DATA MINING AND MACHINE LEARNING

HOUSE PRICE PREDICTOR

Leena Aizdi

Table of Contents

1. INTRODUCTION

House Price Predictor is an application which allows a user to know the expected price of a house depending upon certain attributes. The user is asked to fill in the values of a set of parameters and in return he gets the estimated value of price of the house containing those specific features.

In particular, the goal of the project is to implement a regression model which is able to predict price of the house containing the features specified by the user. The application takes certain inputs regarding attributes from a user and then predict the price of house based on the input attributes entered by the user with the help of the already trained model.

The implementation of the project can be located in the following repository:

https://github.com/leenaaizdee/DMMLProject.git

2. REQUIREMENTS AND DESIGN

In this section, the requirements of the application will be stated. Moreover, a design analysis would be conducted.

2.1. REQUIREMENTS OF THE APPLICATION

Considering the functional requirements, the application should:

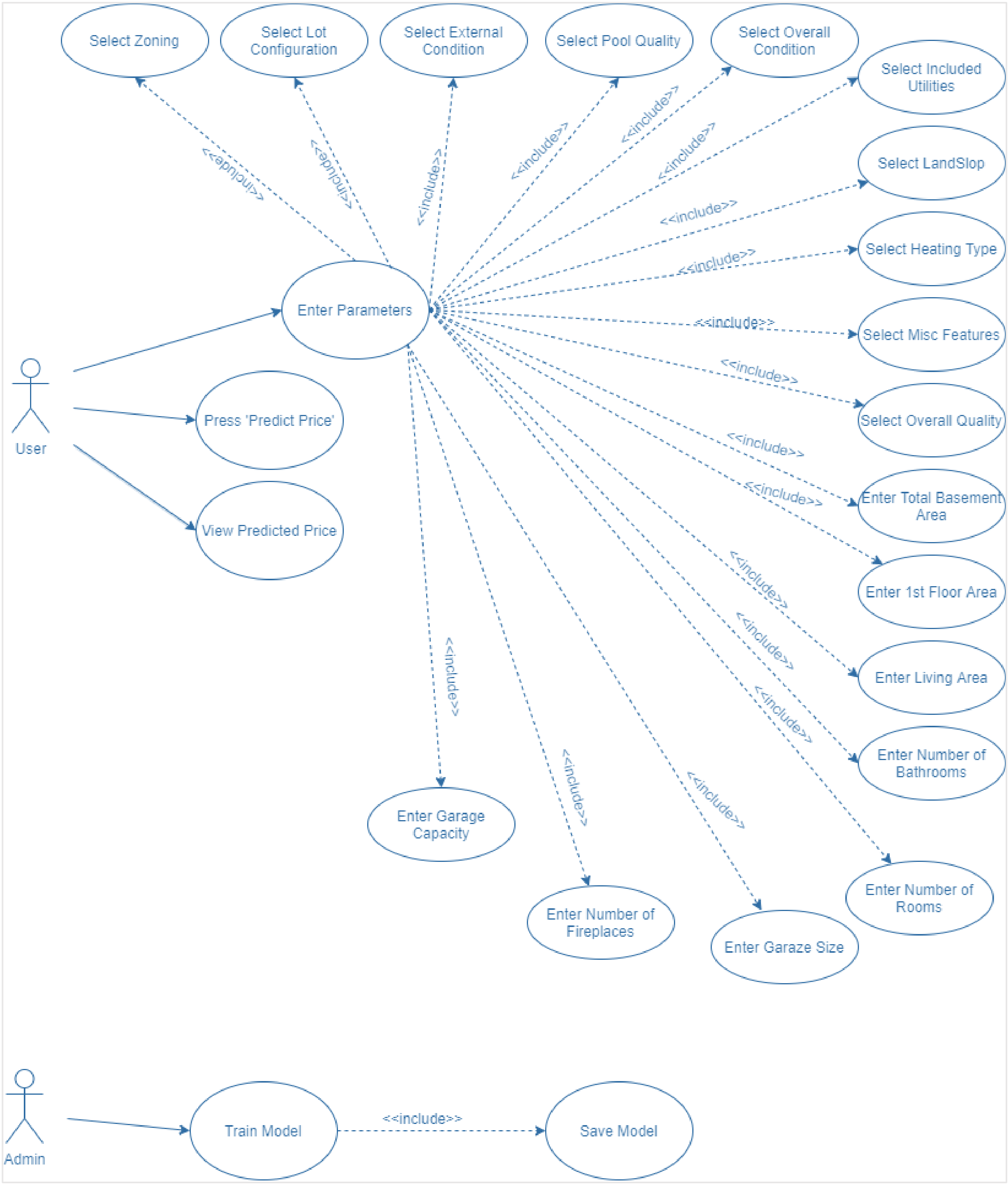
- Take (and perform checks on) inputs from the user about relative attributes of the house
- Compute the price of the house based on the attribute values entered by the user, through a trained model
- Show the output to the user

