House Price Prediction using Machine Learning Techniques

A Machine Learning Approach to Predicting Real Estate Prices

Authors:
Bruno Francesco Barbara
Giovanni Beraldi
Andrea Cuteri
Salvatore Fiorentino
Agostino Rizzo

Contents

1. Business Understanding

1.1 Determine Business Objectives

It is important for a real estate agency to price properties according to their characteristics. The provided dataset describes quite every aspect of a sale of residential buildings in Ames, Iowa and was introduced by Dean De Cock in [1] for educational purposes. The original dataset came from the Ames City Assessor's Office with more the 100 variables and almost 4000 sales between 2006 and 2010. This section describes the business problems and goals behind this project

1.1.1 Business Background

The problem area pertains to the real estate industry, specifically a real estate agency aiming to accurately predict house prices. This information is crucial for properly assessing properties and providing advice to potential buyers or sellers. The real estate agency could use such predictions to enhance decision-making processes, optimize property valuation, and offer a better customer experience. The problem involves the accurate prediction of house prices using data mining techniques. The goal is to develop a predictive model that can analyze a dataset and provide a reliable estimate of the price for a new property.

1.1.2 Business Objectives

The goal of this project is therefore to be able to accurately classify in the right price range of a property in order to:

- Increase the interest and satisfaction of buyers by proposing houses accordingly to the economic availability of the buyers.
- Support the agent in the decision of the price of the house with an automatic tool.

1.1.3 Business Success Criteria

The project can be judged successful if:

- The support offered to the agent reduces the time required to make the decision.
- The clients are more involved since there are fewer houses that are proposed to them and they do not match their budget and/or expectations.
- New houses are easily classified according to the houses already analyzed.

1.2 Data Mining Goals

The objective is to define one or more classification models for:

- Assigning to each house a label that collocates it in a prince range basing on its features.
- Correctly classifying the majority of instances with a low level of error.
- From the company's point of view, it is more desirable that the errors are more about overestimation of the price, than about underestimation.

1.3 Success Criteria

The goodness of model is evaluated according to the following criteria

performance measure	desired value
accuracy	0.80
precision	0.80
recall	0.80
ROC area	0.80

2. Data Understanding

2.1 Data Description

The training set consists of 1460 rows. The number of features is 81 and among those, *SalePrice* is the target. Since the objective of the model is to classify house prices in 3 categories we discretized the target. Most of the attributes describe the characteristics of the houses, both from a qualitative and quantitative point of view. By giving a first read of the data description that was included in the zip file downloaded from *Kaggle*, we noticed that approximately one half of the attributes is categorical of either ordinal or non-ordinal type, and the rest is split into a big chunk of numeric attributes and a few binary ones. Moreover, when referring to categorical attributes, the description also includes the possible value *NA* for most of them. The meaning of such a value turns out to be a special category that refers to a defect in some feature of the house. Just to give an example, *NA* in the attribute Alley is supposed to say that the house has no alley access. However, it is to be investigated whether *NA* values inside other attributes, categorical or not, have the classic meaning of null values.

2.2 Data Exploration

The goal of this phase is to conduct some preliminary analyses on the data with the intent of making a first idea of how each feature of the dataset impacts on the class label. The approach we adopt consists in doing different kinds of plots for each feature, according to their data type. Moreover, the methods `stattest_quali` and `stattest_quanti` are used to understand if both categorical and numeric attributes are statistically significant on the class label or not. First things first, we plot the count of the three possible values for the class label and we notice that the data set is strongly imbalanced as Figure 2.1 shows.

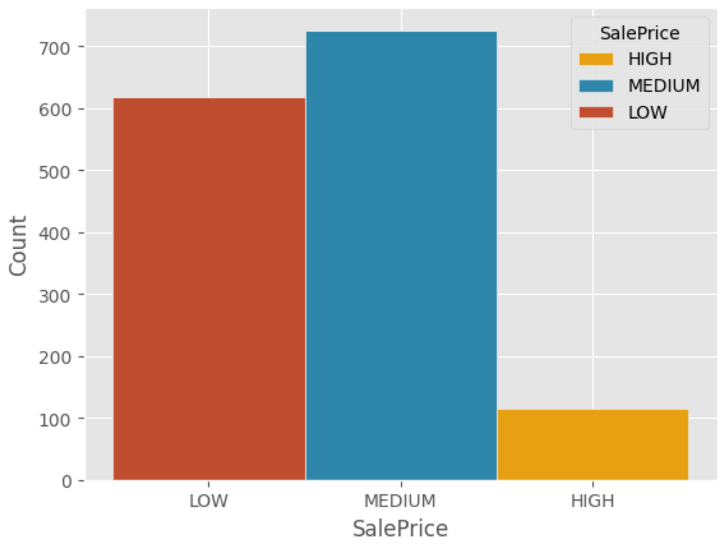


Figure 2.1: class label distribution

2.3 More about statistic tests

In this section we explain a bit more in detail what are the statistic tests conducted on the attributes of the data set in order to identify the prediction value on the target and how non-target attributes are dependent from each other.

2.3.1 `stat_test_quanti` function

The objective of this function is to understand, given a numeric feature and the target feature whether the mean of the groups (determined by the values that the target feature can assume) is different from each other. To this extend we used the method `f_oneway` provided by *scipy* in order to perform the one-way ANOVA test. A low *p-value* (usually < 0.05) for the test indicates that at least a mean of a group (one for each target value) is significantly different that the others hence at least a group influences the numeric feature showing a significant prediction value.

2.3.2 `stat_test_quali` function

The objective of this function is to understand, given a categorical feature and the target whether the groups of the feature and the groups of the target are independent. The type of test is the independent-test (*chiSquare*). A low *p-value* means that at least two groups are dependent, so at least one value of the feature influences the value of the target.

2.3.3 `understand` function

The objective of this function is to perform all the procedures needed to understand how the values of a given feature are distributed and how they are affected by the target. A part from the type the function gives in output: the number of missing values, the Cramer' V coefficent for the correlation (for the numeric attributes we categorize them).

`understand` categorical feature

In the case of a categorical feature there are additional information as the Information Gain, the output of the independence test (*chiSquare*) and for each group of the feature the majority class with the probability to be of that class given that group and its frequencies. As shown in 2.18

`understand` numerical feature

In the case of a numeric feature there are additional information as the output of the independence test (*chiSquare*).

column name	type	description
Id	numeric	Unique Identifier of each house
MSSubClass	categoric ordinal	Identifies the type of dwelling: 20, 30...
MSZoning	categoric	General zoning classification: A, C, ...
LotFrontage	numeric	Linear feet of street connected to property
LotArea	numeric	Lot size in square feet
Street	categoric	Type of road access to property
Alley	categoric	Type of alley access to property
LotShape	categoric	General shape of property
LandContour	categoric	Flatness of the property
Utilities	categoric	Type of utilities available
LotConfig	categoric	Lot configuration: Inside, Corner
LandSlope	categoric	Slope of property
Neighborhood	categoric	Neighborhood of Ames city
Condition1	categoric	Proximity to various conditions
Condition2	categoric	Additional condition (Condition1)
BldgType	categoric	Typology of the structure of the house
HouseStyle	categoric	Style of the structure of the house
OverallQual	categoric ordinal	Rating of the overall condition of the house
OverallCond	categoric ordinal	Rating of the overall condition of the house
YearBuilt	numeric	Year of construction of the house
YearRemodAdd	numeric	Year of remodeling of the house
RoofStyle	categoric	Type of roof: Flat, Gable...
RoofMatl	categoric	Roof material: ClyTile, CompShg...
Exterior1st	categoric	Exterior covering: AsbShng, AsphShn
Exterior2nd	categoric	Exterior covering: AsbShng, AsphShn, ...
MasVnrType	categoric	Masonry veneer type: BrkCmn, BrkFace
MasVnrArea	numeric	Masonry veneer area in square feet
ExterQual	categoric ordinal	Quality of the material on the exterior
ExterCond	categoric ordinal	Material condition of the exterior
Foundation	categoric	Type of foundation: BrkTil, CBlock...
BsmtQual	categoric ordinal	Evaluates the height of the basement
BsmtCond	categoric ordinal	General condition of the basement
BsmtExposure	categoric ordinal	Walkout or garden level walls
BsmtFinType1	categoric ordinal	Rating of basement finished area
BsmtFinSF1	numeric	Square feet of Type 1 basement
BsmtFinType2	categoric ordinal	Rating of basement finished area of type 2
BsmtFinSF2	numeric	Square feet of Type 2 basement
BsmtUnfSF	numeric	Unfinished square feet of basement area
TotalBsmtSF	numeric	Total square feet of basement area
Heating	categoric	Type of heating of the house
HeatingQC	categoric	Heating quality and condition

