IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Final Project

Master’s in Engenharia de Software

And

Master’s in Engenharia Eletrotécnica e Computadores

Professor: Vala Ali Rohani

Student: Jorge Luís Nr: 201300773

Student: João Ribeiro Nr: 201701827

Curricular Unit: Data Analytics

Year: 2022/2023

Índex

1 Introduction

This project was made for the curricular unit of Data Analytics.

The aim of this project is to predict the sale prices of the houses using the Machine Learning model.

We have use the SMART methodology, Specific to predict the house price, Measurable to improve the results up to approximately 90%, Attainable to consume the dataset files, Relevant to train the machine to more concretely predict the value of the houses and Timely to finish the project in 14 days.

To manage the information we have segmented in two variables, the dependent and the independent.

The dependent variable is the sales price and the independent variable LotFrontage, LotArea, OverallQual, OverallCond, YearBuilt, YearRemodAdd, BedroomAbvGr, KitchenAbvGr, TotRmsAbvGrd, GarageArea, PoolArea, MSZoning, Utilities, BldgType, ExterCond, Foundation, BsmtFinType1, Heating, HeatingQC, CentralAir, Electrical, KitchenQual, Functional, PoolQC.

We have chosen those independent variables to better predict the sales prices.

The outputs of this projects were obtained in the jupyter platform environment using python language.

It will be present a theory explanation, the code and the display of the jupyter information.

2 Configuration

To start the project in Jupyter we have import the libraries and the configurations, and than loaded the the CSV file with the dataset to start the machine learning model.

The CSV file dataset has an information of 1460 items and 81 characteristics to work.

2.1 Imports of the libraries

```
import pandas as pd

import warnings

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn import metrics

warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)
```

2.2 Load of the CSV file

```
dfRawData = pd.read_csv("HousePrices-ToBuildAndTestTheModel.csv")

dfRawData.shape

(1460, 81)
```

3 Data Cleaning and Exploratory Analysis

We have read and display the information in the dataset to get the information inside the file to analyze.

The dataset have shown null values which means that it has to be cleaned and filled with the information provided.

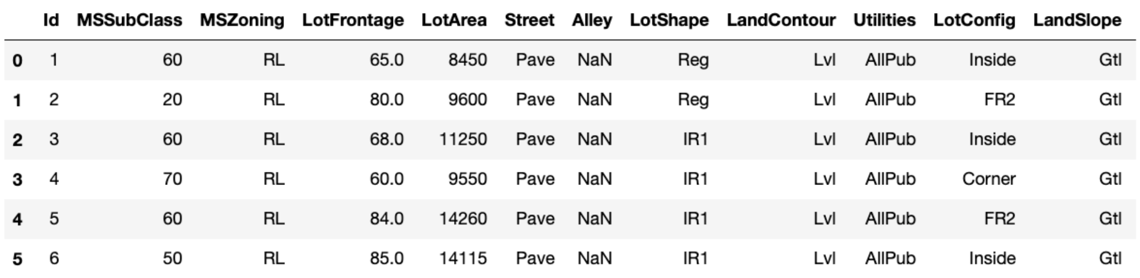
Confirming the null and fill the values it was necessary to check for duplicates to clean the information by removing the repeated ones.

3.1 Read of the Columns type

```
dfRawData.dtypes

Id              int64

MSSubClass        int64

MSZoning         object

LotFrontage      float64

…
```

3.2 Display of the samples

```
dfRawData.head(10)
```



	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl
5	6	50	RL	85.0	14115	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl

3.3 Display of the columns with null values

```
dfToClean = dfRawData

dfToClean.columns[dfToClean.isnull().any()]

Index(['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual',

     'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',

     'Electrical', 'FireplaceQu', 'GarageType', 'GarageYrBlt',

     'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence',

     'MiscFeature'],

    dtype='object'
```

3.4 Fill of the data

```
dfToClean["LotFrontage"].fillna(dfToClean['LotFrontage'].mean(), inplace = True)
```

```
dfToClean["Alley"].fillna("NA", inplace = True)

dfToClean["MasVnrType"].fillna("None", inplace = True)

dfToClean["MasVnrArea"].fillna(0, inplace = True)

…

dfToClean["MiscFeature"].fillna("NA", inplace = True)
```

3.5 Check for null values

```
dfToClean.columns[dfToClean.isnull().any()]

Index([], dtype='object')
```

3.6 Check for duplicates

```
dfToClean[dfToClean.duplicated()]
```

```
 Id    MSSubClass    MSZoning    LotFrontage    LotArea    Street    Alley    LotShape ...
```

4 Dataframes

We had to separate the dataframes in two dataframes, and than transform the object values into numeric values, after that, we've transformed the object values into string.

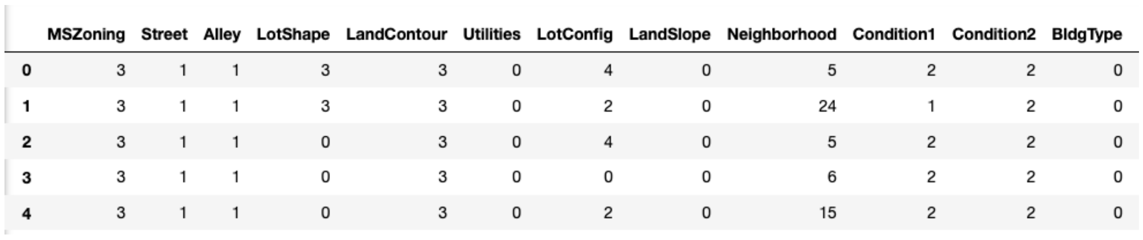
We have created a new dataframe by concatenating the new converted dataframe and the previous numeric.