And for the non-functional requirements, the application should:

- Have a simple, interactive, and understandable GUI
- Have good accuracy

2.2. USE CASE DIAGRAM

The following diagram illustrates the use case:

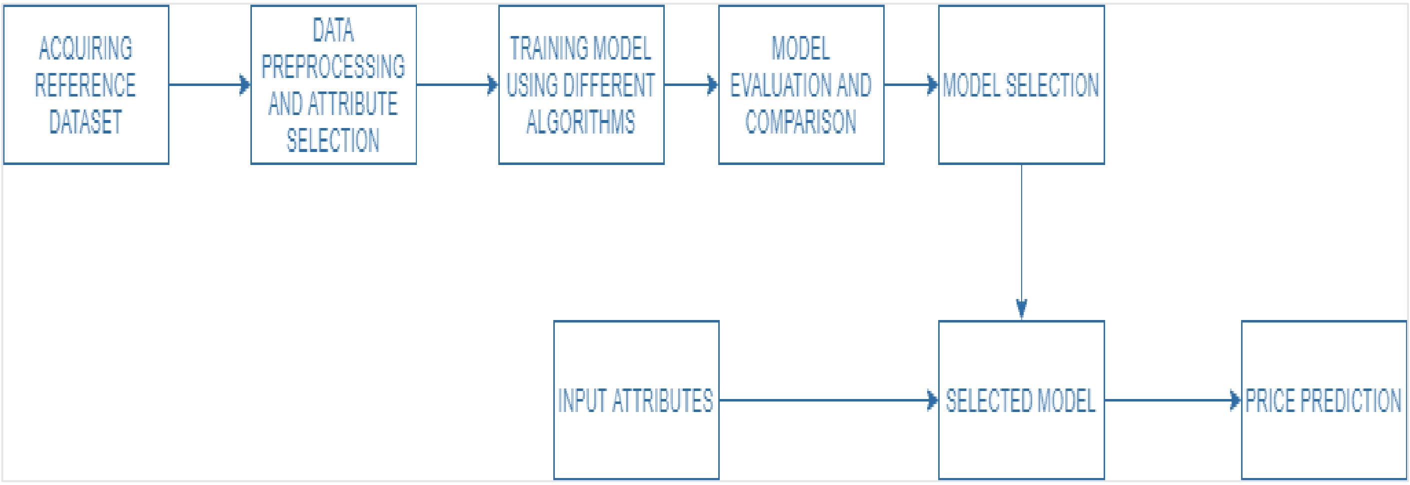


Select Zoning
Select Lot Configuration
Select External Condition
Select Pool Quality
Select Overall Condition
Select Included Utilities
Select LandSlop
Select Heating Type
Select Misc Features
Select Overall Quality
Enter Total Basement Area
Enter 1st Floor Area
Enter Living Area
Enter Number of Bathrooms
Enter Number of Rooms
Enter Garaze Size
Enter Number of Fireplaces
Enter Garage Capacity
Enter Parameters
Press 'Predict Price'
View Predicted Price
User
<<include>>
Admin
Train Model
Save Model

3. DETAILED KDD PROCESS

This section will put light on all the stages of the analysis.

The general workflow is as follows:



3.1. ABOUT THE DATASET

The reference dataset used to train the model has been taken from Kaggle. It contains 1460 instances and 72 attributes.

The attributes present in the dataset are:

1. SalePrice - the property's sale price in dollars. This is the target variable that we're trying to predict.
2. ID – Unique identifier for each instance
3. MSSubClass: The building class
4. MSZoning: The general zoning classification
5. LotArea: Lot size in square feet
6. Street: Type of road access
7. Alley: Type of alley access
8. LotShape: General shape of property
9. LandContour: Flatness of the property
10. Utilities: Type of utilities available
11. LotConfig: Lot configuration

12. LandSlope: Slope of property
13. Neighborhood: Physical locations within Ames city limits
14. Condition1: Proximity to main road or railroad
15. Condition2: Proximity to main road or railroad (if a second is present)
16. BldgType: Type of dwelling
17. HouseStyle: Style of dwelling
18. OverallQual: Overall material and finish quality
19. OverallCond: Overall condition rating
20. YearBuilt: Original construction date
21. YearRemodAdd: Remodel date
22. RoofStyle: Type of roof
23. RoofMatl: Roof material
24. Exterior1st: Exterior covering on house
25. Exterior2nd: Exterior covering on house (if more than one material)
26. MasVnrType: Masonry veneer type
27. MasVnrArea: Masonry veneer area in square feet
28. ExterQual: Exterior material quality
29. ExterCond: Present condition of the material on the exterior
30. Foundation: Type of foundation
31. BsmtUnfSF: Unfinished square feet of basement area
32. TotalBsmtSF: Total square feet of basement area
33. Heating: Type of heating
34. HeatingQC: Heating quality and condition
35. CentralAir: Central air conditioning
36. Electrical: Electrical system
37. 1stFlrSF: First Floor square feet
38. 2ndFlrSF: Second floor square feet
39. LowQualFinSF: Low quality finished square feet (all floors)
40. GrLivArea: Above grade (ground) living area square feet
41. BsmtFullBath: Basement full bathrooms
42. BsmtHalfBath: Basement half bathrooms
43. FullBath: Full bathrooms above grade

