 1.

The function implemented to perform the transformation can be found in section 1.5.2 of the code under the name "categorical_encoding".

Categorical features transformation

The consecutive step is to check if these features can benefit of a transformation in order to try to reach normal distributions. Four transformations have been tried on these data (Natural Logarithm, square root, square and cubic root). After testing them, it has been found out that the following four numerical encoded features benefit of these transformations:

- 'MSZoning_square': square transformation,
- 'ExterQual_croot': cubic root transformation,
- 'KitchenQual_croot': natural logarithm transformation,
- 'BsmtQual_root': square root transformation.

With this step, the feature engineering part is concluded. Figure 17 shows the correlation matrix of all the selected features (twelve in total) plus the target feature.

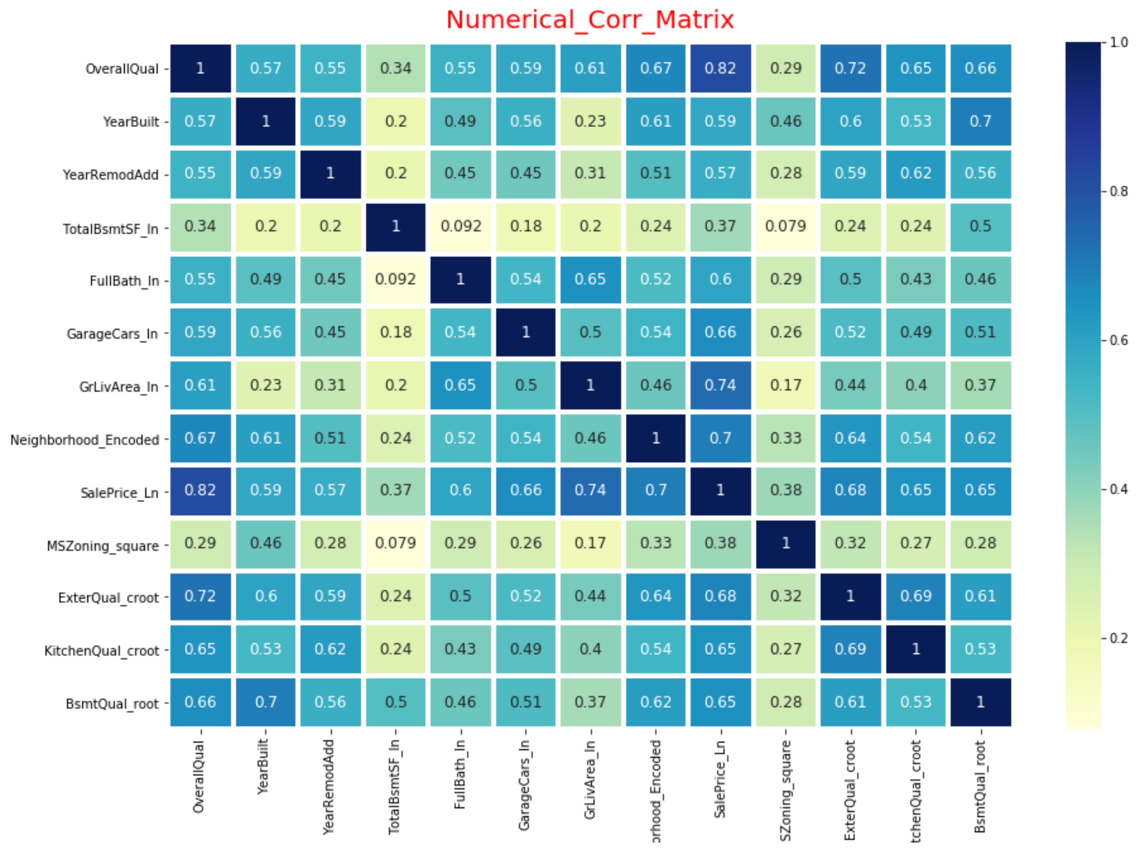


Figure 17: Correlation Matrix with all Categorical and Numerical selected features

Standardizing and splitting the data

Before running machine learning models, it is necessary to standardize and split the data into training and test set. For this scenario it has been decided to use 70% of the data as training set and 30% for the test set. Under section 2 of the .ipynb file, there is the code used to carry out the standardization and data splitting.

Hyperparameter tuning

In order to find the best values of parameters for every chosen model, hyperparameter tuning process is executed. In this step, the 'GridSearchCV' function implemented into the sklearn library is used to tune the parameters. Here, for every combination of parameters the models are trained by using cross validation in order to compute the R^2 for every combination. At the end of the process it is possible to find the best set of values of parameters for every model. As the best set of parameters we consider the one providing the highest R^2. Table 1 shows the tested and chosen values for tested parameters.

