Machine Learning - House Prices Analysis

Daniele Avolio : 242423 Alessandro Fazio : 242422 Merem Hassem Indiris

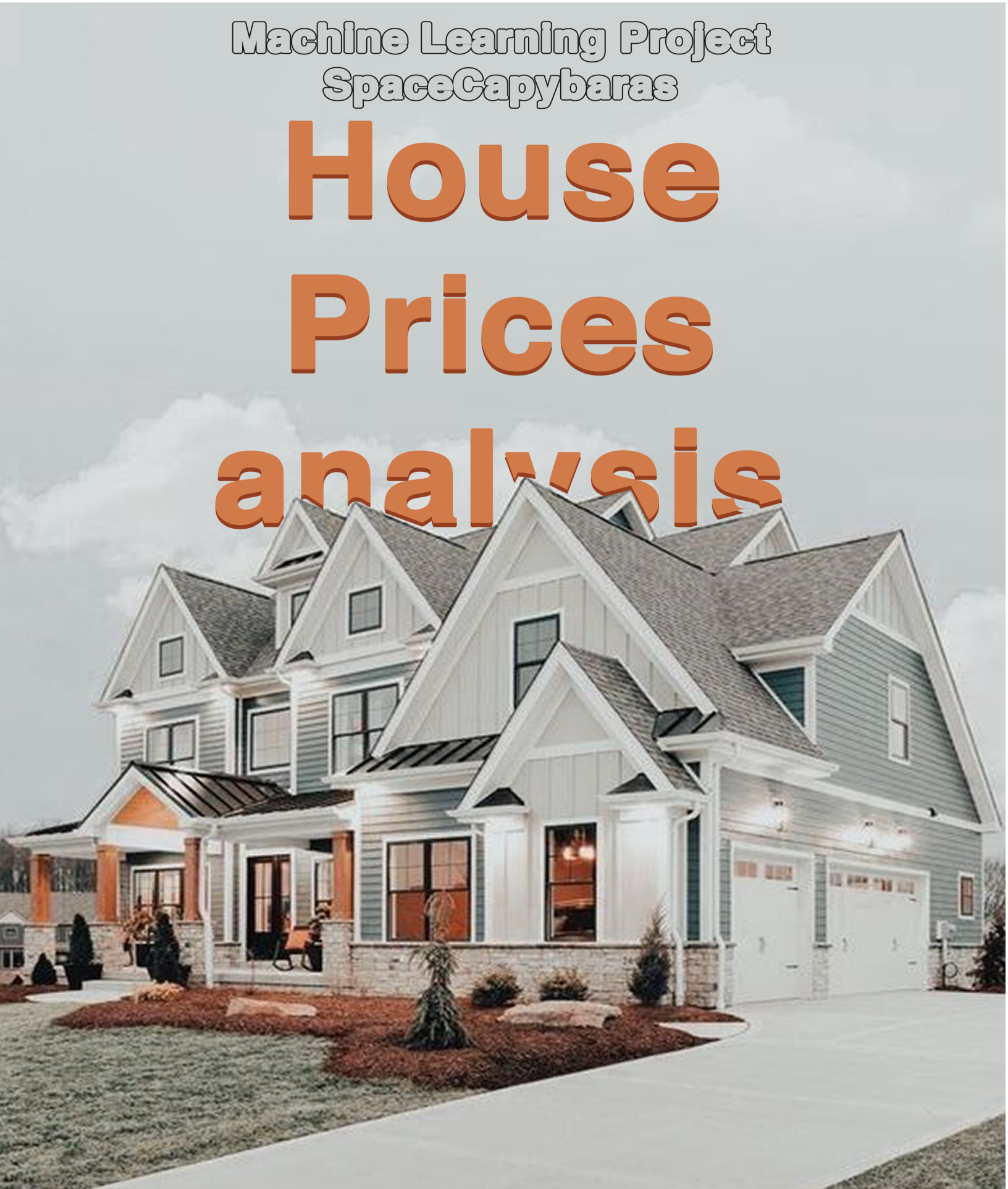
Michele Vitale: 247410 Lorenzo Piro

A.Y. 2022/2023

Machine Learning Project

SpaceCapybaras

House Prices analysis



Contents

Chapter 1

Introduction

To develop this **Machine Learning** project we are going to use the CRISP-DM methodology, which is a well-known and widely used methodology for data mining projects. It is an iterative process that is composed of six phases.

In particulal, the phases are:

1. **Business Understanding**: in this phase we will try to understand the problem and the objectives of the project. We will also try to understand the data that we have at our disposal and how we can use it to solve the problem.

2. **Data Understanding**: in this phase we will try to understand the data that we have at our disposal. We will try to understand the meaning of the data and how we can use it to solve the problem.

3. **Data Preparation**: in this phase we will try to prepare the data for the next phases. We will try to clean the data and to transform it in a way that will be useful for the next phases.

4. **Modeling**: in this phase we will try to build a model that will be able to solve the problem. We will try to find the best model for our problem.

5. **Evaluation**: in this phase we will try to evaluate the model that we have built. We will try to understand if the model is good enough to solve the problem.

6. **Deployment**: in this phase we will try to deploy the model that we have built. We will try to understand how we can use the model to solve the problem.

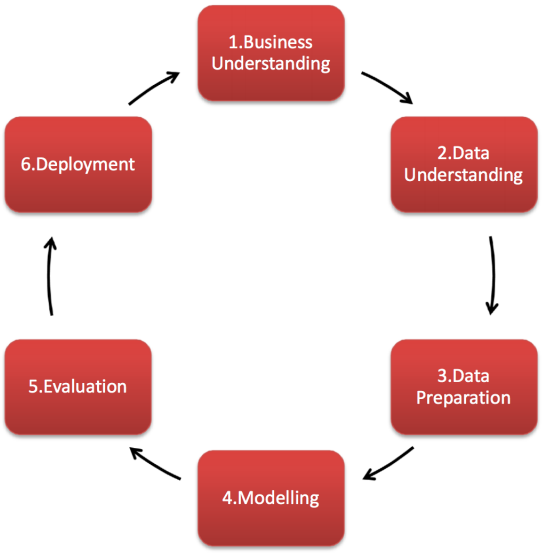


Figure 1.1: CRISP-DM methodology

Chapter 2

Business understanding

2.1 Background

Our project is based using the dataset of the Kaggle competition House Prices: Advanced Regression Techniques.

The goal of the competition is to predict the final price of each home based on 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa. The dataset is composed by 1460 rows and 81 columns, with an ID column and the last one which is the target variable, the Sale Price.

A small list of the variables is the following:

- **LotArea**: Lot size in square feet
- **OverallQual**: Overall material and finish quality
- **OverallCond**: Overall condition rating
- **YearBuilt**: Original construction date
- **YearRemodAdd**: Remodel date
- **RoofStyle**: Type of roof
- **Exterior1st**: Exterior covering on house
- **Exterior2nd**: Exterior covering on house (if more than one material)
- **MasVnrType**: Masonry veneer type
- **MasVnrArea**: Masonry veneer area in square feet
- ...