4.1 Separation the dataframe into two dataframes

```
objList = dfToClean.select_dtypes(include=np.object).columns.tolist()

dfToCleanNum = dfToClean.drop(objList, axis = 1)

numList = dfToClean.select_dtypes(include=np.number).columns.tolist()

dfToCleanObj = dfToClean.drop(numList, axis = 1)
```

4.2 Turns all objects into strings and transforms the strings columns into numeric columns

```
dfToCleanObj = dfToCleanObj.astype(str)

dfToCleanObj = dfToCleanObj.apply(LabelEncoder().fit_transform)

dfToCleanObj
```



	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType
0	3	1	1	3	3	0	4	0	5	2	2	0
1	3	1	1	3	3	0	2	0	24	1	2	0
2	3	1	1	0	3	0	4	0	5	2	2	0
3	3	1	1	0	3	0	0	0	6	2	2	0
4	3	1	1	0	3	0	2	0	15	2	2	0

4.3 Creation of the dataframe

```
dfBasicCleaning = pd.concat([dfToCleanNum, dfToCleanObj], axis=1, join="outer")

dfToClean = dfBasicCleaning
```

4.4 Display of the shape

```
dfToClean.shape

(1460, 81)
```

5 Outliers

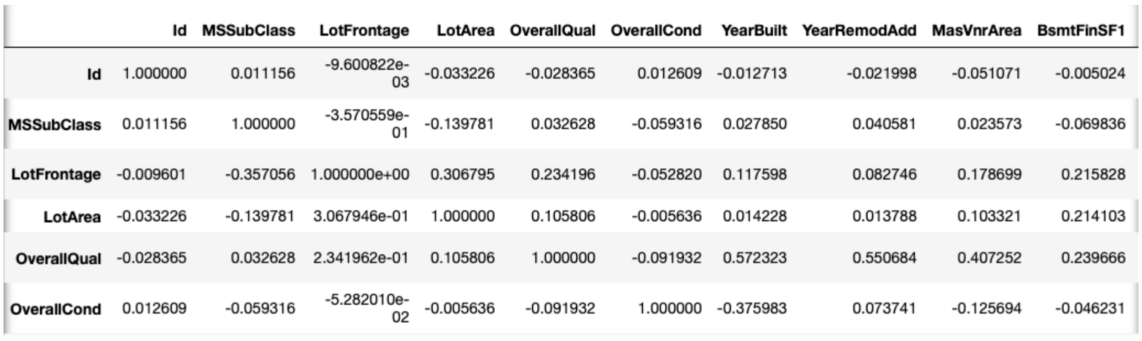
We had to find the outliers and removed the outliers and display the correlation and create a new dataframe.

5.1 Find the outliers and remove them

```
def find_outliers(df):

  q1=df.quantile(0.25)

  q3=df.quantile(0.75)

  IQR=q3-q1

  outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]

  return outliers

def remove_outliers(df, column):

   for col in list(column):

       outliers = find_outliers(df[col]).index

   for i in outliers:

       i += 1

       df = df.drop(df[(df['Id'] == i)].index)

   return df
```

5.2 Display of the correlation

dfToClean.corr()



	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1
Id	1.000000	0.011156	-9.600822e-03	-0.033226	-0.028365	0.012609	-0.012713	-0.021998	-0.051071	-0.005024
MSSubClass	0.011156	1.000000	-3.570559e-01	-0.139781	0.032628	-0.059316	0.027850	0.040581	0.023573	-0.069836
LotFrontage	-0.009601	-0.357056	1.000000e+00	0.306795	0.234196	-0.052820	0.117598	0.082746	0.178699	0.215828
LotArea	-0.033226	-0.139781	3.067946e-01	1.000000	0.105806	-0.005636	0.014228	0.013788	0.103321	0.214103
OverallQual	-0.028365	0.032628	2.341962e-01	0.105806	1.000000	-0.091932	0.572323	0.550684	0.407252	0.239666
OverallCond	0.012609	-0.059316	-5.282010e-02	-0.005636	-0.091932	1.000000	-0.375983	0.073741	-0.125694	-0.046231

5.3 Removal of the outliers

dfToClean = remove_outliers(dfToClean, ["OverallQual"])

dfToClean = remove_outliers(dfToClean, ["TotalBsmtSF"])

dfToClean = remove_outliers(dfToClean, ["1stFlrSF"])

dfToClean = remove_outliers(dfToClean, ["GrLivArea"])

dfToClean = remove_outliers(dfToClean, ["FullBath"])

dfToClean = remove_outliers(dfToClean, ["GarageCars"])

dfToClean = remove_outliers(dfToClean, ["GarageArea"])

5.4 Creation of the dataframe

dfCleaned = dfToClean

dfCleaned.shape

(1345, 81)

6 Display

At this stage we have display the data to check all the information using the graphical representation on the jupyter to understand the sale price and the correlation between the variables.