Column Name	Type	Description
CentralAir	binary	Equipped with central air conditioning
Electrical	categoric	Type of electrical system
1stFlrSF	numeric	First Floor square feet
2ndFlrSF	numeric	Second floor square feet
LowQualFinSF	numeric	Low quality finished square feet (all floors)
GrLivArea	numeric	Above grade (ground) living area square feet
BsmtFullBath	categoric ordinal	Number of basement full bathrooms
BsmtHalfBath	categoric ordinal	Number of basement half bathrooms
FullBath	categoric ordinal	Number of full bathrooms above grade
HalfBath	categoric ordinal	Number of half baths above grade
Bedroom	categoric ordinal	Number of bedrooms above grade
Kitchen	categoric ordinal	Number of kitchens above grade
KitchenQual	categoric ordinal	Rating of the quality of the kitchen
TotRmsAbvGrd	numeric	Number of rooms above grade (no bathrooms)
Functional	categoric	Generic home functionalities
Fireplaces	categoric ordinal	Number of fireplaces
FireplaceQu	categoric ordinal	Quality of the fireplaces
GarageType	categoric	Location of the garage
GarageYrBlt	numeric	Year garage was built
GarageFinish	categoric	Status of completeness of the garage
GarageCars	categoric ordinal	Size of garage in car capacity
GarageArea	numeric	Size of garage in square feet
GarageQual	categoric ordinal	Rating of the quality of the garage
GarageCond	categoric ordinal	Rating of the condition of the garage
PavedDrive	categoric	Paved driveway typology
WoodDeckSF	numeric	Wood deck area in square feet
OpenPorchSF	numeric	Open porch area in square feet
EnclosedPorch	numeric	Enclosed porch area in square feet
3SsnPorch	numeric	Three season porch area in square feet
ScreenPorch	numeric	Screen porch area in square feet
PoolArea	numeric	Pool area in square feet
PoolQC	categoric ordinal	Quality of the pool
Fence	categoric ordinal	Fence quality based on material and privacy
MiscFeature	categoric	Various feature not covered in other categories
MiscVal	numeric	Value of miscellaneous feature
MoSold	categoric ordinal	Month Sold
YrSold	numeric	Year Sold
SaleType	categoric	Typology of sale
SaleCondition	categoric ordinal	Condition of sale: Normal, Abnormal...
SalePrice	categoric ordinal	Sale price of the house (target feature)

2.4 Attributes

2.4.1 ID

By starting to explore the features of the data set, we notice that the attribute ID is probably a primary key since it has 1460 distinct values(the corresponding entropy is maximum). Moreover, we plotted the variation of house prices w.r.t the ids in Figure 2.2, as we were expecting, no correlation emerged. From Figure 2.2 it emerges that this attribute has no effect on the sale price and it is very likely that it will be dropped.

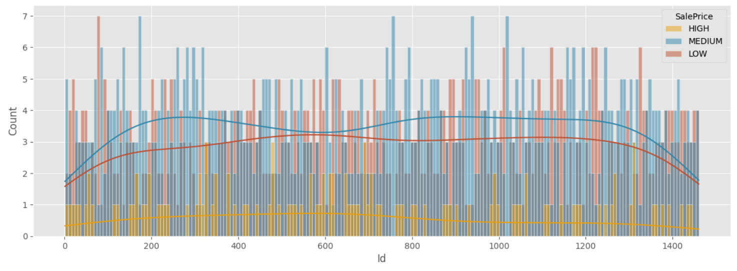


Figure 2.2: Id-SalePrice

2.4.2 MSSubClass

The attribute *MSSubClass* is a categoric ordinal attribute that collects several information about the house. Essentially it describes the style and the age of the house. By looking at plots in Figures [2.3, 2.4] it is clear that this attribute has a significant impact in determining the class label. For example it is clear that the class with value *60* contains very few examples that are classified as *LOW*, while class *50* is a good predictor for the class *LOW*.

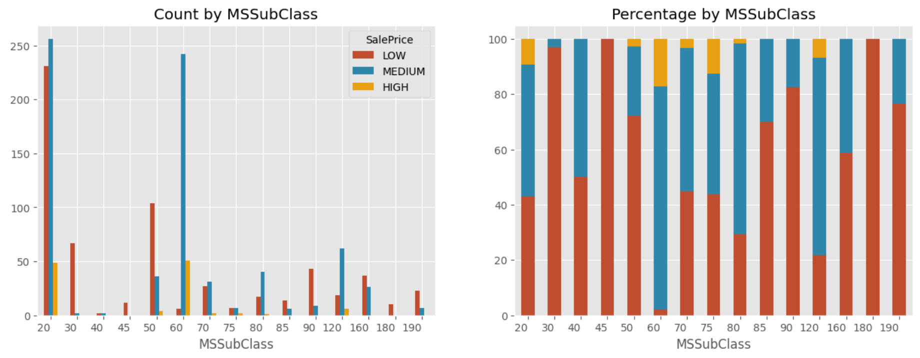


Figure 2.3: MSSubClass-SalePrice

```
Chi-square Test - Outcome
P-Value: 1.122234664911419e-81 [<0.05]
Cramer's V - Outcome
Coeff: 0.4010822438878207
(1.122234664911419e-81, '[<0.05]')
```

Figure 2.4: Independence test for MSSubClass

2.4.3 OverallQual

OverallQual represents a good measure of evaluation. As shown in figure 2.5 there should be a strong correlation between the quality of the house and the corresponding class assigned to them.

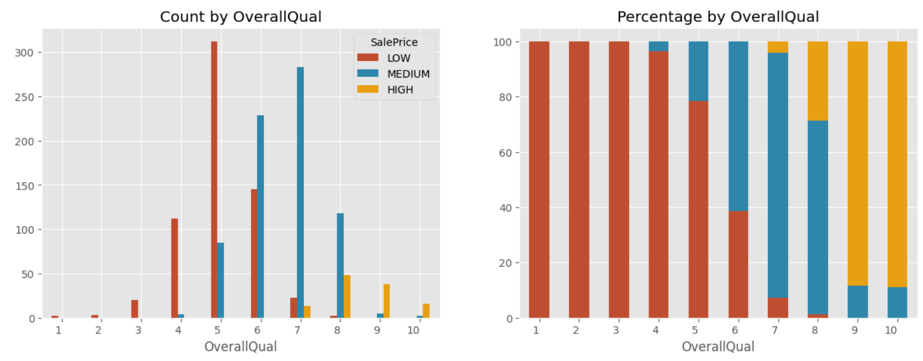


Figure 2.5: OverallQual-SalePrice

2.4.4 Neighborhood

Neighborhood describes where the house is located. The plot in Figure 2.6 shows that there is a strong correlation between the value of this attribute and the class label.

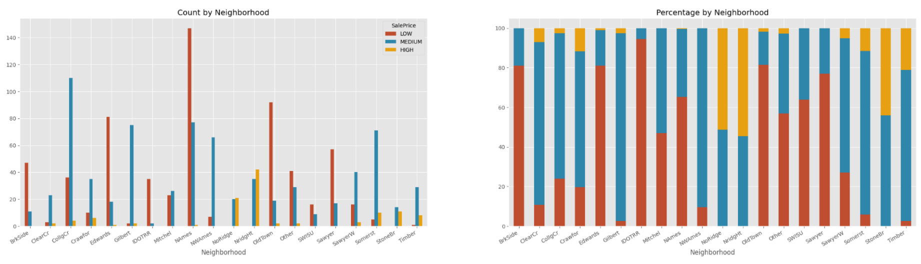


Figure 2.6: Neighborhood-SalePrice

From the plot, we can see that `NAmes` and `CollgCr` are the most populated neighborhoods while `NoRidge`, `NridgHt` and `StoneBr` are the most expensive ones.

2.4.5 House Style

The house style briefly describes the structure of the house. It can be observed by figure 2.7 that it discriminates the houses by the sale price, thus we carried on a statistical test that can confirm the correlation. In figure 2.8 it can be observed the difference between the values observed with respect to the expected value in case of independence. The results of the test confirm the presence of a correlation.

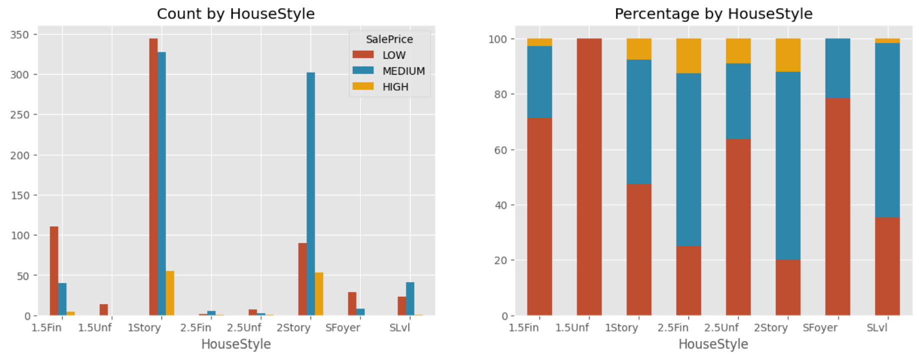
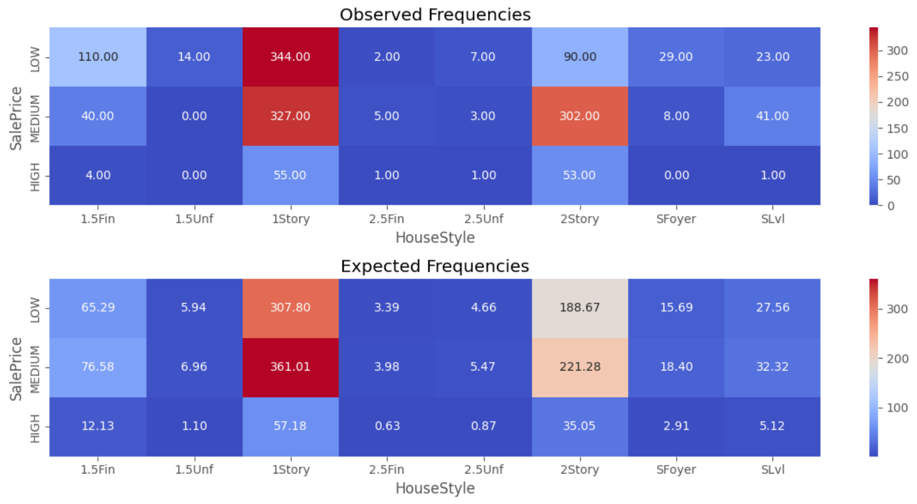


Figure 2.7: House Style barplot by SalePrice



SalePrice	1.5Fin	1.5Unf	1Story	2.5Fin	2.5Unf	2Story	SFoyer	SLvl
LOW	110.00	14.00	344.00	2.00	7.00	90.00	29.00	23.00
MEDIUM	40.00	0.00	327.00	5.00	3.00	302.00	8.00	41.00
HIGH	4.00	0.00	55.00	1.00	1.00	53.00	0.00	1.00

SalePrice	1.5Fin	1.5Unf	1Story	2.5Fin	2.5Unf	2Story	SFoyer	SLvl
LOW	65.29	5.94	307.80	3.39	4.66	188.67	15.69	27.56
MEDIUM	76.58	6.96	361.01	3.98	5.47	221.28	18.40	32.32
HIGH	12.13	1.10	57.18	0.63	0.87	35.05	2.91	5.12

Figure 2.8: HouseStyle-SalePrice indipendence test representation

2.4.6 CentralAir

Most houses in the dataset have a centralized air conditioning system. Positive instances have not a discriminant effect on the sale price hence knowing that a house is equipped with air conditioning it is not a relevant feature. As we expected, the *information gain* over this attribute is around 0.06. On the contrary, more than 80% of the ones that do not have air conditioning have a low sale price becoming a relevant information.