Table 1: Tested and chosen values of parameters

model	**parameter**	**tested values**	**chosen value**
kNN	number of neighbors	1, 50, 100, 150, 200	50
kNN	weights	'uniform', 'distance'	'distance'
dtr	min samples split	2, 21, 41	2
dtr	max depth	10, 40, 70, 100	70
dtr	min samples leaf	1, 2, 3, 4, 5	5
dtr	max features	'log2', 'sqrt', 5, None	5
svm	kernel	'rbf', 'linear'	'rbf'
svm	gamma	0.01, 0.001, 0.0001	0.001
svm	C	10, 100, 1000	100
rf	number of estimators	100, 200	200
rf	max depth	10, 100	10
rf	max features	'auto', 'sqrt', 'log2'	'sqrt'
rf	boostrap	True	True
NN	number of hidden layers	2, 3, 4, 5	2
NN	number of neurons	200, 225, 250, 275	200
NN	number of iterations	10000, 15000, 20000, 25000	10000
AdaBoost	n estimators	500, 1000, 2000	1000
AdaBoost	learning rate	0.001, 0.01, 0.1	0.01

Model comparison and results

Having tuned the parameters, it is necessary compare all the models in order to find the one that defines a predictive function for the 'SalePrice' target feature in the best way. Figure 18 shows the box plots for the 7 tested models after cross validation results with the selected values of parameters. As can be seen, the best models seem to be Random Forest and Support Vector Machine. Also Multiple Regression, Knn and AdaBoost perform quite well, which cannot be said the same for Decision Tree Regressor and Neural Network, that in this case returns worse results than the remaining models.

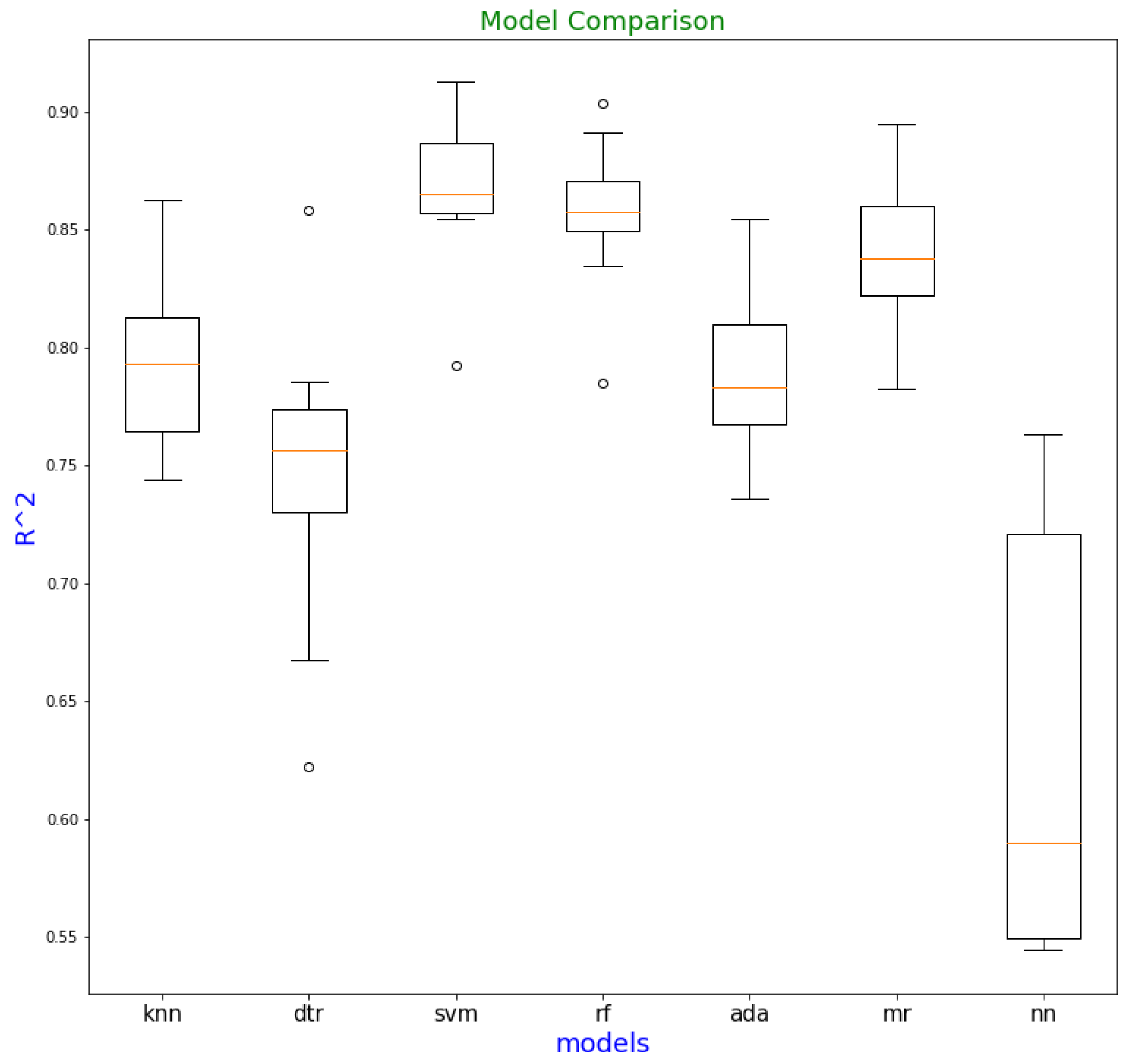


Figure 18: R^2 cross validation results

The next thing to do is compare the models based on the errors that they make on the test set. Figure 19 returns Mean Absolute Error, Mean Square Error, Root Mean Square Error and R^2. This last measure is also reported to shows that the models don't over-fit the data. In fact, as reported in table two, R^2 results on the test data are greater than the average cross validation R^2.