It was made analysis and test to discover the meaningful patterns in the data to understand sale price variations according to the representation. We can see that the maximum price is 451950.00, the minimum is 34900.00 and the standard is 64673.28.

We can see that the correlation between the sales rice and the density is 0.8.

In the heatmap, we spot the highest correlated values between the variables, to understand better which variables influences the most .

In the box plot the correlation between the sales price and the overall quality, we can understand the influence of the overall quality in the sales price.

In histograms we have some variable displayed to understand the influence they have

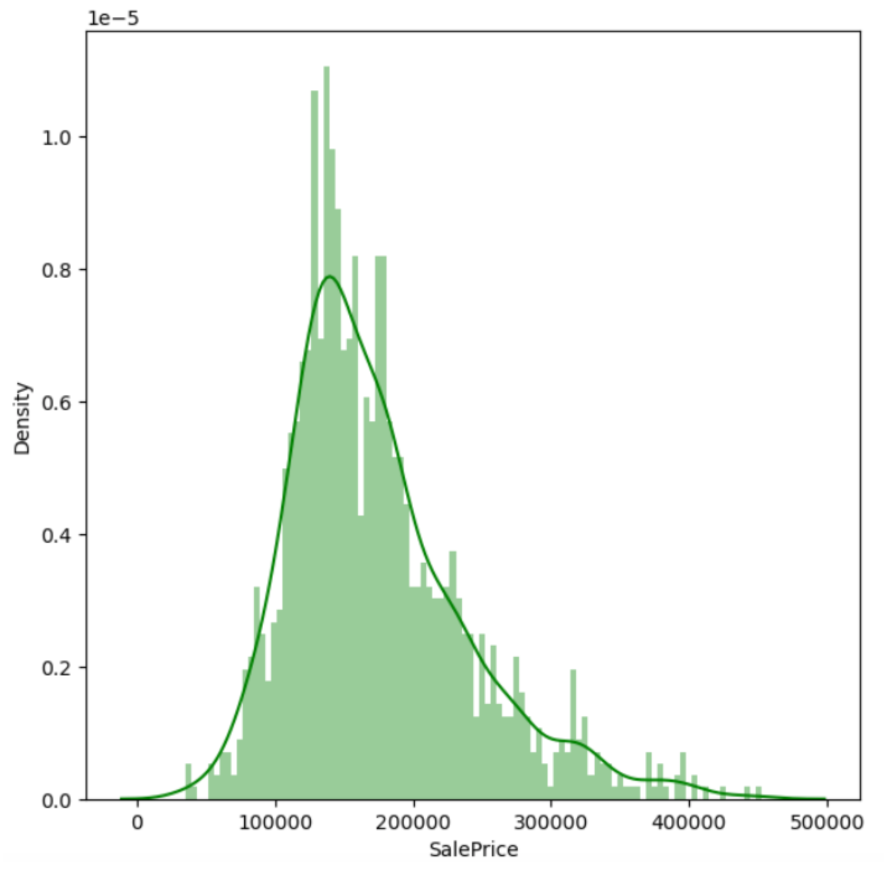
6.1 Display the SalePrice

```
dfCleaned['SalePrice'].describe()
```

```
count     1345.000000
mean    174217.515242
std      64673.287284
min      34900.000000
25%     130000.000000
50%     160000.000000
75%     205950.000000
max     451950.000000
Name: SalePrice, dtype: float64
```

6.2 Display the values of SalePrice

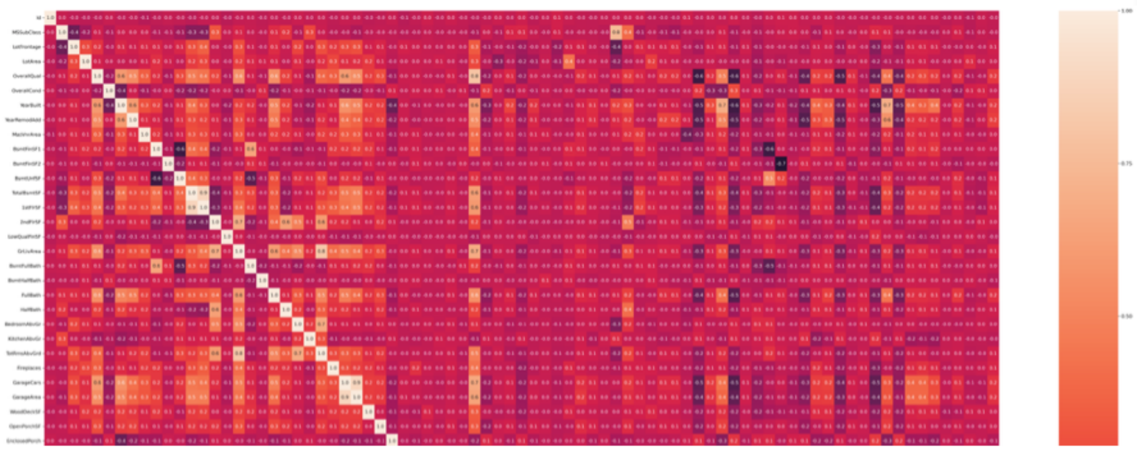
```
plt.figure(figsize=(7, 7))

sns.distplot(dfCleaned['SalePrice'], color='g', bins=100, hist_kws={'alpha': 0.4});
```