44. HalfBath: Half baths above grade
45. Bedroom: Number of bedrooms above basement level
46. Kitchen: Number of kitchens
47. KitchenQual: Kitchen quality
48. TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
49. Functional: Home functionality rating
50. Fireplaces: Number of fireplaces
51. FireplaceQu: Fireplace quality
52. GarageType: Garage location
53. GarageFinish: Interior finish of the garage
54. GarageCars: Size of garage in car capacity
55. GarageArea: Size of garage in square feet
56. GarageQual: Garage quality
57. GarageCond: Garage condition
58. PavedDrive: Paved driveway
59. WoodDeckSF: Wood deck area in square feet
60. OpenPorchSF: Open porch area in square feet
61. EnclosedPorch: Enclosed porch area in square feet
62. 3SsnPorch: Three season porch area in square feet
63. ScreenPorch: Screen porch area in square feet
64. PoolArea: Pool area in square feet
65. PoolQC: Pool quality
66. Fence: Fence quality
67. MiscFeature: Miscellaneous feature not covered in other categories
68. MiscVal: $Value of miscellaneous feature
69. MoSold: Month Sold
70. YrSold: Year Sold
71. SaleType: Type of sale
72. SaleCondition: Condition of sale

A quick view on the attributes tells that the dataset is filled with both categorical and numerical attributes. Both of these types of attributes will have a certain effect on the target variable i.e., Sale Price of the house.

3.2. DATA PRE-PROCESSING AND ATTRIBUTE SELECTION

After loading the dataset, the first task in Data Pre-processing is to know the data. By observing the dataset, it is obvious that it contained 43 categorical attributes and 29 numerical attributes. A lot of attributes seem very redundant but in order to achieve good results, we have to perform certain analyses to be certain!

But, the ID attribute clearly plays no part in the prediction of the sale price because it is just a unique identifier for each instance, so I decided to remove it.

After that I check for the missing or null values in the dataset so that a decision can be made on how to tackle them. Fortunately, there were no missing values for all the columns except just one; In the attribute column MasVNrArea, there were more than half values equal to null. In this case, it was not suggested to replace the values by the mean, so I decided to remove that column. Also, because in the dataset, there had already a lot of redundant attributes so it seemed a better choice to remove this column rather than filling it with dummy values which might have a different effect on the price.

After that I also check the datatype of all the attributes. Some of the attributes, although they were categorical but were being treated as numerical attribute. The attributes containing year values were being treated as numeric attributes because their datatype was int. Although, we know that year is a categorical attribute. The other attributes were categorical but already encoded in numerical values in the dataset and hence were being treated as a numerical attribute. Henceforth, I decided to change the datatype of those attributes to String. The attributes were:

- YearBuilt
- YearRemodAdd
- OverallQual
- OverallCond

After this stage, I encoded all the categorical attributes to unique numerical values. This is done because during certain algorithms applied on categorical attributes do not take string values as an input. So, each value of a categorical attribute was mapped to a numerical value.

The biggest task in this phase was to handle categorical and numerical data and find a chunk of attributes having a visible effect on the target variable.