Table 2: R^2 values for tested models

model	**CV mean** R^2	R^2 **on test data**
nn	0.6324	0.6883
dtr	0.7450	0.7725
ada	0.7901	0.8143
knn	0.7931	0.8172
mr	0.8393	0.8513
rf	0.8568	0.8758
svm	0.8675	0.8855

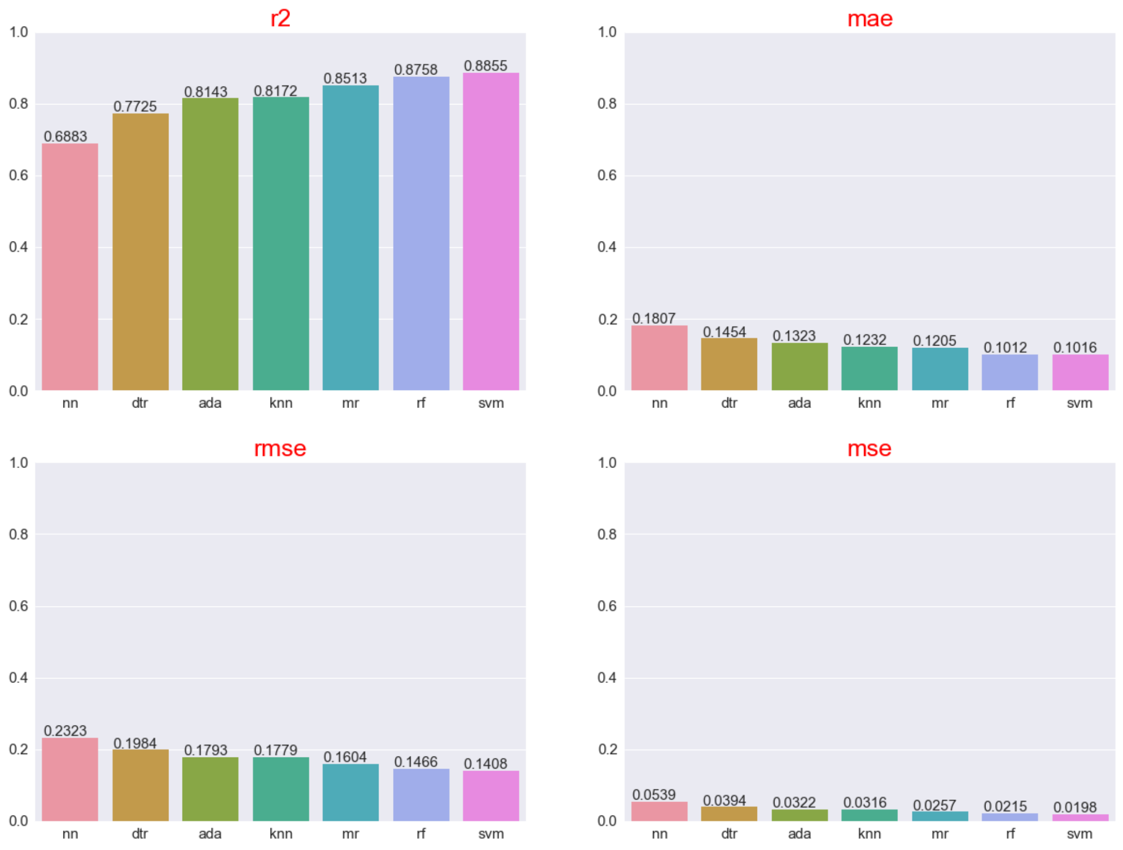


Figure 19: R^2, MAE, RMSE, MSE

Selecting the model and uploading prediction on Kaggle

After having compared the models, it is necessary to select one of them. As can be seen from table 2 and also from Figure 14, Support Vector Machine always returns lower errors and also the higher R^2, so it seems to be the most probable candidate to be selected as best model. However, a keen eye can see that Random Forest performs slightly better in terms of generalization. In fact, the average cross validation R^2 is about 0.8568 and R^2 on the test data is 0.8758 meaning that there is an improvement on the R^2 of 0.0190. Instead, for Support Vector Machine, it is of 0.0180. For that reason, the selected model for this task is Random Forest.

Now that the model has been selected, it is the time to make predictions on the 'test.csv' data provided by Kaggle. This dataset consists of 1459 samples and 80 features, because the target feature 'Sale price' is not included, in fact it has to be predicted by the participants. Under section 7 of the .ipynb file, there is the code used to clean this dataset by calling the same functions used for the 'train.csv' data set. Once the 'test.csv' has been cleaned the Random Forest model estimator is applied and the predicted Sale Prices are stored into 'submission_rf.csv' file and uploaded on the Kaggle platform. The score returned by Kaggle is: RMAE = 0.19135