2.2 Business objectives

In this part we are going to analyze the business objective of the project. Our goal is different from the one of the competition, in fact we are requested to **convert the Sale Price variable into 3 ranges**:

1. **LOW**: From 0 to 150000
2. **MEDIUM**: From 150000 to 300000
3. **HIGH**: From 300000 and beyond

So, our goal is to predict a categorical variable with 3 possible values, instead of a continuous variable. This is a very important difference, because we are not interested in the exact value of the Sale Price, but only in the range in which it falls.

2.3 Business success criteria

The success criteria of the project is to obtain a model that is able to predict the Sale Price range with a good accuracy. In particular, we want to create a model that is able to predict the Sale Price range with an accuracy of at least 0.8.

2.4 Assessment of the situation

There is a very important aspect that we have to consider: there are a lot of variables in the dataset that contains a values of **NA**, that could leat to thinking that the value is missing.

Actually, this is not always true. In particular, the description of the dataset contains information about the meaning of the **NA** value for each variable. For example, the **NA** value for the **PoolQC** variable means that the house doesn't have a pool, so the value is not missing, but it is a value that has a meaning.

There are more variables that have a similar meaning for the **NA** value, so we have to be careful when we are going to handle the missing values. For this project, we are going to assume that the **NA** value is not missing but is just a value that has a meaning. If we need to drop the column, will be specified in the particular section

2.5 Inventory of resources

For this project, we are going to use the following resources:

- **Python 3.10.6**: The programming language used for the project
- **Jupyter Notebook**: The IDE used for the project
- **Pandas**: used for data manipulation
- **Numpy**: used for data manipulation
- **Matplotlib**: used for data visualization
- **Seaborn**: used for data visualization
- **Scikit-learn**: used for machine learning

2.6 Project Plan

The development of the project will be divided into 5 main phases, that are split like this:

- **Phase 1**: Data understanding (1 week)
- **Phase 2**: Data preparation (1 week)
- **Phase 3**: Modeling (1 week)
- **Phase 4**: Evaluation (1 week)
- **Phase 5**: Deployment (1 week)

Chapter 3

Data understanding

3.1 Data description

The dataset we are using, House Prices - Advanced Regression Techniques, is a dataset containing 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa. The dataset is composed of 1460 rows, each one representing a house, and 81 columns, 80 of which are features and 1 is the target variable, the SalePrice.

With the dataset is provided a *data description* with the description for each variable in thedataset.

Table 3.1: Property Features

Feature	**Description**
SalePrice	The property's sale price in dollars. This is the target variable that you're trying to predict.
MSSubClass	The building class
MSZoning	The general zoning classification
LotFrontage	Linear feet of street connected to property
LotArea	Lot size in square feet
Street	Type of road access
Alley	Type of alley access
LotShape	General shape of property

Continued on next page

Table 3.1 – continued from previous page

Feature	**Description**
LandContour	Flatness of the property
Utilities	Type of utilities available
LotConfig	Lot configuration
LandSlope	Slope of property
Neighborhood	Physical locations within Ames city limits
Condition1	Proximity to main road or railroad
Condition2	Proximity to main road or railroad (if a second is present)
BldgType	Type of dwelling
HouseStyle	Style of dwelling
OverallQual	Overall material and finish quality
OverallCond	Overall condition rating
YearBuilt	Original construction date
YearRemodAdd	Remodel date
RoofStyle	Type of roof
RoofMatl	Roof material
Exterior1st	Exterior covering on house
Exterior2nd	Exterior covering on house (if more than one material)
MasVnrType	Masonry veneer type
MasVnrArea	Masonry veneer area in square feet
ExterQual	Exterior material quality
ExterCond	Present condition of the material on the exterior

Continued on next page

Table 3.1 – continued from previous page

Feature	**Description**
Foundation	Type of foundation
BsmtQual	Height of the basement
BsmtCond	General condition of the basement
BsmtExposure	Walkout or garden level basement walls
BsmtFinType1	Quality of basement finished area
BsmtFinSF1	Type 1 finished square feet
BsmtFinType2	Quality of second finished area (if present)
BsmtFinSF2	Type 2 finished square feet
BsmtUnfSF	Unfinished square feet of basement area
TotalBsmtSF	Total square feet of basement area
Heating	Type of heating
HeatingQC	Heating quality and condition
CentralAir	Central air conditioning
Electrical	Electrical system
1stFlrSF	First Floor square feet
2ndFlrSF	Second floor square feet
LowQualFinSF	Low quality finished square feet (all floors)
GrLivArea	Above grade (ground) living area square feet

Continued on next page

Table 3.1 – continued from previous page

Feature	**Description**
BsmtFullBath	Basement full bathrooms
BsmtHalfBath	Basement half bathrooms
FullBath	Full bathrooms above grade
HalfBath	Half baths above grade
Bedroom	Number of bedrooms above basement level
Kitchen	Number of kitchens
KitchenQual	Kitchen quality
TotRmsAbvGrd	Total rooms above grade (does not include bathrooms)
Functional	Home functionality rating
Fireplaces	Number of fireplaces
FireplaceQu	Fireplace quality
GarageType	Garage location
GarageYrBlt	Year garage was built
GarageFinish	Interior finish of the garage
GarageCars	Size of garage in car capacity
GarageArea	Size of garage in square feet
GarageQual	Garage quality
GarageCond	Garage condition
PavedDrive	Paved driveway
WoodDeckSF	Wood deck area in square feet
OpenPorchSF	Open porch area in square feet
EnclosedPorch	Enclosed porch area in square feet

Continued on next page

Table 3.1 – continued from previous page

Feature	**Description**
3SsnPorch	Three season porch area in square feet
ScreenPorch	Screen porch area in square feet
PoolArea	Pool area in square feet
PoolQC	Pool quality
Fence	Fence quality
MiscFeature	Miscellaneous feature not covered in other categories
MiscVal	$Value of miscellaneous feature
MoSold	Month Sold
YrSold	Year Sold
SaleType	Type of sale
SaleCondition	Condition of sale

An important thing to specify is that the dataset is given with an additional "data_description.txt" file that describes the features in detail. This file is used to understand the features and their values. From here, some variables have a specific value to point the fact the the house doesn't have that feature. For example, the feature "Alley" has the value "NA" to point the fact that the house doesn't have an alley. This is important to know. The next list contains all the features that have the value 'NA' indicating the non-existence of the feature.

1. Alley
2. BsmtQual
3. BsmtCond
4. BsmtExposure
5. BsmtFinType1
6. BsmtFinType2
7. FireplaceQu
8. GarageType
9. GarageFinish
10. GarageQual
11. GarageCond

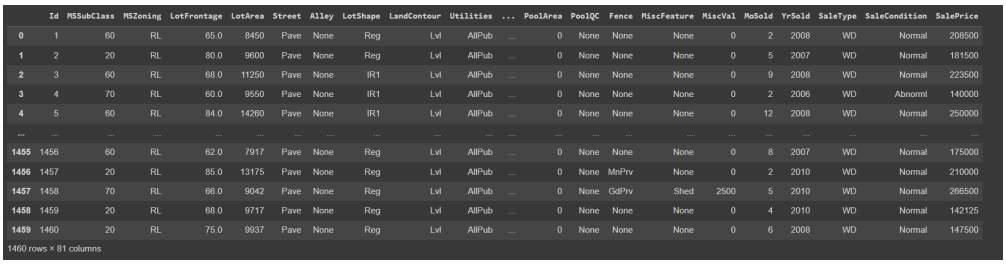
12. PoolQC
13. Fence
14. MiscFeature

In this project, as a team we decided to consider those variables **not null**, and considering the value 'NA' always as the non-existence of the feature. This is important to know because it will be used in the data cleaning phase.