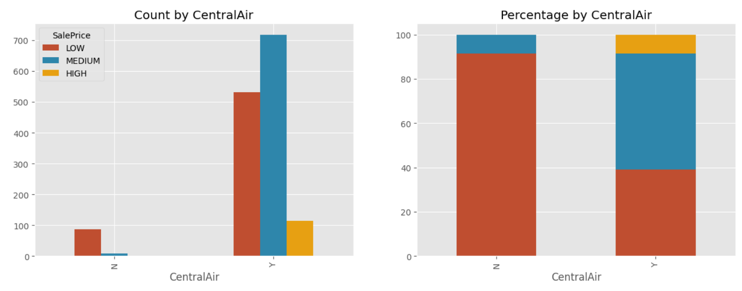


Figure 2.9: CentralAir-SalePrice

The centralized air conditioning system is a privilege of the most recent houses. It is plausible that in recent years houses are more probable to have this system. As expected the `CentralAir` and `YearBuilt` attributes are not indepen-

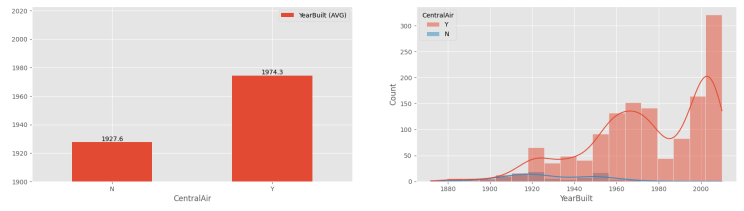


Figure 2.10: CentralAir-YearBuilt

dent since the indipendence Chi-square test admits a *p-value* $\approx 1.27 \times 10^{22}$. At the end, the `YearBuilt` attribute is more indicative than the `CentralAir` when we consider the sale price as the target one. Based on the previous observations, we will consider the `CentralAir` attribute as a candidate to be removed.

2.4.7 Electrical

The `Electrical` attribute describes the type of electrical system installed into the house. The most installed system is `SBrkr (Standard Circuit Breakers & Romex)` while other systems are less frequent to be installed. This unbalance in the frequencies leads to have no relevance in predicting the sale price as justified by a low *information gain* of around 0.0601. Therefore, a reasonable choice would be to remove it given the low prediction value.

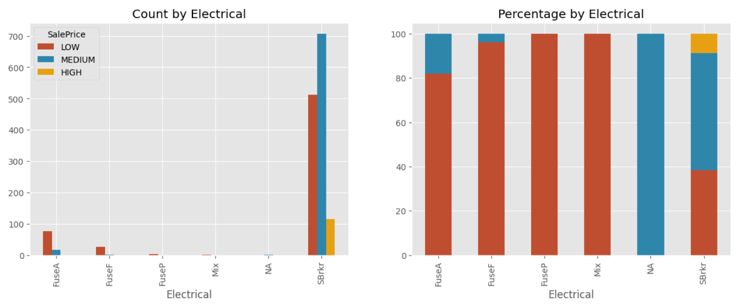


Figure 2.11: Electrical System

2.4.8 1stFlrSF

The square feet of a building do affect its sale price as anyone expects. From the following plots, it is evident that the sale price follows a linear trend with respect to the first floor square feet. A smaller first floor will more likely induce a lower price for the building. On the contrary, if one likes more space then more he needs to pay for it.

In addition, an analysis of the correlation between the first floor area (`1stFlrSF`) and the length of street connected to the property (`LotFrontage`) has been performed in order to validate an expected dependence by performing the *Pearson Correlation Test*. However, the two attributes are correlated with a coefficient of around 0.4 validating only partially what we expected. In fact, the `LotFrontage` may not only be due to the first floor square feet but make sense to take into account the overall occupied surface of the dwelling including, for example, other attributes like the pool area (`PoolArea`), the open porch area (`OpenPorchSF`) and the size of the garage (`GarageArea`).

2.4.9 FullBath

The attribute *FullBath* describes how many baths with complete services are at the floors above the ground floor. As evidenced in the plot in Figure 2.12 there is a strong correlation between the number of full baths and the corresponding class label. Another strong result is reported from the statistic test shown in Figure 2.13

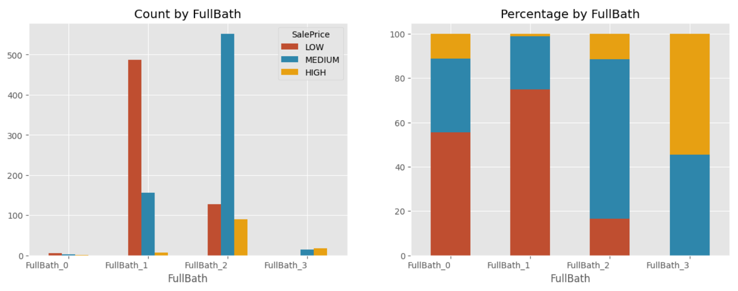


Figure 2.12: FullBath-SalePrice

```
Chi-square Test - Outcome
P-Value: 4.018443251096999e-127 [<0.05]
Cramer's V - Outcome
Coeff: 0.45463384659527495
0.29826715827233946
Understanding FullBath:
    Missing values: 0
    Independence Test: 4.02e-127[<0.05]
    Information Gain: 0.30
    When the FullBath is:
       * FullBath_1, then the sale price is 70% likely to be LOW
         (44.52% of the times)

       * FullBath_2, then the sale price is 70% likely to be MEDIUM
         (52.60% of the times)

       * FullBath_0, then the sale price is 55% likely to be LOW
         (0.62% of the times)

       * FullBath_3, then the sale price is 50% likely to be HIGH
         (2.26% of the times)
```

Figure 2.13: FullBath independence test

2.4.10 YearBuilt

This attribute, as its name suggests, represents the year in which the house was built. Again, the plots in Figures [2.14, 2.15] show that the value of this attribute is a good discriminator for the class label.

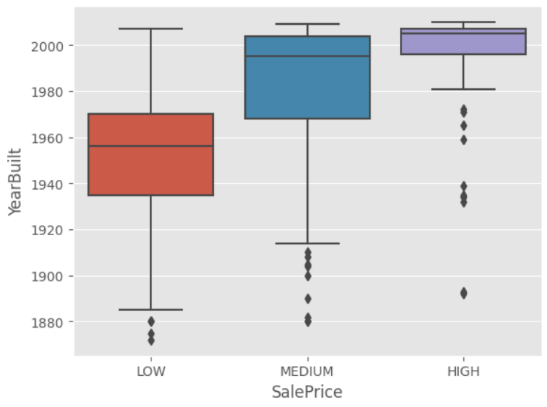


Figure 2.14: YearBuilt-SalePrice

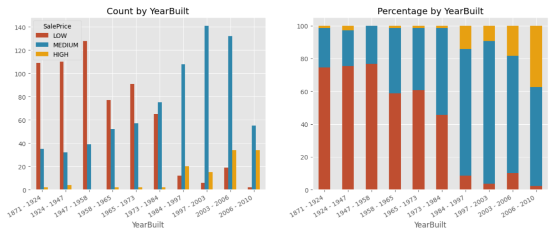


Figure 2.15: YearBuilt-SalePrice

2.4.11 GarageYrBlt

The attribute indicates the year of construction of the garage. Accordingly to the actual presence of a garage, it can assume a numeric value or *NA* as well as the other attributes describing the garage. Moreover, the attribute is considered to be directly related to the sale price of the house (of course a newer garage has an higher value).

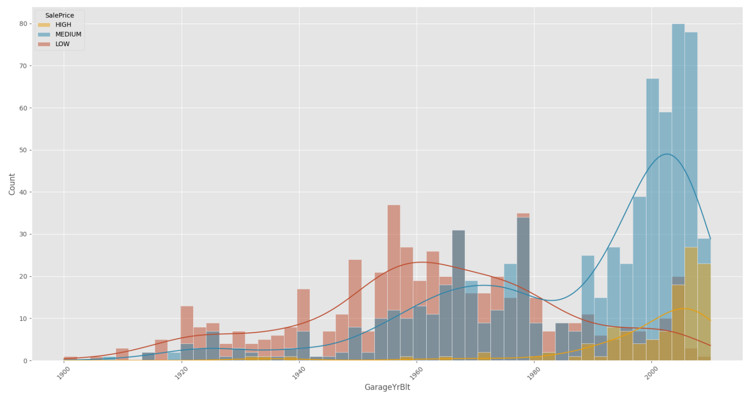


Figure 2.16: Garage Year Built

2.4.12 BsmtFinType1

Relatively to the basement, the attribute describes the quality of the finished area. As the plots in Figure 2.17 show, the higher ratings, such as *GLQ* and *ALQ*, and respectively the lower ratings, such as *BLW* and *LwQ*, directly affect the sale price of the house. However, it can be noticed a relevant numbers of houses with either unfinished or missing basement.