6.3 Display the heatmap of the correlation

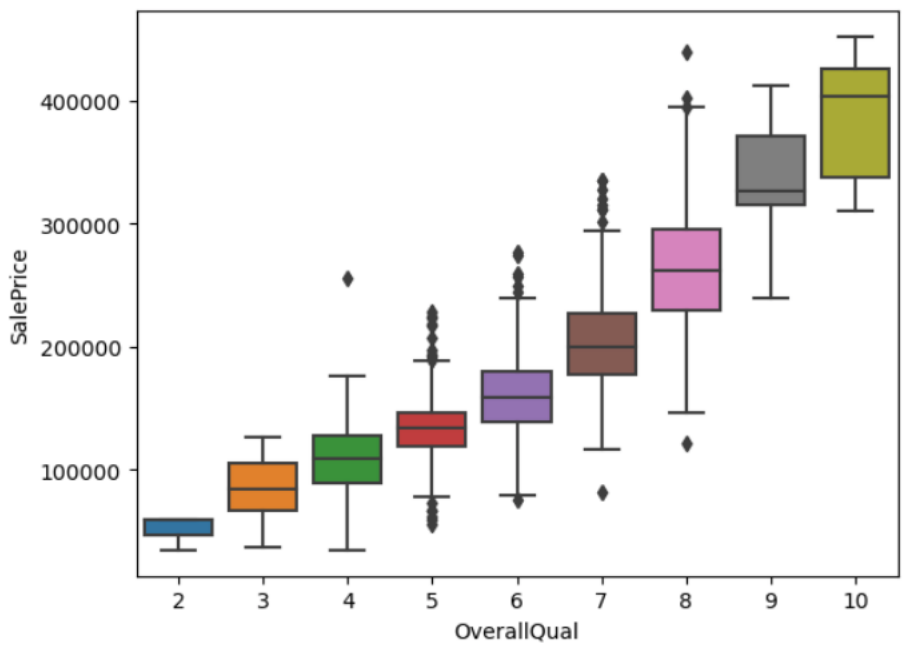
fig = plt.figure(figsize=(45,45), dpi = 300)

sns.heatmap(dfToClean.corr(), annot = True, fmt = '.1f')



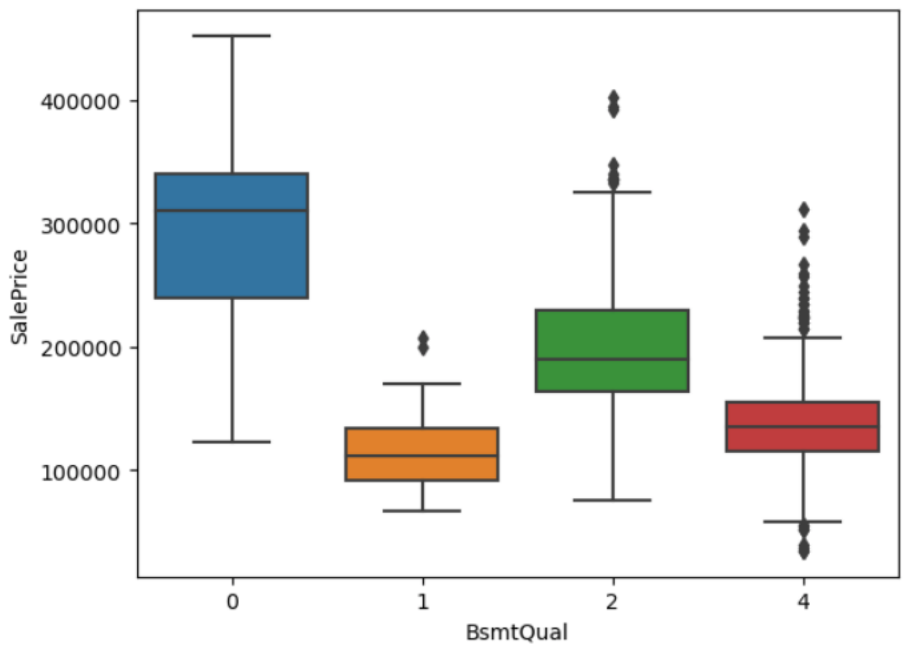
6.4 Display the relation between SalePrice and OverallQual

```
sns.boxplot(x='OverallQual', y='SalePrice', data=dfCleaned)

plt.show()
```