Note that this is not related to the other variables, since the variables not mentioned in the above list have the value 'NA' to indicate **null value**. For example, if in some rows the feature "MasVnrType" has the value "NA", it means that the value is null and not that the house doesn't have a masonry veneer type.

3.2 Data exploration

The dataset contains 81 features, and 1460 rows. The attributes gives information about the house, and the target variable is the SalePrice. The goal of this project is to predict the SalePrice of a house given the features. The features are a mix of categorical and numerical variables.



	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	1	60	RL	65.0	8450	Pave	None	Reg	Lvl	AllPub	...	0	None	None	None	0	2	2008	WD	Normal	208500
1	2	20	RL	80.0	9600	Pave	None	Reg	Lvl	AllPub	...	0	None	None	None	0	5	2007	WD	Normal	181500
2	3	60	RL	68.0	11250	Pave	None	IR1	Lvl	AllPub	...	0	None	None	None	0	9	2008	WD	Normal	223500
3	4	70	RL	60.0	9550	Pave	None	IR1	Lvl	AllPub	...	0	None	None	None	0	2	2006	WD	Abnorml	140000
4	5	60	RL	84.0	14260	Pave	None	IR1	Lvl	AllPub	...	0	None	None	None	0	12	2008	WD	Normal	250000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456	60	RL	62.0	7917	Pave	None	Reg	Lvl	AllPub	...	0	None	None	None	0	8	2007	WD	Normal	175000
1456	1457	20	RL	85.0	13175	Pave	None	Reg	Lvl	AllPub	...	0	None	MnPrv	None	0	2	2010	WD	Normal	210000
1457	1458	70	RL	66.0	9042	Pave	None	Reg	Lvl	AllPub	...	0	None	GdPrv	Shed	2500	5	2010	WD	Normal	266500
1458	1459	20	RL	68.0	9717	Pave	None	Reg	Lvl	AllPub	...	0	None	None	None	0	4	2010	WD	Normal	142125
1459	1460	20	RL	75.0	9937	Pave	None	Reg	Lvl	AllPub	...	0	None	None	None	0	6	2008	WD	Normal	147500

1460 rows × 81 columns

Figure 3.1: Dataset print

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
    Data columns (total 81 columns):
    #    Column          Non-Null Count   Dtype
    ---  ------          --------------   -----
    0    Id              1460 non-null    int64
    1    MSSubClass      1460 non-null    int64
    2    MSZoning        1460 non-null    object
    3    LotFrontage     1201 non-null    float64
    4    LotArea         1460 non-null    int64
    5    Street          1460 non-null    object
    6    Alley           91 non-null      object
    7    LotShape        1460 non-null    object
    8    LandContour     1460 non-null    object
    9    Utilities       1460 non-null    object
    10   LotConfig       1460 non-null    object
    11   LandSlope       1460 non-null    object
```

```
12  Neighborhood    1460 non-null    object
13  Condition1      1460 non-null    object
14  Condition2      1460 non-null    object
15  BldgType        1460 non-null    object
16  HouseStyle      1460 non-null    object
17  OverallQual     1460 non-null    int64
18  OverallCond     1460 non-null    int64
19  YearBuilt       1460 non-null    int64
20  YearRemodAdd    1460 non-null    int64
21  RoofStyle       1460 non-null    object
22  RoofMatl        1460 non-null    object
23  Exterior1st     1460 non-null    object
24  Exterior2nd     1460 non-null    object
25  MasVnrType      588 non-null     object
26  MasVnrArea      1452 non-null    float64
27  ExterQual       1460 non-null    object
28  ExterCond       1460 non-null    object
29  Foundation      1460 non-null    object
30  BsmtQual        1423 non-null    object
31  BsmtCond        1423 non-null    object
32  BsmtExposure    1422 non-null    object
33  BsmtFinType1    1423 non-null    object
34  BsmtFinSF1      1460 non-null    int64
35  BsmtFinType2    1422 non-null    object
36  BsmtFinSF2      1460 non-null    int64
37  BsmtUnfSF       1460 non-null    int64
38  TotalBsmtSF     1460 non-null    int64
39  Heating         1460 non-null    object
40  HeatingQC       1460 non-null    object
41  CentralAir      1460 non-null    object
42  Electrical      1459 non-null    object
43  1stFlrSF        1460 non-null    int64
44  2ndFlrSF        1460 non-null    int64
45  LowQualFinSF    1460 non-null    int64
46  GrLivArea       1460 non-null    int64
47  BsmtFullBath    1460 non-null    int64
48  BsmtHalfBath    1460 non-null    int64
49  FullBath        1460 non-null    int64
50  HalfBath        1460 non-null    int64
51  BedroomAbvGr    1460 non-null    int64
52  KitchenAbvGr    1460 non-null    int64
53  KitchenQual     1460 non-null    object
54  TotRmsAbvGrd    1460 non-null    int64
```

```
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

3.2.1 Preliminary operation on data preparation

In order to achieve **better results looking at data and graphs**, we decided to do some data transformation and cleaning before the data processing part. This choice brings an easier and more intuitive data visualization and analysis for us, in order to get a better insight of the data.

The action carried out in this part are the following:

1. Trasnforming **NA** values into **no_value** for the categorical features written in the precedent section.
2. Setting GarageYrBlt, MasVnrArea, LotFrontage null values to **0**
3. Changing all the columns with attribute **object** to **category** type
4. Changed MasVnrArea and LotFrontage to **int64** type
5. Dropped one row with Electrical null value
6. Others changes during data visualization part (Will be reported in the data processing section)

3.2.2 Data Exploration - More in depth

	count	mean	std	min	25%	50%	75%	max
Id	1460.0	730.500000	421.610009	1.0	365.75	730.5	1095.25	1460.0
MSSubClass	1460.0	56.897260	42.300571	20.0	20.00	50.0	70.00	190.0
LotFrontage	1460.0	57.623288	34.664304	0.0	42.00	63.0	79.00	313.0
LotArea	1460.0	10516.828082	9981.264932	1300.0	7553.50	9478.5	11601.50	215245.0
OverallQual	1460.0	6.099315	1.382997	1.0	5.00	6.0	7.00	10.0
OverallCond	1460.0	5.575342	1.112799	1.0	5.00	5.0	6.00	9.0
YearBuilt	1460.0	1971.267808	30.202904	1872.0	1954.00	1973.0	2000.00	2010.0
YearRemodAdd	1460.0	1984.865753	20.645407	1950.0	1967.00	1994.0	2004.00	2010.0
MasVnrArea	1460.0	103.117123	180.731373	0.0	0.00	0.0	164.25	1600.0
BsmtFinSF1	1460.0	443.639726	456.098091	0.0	0.00	383.5	712.25	5644.0
BsmtFinSF2	1460.0	46.549315	161.319273	0.0	0.00	0.0	0.00	1474.0
BsmtUnfSF	1460.0	567.240411	441.866955	0.0	223.00	477.5	808.00	2336.0
TotalBsmtSF	1460.0	1057.429452	438.705324	0.0	795.75	991.5	1298.25	6110.0
1stFlrSF	1460.0	1162.626712	386.587738	334.0	882.00	1087.0	1391.25	4692.0
2ndFlrSF	1460.0	346.992466	436.528436	0.0	0.00	0.0	728.00	2065.0
LowQualFinSF	1460.0	5.844521	48.623081	0.0	0.00	0.0	0.00	572.0
GrLivArea	1460.0	1515.463699	525.480383	334.0	1129.50	1464.0	1776.75	5642.0
BsmtFullBath	1460.0	0.425342	0.518911	0.0	0.00	0.0	1.00	3.0
BsmtHalfBath	1460.0	0.057534	0.238753	0.0	0.00	0.0	0.00	2.0
FullBath	1460.0	1.565068	0.550916	0.0	1.00	2.0	2.00	3.0
HalfBath	1460.0	0.382877	0.502885	0.0	0.00	0.0	1.00	2.0
BedroomAbvGr	1460.0	2.866438	0.815778	0.0	2.00	3.0	3.00	8.0
KitchenAbvGr	1460.0	1.046575	0.220338	0.0	1.00	1.0	1.00	3.0