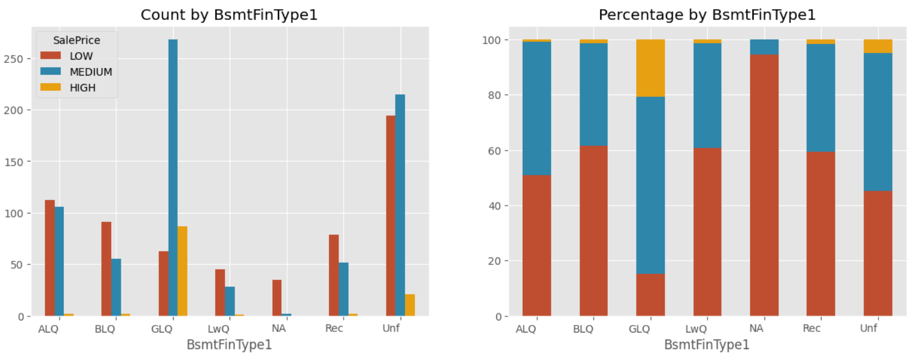


Figure 2.17: Finished basement rating categories

Evetually, it is reported a statistical test in figure 2.18, in which the low p-value strengthens the hypothesis of a possible relation between *BsmtFinType1* and *SalePrice*.

```
Understanding BsmtFinType1:
    Missing values: 37
    Independence Test: 3.52e-57[<0.05]
    Information Gain: 0.16
    When the BsmtFinType1 is:
        * NA, then the sale price is 90% likely to be LOW
          (2.53% of the times)

        * GLQ, then the sale price is 60% likely to be MEDIUM
          (28.63% of the times)

        * BLQ+LwQ, then the sale price is 60% likely to be LOW
          (15.21% of the times)

        * Rec, then the sale price is 55% likely to be LOW
          (9.11% of the times)

        * ALQ, then the sale price is 50% likely to be LOW
          (15.07% of the times)

        * Unf, then the sale price is 50% likely to be MEDIUM
          (29.45% of the times)
```

Figure 2.18: Statistical test of the relation between *BsmtFinType1* and SalePrice with a significance level of 5%

2.4.13 KitchenQual

As reported in the plots in Figure 2.19, the quality of the kitchen seems to accurately distinguish between *HIGH* and the other class labels. Nevertheless, it has to be taken in account the notable imbalance between the classes.

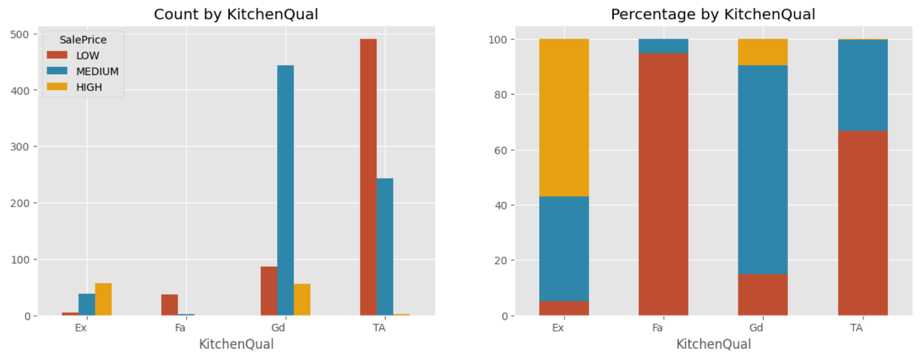


Figure 2.19: Kitchen quality barplot

2.4.14 GarageArea $\iff$ GarageCars

We also focused in the correlation between attributes, here an example of correlation of GarageArea and GarageCars. We can see in figure 2.20 that the squares fit of garage is affected of the number of cars people place in it.

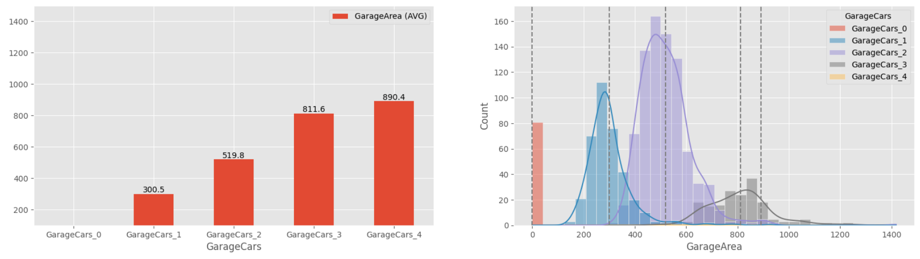


Figure 2.20: Correlation between GarageArea and GarageCars

2.5 Data Quality

The data set does not contain any null value, in the sense that there is no empty cell for any example and for any attribute. However, this does not mean that there are no nulls. From a first inspection of the data we noticed that there are many attributes valued to *NA*. As written in the file *data_description.txt*, the meaning of this value is often intended as a defect. In most cases *NA* stands for a feature that in a certain house is missing. Just as an example, the value *NA* in the attribute *Alley* means that the house has no alley access.

attribute	count of NA	percentage of NA
LotFrontage	**259**	**17.74%**
Alley	**1369**	**93.77%**
MasVnrType	8	0.55%
MasVnrArea	8	0.55%
BsmtQual	37	2.53%
BsmtCond	37	2.53%
BsmtExposure	38	2.60%
BsmtFinType1	37	2.53%
BsmtFinType2	38	2.60%
Electrical	1	0.06%
FireplaceQual	**690**	**47.3%**
GarageType	81	5.55%
GarageYrBlt	81	5.55%
GarageFinish	81	5.55%
GarageQual	81	5.55%
GarageCond	81	5.55%
PoolQC	**1453**	**99.5%**
Fence	**1179**	**80.8%**
MiscFeature	**1406**	**96.3%**

Table 2.1: Count and percentages of *NA* values by attributes (if present)

In Table 2.1 we report the list of attributes containing *NA* values along with the percentage and whether it refers to a null/missing or to a categorical value. For example, that is the case of the attribute *GarageType*. In fact, a NA value for this attribute does not refer to a missing value but it is used in case the house is not equipped with any garage. Another misleading example is the case of the *MasVnrType* attribute. It is a categorical attribute admitting the value *None* in case there is no masonry veneer. However, 8 (around 0.55%) of its values are *NA* making a possible interpretation ambiguous. Since it happens only 0.55% of the times, the *NA* values of this attribute are likely to be mapped to *None* during the *data preparation* phase in order to clear this ambiguity.
Regarding inconsistencies, the data set contains a few of them. Most of the times the problem was having *NA* values inside numeric features (that were not seen as numeric in the data frame just because this value was present). Another

problem, this time a bit more complicated is that there are a couple of houses having *1story* as style, but not zero as *2ndFloorSF*. The last one concerns the pair of attributes *MasVnrType* and *MasVnrArea*. This time *MasVnrType* was valued to *None*, which means no veneer, but *MasVnrArea* was not zero as it should have been.

2.6 Data Understanding Results

In tables [2.2, 2.3] we show what are our preliminary considerations after the data understanding phase.

column name	description
Id	Unique Identifier, No visible relation with SalePrice
MSSubClass	Good predictor
MSZoning	Redundant
LotFrontage	No visible relation with SalePrice
LotArea	Good Predictor, but highly correlated with SF
Street	No visible relation with SalePrice
Alley	No visible relation with SalePrice
LotShape	No visible relation with SalePrice
LandContour	No visible relation with SalePrice
Utilities	No visible relation with SalePrice
LotConfig	No visible relation with SalePrice
LandSlope	No visible relation with SalePrice
Neighborhood	Good predictor
Condition1	No visible relation with SalePrice
Condition2	No visible relation with SalePrice
BldgType	No visible relation with SalePrice
HouseStyle	Good predictor
OverallQual	Good predictor
OverallCond	Redundant, OverallQual is more important
YearBuilt	Good predictor
YearRemodAdd	Good predictor
RoofStyle	No visible relation with SalePrice
RoofMatl	No visible relation with SalePrice
Exterior1st	Good predictor
Exterior2nd	Redundant (considered Exterior1st)
MasVnrType	Good predictor
MasVnrArea	Good predictor
ExterQual	Good predictor
ExterCond	Already considered ExterQual
Foundation	Good predictor
BsmtQual	Good predictor
BsmtCond	No visible relation with SalePrice
BsmtExposure	No visible relation with SalePrice
BsmtFinType1	Good predictor
BsmtFinSF1	No visible relation with SalePrice
BsmtFinType2	Already considered Type1 which is more relevant
BsmtFinSF2	aldready Considered Type1
BsmtUnfSF	No visible relation with SalePrice
TotalBsmtSF	Good predictor (is part of the total SF area of the house)
Heating	No visible relation with SalePrice
HeatingQC	No visible relation with SalePrice

Table 2.2: Classification of attributes after Data Understanding(1)