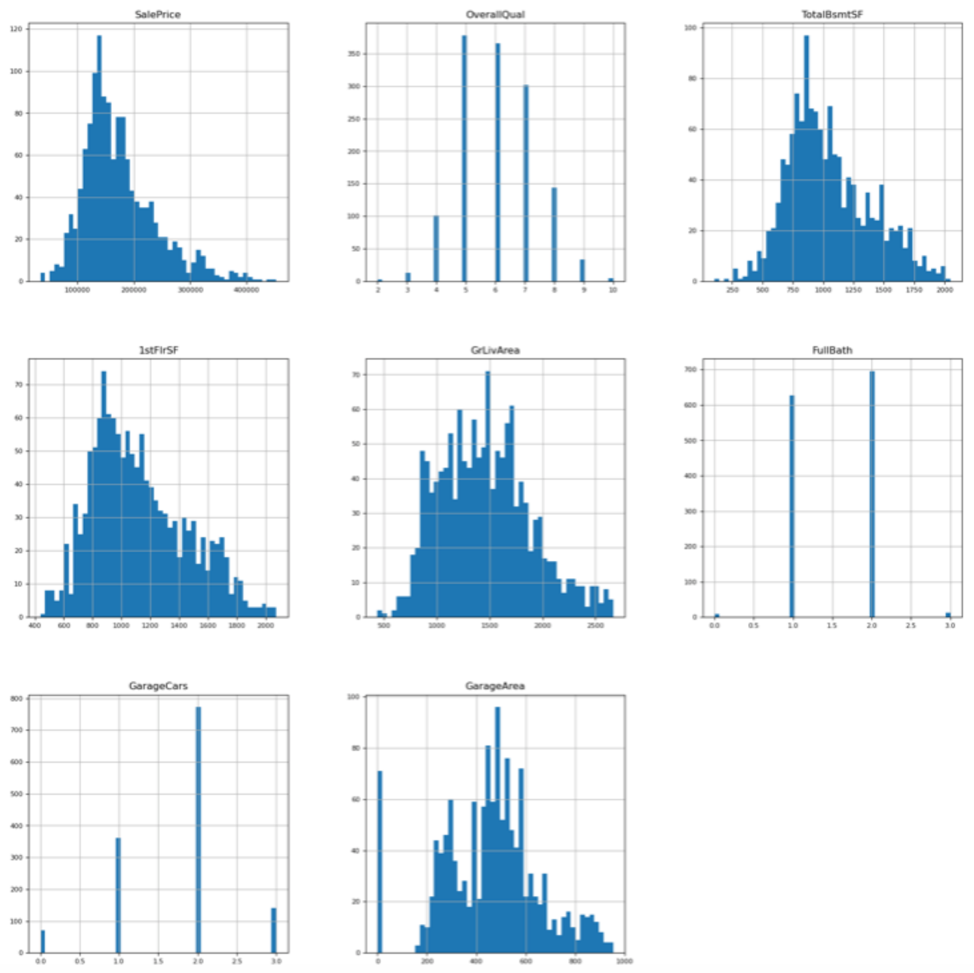
6.5 Display the relation between thw SalePrice and the BsmtQual

```
sns.boxplot(x='BsmtQual', y='SalePrice', data=dfCleaned)

plt.show()
```



6.6 Display the histograms

```
dfCleanedNum = dfCleaned[["SalePrice", "OverallQual", "TotalBsmtSF", "1stFlrSF", "GrLivArea", "FullBath","GarageCars","GarageArea"]]

dfCleanedNum.hist(figsize=(20, 20), bins=50, xlabelsize=8, ylabelsize=8);
```



7 Variables

We have to separate the dependent variable from the independent to predict the sales price, and create the regression model to provide a function that describes the relationship between the independent variables.

7.1 Separate the Dependent variable from the Independent variables

```
y = dfCleaned.SalePrice

col = dfCleaned.columns.tolist()

col.remove('SalePrice')

col.remove('Id')
col.remove('LotShape')

col.remove('LandContour')

col.remove('LotConfig')

col.remove('LandSlope')

col.remove('Neighborhood')

col.remove('Condition1')

…

x = dfCleaned[col]
```

7.2 Display the Independent variables

```
print(x.columns)

Index(['LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
'YearRemodAdd', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'GarageArea',
'PoolArea', 'MSZoning', 'Utilities', 'BldgType', 'BsmtQual', 'BsmtFinType1', 'Heating',
'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'PoolQC'],
 dtype='object')
```

7.3 Separate the train and test data

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

7.4 Regression model

```
finalModel = LinearRegression()

finalModel.fit(x_train, y_train)

LinearRegression()
```

8 Prediction

We had to create the regression model to create a better prediction, as we can confirm is the display of the graphic.

The table in below also display relation of the dataframe between the actual and the predicted value.

For a better correlation of the values we had to remove the outliers and the items with least correlation.

8.1 Create predictions

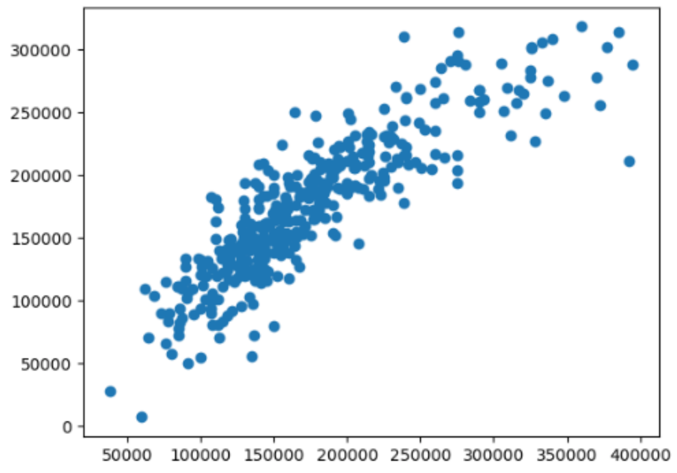
predictions = finalModel.predict(x_test)

8.2 Display the predictions

dfCompare = pd.DataFrame({'Actual': y_test, 'Predicted':predictions})

plt.scatter(y_test, predictions)