Figure 3.2: Numerical info 1

TotRmsAbvGrd	1460.0	6.517808	1.625393	2.0	5.00	6.0	7.00	14.0
Fireplaces	1460.0	0.613014	0.644666	0.0	0.00	1.0	1.00	3.0
GarageYrBlt	1460.0	1868.739726	453.697295	0.0	1958.00	1977.0	2001.00	2010.0
GarageCars	1460.0	1.767123	0.747315	0.0	1.00	2.0	2.00	4.0
GarageArea	1460.0	472.980137	213.804841	0.0	334.50	480.0	576.00	1418.0
WoodDeckSF	1460.0	94.244521	125.338794	0.0	0.00	0.0	168.00	857.0
OpenPorchSF	1460.0	46.660274	66.256028	0.0	0.00	25.0	68.00	547.0
EnclosedPorch	1460.0	21.954110	61.119149	0.0	0.00	0.0	0.00	552.0
3SsnPorch	1460.0	3.409589	29.317331	0.0	0.00	0.0	0.00	508.0
ScreenPorch	1460.0	15.060959	55.757415	0.0	0.00	0.0	0.00	480.0
PoolArea	1460.0	2.758904	40.177307	0.0	0.00	0.0	0.00	738.0
MiscVal	1460.0	43.489041	496.123024	0.0	0.00	0.0	0.00	15500.0
MoSold	1460.0	6.321918	2.703626	1.0	5.00	6.0	8.00	12.0
YrSold	1460.0	2007.815753	1.328095	2006.0	2007.00	2008.0	2009.00	2010.0
SalePrice	1460.0	180921.195890	79442.502883	34900.0	129975.00	163000.0	214000.00	755000.0

Figure 3.3: Numerical info 2

	count	unique	top	freq
MSZoning	1460	5	RL	1151
Street	1460	2	Pave	1454
Alley	1460	3	no_value	1369
LotShape	1460	4	Reg	925
LandContour	1460	4	Lvl	1311
Utilities	1460	2	AllPub	1459
LotConfig	1460	5	Inside	1052
LandSlope	1460	3	Gtl	1382
Neighborhood	1460	25	NAmes	225
Condition1	1460	9	Norm	1260
Condition2	1460	8	Norm	1445
BldgType	1460	5	1Fam	1220
HouseStyle	1460	8	1Story	726
RoofStyle	1460	6	Gable	1141
RoofMatl	1460	8	CompShg	1434
Exterior1st	1460	15	VinylSd	515
Exterior2nd	1460	16	VinylSd	504
MasVnrType	1460	4	no_value	872
ExterQual	1460	4	TA	906
ExterCond	1460	5	TA	1282
Foundation	1460	6	PConc	647
BsmtQual	1460	5	TA	649

Figure 3.4: Categorical info 1

BsmtCond	1460	5	TA	1311
BsmtExposure	1460	5	No	953
BsmtFinType1	1460	7	Unf	430
BsmtFinType2	1460	7	Unf	1256
Heating	1460	6	GasA	1428
HeatingQC	1460	5	Ex	741
CentralAir	1460	2	Y	1365
Electrical	1459	5	SBrkr	1334
KitchenQual	1460	4	TA	735
Functional	1460	7	Typ	1360
FireplaceQu	1460	6	no_value	690
GarageType	1460	7	Attchd	870
GarageFinish	1460	4	Unf	605
GarageQual	1460	6	TA	1311
GarageCond	1460	6	TA	1326
PavedDrive	1460	3	Y	1340
PoolQC	1460	4	no_value	1453
Fence	1460	5	no_value	1179
MiscFeature	1460	5	no_value	1406
SaleType	1460	9	WD	1267
SaleCondition	1460	6	Normal	1198

Figure 3.5: Categorical info 2

3.2.3 Plotting

Target variable

As it can be seen in the graph, the target variable is strongly unbalanced, and in particular the label "HIGH" has very few data compared to the other two.
This will be taken into account during the modeling phase, considering also some oversampling techniques.

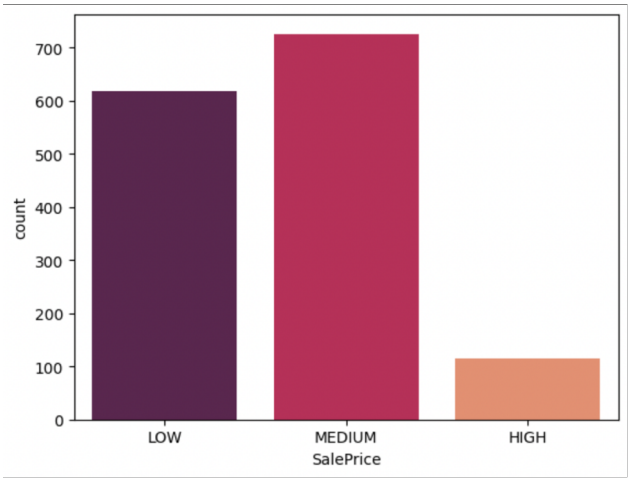


Figure 3.6

Features

After dropping some attributes for better visualization, we proceeded to analyze the distribution of the numerical attributes. This analysis provided us with valuable insights, which aided us in making decisions regarding the removal or transformation of certain numerical attributes into categorical ones.
By examining the distribution of numerical attributes, we gained a deeper understanding of their characteristics. These insights allowed us to determine which attributes could be removed without significant loss of information and which attributes could be better represented as categorical variables.

During the analysis of the numerical attributes, we discovered that about eleven of these attributes exhibited a right-skewed distribution; some just even had one value (LowQualityFinSF). Upon identifying this right-skewed behavior, we evaluated the relevance of these attributes to our analysis objectives.

Among the numerical attributes, we observed that certain attributes exhibited interesting distributions. In particular, we noticed that the data were distributed almost evenly across the entire range, this attributes will help us design a good model.

We make reference to the notebook to get more informations about the cited variables and graphs.