Column Name	Description
CentralAir	No visible relation with SalePrice
Electrical	No visible relation with SalePrice
1stFlrSF	Good predictor
2ndFlrSF	Good predictor
LowQualFinSF	No visible relation with SalePrice
GrLivArea	Good predictor, correlated with SF
BsmtFullBath	Considered FullBath
BsmtHalfBath	Considered FullBath
FullBath	Good predictor
HalfBath	No visible relation with SalePrice, already considered FullBath
Bedroom	No visible relation with SalePrice (at least if considered alone)
Kitchen	No visible relation with SalePrice
KitchenQual	Good predictor
TotRmsAbvGrd	Good predictor
Functional	Generic home functionalities
Fireplaces	Good predictor
FireplaceQu	No visible relation with SalePrice
GarageType	Good predictor
GarageYrBlt	No strong correlation with salePrice (the wrapper feature selection says something different
GarageFinish	Good predictor
GarageCars	related to GarageArea, redundant
GarageArea	Good predictor
GarageQual	Considered GarageTyple
GarageCond	partially inside GarageType
PavedDrive	No visible relation with SalePrice
WoodDeckSF	No visible relation with SalePrice
OpenPorchSF	No visible relation with SalePrice
EnclosedPorch	No visible relation with SalePrice
3SsnPorch	No visible relation with SalePrice
ScreenPorch	No visible relation with SalePrice
PoolArea	No visible relation with SalePrice
PoolQC	No visible relation with SalePrice
Fence	No visible relation with SalePrice
MiscFeature	No visible relation with SalePrice
MiscVal	No visible relation with SalePrice
MoSold	No visible relation with SalePrice
YrSold	No visible relation with SalePrice
SaleType	No visible relation with SalePrice
SaleCondition	No visible relation with SalePrice
SalePrice	Sale price of the house (target feature)

Table 2.3: Classification of attributes after Data Understanding(2)

2.6.1 Correlation

In this phase we concentrated in understanding the correlation between the selected attributes and the target feature. In addition we have also taken into account how those features are correlated one another, in order to identify possible redundant information to exploit during the feature selection phase, we provided a correlation matrix and a correlation graph. For each pair of features (f_1, f_2) a different correlation coefficient is computed based of their types. If both are numeric the *Pearson's coefficient* is used otherwise the *Cramer's V coefficient*. In the last case, the one that is numeric is previously categorized.

Matrix correlation

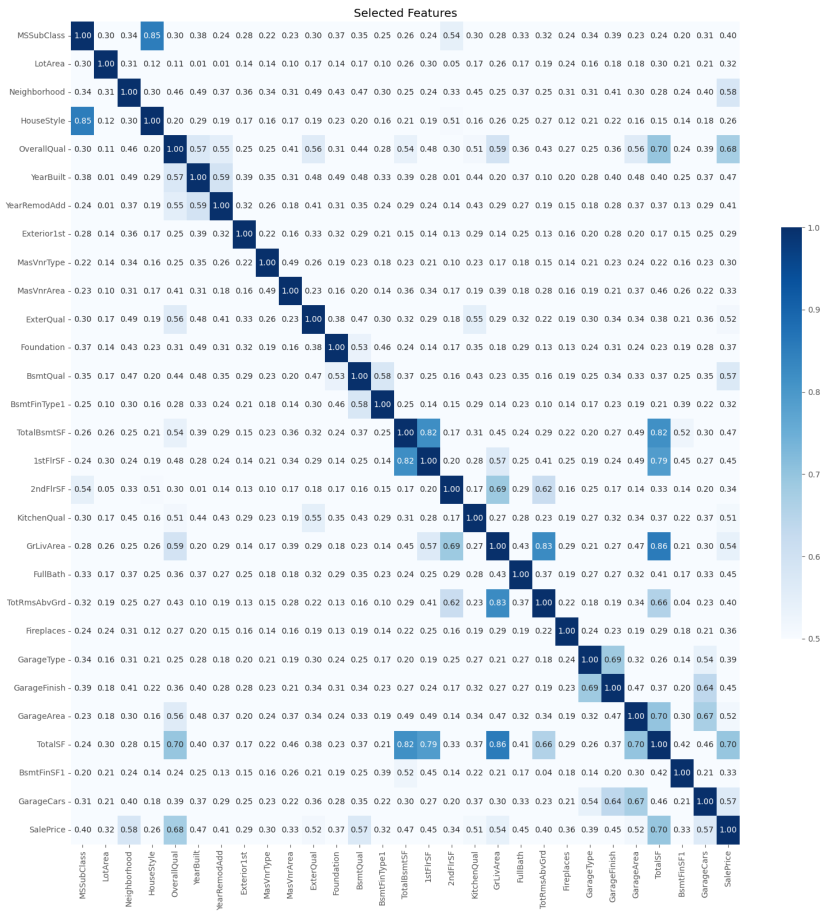


Figure 2.21: Correlation matrix

Graph correlation

In figure 2.22 is the correlation graph. Each vertex is a feature that we previously selected as a consequence of the data understanding phase (refer section 3.1 for more info). Then the graph is created as follows:

1. the size of each node represents the level of correlation with the target
2. an edge is created iff the correlation between the features it connects is above some threshold and its weight is the correlation
3. with a greedy approach, the edge with the highest weight is chosen identifying the pair of attributes that are similar the most meaning that one can safely substitute the other.
4. the feature with the highest correlation with the target is kept and the it is marked with a cadet blue color, while the other is marked with a light-gray color and all its outgoing edges are removed
5. the procedure is repeated from point 3 until the stopping condition (no more edges) is met.

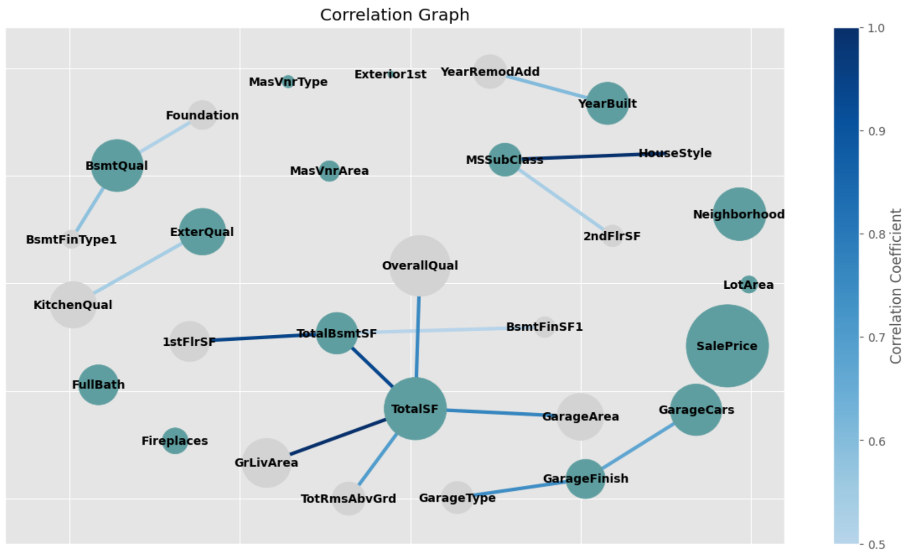


Figure 2.22: Correlation graph

From the provided correlation graph, we can see that different features have a good correlation with the target (size of the node) such as `TotalSF`, `Neighborhood`, `OverallQual` and `BsmtQual`. The analogous result can also be seen from the correlation matrix. In addition, many attributes are strongly correlated one another suggesting the possibility to remove one of them. Examples of this kind are the correlations (`TotalSF`, `TotalBsmtSF`), (`MSSubClass`, `HouseStyle`) and (`TotalBsmtSF`, `1stFlrSF`).

3. Data Preparation

3.1 Feature Selection - manual

In this step we select a bunch of attributes according to the result that came from the *data understanding* phase. As selection criteria we consider both the evidence of the statistic tests, that were conducted on most of the attributes, and the meaning of the attribute values distribution w.r.t the class label.

3.2 Data Cleaning

The method `data_cleaning_reduction` works on replacing nulls values and fixing inconsistencies that were found among the most important attributes. The list of attributes that need to be cleaned by this method comes from the analyses conducted in the data understanding step. In Figure 3.1 we show the map of types that is needed in the cleaning phase. In this map each key is the name of an attribute while the value is either a type or a type and the corresponding order for categorical ordinal attributes. Moreover, that map contains null values substitutions is used by the method for determining what values are to be replaced. The last part of the method also deals with null values, but this time their corrections are more related to the semantic of the data. Just to give an example, this method has to replace the value *NA* of the attribute *MasVnrArea* with the value 0 whenever *MasVnrType* for the same example is valued to *None*. The reasoning in this case is the following: a house that has no veneer decoration must have 0 as the decorated area.
Another method that deals with data cleaning is called `inconsistencies` in the notebook. Such method has the responsibility of fixing all the inconsistencies that we discovered on the good features. The method tries to detect possible inconsistencies in the data and, if they are found, they are printed and the associated correction is made. Again, we focused on a restricted subset of relevant attribute because checking all the features was meaningless and time wasting. More details are on the notebook, and the inconsistencies that we corrected are the same as referred in the last section.

```
GOOD_FEATURES_TYPE_MAP = \
    {"MSSubClass": ["ordinal integer"],
    "Neighborhood": ["category"],
    "HouseStyle": ["category"],
    "OverallQual": ["int64"],
    "YearBuilt": ["int64"],
    "YearRemodAdd": ["int64"],
    "Exterior1st":["category"],
    "MasVnrType":["category"],
    "MasVnrArea":["int64"],
    "ExterQual":["ordinal category",["Po", "Fa","TA","Gd","Ex"]],
    "Foundation":["category"],
    "BsmtQual":["ordinal category",["NA","Po","Fa","TA","Gd","Ex"]],
    "BsmtFinType1":["ordinal category",["NA","Unf","LwQ","Rec","BLQ","ALQ","GLQ"]],
    "TotalBsmtSF":["int64"],
    "1stFlrSF":["int64"],
    "2ndFlrSF":["int64"],
    "GrLivArea":["int64"],
    "FullBath":["int64"],
    "KitchenQual":["ordinal category", ["Po","Fa","TA","Gd","Ex"]],
    "TotRmsAbvGrd":["int64"],
    "Fireplaces":["int64"],
    "GarageType":["category"],
    "GarageFinish":["ordinal category",["NA","Unf","RFn","Fin"]],
    "GarageArea":["int64"],
    "LotArea": [’int64’],

```

Figure 3.1: Map for data cleaning

3.3 Data Aggregation & Normalization

As a result of the *data understanding* phase, we decided to aggregate some of the values of attributes based on how they are distributed. The method that deals with aggregation and normalization is called `data_transformation`. Such method takes as input a map that we constructed according to the distribution of the attribute values that emerged from the plots in the *data understanding* phase. This map is used in the aggregation phase, and has as keys the name of the attributes that require aggregation and as values another map [3.2] . This second map represents how the values in the domain of the attribute should be mapped to other values. This mapping can either be done towards values that were already in the domain of the attribute or to new values. The other step of normalization is instead done by simply iterating over all numeric attributes and by calling the `StandardScaler` on them.

```
GOOD_FEATURES_AGGREGATED_VALUES = {
"HouseStyle": {
                "1Story" : "other",
                "1.5Unf" : "other",
                "2.5Fin" : "other",
                "2.5Unf" : "other",
                "Sfoyer" : "other",
                "SLvl" : "other"
                },
"OverallQual": {
                1 : 4,
                2 : 4,
                3 : 4,
                10 : 9,
                },
"Exterior1st": {
                "AsbShng" : "other",
                "AsphShn" : "other",
                "BrkComm" : "other",
                "BrkFace" : "other",
                "CBlock" : "other",
                "CemntBd" : "other",
                "HdBoard" : "other",
...