In order to find redundancy among attributes I had to consider that there are different algorithms for numerical and categorical attributes. So first, I applied Chi Square Test on all the categorical attributes. I generated a 43x43 matrix depicting the mutual correlation between the categorical attributes. A snapshot of a part of the matrix is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		MSZonir	Street	Alley	LotShap	LandCor	Utilities	LotConfi	LandSlo	Neighbor	Conditior	Conditior	BldgTyp	HouseSt	OverallQ	OverallC
2	MSZonir	0	0	0	0	0	0.99175	0.00085	0.00332	0	0.00125	2	0	0	0	0
3	Street	0	0	0.81854	0.19796	7.00E-05	1	0.84288	0	0	0	2	0.00016	0.3739	0.05375	0.06378
4	Alley	0	0.81854	0	0.00041	0.00192	0.96729	0.17234	0.80215	0	0	2	0	0	0.00013	0
5	LotShap	0	0.19796	0.00041	0	0	0.5687	0	0	0	0	2	2.00E-05	0.00203	0	0.02368
6	LandCor	0	7.00E-05	0.00192	0	0	0.99014	0.006	0	0	0.68038	2	0.00105	0	0	0
7	Utilities	0.99175	1	0.96729	0.5687	0.99014	0	0.00575	0.97216	0.03947	1	2	0.99546	0.00313	0.96792	0.77904
8	LotConfi	0.00085	0.84288	0.17234	0	0.006	0.00575	0	0.00095	0	0	2	0.00023	0.5691	0.39268	0.47691
9	LandSlo	0.00332	0	0.80215	0	0	0.97216	0.00095	0	0	0.90251	2	0.26294	0.50307	0	0
10	Neighbor	0	0	0	0	0	0.03947	0	0	0	2	2	0	0	0	0
11	Conditior	0.00125	0	0	0	0.68038	1	0	0.90251	2	0	2	0.00042	0	0.00083	2
12	Conditior	2	2	2	2	2	2	2	2	2	2	0	2	2	2	2
13	BldgTyp	0	0.00016	0	2.00E-05	0.00105	0.99546	0.00023	0.26294	0	0.00042	2	0	0	0	0
14	HouseSt	0	0.3739	0	0.00203	0	0.00313	0.5691	0.50307	0	0	2	0	0	0	0
15	OverallQ	0	0.05375	0.00013	0	0	0.96792	0.39268	0	0	0.00083	2	0	0	0	0
16	OverallC	0	0.06378	0	0.02368	0	0.77904	0.47691	0	0	2	2	0	0	0	0
17	YearBuil	2	0.0001	0	0.01876	2	0.99998	2	0	2	2	2	2	2	2	2
18	YearRem	0	0	0.00395	0.01414	1.00E-05	0.002	0.03332	6.00E-05	2	2	2	0	2	2	2
19	RoofStyl	0.00016	0.97582	0	0.15446	0	0.99801	9.00E-05	0	0	2.00E-05	2	0.03144	0	0	0.07039
20	RoofMatl	2	1	0.99996	2	2	2	2	0	2	2	2	2	2	2	2
21	Exterior1	0	0.95147	0	0.00334	0	0.97599	0.07327	0	0	2	2	0	0	0	0
22	Exterior2	0	0.97212	0	0.00048	0	0.97877	0.00369	6.00E-05	0	2	2	0	0	0	0
23	MasVnrT	0	0.95178	4.00E-05	0.00274	1.00E-05	0.68396	0.81332	0.11005	0	0.31935	2	5.00E-05	0	0	0
24	ExterQu	0	0	0	0	0	0.8937	0.37691	2.00E-05	0	0	2	0	0	0	0
25	ExterCor	1.00E-05	0.98135	0.10082	0.4857	0.98162	0.9977	0.88572	0.81748	0	0.36447	2	0	0	0	0
26	Foundati	0	0.16474	0	0	0	0.93455	0.06164	0.07088	0	3.00E-05	2	0	0	0	0
27	Heating	0.01064	0.99966	0	0.2749	0.99216	1	0.93768	0.69866	2	2	2	0	0	2	2
28	HeatingC	0	0.34816	0.00019	0.019	0.01607	0.28104	0.41384	0.05353	0	0	2	0	0	0	0
29	CentralAi	0	0.06571	0	0.00018	1.00E-05	1	0.04363	0.90921	0	0.24011	2	0	0	0	0
30	Electrical	0	0.94925	0	0	0.06855	0.01251	0.67091	0.99905	0	2	2	6.00E-05	0	0	0
31	KitchenC	0	0.03589	2.00E-05	0	0	0.684	0.50214	0.06584	0	0.00043	2	0	0	0	0
32	Function	0.74143	0.99846	0.22552	0.9941	0.75976	0.99999	0.99586	0.00508	2	2	2	0.01498	0.13395	0	0
33	Fireplace	0	0.95718	0.00456	0	0.00049	0.72401	0.02736	0.22349	0	0.62576	2	0	0	0	0
34	GarageT	0	0	0	0	0	0	0.01067	1.00E-05	0	0	2	0	0	0	0
35	GarageFi	0	0.65546	0	0	0	0.48231	0.16734	0.70115	0	0	2	0	0	0	0
36	GarageQ	0	0.895	0	0.00043	0.13421	0.99978	0.37801	0.99942	2	2	2	0	0	0	0
37	GarageC	0	0.90144	0	0.0335	0.57547	0.99983	0.01733	0.85358	2	0.21398	2	0	0	0	0
38	PavedDri	0	0.53563	0	0.00093	0	0.95618	0.23366	0.76803	0	7.00E-05	2	0	0	0	0
39	PoolQC	0.99956	0.9987	0.99821	0	0.671	2	0.27658	0.99888	0.16579	2	2	0.99992	0.04283	0	2
40	Fence	0.23015	0.83793	0.5399	0.19154	0.17398	0.99343	0.48384	0.66613	0	0.22462	2	0.001	0	0	0
41	MiscFeat	0.5582	0	0.74735	0.79055	0.99968	0.99982	0.88576	0.59038	0.88945	2	2	0.00052	0.97184	0.87488	0.38279
42	SaleTyp	0	0.00107	0.23423	0.96755	0.25923	6.00E-05	0.72083	0.93269	2	2	2	0	2	2	2
43	SaleCon	0	0.00159	0.01421	0.44951	0	0.01939	0.14124	0.1669	0	0.75151	2	0	0	0	0