Chapter 4

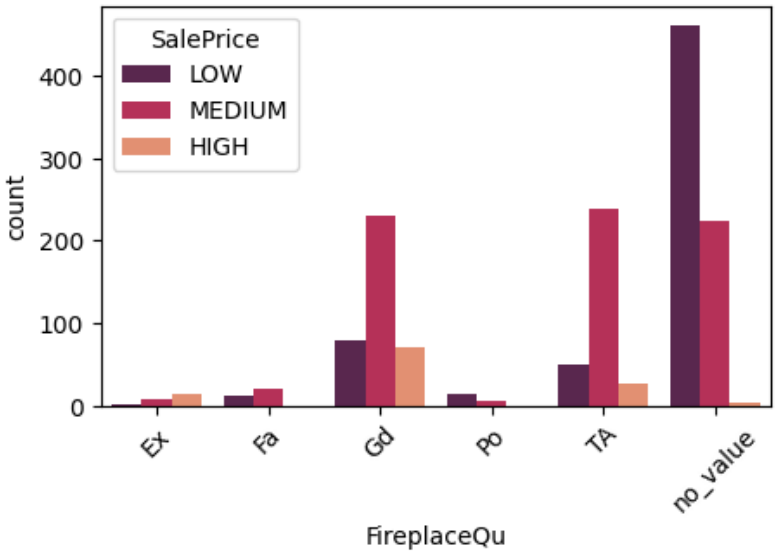
Data preparation

4.1 Data pruning

As previosly seen, many of the features were not encoding patterns, evenly distributed and significant into the specific domain, so we dropped them. This reduced the total amount of features, from the 79 starting ones, to 43.
An interesting consideration comes out of the FirePlaceQ attribute, which in the data description corresponds to:

```
FireplaceQu: Fireplace quality

Ex Excellent - Exceptional Masonry Fireplace
Gd Good - Masonry Fireplace in main level
TA Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement
Fa Fair - Prefabricated Fireplace in basement
Po Poor - Ben Franklin Stove
NA No Fireplace
```



This might seem an interesting attribute to take into account. However, looking at the distribution based on target category, we can note that each one of the labels different from the one encoding "No Fireplace" have a similar distribution, which does not seems to be strictly related to the category itself. In practise, the overall quality of the fireplace might not be directly related to the final price, but instead the presence of a fireplace definitely is important into the evaluation process. Since that information is already expressed by the *Fireplaces* attribute, we can drop *FireplaceQu*.

4.2 Conversion,binarization, encoding

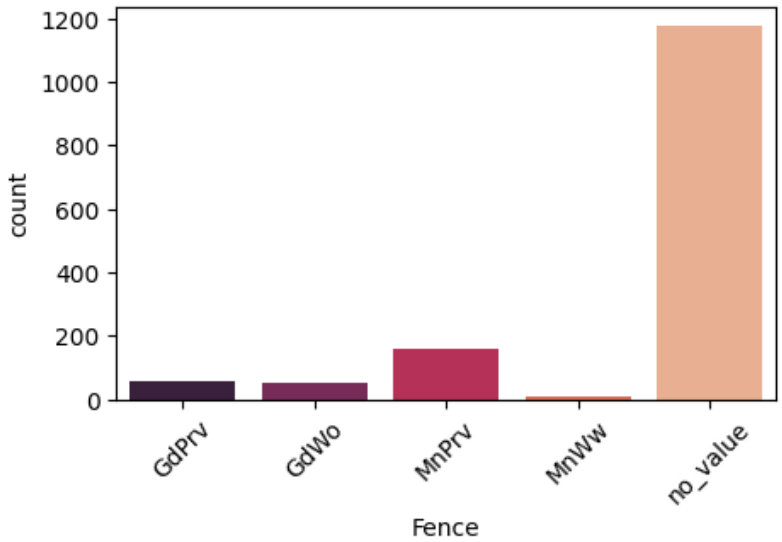
Some variables, even if scaled on numbers, were encoding a categorical information. Specifically, the following have been converted in categorical object type:

- Fireplaces
- FullBath
- HalfBath
- MSSubClass
- YrSold

We decided to binarize some features that are unbalanced, but relevant in the context:

- Fence
- LotConfig
- LotShape

To better understand the reasons that lead us to that, let's consider the Fence attribute.



The majority of the houses have no fence at all. Those who have them are split into different fence types, which generate as result that classes are very unbalanced, but conceptually divisible in two macro-categories *Has_Fence/No_Fence*. Also, from business knowledge, we can say that probably the specific type of fence does not reasonably infer a lot into the final sale price,even less not on an integer but on a price category. We then decided to keep the feature, but grouping the values in the two previously quoted categories.

Finally, we applied the one-hot-encoding on the remaining categorical variables, creating a dummy binary value for every category on each variable in the dataset.

The final shape of our dataset, after all the data cleaning process, is

```
df.shape
(1460, 44)
```

At the end of this phase we decided to save, just for practical reasons, the edited dataframe in a new .parquet file, a specific type designed to store and load data in a space and time efficient way.

4.3 Data normalization

In our project, despite thorough exploration, data normalization did not lead to significant improvements in model performance. As a result, we opted not to include normalization in our workflow and focused on alternative preprocessing techniques that showed more promising results. So, we decided to skip this step, and proceed with the model building.

Chapter 5

Modeling and model evaluation

5.1 Preliminary operations

First steps in this part of the modeling was to split our dataset in a train set, used to train every model, and a test set, to compute the evaluation metrics of them.
Then, to avoid bias due to the unbalanced target classes, we did an oversampling over the sets using the SMOTE technique. Finally, we have built many models, with an iterative approach over each one to find a proper parameters setting that could achieve good results.

5.1.1 Decision Tree Classifier

A Decision Tree is a supervised learning model that employs a hierarchical structure of nodes to make decisions based on input features. Each internal node represents a feature, and the branches represent possible outcomes based on the feature's value. Decision Trees excel in capturing complex relationships and are easy to interpret. However, they are prone to overfitting and can struggle with handling continuous data or outliers.

5.1.2 Random Forest

The Random Forest algorithm is an ensemble learning method that combines multiple decision trees to make predictions. It generates a multitude of decision trees and aggregates their outputs to produce a final prediction. Random Forests are known for their robustness, versatility, and resistance to overfitting. They are particularly effective for handling high-dimensional data and are widely used for classification and regression tasks

5.1.3 Gradient Boosting Classifier

Gradient Boosting is an ensemble learning technique that combines multiple weak learners, typically decision trees, to create a strong predictive model. It works by iteratively training new models that focus on the errors made by the previous models, gradually reducing the overall error. Gradient Boosting is highly effective in handling tabular data and has achieved remarkable success in various machine learning competitions.

5.1.4 AdaBoost Classifier

Adaboost, short for Adaptive Boosting, is another ensemble learning method that sequentially combines weak learners to create a strong predictive model. It assigns higher weights to misclassified instances in each iteration, enabling subsequent models to focus on those instances. Adaboost is versatile, relatively easy to implement, and can be used for both classification and regression tasks. However, it is sensitive to noisy data and outliers.

5.1.5 Gaussian Naive Bayes Classifier

Gaussian Naive Bayes is a probabilistic classifier that applies Bayes' theorem with the assumption of independence between features. It assumes that the features follow a Gaussian distribution and calculates the probability of each class given the observed feature values. Despite its simplicity, Gaussian Naive Bayes can perform well in many real-world applications, especially when the independence assumption holds.