```

Figure 3.2: Map for data aggregation

3.4 Feature Creation

As it emerged from the Data Understanding, there are several attributes that describe at a very high level of detail some main characteristics of houses. For this reason we decided to create the attribute *TotalSF* that is a summary of several aspects related to the dimensions of houses.
With the intent of splitting as much as possible the three classes we also created the attributes *TotalSF_prop* and *TotalSF_high*. All the three created features are explained in the following subsections

3.4.1 TotalSF

The attributes `1stFlrSF`, `2ndFlrSF`, `TotalBsmtSF` converge in determining what is the total dimension of houses. So, we decided to create an attribute that collects those information in a single numeric feature that we called *TotalSF*. As it can be seen from the plots in Figures [3.3, 3.4] there is a trend.

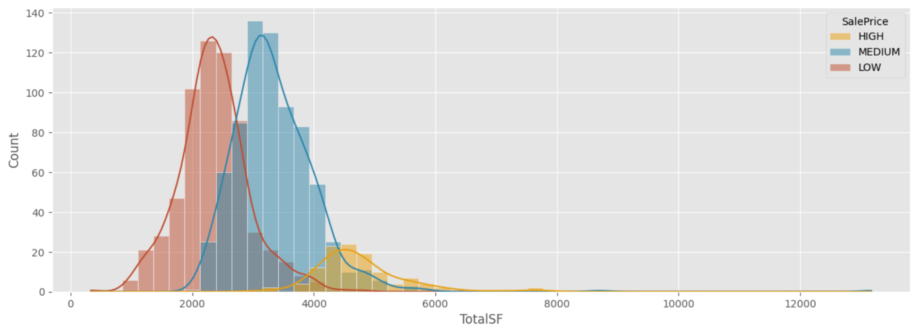


Figure 3.3: Total square feet

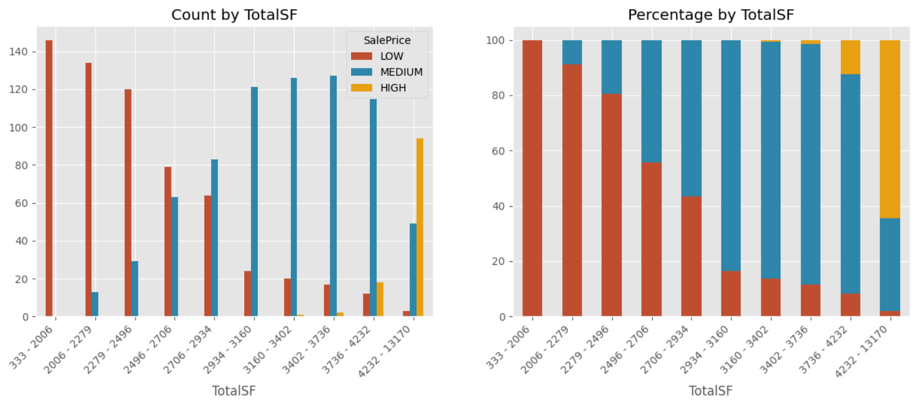


Figure 3.4: Total square feet

3.4.2 TotalSF_prop

This attribute simulates the interaction with total square feet, neighborhood and the overall quality of the house. Each neighborhood is associated with a weight representing an order of quality. TotalSF_prop is the multiplication between the neighborhood (the corresponding weight), totalSF and overallQual.

3.4.3 TotalSF_high

This attribute is boolean and its intent is to identify a house with value high and it depends on TotalSF_prop. Since we noticed that the value of the property based only on the square feet and the neighborhoods is sufficient to be considered high.

3.4.4 Others

Additional feature creations that we did not take into consideration since they are not that much related to the target features are:

- number of bathroom given by the sum of `TotalBsmtSF`, `1stFlrSF` and `2ndFlrSF`
- how old (in years) was the house when it was sold given by the difference between `YrSold` and `YearBuilt`

3.5 Feature Selection - Wrapper

This step exploits a wrapper approach that, starting from the selected kernel of attributes in the manual feature selection, tries to both add and remove other attributes of the data set. As black-box model we use a *Random Forest Classifier*. The way we compare the model with the one having one more or one less feature is based on the computation of different metrics such as the *accuracy*, *balanced_accuracy*, *precision*, *recall* and the *f1_score* computed over a *K-Fold* evaluation method. In addition, for each metric a corresponding *weight* can also be specified guiding the feature selector to improve all the metrics in a weighted fashion. Moreover, the selector can be guided to improve the metrics (except for the *accuracy*) for a specific class only, between `LOW`, `MEDIUM` and `HIGH`. The backward, or deletion phase is implemented in the `validate_backward` method while the addition phase is in the `validate_forward` method. Both methods are implemented following a *greedy* approach.

3.5.1 `validate_backward` method

For deciding the exclusion of an attribute, first a dataset D with the starting set of features F along with a set of metrics ME and the number n of iterations are provided. Let t be the target feature. Then, the method proceeds as follows:

1. for each feature $f_i \in F$, the correlation c_i between f_i and t is computed as described in the figure 2.21
2. a list C of all computed coefficients is sorted in **ascending** order
3. following a *greedy* approach, the first element c' in C is taken corresponding to the feature f' that less influences the target t
4. the feature f' is removed from D obtaining a new candidate datatest D'
5. let M, M' be two models for D and D' respectively, then both are evaluated n times using the *KFold* method obtaining the two samples S_M and $S_{M'}$ of metrics in ME (when $|ME| > 1$ the mean is considered)

6. a statistic test for the comparison of the means $\mu(S_M)$ and $\mu(S_{M'})$ is performed choosing $\mu(S_{M'}) < \mu(S_M)$ as alternative hypothesis (decrease in performance) and obtaining the corresponding *pvalue* p to compare with an error rate[1] $\alpha = 0.1$
   - if $p \leq \alpha$, then dropping the feature will decrease the metrics hence it is kept
   - otherwise, there is no evidence in a decrease of the metrics hence the feature is dropped and $D = D'$
7. c' is removed from C
8. while $|C| > 0$, the process is repeated from point 3

```
Feature Selection - backward phase
-----------------------------------

Current dataframe width: 12
Model name DT
Staged dataframe score (avg): 0.823685

TotalSF selection - Validation outcome:
  - keep candidate
  - pvalue: 0.000029
  - candidate score: 0.799488
  - current score: 0.823685

MasVnrArea selection - Validation outcome:
  - drop candidate
  - pvalue: 0.623791
  - candidate score: 0.824918
  - current score: 0.824918

Model name RFST
Final dataframe score (avg): 0.826359

Features to keep: ['TotalSF']
Features to remove: ['MasVnrArea']
```

Figure 3.5: Execution output of the `validate_backward` method

3.5.2 `validate_forward` method

For deciding the inclusion of a new attribute, first a dataset D with the starting set of features F along with a set of metrics ME and the number n of iterations are provided. Let t be the target feature. Then, the method proceeds as follows:

1. for each feature $f_i \notin F$, the correlation c_i between f_i and t is computed as described in the figure 2.21

[1]The value of α is high since the null hypothesis is to drop the feature so it is preferred to increase α to be safer

2. a list C of all computed coefficients is sorted in **descending** order

3. following a *greedy* approach, the first element c' in C is taken corresponding to the feature f' that mostly influences the target t

4. the feature f' is now inserted into D obtaining a new candidate datatest D'

5. let M, M' be two models for D and D' respectively, then both are evaluated n times using the *KFold* method obtaining the two samples S_M and $S_{M'}$ of metrics in ME (when $|ME| > 1$ the mean is considered)

6. a statistic test for the comparison of the means $\mu(S_M)$ and $\mu(S_{M'})$ is performed choosing $\mu(S_{M'}) > \mu(S_M)$ as alternative hypothesis (increase in performance) and obtaining the corresponding *pvalue* p to compare with an error rate $\alpha = 0.05$

    - if $p \leq \alpha$, then inserting the feature will increase the metrics hence $D = D'$
    - otherwise, there is no evidence in an increase of the metrics hence the feature is discarded

7. c' is removed from C

8. while $|C| > 0$, the process is repeated from point 3

```
Feature Selection - forward phase
-----------------------------------

Current dataframe width: 12
Model name RFST
Staged dataframe score (avg): 0.825566

Fireplaces selection - Validation outcome:
  - discard candidate
  - increase: -0.001777
  - candidate score: 0.823789
  - pvalue: 0.599245
  - current score: 0.825566

1stFlrSF selection - Validation outcome:
  - discard candidate
  - increase: -0.002805
  - candidate score: 0.822761
  - pvalue: 0.842761
  - current score: 0.825566

...

2ndFlrSF selection - Validation outcome:
  - discard candidate
  - increase: -0.007042
  - candidate score: 0.818524
  - pvalue: 0.883400
  - current score: 0.825566

BsmtFinSF1 selection - Validation outcome:
  - include candidate
  - increase: 0.005663
  - candidate score: 0.831230
  - pvalue: 0.035269
  - current score: 0.831230

Model name RFST
Final dataframe score (avg): 0.830643

Features to include: ['BsmtFinSF1']
```

Figure 3.6: Execution output of the `validate_forward` method

The two methods described above are recursively applied in an alternative fashion. At the beginning, a starting dataset is defined over a set of initial features based on a "manual" and correlation-based analysis as described in the sections 2.6.1 and 3.1.