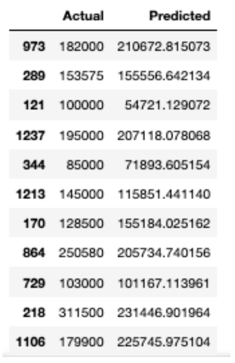
```
<matplotlib.collections.PathCollection at 0x7fc0955f3280>
```



8.3 Create the predictions for the test data

dfCompare = pd.DataFrame({'Actual': y_test, 'Predicted':predictions})

dfCompare



	Actual	Predicted
973	182000	210672.815073
289	153575	155556.642134
121	100000	54721.129072
1237	195000	207118.078068
344	85000	71893.605154
1213	145000	115851.441140
170	128500	155184.025162
864	250580	205734.740156
729	103000	101167.113961
218	311500	231446.901964
1106	179900	225745.975104

8.4 Display the metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))

print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))

```
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
predictions)))
```

```
Mean Absolute Error: 23316.33524577359

Mean Squared Error: 990026435.9419634

Mean Absolute Percentage Error: 0.1424496823167245

Root Mean Squared Error: 31464.6855369947
```

8.5 Display the model

```
finalModel.score(x_train, y_train)
```

```
0.8316433311109943
```

8.6 Creation of the new linear regression

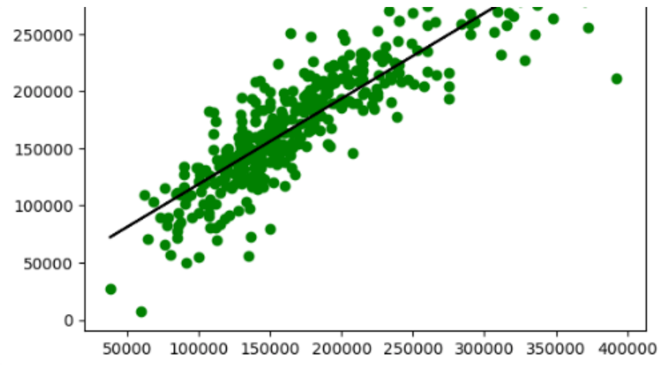
```
y_test_array = y_test.array.reshape(-1, 1)

regressor = LinearRegression()

regressor.fit(y_test_array, predictions)

y_regressor = regressor.predict(y_test_array)

print(regressor.coef_)

print(regressor.intercept_)
```

```
[0.74798476]

43763.57029332046
```

8.7 Display of the of the Plots of the predictions

```
plt.scatter(y_test, predictions,color='g')

plt.plot(y_test_array, y_regressor,color='black')
```

```
[<matplotlib.lines.Line2D at 0x7fc0962afaf0>]
```



8.8 Change the train and test split proportion

```
y = dfCleaned.SalePrice

col = dfCleaned.columns.tolist()

col.remove('SalePrice')

x = dfCleaned[col]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.14, random_state=1)

model = LinearRegression()

model.fit(x_train, y_train)

predictions = model.predict(x_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))

print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))
```

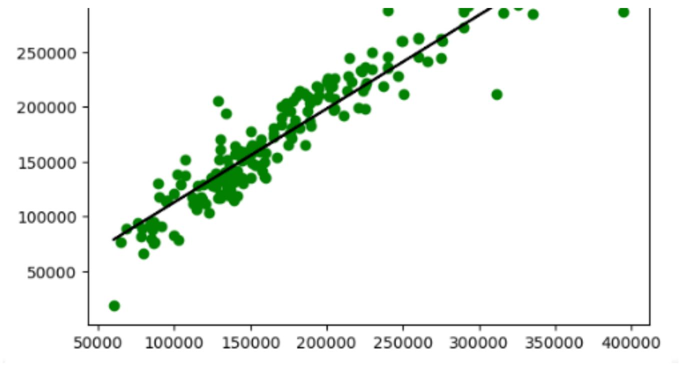
```
Mean Absolute Error: 15871.71051900959

Mean Squared Error: 493841137.4366765

Mean Absolute Percentage Error: 0.10112593066571478
```

```
model.score(x_train, y_train)
```

```
0.9040907227158079
```

```
y_test_array = y_test.array.reshape(-1, 1)

regressor = LinearRegression()
```

```
regressor.fit(y_test_array, predictions)

y_regressor = regressor.predict(y_test_array)

plt.scatter(y_test, predictions,color='g')

plt.plot(y_test_array, y_regressor,color='k')

[<matplotlib.lines.Line2D at 0x7fc09561c6a0>]
```



8.9 Remove the outliers

```
dfCleanedTest = dfCleaned

dfCleanedTest = remove_outliers(dfCleanedTest, ["YearBuilt"])

dfCleanedTest = remove_outliers(dfCleanedTest, ["YearRemodAdd"])

dfCleanedTest = remove_outliers(dfCleanedTest, ["MasVnrArea"])

dfCleanedTest = remove_outliers(dfCleanedTest, ["TotRmsAbvGrd"])

dfCleanedTest = remove_outliers(dfCleanedTest, ["Fireplaces"])

y = dfCleanedTest.SalePrice

col = dfCleanedTest.columns.tolist()

col.remove('SalePrice')

x = dfCleanedTest[col]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=1)

model = LinearRegression()

model.fit(x_train, y_train)

predictions = model.predict(x_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))
print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))
Mean Absolute Error: 15214.753573758997
Mean Squared Error: 477241810.61568594
Mean Absolute Percentage Error: 0.08607943868711154

model.score(x_train, y_train)
0.8976257610078535
```