5.1.6 K-Neighborhood classifier

K-Nearest Neighbors is a non-parametric algorithm used for both classification and regression tasks. It classifies new instances by finding the closest labeled instances in the feature space and assigning the majority class or predicting the average value of their targets. KNN is easy to understand and implement, but it can be computationally expensive, especially with large datasets. It is also sensitive to the choice of the number of neighbors (K) and the distance metric.

5.1.7 Support Vector Classifier

Support Vector Machine is a powerful supervised learning algorithm that separates data points into different classes by constructing hyperplanes in a high-dimensional feature space. It aims to find an optimal hyperplane that maximizes the margin between classes while minimizing classification errors. SVMs can handle both linear and non-linear data through the use of kernel functions. They are effective in high-dimensional spaces and have good generalization capabilities. However, they can be sensitive to the choice of hyperparameters and require careful preprocessing of data.

5.2 Train and test splits and oversampling

The dataset used in this project was divided into a 70% training set and a 30% testing set. This split ensured that a substantial portion of the data was dedicated to training the machine learning models, while also providing an independent subset for evaluating their performance.

Furthermore, it was observed that the dataset exhibited class imbalance, where the minority class had significantly fewer instances compared to the majority class, **HIGH** class have very few instances compared to **LOW and MEDIUM**. To address this issue and ensure fair representation of all classes during model training, oversampling techniques were applied.

Specifically, oversampling was performed for the minority class using the BorderlineSMOTE method.

5.2.1 BorderlineSMOTE

BorderlineSMOTE generates synthetic samples for the minority class, effectively increasing its representation in the dataset. By incorporating these synthetic samples, the training data became more balanced, allowing the models to learn from both the majority and minority classes more effectively.

By splitting the data into training and testing sets and applying oversampling techniques, we aimed to improve the robustness and performance of the machine learning models, particularly in handling class imbalance challenges. The resulting datasets were then used to train and evaluate the models, providing a comprehensive assessment of their effectiveness in addressing the problem at hand.

5.3 Parameter tuning

We wanted to see how different machine learning models perform without changing their default settings. We thought about tweaking the parameters to potentially improve the models, but we hit a roadblock. The problem was that our resources, like computing power and time, weren't enough to handle the long runtimes that come with parameter tuning.

Even though we couldn't fine-tune the models as much as we wanted, we still went ahead and evaluated them using their default configurations.

In our quest for finding the optimal configuration for the Random Forest model, we enlisted the help of sklearn's random search cross-validation (CV) technique. This approach allowed us to explore different combinations of parameters in a randomized manner, hoping to strike gold with a winning combination.

However, despite our efforts, the results didn't quite meet our expectations. The random search CV didn't yield a significant number of promising outcomes that would greatly enhance the model's performance on our specific dataset.

5.4 Model Evaluation

	accuracy	precision	recall	f1-score	confusion_matrix
RF	0.863014	0.864321	0.863014	0.861503	[[28, 0, 14], [0, 163, 15], [3, 28, 187]]
GB	0.860731	0.860755	0.860731	0.860276	[[32, 0, 10], [0, 160, 18], [6, 27, 185]]
AB	0.787671	0.791997	0.787671	0.786487	[[29, 0, 13], [0, 155, 23], [6, 51, 161]]
NB	0.639269	0.733426	0.639269	0.655490	[[37, 0, 5], [2, 144, 32], [93, 26, 99]]
KNN	0.792237	0.804682	0.792237	0.795434	[[35, 0, 7], [3, 142, 33], [24, 24, 170]]
DT	0.801370	0.802925	0.801370	0.801781	[[28, 0, 14], [1, 144, 33], [14, 25, 179]]
SVC	0.776256	0.779255	0.776256	0.777318	[[30, 2, 10], [1, 141, 36], [18, 31, 169]]

Figure 5.1: Models initial results

The Random Forest and Gradient Boosting models showed similar and consistent results across accuracy, precision, recall, and F1-score. These models demonstrated balanced and robust performance with relatively few misclassifications, as evident from the confusion matrices.

On the other hand, the Adaboost model had slightly lower values for accuracy, precision, recall, and F1-score compared to the Random Forest and Gradient Boosting models. It faced challenges in correctly classifying instances, particularly in the minority class.

The Gaussian Naive Bayes model achieved the lowest accuracy among all models, indicating a lower overall correctness in its predictions. Although it had relatively high precision for positive instances, it struggled with recall, resulting in a higher number of false negatives. The confusion matrix revealed significant misclassifications, particularly in the minority class.

The K-nearest Neighbors and Decision Tree models exhibited similar performance with reasonably good accuracy, precision, recall, and F1-score. The confusion matrices indicated some misclassifications across the classes, but overall, these models performed adequately.

In contrast, the Support Vector Machine model had lower accuracy compared to most other models. The precision, recall, and F1-score aligned with the accuracy, suggesting a moderate performance. The confusion matrix highlighted misclassifications, especially with false positives in the majority class.

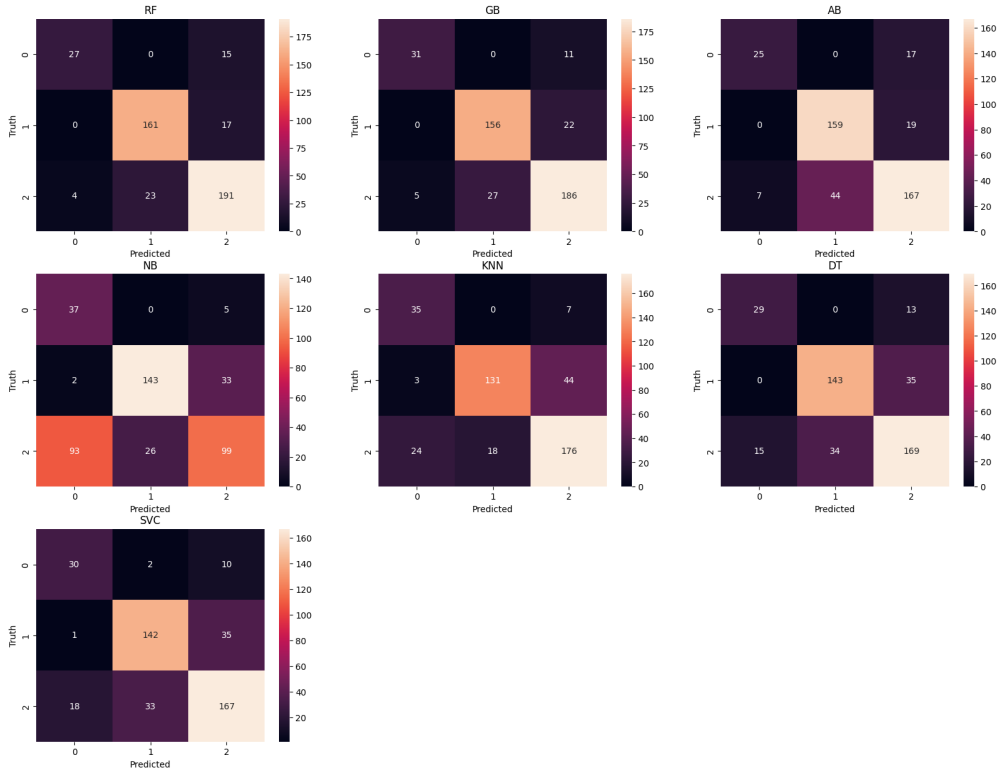


Figure 5.2: Confusion Matrix

5.4.1 K-fold cross-validation

KFold is a technique used in machine learning for cross-validation. It involves splitting the dataset into k equal-sized folds or subsets. The model is then trained and evaluated k times, each time using a different fold as the validation set and the remaining folds as the training set. This ensures that each data point is used for both training and validation, providing a more reliable estimation of the model's performance.

cross_val_score is a function provided by the scikit-learn library that automates the process of performing

KFold cross-validation and computing evaluation metrics. It simplifies the process by allowing you to specify the number of folds (k) and the evaluation metric of interest, such as accuracy or mean squared error. The function takes care of splitting the data, training the model, and evaluating its performance on each fold. It returns an array of scores, representing the performance metric for each fold.