	A	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
1		YearBuil	YearRem	RoofStyl	RoofMatl	Exterior1	Exterior2	MasVnrT	ExterQu	ExterCor	Foundati	Heating	HeatingC	CentralAi	Electrical	KitchenC
2	MSZonir	2	0	0.00016	2	0	0	0	0	1.00E-05	0	0.01064	0	0	0	0
3	Street	0.0001	0	0.97582	1	0.95147	0.97212	0.95178	0	0.98135	0.16474	0.99966	0.34816	0.06571	0.94925	0.03589
4	Alley	0	0.00395	0	0.99996	0	0	4.00E-05	0	0.10082	0	0	0.00019	0	0	2.00E-05
5	LotShap	0.01876	0.01414	0.15446	2	0.00334	0.00048	0.00274	0	0.4857	0	0.2749	0.019	0.00018	0	0
6	LandCor	2	1.00E-05	0	2	0	0	1.00E-05	0	0.98162	0	0.99216	0.01607	1.00E-05	0.06855	0
7	Utilities	0.99998	0.002	0.99801	2	0.97599	0.97877	0.68396	0.8937	0.9977	0.93455	1	0.28104	1	0.01251	0.684
8	LotConfi	2	0.03332	9.00E-05	2	0.07327	0.00369	0.81332	0.37691	0.88572	0.06164	0.93768	0.41384	0.04363	0.67091	0.50214
9	LandSlo	0	6.00E-05	0	0	0	6.00E-05	0.11005	2.00E-05	0.81748	0.07088	0.69866	0.05353	0.90921	0.99905	0.06584
10	Neighbor	2	2	0	2	0	0	0	0	0	0	2	0	0	0	0
11	Conditior	2	2	2.00E-05	2	2	2	0.31935	0	0.36447	3.00E-05	2	0	0.24011	2	0.00043
12	Conditior	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
13	BldgTyp	2	0	0.03144	2	0	0	5.00E-05	0	0	0	0	0	0	6.00E-05	0
14	HouseSt	2	2	0	2	0	0	0	0	0	0	0	0	0	0	0
15	OverallQ	2	2	0	2	0	0	0	0	0	0	2	0	0	0	0
16	OverallC	2	2	0.07039	2	0	0	0	0	0	0	2	0	0	0	0
17	YearBuil	0	2	2	2	2	2	2	0	2	2	2	2	0	2	0
18	YearRem	2	0	2	2	2	2	0	0	0.00107	0	2	0	0	2	0
19	RoofStyl	2	2	0	2	0	0	0	0	0	0	2	0.89348	0.09344	0.9999	0
20	RoofMatl	2	2	2	0	2	2	2	2	2	2	2	2	0.9916	2	2
21	Exterior1	2	2	0	2	0	0	0	0	3.00E-05	0	2	0	0	2	0
22	Exterior2	2	2	0	2	0	0	0	0	0.01429	0	2	0	0	2	0
23	MasVnrT	2	0	0	2	0	0	0	0	0.01292	0	0.79985	0	0	0.00216	0
24	ExterQu	0	0	0	2	0	0	0	0	0	0	0.08662	0	0	0	0
25	ExterCor	2	0.00107	0	2	3.00E-05	0.01429	0.01292	0	0	0	2	0.00134	0	0	0
26	Foundati	2	0	0	2	0	0	0	0	0	0	2	0	0	0	0
27	Heating	2	2	2	2	2	2	0.79985	0.08662	2	2	0	0	0	2	0
28	HeatingC	2	0	0.89348	2	0	0	0	0	0.00134	0	0	0	0	0	0
29	CentralAi	0	0	0.09344	0.9916	0	0	0	0	0	0	0	0	0	0	0
30	Electrical	2	2	0.9999	2	2	2	0.00216	0	0	0	2	0	0	0	0
31	KitchenC	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
32	Function	2	2	0	2	2	2	0	0	0	0	2	0.29986	0.00787	2	8.00E-05
33	Fireplace	2	0	0	2	0	0	0	0	0.4109	0	2	0	0	0.00929	0
34	GarageT	2	2	0.00035	2	0	0	0	0	0	0	2	0	0	0	0
35	GarageFi	0	0	1.00E-05	2	0	0	0	0	0	0	0	0	0	0	0
36	GarageQ	2	2	0.54177	2	0	0	0	0	0	0	2	0	0	2	0
37	GarageC	2	2	0.02925	2	0	2	0	0	0	0	2	0	0	0	0
38	PavedDri	0	0	0	0.99977	0	0	0	0	0	0	0	0	0	0	0
39	PoolQC	2	0.19274	0.00063	2	2	2	0.70307	0.15751	2	0.98062	2	0.09246	0.92119	2	0.16611
40	Fence	2	0	0.43329	2	0	0	0.00083	0	0	0	0.95179	0	0.43927	0.99227	0
41	MiscFeat	2	2	0	2	2	0.58217	0.45125	0	2	0	2	0.14443	0.16845	2	0.00039
42	SaleTyp	2	2	2	2	2	2	0	0	2	2	2	0	0.0001	2	0
43	SaleCon	2	2	0	2	0	0	0	0	0.01813	0	2	0	0.00026	0	0