Mitigate the limits of the *greedy* approach

As described above, both methods `validate_backward` and `validate_forward` follow a *greedy* approach in selecting, respectively, a candidate feature to drop or insert based on the most promising correlation coefficient. However, an early choice of a feature can penalize a future gain in performance compromising the optimality of the feature selection. In order to mitigate this inconvenient, both methods are recursively applied one after the other until a *fix point* is reached.

On the starting dataset the `validate_backward` method is applied in order to possibly remove useless features (no evidence in a decrease of the metrics). After the *backward* phase, `validate_forward` method is applied on the resulting dataset in order to possibly insert additional features that significantly increase the metrics obtaining a new candidate dataset and then the process is repeated again until the two methods will converge to a *fix point* (iteration in which it is no longer convenient to perform a drop or an insert).

The resulting set of features when the *fix point* is reached are those that we finally selected.

3.6 Outlier analysis

Aside from the treatment of `NA` and missing values, in order to avoid a misleading effect in the modelling phase, we also performed an analysis of possible outliers in the distribution of the selected features. In particular, we firstly differentiated the analysis between the different data types.

For numeric features, outliers are highlighted using the *interquartile range* method since most of them have a skewed distribution:

- the upper and lower fences are computed as $Q_3 + (1.5 \times IQR)$ and $Q_1 - (1.5 \times IQR)$, respectivelty, where Q_1 and Q_3 are the first and third quartile while IQR is the interquartile range.
- the values of the feature falling outside the two fences are considered to be outliers.

For categorical features, outliers are instead highlighted analyzing the frequencies for each category. Frequencies below a certain threshold (usually 1%) are considered to be outliers.

During the *data understanding* phase, we noted that many attributes have outliers when cross-plotted with the classes of the target feature. For example, it is evident that high price houses very unlikely have a detached garage (very few instances) as shown in the following plot.

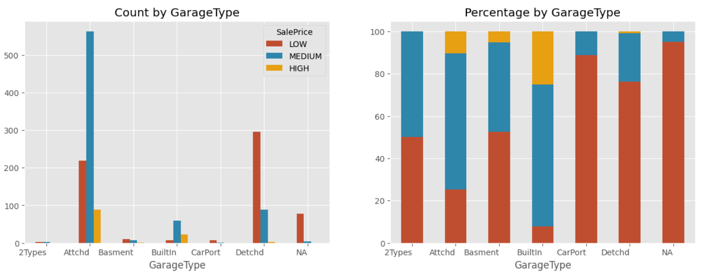


Figure 3.7: Type of garage for each price class

In particular, for categorical and numerical features the correlation outlier analysis is differentiated w.r.t. each value of the target simply by reapplying the methods described above.

Outliers removal vs limited dataset size

Even though highlight and remove outliers can avoid to mislead the modelling phase, removing too much of them can be an issue especially in case of a very limited dataset size as in our case. In fact, if on one hand we have misleading effect due to the presence of outliers, on the other, reducing further the dataset will lead to an *underfitting* issue. Based on those considerations, we mostly decided to disable the `OutlierAnalyzer` or to use it with very low thresholds.

4. Modelling and Validation

4.1 Selection of Modelling Techniques

For the selection of the best modelling techniques we reused the function that computes the metrics of a given model via the **K-fold** technique. We passed to the function different types of models, and according to the returned metrics we decided which were the most suitable models for our classification task. We show in Table 4.2 the resulting metrics for each type of model. As metrics we used a balanced accuracy score, precision, recall and F1-score.
Since the the data set is strongly imbalanced (as already shown in the plot in Figure 2.1) we decided to apply a re sampling technique for balancing the amount of examples belonging to the minority class, which in our case was the class *HIGH*. We show the resulting distribution of the class labels in the Figure 4.1. In particular we used `SMOTENC` that comes with the library `imblearn`. Of course, the sampling was only done on the training training set for the K-fold analysis and on the training set for the final models.

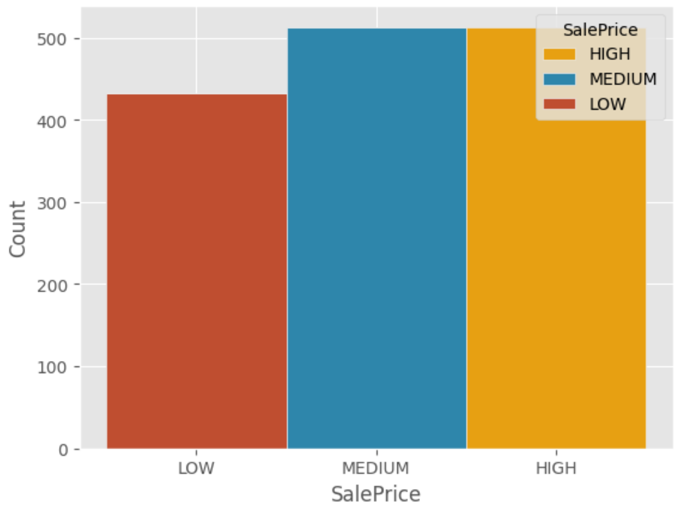


Figure 4.1: Results of re sampling for the minority class

4.2 Validation Report

As outcome of the validation phase, in the following tables we reported the main monitored measures of quality. In addition, we also collected all the fit and prediction times for each of the validated models.

Model	Accuracy (avg)	Precision (avg)	Recall (avg)	F1-Score (avg)
Decision Tree-EN	0.79	0.77	0.76	0.75
Decision Tree-GI	0.76	0.75	0.76	0.76
Random Forest	**0.85**	**0.84**	**0.82**	**0.82**
Random Forest-BL	**0.85**	**0.85**	**0.82**	**0.83**
ADAB	0.77	0.77	0.73	0.73
Gradient Boosting	**0.85**	**0.83**	**0.82**	**0.82**
Naive Bayes	0.63	0.61	0.72	0.57
Neural Network	0.77	0.75	0.68	0.71

Table 4.1: Selection of best models

Model	Fit Time (avg - sec)	Prediction Time (avg - sec)
Decision Tree-EN	0.0083	0.0014
Decision Tree-GI	0.0078	0.0018
Random Forest	**0.2825**	**0.0149**
Random Forest-BL	**0.8206**	**0.8316**
ADAB	0.1881	0.0138
Gradient Boosting	**1.1461**	**0.0030**
Naive Bayes	0.0037	0.0020
Neural Network	0.5057	0.0022

Table 4.2: Fit and Prediction Timings

We can note that the ones that perform better in terms of quality require more time. In particular, we noted an increase in time when we pass from the `Random Forest` to the `Random Forest-BL` even measures like *accuracy*, *precision*, *recall* and *F1-Score* remain the same.
As a final consideration, the `Gradient Boosting` model is the one that takes more time than others during the fitting phase but it will be much quicker to make a predict keeping the almost the same quality measures of the others that are selected in bold.

5. Evaluation

5.1 Build Models

As already mentioned, we selected the final models according to the results obtained in the previous step. For simplicity we consider only Random Forest and not Random Forest-BL because the results were similar. Also the Decision Tree classifier is considered, but it is not among the best models.

5.2 Model Tuning

As mentioned above, the Random Forest classifier led to the most promising performances during the *validation phase*. In order to tune its best parameters, we exploited the `RandomizedSearchCV` function of the `sklearn` module. In particular, for each parameter (`n_estimators`, `max_features`, `max_depth`, `min_samples_split`, `min_samples_leaf` and `bootstrap`) we fixed a set of possible values. Thereafter, the function `RandomizedSearchCV` will sample different parameter configurations from the specified distributions. Each sampled configuration is validated by fitting and evaluating the corresponding model. The most promising configuration is returned.

5.3 Model Assessment

The objective of this phase is to measure the goodness of each type of model that was selected in the previous step. In table Tab 5.1 we show the final results that we obtained with the best two models plus the Decision Tree Classifier. Moreover, we report the ROC Curve and the Confusion Matrix. As we can see from both the plots for the Confusion Matrix in Figures[5.1, 5.2, 5.3] and the ROC Curve in Figures[5.4, 5.5, 5.6], the best models are approximately equivalent in terms of performances, while the Decision Tree Classifier is quite behind them.
In conclusion, Random Forest and Gradient Boosting are not so far from each other. However, our preferred model remains the Random Forest

We also collected all the fit and prediction times for each of the tested models. We can note that the ones that perform better in terms of quality require more time. For the `Gradient Boosting` model, we observed the same effect: it is the one that takes more time than others during the fitting phase but it will be much quicker to make a prediction with respect to the `Random Forest`

Model	Accuracy	P. High	P. Low	P. Medium	R. High	R. Low	R. Medium
Decision Tree	0.78	0.69	0.80	0.78	0.63	0.82	0.77
Random Forest	**0.90**	**0.89**	**0.90**	**0.90**	**0.87**	**0.92**	**0.89**
Gradient Boosting	**0.90**	**0.83**	**0.90**	**0.91**	**0.86**	**0.95**	**0.88**

Table 5.1: Best models evaluation + DT

Model	Fit Time (s)	Prediction Time (s)
Decision Tree	0.0185	0.0029
Random Forest	**0.3951**	**0.0209**
Gradient Boosting	**1.4010**	**0.0050**

Table 5.2: Fit and Prediction Timings of best models

Now we analyze the difference between the confusion matrix of the Decision Tree Classifier and the Random Forest. We can see that the Random Forest beats the DT from every point of view, just as we were expecting and as it emerged from the results in Tab 5.1.