In our project, we utilized KFold cross-validation along with cross_val_score to assess the performance of our machine learning models. By employing this technique, we were able to obtain a more robust evaluation of the models, taking into account variations in performance across different subsets of the data. This helped us gain a better understanding of how well our models generalize to unseen data and allowed us to make more informed decisions about their effectiveness.

	RF	GB	AB	NB	KNN	DT
0	0.924658	0.931507	0.808219	0.773973	0.835616	0.876712
1	0.900685	0.914384	0.815068	0.732877	0.828767	0.849315
2	0.924399	0.914089	0.725086	0.793814	0.828179	0.855670
3	0.920962	0.917526	0.835052	0.804124	0.852234	0.855670
4	0.914089	0.910653	0.835052	0.721649	0.817869	0.841924

Figure 5.3: K-fold cross-validation

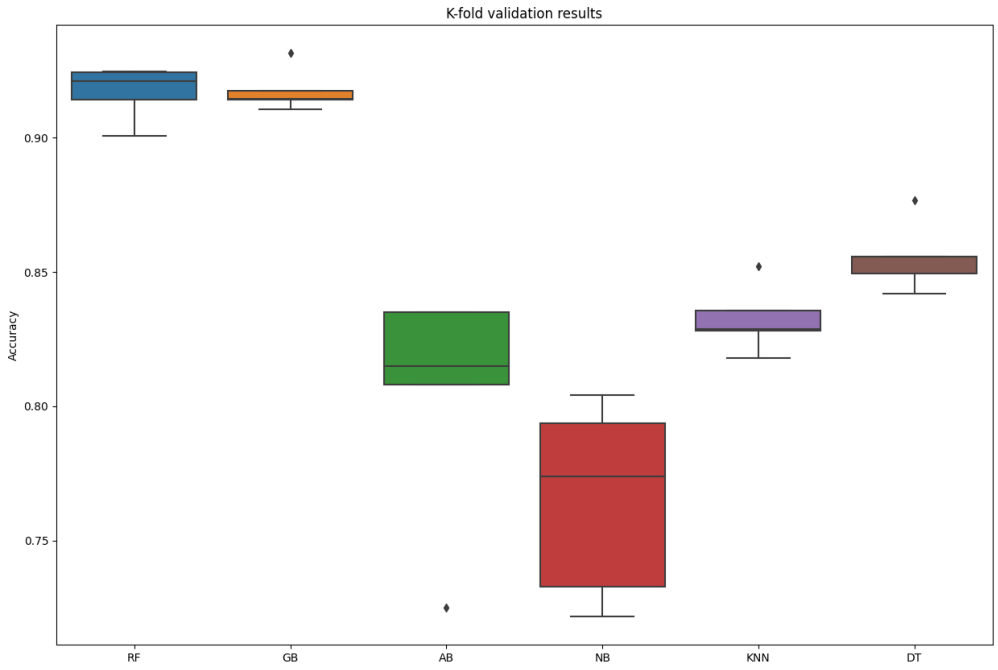


Figure 5.4: K-fold box plot

Analyzing the scores, we can observe the following:

- Random Forest (RF) and Gradient Boosting (GB) consistently achieved high scores across all folds, indicating strong performance and good generalization ability. These models consistently outperformed the other models in terms of accuracy.
- Adaboost (AB) had relatively lower scores compared to RF and GB, but it still displayed decent performance across most folds. It showed some variability in performance, with scores ranging from moderate to good.
- Gaussian Naive Bayes (NB) had mixed performance, with scores varying across folds. It achieved relatively lower scores compared to RF, GB, and AB, indicating that it struggled to capture the complexities of the dataset.
- K-nearest Neighbors (KNN) and Decision Tree (DT) models had scores that varied across folds but generally performed at a moderate level. They were outperformed by RF and GB but showed comparable performance to AB and NB.

We can see that Random Forest and Gradient Boosting have the most consistent performance across all folds, with scores ranging from good to excellent. They also achieved the highest accuracy among all models, indicating that they were able to correctly classify the majority of instances in the dataset. These models demonstrated strong generalization ability, suggesting that they would perform well on unseen data.

5.5 Feature Selection — SelectKBest

SelectKBest is a feature selection algorithm that selects the best features from a dataset based on univariate statistical tests. It can be used for classification or regression problems. The algorithm works by scoring each feature based on a statistical test, such as the chi-squared. It then selects the top k features with the highest scores.

We tried to prune the model selecting the best features, but we didn't get any significant results. The model performed better with all the features we selected in the first place.

We left the code in the notebook for reference.

5.6 ROC Curve

The ROC curve is like a graph that shows how well a machine learning system can tell things apart. It plots the true positive rate (how well it detects the right things) against the false positive rate (how often it makes mistakes) as we change the threshold for making decisions. The curve shows how the sensitivity (ability to detect) changes as the fall-out (making mistakes) varies.

Basically, if we know the probabilities of detecting something and making a mistake, we can create the ROC curve by looking at the cumulative probabilities. We plot the detection probability on the y-axis and the false-alarm probability on the x-axis. It helps us see how well a classifier can distinguish between positive and negative cases. We can look at the shape of the curve and the area under it to evaluate the performance. A larger area under the curve means a better classifier that can tell things apart more accurately. So it is a handy way to measure how well a classifier can do its job. It lets us compare different models and see how good they are at separating the good from the bad.