Figure 1: chi2 matrix of categorical features

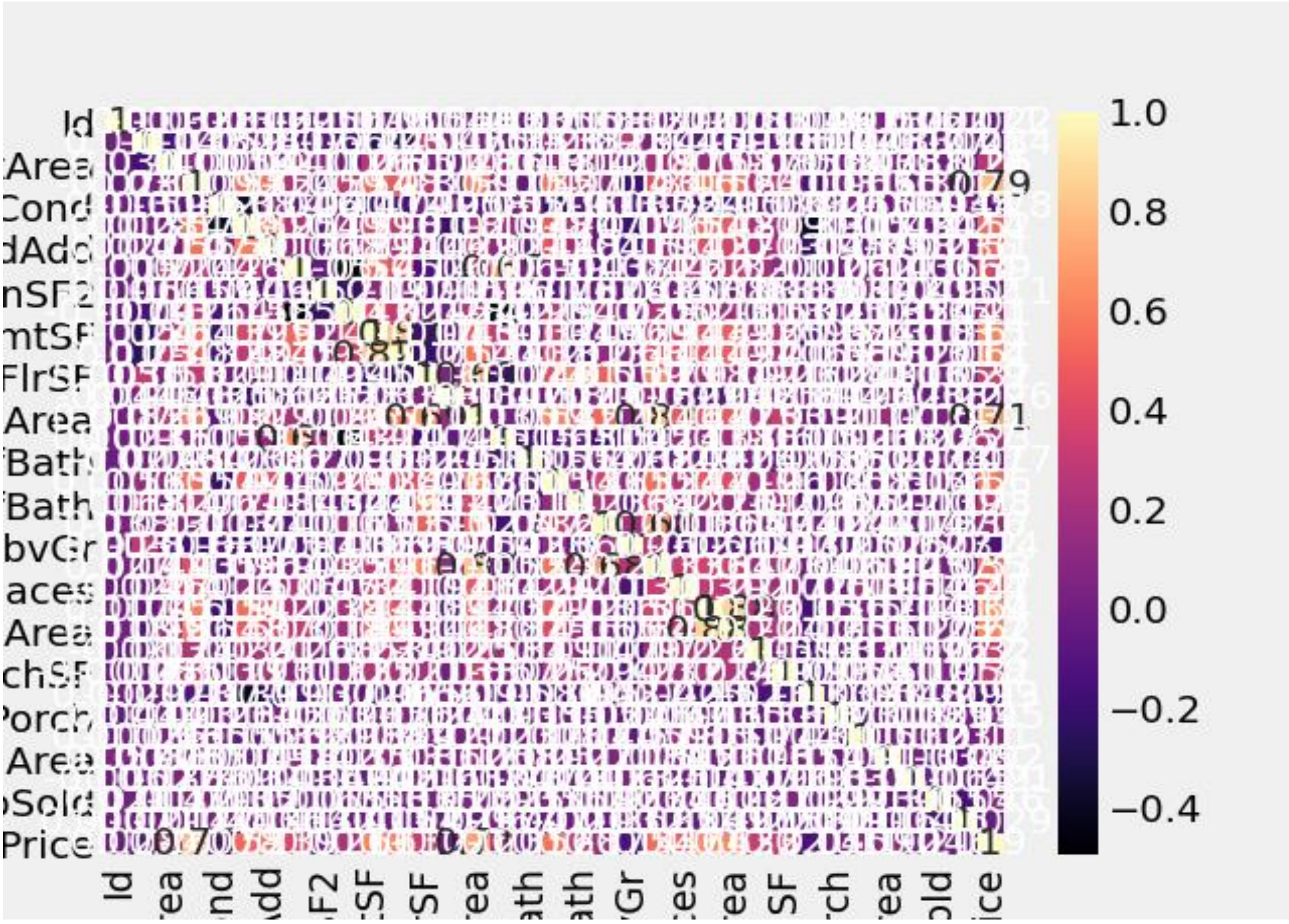
The chi square values were encoded in such a way that if the p-value[1] is very large (i.e., there exists no correlation between the two attributes), I change the value to 2 in the matrix. And when the p-value is very small (i.e., the two attributes are highly correlated), I change the value to 0. This is done for simplicity in evaluating the results. After the values are generated, I store them in the matrix which will then be used to evaluate the results and eliminate redundant attributes. In order to eliminate the redundant and highly correlated attributes, I perform a traversal on the matrix