8.10 Creation of the new predictions

```
y = dfBasicCleaning.SalePrice
col = dfBasicCleaning.columns.tolist()
col.remove('SalePrice')
x = dfBasicCleaning[col]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.12, random_state=1)
model = LinearRegression()
model.fit(x_train, y_train)
predictions = model.predict(x_test)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))
print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))
model.score(x_train, y_train)
```

8.11 Remove the outliers

```
dfBasicCleaningTest = dfBasicCleaning
dfBasicCleaningTest = remove_outliers(dfBasicCleaningTest,
dfBasicCleaningTest.columns.tolist())
y = dfBasicCleaningTest.SalePrice
col = dfBasicCleaningTest.columns.tolist()
```

```
col.remove('SalePrice')

x = dfBasicCleaningTest[col]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.18, random_state=1)

model = LinearRegression()

model.fit(x_train, y_train)

predictions = model.predict(x_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))

print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))

model.score(x_train, y_train)

y_test_array = y_test.array.reshape(-1, 1)

regressor = LinearRegression()

regressor.fit(y_test_array, predictions)

y_regressor = regressor.predict(y_test_array)

plt.scatter(y_test, predictions,color='g')

plt.plot(y_test_array, y_regressor,color='k')
```

8.12 Remove the items with the least correlation

```
dfCleanedTest2 = dfCleaned

dfCleanedTest2 = dfCleanedTest2.drop(['GarageFinish', 'GarageType', 'KitchenQual',
'HeatingQC', 'BsmtQual', 'ExterQual'], axis=1)

y = dfCleanedTest2.SalePrice

col = dfCleanedTest2.columns.tolist()

col.remove('SalePrice')

x = dfCleanedTest2[col]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.14, random_state=1)

model = LinearRegression()

model.fit(x_train, y_train)
```

```
predictions = model.predict(x_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))

print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))

model.score(x_train, y_train)
```

8.13 Remove of the significant outliers

```
dfCleanedTest2x = dfCleanedTest2

dfCleanedTest2x = remove_outliers(dfCleanedTest2x, ["OverallQual"])

dfCleanedTest2x = remove_outliers(dfCleanedTest2x, ["TotalBsmtSF"])

dfCleanedTest2x = remove_outliers(dfCleanedTest2x, ["1stFlrSF"])

dfCleanedTest2x = remove_outliers(dfCleanedTest2x, ["GrLivArea"])

dfCleanedTest2x = remove_outliers(dfCleanedTest2x, ["FullBath"])

dfCleanedTest2x = remove_outliers(dfCleanedTest2x, ["GarageCars"])

dfCleanedTest2x = remove_outliers(dfCleanedTest2x, ["GarageArea"])

y = dfCleanedTest2x.SalePrice

col = dfCleanedTest2x.columns.tolist()

col.remove('SalePrice')

x = dfCleanedTest2x[col]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.29, random_state=1)

model = LinearRegression()

model.fit(x_train, y_train)

predictions = model.predict(x_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))

print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))

model.score(x_train, y_train)
```

8.14 Remove of the items with the least correlation

```
dfCleanedTest3 = dfCleaned

dfCleanedTest3 = dfCleanedTest3.drop(['SaleType', 'MiscFeature', 'PoolQC',
'GarageFinish', 'GarageType', 'FireplaceQu', 'KitchenQual', 'HeatingQC', 'Heating',
'BsmtFinType1', 'BsmtExposure', 'BsmtQual', 'ExterQual', 'BldgType', 'LotConfig',
'LotShape', 'MSZoning', 'EnclosedPorch', 'KitchenAbvGr', 'OverallCond',
'MSSubClass'], axis=1)

y = dfCleanedTest3.SalePrice

col = dfCleanedTest3.columns.tolist()

col.remove('SalePrice')

x = dfCleanedTest3[col]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.11, random_state=1)

model = LinearRegression()

model.fit(x_train, y_train)

predictions = model.predict(x_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))

print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))

model.score(x_train, y_train)
```

8.15 Remove significant outliers

```
dfCleanedTest3x = dfCleanedTest3

dfCleanedTest3x = remove_outliers(dfCleanedTest3x, ["OverallQual"])

dfCleanedTest3x = remove_outliers(dfCleanedTest3x, ["TotalBsmtSF"])

dfCleanedTest3x = remove_outliers(dfCleanedTest3x, ["1stFlrSF"])

dfCleanedTest3x = remove_outliers(dfCleanedTest3x, ["GrLivArea"])

dfCleanedTest3x = remove_outliers(dfCleanedTest3x, ["FullBath"])

dfCleanedTest3x = remove_outliers(dfCleanedTest3x, ["GarageCars"])

dfCleanedTest3x = remove_outliers(dfCleanedTest3x, ["GarageArea"])
```

```
y = dfCleanedTest3x.SalePrice

col = dfCleanedTest3x.columns.tolist()

col.remove('SalePrice')

x = dfCleanedTest3x[col]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.16, random_state=1)

model = LinearRegression()

model.fit(x_train, y_train)

predictions = model.predict(x_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions))

print('Mean Absolute Percentage Error:',
metrics.mean_absolute_percentage_error(y_test, predictions))

model.score(x_train, y_train)
```

9 Load the CSV file

It was loaded a new CSV file, find and clean the null values. Then we had to covert the object values in the dataframe into numeric to predict the sales price.