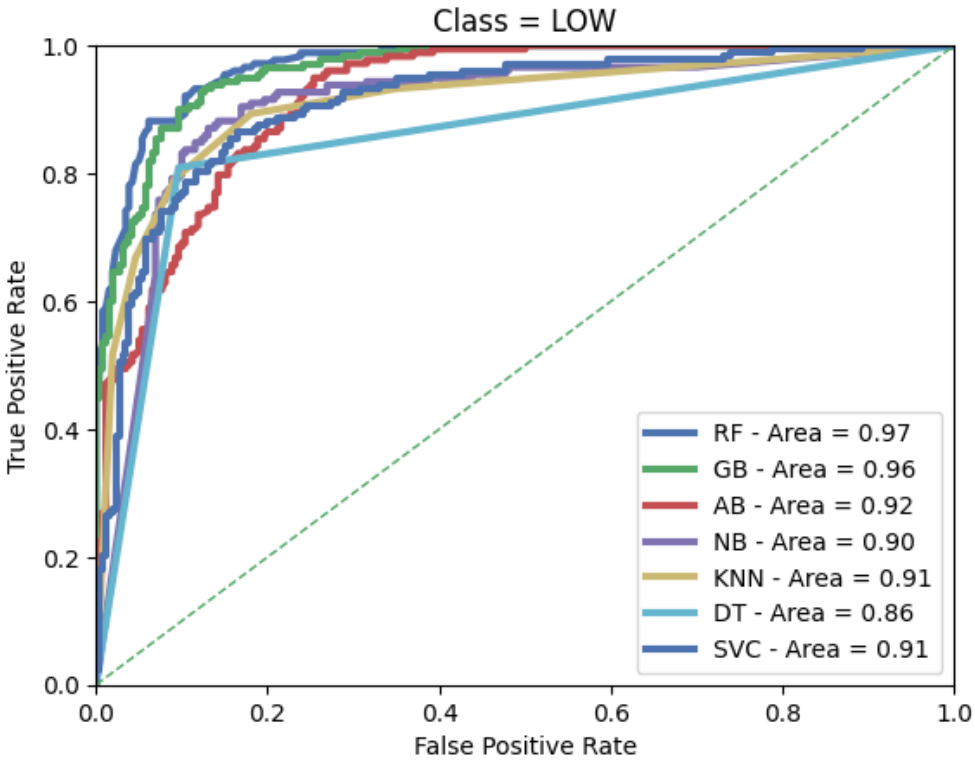


Figure 5.5: ROC Curve - LOW

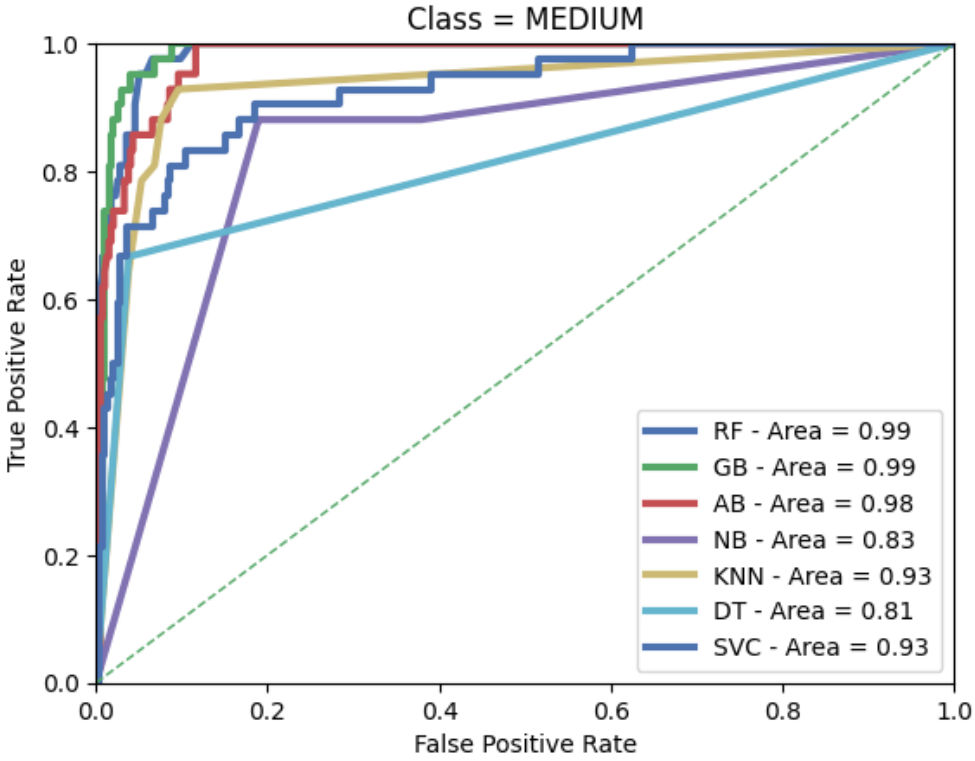


Figure 5.6: ROC Curve - Medium

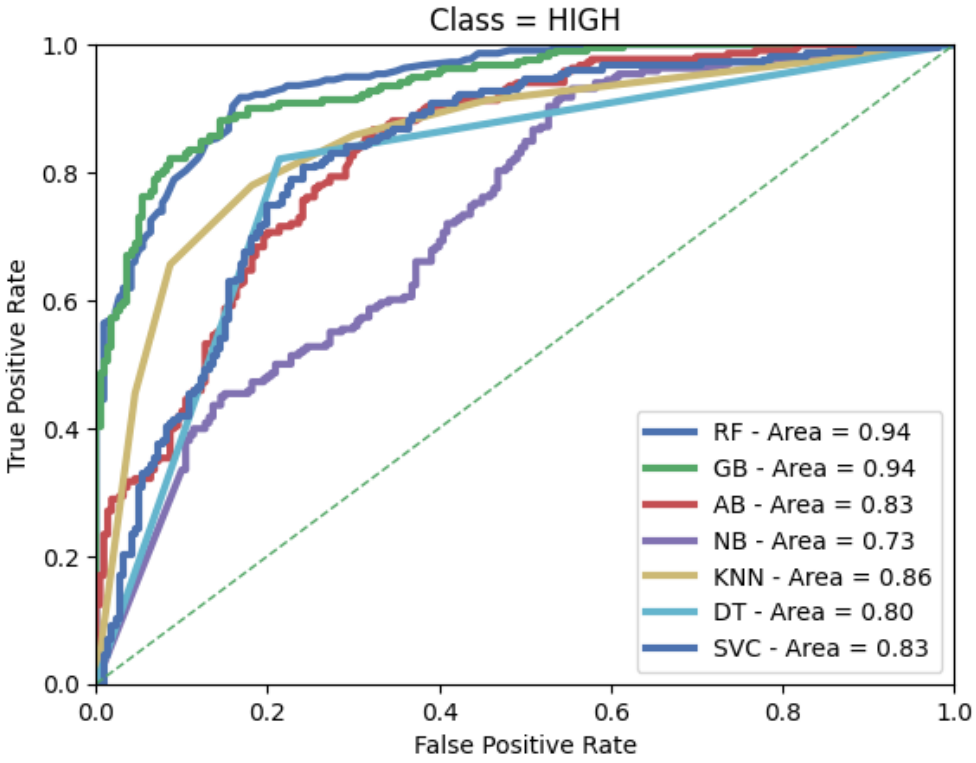


Figure 5.7: ROC Curve - HIGH

5.6.1 Performance

In the **medium category**, the Random Forest (RF), Gradient Boosting (GB) and AdaBoost (AB) models exhibit excellent performance with AUC scores of 0.99, 0.99 and 0.98, respectively. These models demonstrate strong discrimination between classes, making them reliable choices for accurate classification tasks.

In the low category, the Gradient Boosting (GB) and Random Forest (RF) models stand out with a promising AUC score of 0.97 and 0.96 each. These are suitable option for classification tasks where accuracy is crucial.

In the high category, both the Random Forest (RF) and Gradient Boosting (GB) models shine with AUC scores of 0.94. These models demonstrate robust predictive capabilities and strong discrimination between classes. Additionally, the AdaBoost (AB) model performs reasonably well with an AUC score of 0.83.

The Gaussian Naive Bayes (NB) model achieves relatively lower AUC scores across all categories, suggesting limitations in accurately distinguishing between classes. However, it still demonstrates moderate performance, especially in the medium and low categories.

Sp, in the end, the Random Forest (RF) and Gradient Boosting (GB) models consistently showcase top-tier performance across different performance categories, making them the best models among those evaluated.