[1] The p-value is the probability of obtaining results at least as extreme as the observed results of a statistical hypothesis test, assuming that the null hypothesis is correct. A smaller p-value means that there is stronger evidence in favour of the rejection of the null hypothesis.

generated. In this traversal, I check the values between each attribute with all the remaining attributes. If the matrix value is equal to 0 between two attributes, it means that these two attributes are highly correlated, and it is not useful to store both of them for further analyses. So, I eliminate one of them. I do this process for all the attributes and finally print selected features and eliminated features. The final categorical attributes selected are:

- MSZoning
- Utilities
- LotConfig
- LandSlope
- ExterCond
- Heating
- PoolQC
- MiscFeature
- OverallQual
- OverallCond

Now we have to analyse the numerical attributes. At first, I generated a heatmap to analyse the relation between all the numerical attributes. The heatmap would show visually the relation between each numerical attribute with all the other numerical attributes. But since the number of attributes was large, the heatmap did not really show visible results. Moreover, it was hard to analyse the results by visualizing the heatmap, so I decided to remove this part and opt for other measures.



After discarding the heatmap, for the numerical attributes, I performed four different measures to analyse their mutual correlation and then select the best attributes among them. The four methods were:

- Pearson
- Kendall
- Spearman
- SelectKBest

Since in this case, target variable (i.e., price) is also a numerical attribute and the numerical attributes are 28 excluding price, we can directly analyse their affect on the target variable and select the best uncorrelated attributes having effect on price variable. I tried different measures to analyse all the results and then make a final attribute selection. The results of all the methods were same.

Some snapshots of analysis results are shown below:

```
correlation of column  LotArea:--------->Sales Price :--------->(0.2638433538714057, 1.1231391549193063e-24)
correlation of column  BsmtUnfSF:--------->Sales Price :--------->(0.21447910554696886, 1.1829757963602225e-16)
correlation of column  TotalBsmtSF:--------->Sales Price :--------->(0.6135805515591956, 9.484229391505757e-152)
correlation of column  1stFlrSF:--------->Sales Price :--------->(0.6058521846919146, 5.394710618971284e-147)
correlation of column  2ndFlrSF:--------->Sales Price :--------->(0.3193338028320682, 5.764335119181669e-36)
correlation of column  LowQualFinSF:--------->Sales Price :--------->(-0.02560613000067949, 0.32820730984071167)
correlation of column  GrLivArea:--------->Sales Price :--------->(0.7086244776126521, 4.518033646779945e-223)
correlation of column  BsmtFullBath:--------->Sales Price :--------->(0.22712223313149368, 1.5503441372146568e-18)
correlation of column  BsmtHalfBath:--------->Sales Price :--------->(-0.016844154297359006, 0.5201536357280931)
correlation of column  FullBath:--------->Sales Price :--------->(0.5606637627484438, 1.236470066763814e-121)
correlation of column  HalfBath:--------->Sales Price :--------->(0.2841076755947834, 1.650473395572193e-28)
correlation of column  BedroomAbvGr:--------->Sales Price :--------->(0.16821315430073983, 9.927497326188457e-11)
correlation of column  KitchenAbvGr:--------->Sales Price :--------->(-0.1359073708421416, 1.8604260320764677e-07)
correlation of column  TotRmsAbvGrd:--------->Sales Price :--------->(0.5337231555820281, 2.772280932958432e-108)
correlation of column  Fireplaces:--------->Sales Price :--------->(0.46692883675152697, 6.141487461123593e-80)
correlation of column  GarageCars:--------->Sales Price :--------->(0.6404091972583519, 2.4986441671800782e-169)
correlation of column  GarageArea:--------->Sales Price :--------->(0.6234314389183623, 5.265038167974214e-158)
correlation of column  WoodDeckSF:--------->Sales Price :--------->(0.32441344456813015, 3.972216535629091e-37)
correlation of column  OpenPorchSF:--------->Sales Price :--------->(0.3158562271160554, 3.493373623096983e-35)
correlation of column  EnclosedPorch:--------->Sales Price :--------->(-0.12857795792595642, 8.255770475118623e-07)
correlation of column  3SsnPorch:--------->Sales Price :--------->(0.04458366533574851, 0.08858170358062778)
correlation of column  ScreenPorch:--------->Sales Price :--------->(0.11144657114291115, 1.972140019470224e-05)
correlation of column  PoolArea:--------->Sales Price :--------->(0.09240354949187318, 0.0004073489601198664)
correlation of column  MiscVal:--------->Sales Price :--------->(-0.021189579640303314, 0.4184863494083354)
correlation of column  MoSold:--------->Sales Price :--------->(0.04643224522381937, 0.0761275785060908)
correlation of column  YrSold:--------->Sales Price :--------->(-0.028922585168730388, 0.26941319328103863)
correlation of column  SalePrice:--------->Sales Price :--------->(1.0, 0.0)
```