9.1 Load of the document

```
dfToPredict = pd.read_csv("HousePrices-FreshDataToPredict.csv")

dfToPredict.shape
```

9.2 Display the null values

```
dfToPredict.columns[dfToPredict.isnull().any()]

Index(['LotFrontage', 'Alley', 'BsmtExposure', 'KitchenQual', 'FireplaceQu',

       'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageQual', 'GarageCond',

       'PoolQC', 'Fence', 'MiscFeature'],

      dtype='object')
```

9.3 Fill the data

```
dfToPredict["LotFrontage"].fillna(dfToPredict['LotFrontage'].mean(), inplace = True)
```

```
dfToPredict["Alley"].fillna("NA", inplace = True)

dfToPredict["BsmtExposure"].fillna("NA", inplace = True)

dfToPredict["KitchenQual"].fillna("TA", inplace = True)

dfToPredict["FireplaceQu"].fillna("NA", inplace = True)

dfToPredict["GarageType"].fillna("NA", inplace = True)

dfToPredict["GarageYrBlt"].fillna("NA", inplace = True)

dfToPredict["GarageFinish"].fillna("NA", inplace = True)

dfToPredict["GarageQual"].fillna("NA", inplace = True)

dfToPredict["GarageCond"].fillna("NA", inplace = True)

dfToPredict["PoolQC"].fillna("NA", inplace = True)

dfToPredict["Fence"].fillna("NA", inplace = True)

dfToPredict["MiscFeature"].fillna("NA", inplace = True)
```

9.4 Convert objects values in the dataframe into numeric

```
objListPred = dfToPredict.select_dtypes(include=np.object).columns.tolist()

dfToCleanNumPred = dfToPredict.drop(objList, axis = 1)

numListPred = dfToPredict.select_dtypes(include=np.number).columns.tolist()

dfToCleanObjPred = dfToPredict.drop(numListPred, axis = 1)

dfToCleanObjPred = dfToCleanObjPred.astype(str)

dfToCleanObjPred = dfToCleanObjPred.apply(LabelEncoder().fit_transform)

dfBasicCleaningPred = pd.concat([dfToCleanNumPred, dfToCleanObjPred], axis=1,
join="outer")
```

10 Prediction

After all the process conclude we’ve created the prediction of the sales price base on the variables values.

10.1 Prediction of the SalePrice

```
dfBasicCleaningPred=dfBasicCleaningPred.drop(['Id'], axis=1)

dfBasicCleaningPred=dfBasicCleaningPred.drop(['LotShape'], axis=1)

dfBasicCleaningPred=dfBasicCleaningPred.drop(['LandContour'], axis=1)
```

dfBasicCleaningPred=dfBasicCleaningPred.drop(['LotConfig'], axis=1)

…

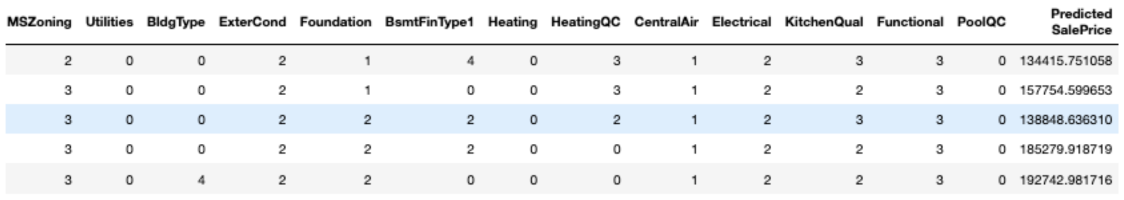
10.2 Prediction of the SalePrice

predToPredict = finalModel.predict(dfBasicCleaningPred)

predToPredict = pd.concat([dfBasicCleaningPred, pd.DataFrame(predToPredict)], axis=1)

predToPredict.columns = [*predToPredict.columns[:-1], 'Predicted SalePrice']

predToPredict.head()



MSZoning	Utilities	BldgType	ExterCond	Foundation	BsmtFinType1	Heating	HeatingQC	CentralAir	Electrical	KitchenQual	Functional	PoolQC	Predicted SalePrice
2	0	0	2	1	4	0	3	1	2	3	3	0	134415.751058
3	0	0	2	1	0	0	3	1	2	2	3	0	157754.599653
3	0	0	2	2	2	0	2	1	2	3	3	0	138848.636310
3	0	0	2	2	2	0	0	1	2	2	3	0	185279.918719
3	0	4	2	2	0	0	0	1	2	2	3	0	192742.981716

11 Conclusion

With this work we’ve concluded that the machine learning project aimed to predict houses sales prices can be a complex and challenging task, but also useful and powerful. However, with the right approach and techniques, it can yield valuable insights and predictions that can assist in the real estate market. The key success factors for a such project includes having a large and relevant dataset, selecting an appropriate model, and incorporating the domain knowledge.

Additionally, it’s important to continuously monitor and update the model as new data becomes available to ensure its ongoing performance.

Overall, using machine learning to predict house prices is a promising area of research that has the potential to revolutionize the way we approach and make decisions in the real estate market.