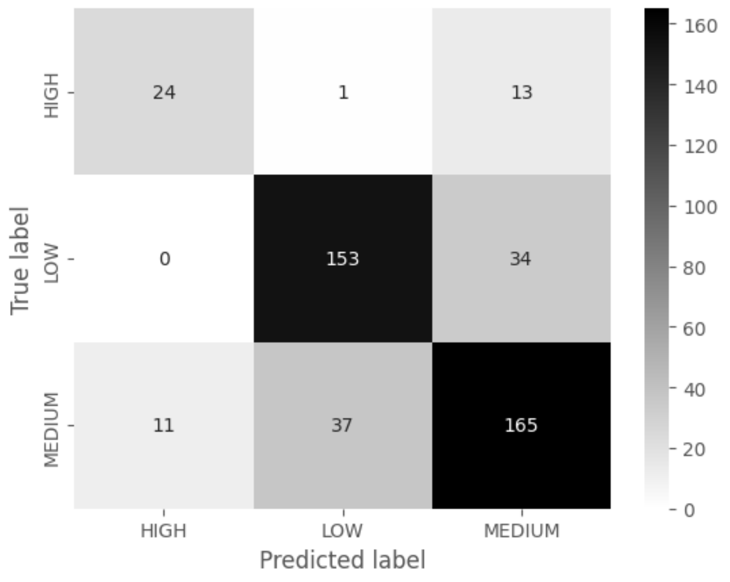


Figure 5.1: Confusion Matrix for Decision Tree Classifier

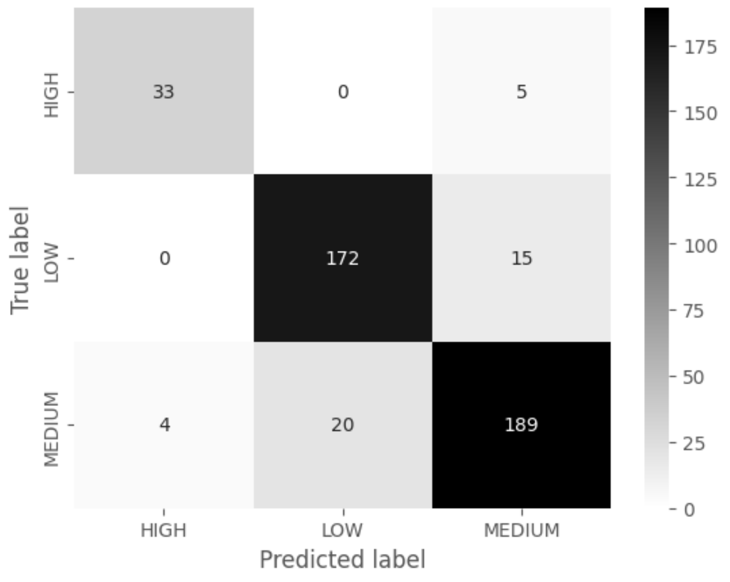


Figure 5.2: Confusion Matrix for Random Forest Classifier

We can make a better comparison between the two best models that are Random Forest and Gradient Boosting. In this case we can see that the class

HIGH is better identified by the Random Forest.On the other hand, the Gradient Boosting Classifier tends to recognize the class *LOW* more precisely.

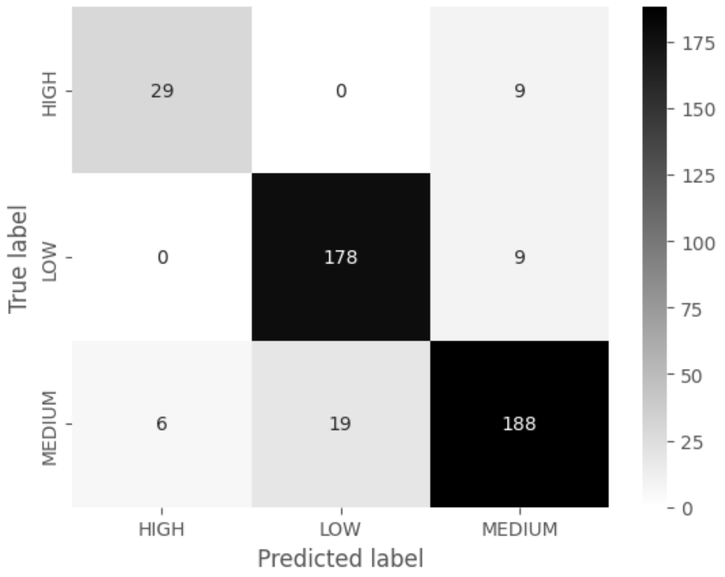


Figure 5.3: Confusion Matrix for Gradient Boosting Classifier

In the ROC Curve shown in Figure 5.4 we can see that the area is almost maximum. We believe that such a good performance is also owed to the re sampling technique we adopted for the minority class `HIGH`. In this way our models were trained on a much higher number of examples with class `HIGH` and thus they are able to correctly recognize most of the houses with label `HIGH`.

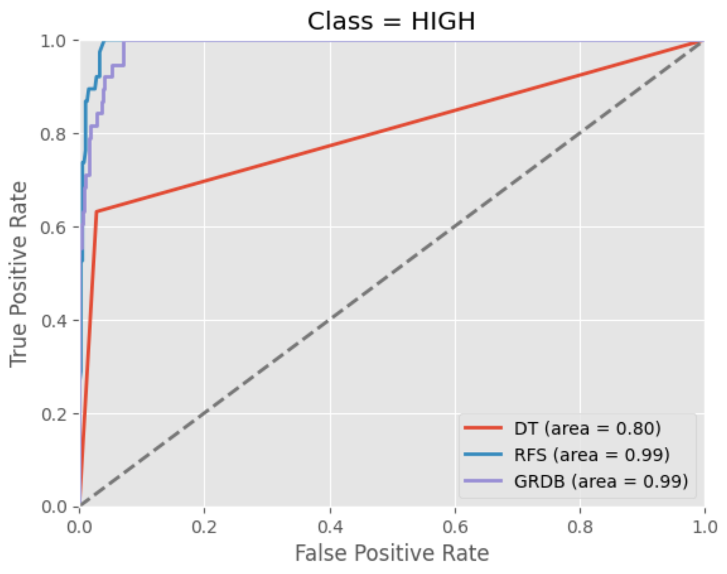


Figure 5.4: ROC Curve for the class *HIGH*

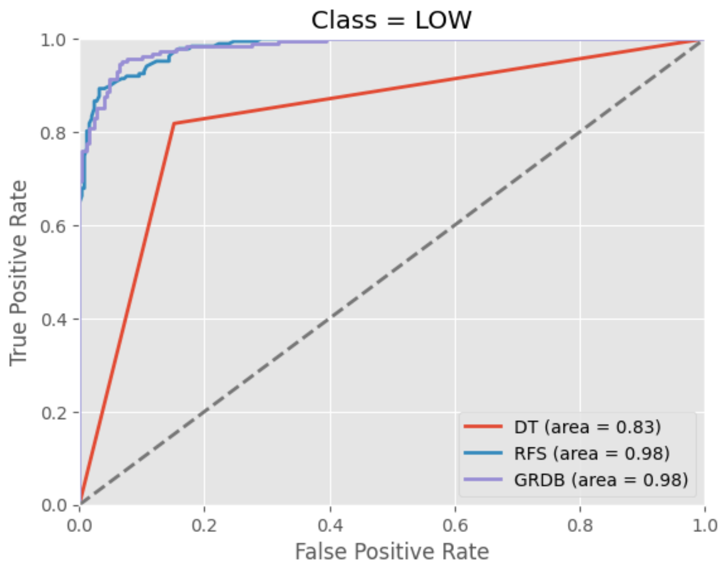


Figure 5.5: ROC Curve for the class *LOW*

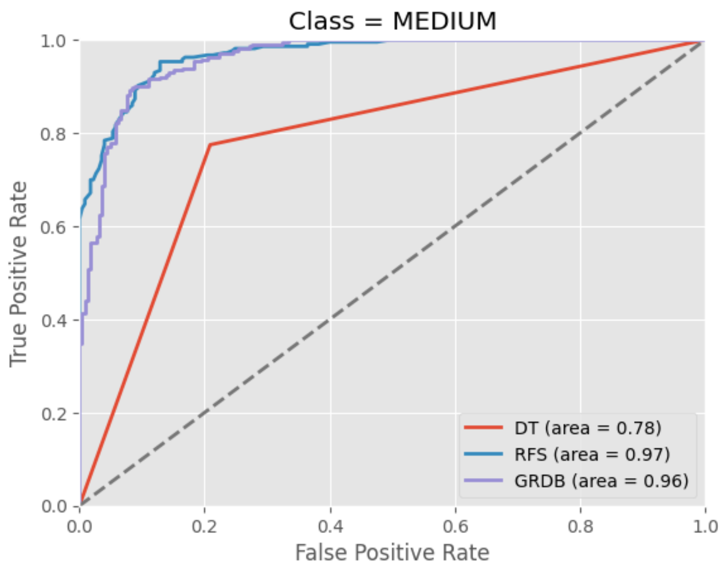


Figure 5.6: ROC Curve for the class *MEDIUM*

Bibliography

[1] Dean De Cock. Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19(3), 2011.