Figure 2: Pearson Correlation Analysis results

```
regression test results:
Feature 0: 7.778938
Feature 1: 70.059502
Feature 2: 59.519614
Feature 3: 611.442092
Feature 4: 594.933467
Feature 5: 103.914795
Feature 6: 4.328238
Feature 7: 962.986439
Feature 8: 55.565212
Feature 9: 0.000448
Feature 10: 419.834954
Feature 11: 93.337287
Feature 12: 18.861649
Feature 13: 24.533020
Feature 14: 315.681762
Feature 15: 319.707826
Feature 16: 650.575510
Feature 17: 649.292951
Feature 18: 134.132223
Feature 19: 92.471965
Feature 20: 15.983062
Feature 21: 2.584612
Feature 22: 9.942398
Feature 23: 14.649498
Feature 24: 0.974034
Feature 25: 0.798349
Feature 26: 0.845385
```

Figure 3: SelectKBest test results

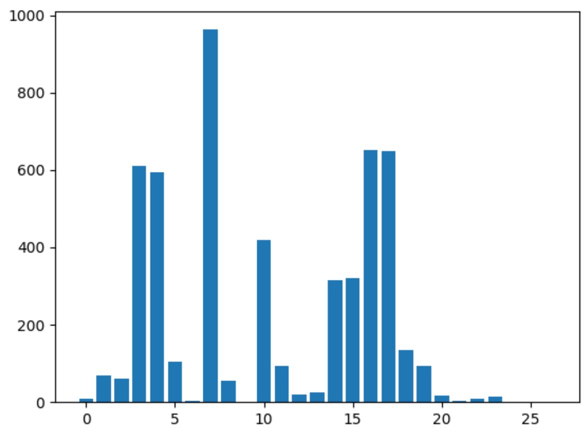


Figure 4: SelectKBest test result (B)

In snapshots shown earlier the higher score indicates higher correlation between attributes.

After performing these correlation tests on numerical features, I observed that all these tests were indicating almost same results.

I selected the following numerical features based on their test scores:

- TotalBsmtSF
- 1stFlrSF
- GrLivArea
- FullBath
- TotRmsAbvGrd
- Fireplaces
- GarageCars
- GarageArea

After performing the set of different analyses, the final attribute selection contained:

- MSZoning
- Utilities
- LotConfig
- LandSlope
- ExterCond
- Heating
- PoolQC
- MiscFeature
- OverallQual
- OverallCond
- TotalBsmtSF
- 1stFlrSF
- GrLivArea
- FullBath
- TotRmsAbvGrd
- Fireplaces
- GarageCars

- GarageArea

These 18 attributes are the final attributes on which the value of price depends. The user will be asked for the values of these 18 attributes.

3.3. MODEL TRAINING AND EVALUATION

In this stage, we apply different algorithms in order to train the model.

We have to note that this is not a classification problem. We are not examining and assigning a class to the instances. But we are providing a continuous attribute value as the output. This makes it a regression problem. So, we have to use algorithms that can perform regression. The algorithms I implemented are:

- Multiple Linear Regression
- Random Forest Regression
- K-Nearest Neighbours based Regression
- ElasticNet Regression
- Lasso Regression

One thing to note is that I performed a 10-fold Cross Validation in all of the above algorithms in order to evaluate the results.

Now we will just analyse shortly how these algorithms:

3.3.1. Multiple Linear Regression

Multiple Linear Regression finds the effect of a set of independent variables on one dependent variable. In our case, the independent variables are all the attributes except price while price is the

```
X_train, X_test, y_train, y_test = train_test_split(df.loc[:, df.columns != 'SalePrice'],
                                                    df['SalePrice'], test_size=0.3, train_size=0.7,
                                                    random_state=np.random.seed(0))
X_trainn = df.loc[:, df.columns != 'SalePrice']
y_trainn = df['SalePrice']

# Regression Model based Multiple Linear Regression Algorithm
lm = linear_model.LinearRegression()
#scores = cross_val_score(lm, X_trainn,y_trainn, scoring='r2', cv= 10)
scores = cross_validate(lm, X_trainn,y_trainn, scoring=('r2','neg_mean_absolute_percentage_error','explained_variance'), cv= 10)
print(mean(scores['test_neg_mean_absolute_percentage_error']))
print(mean(scores['test_r2']))
print(mean(scores['test_explained_variance']))

# fitting model
lm.fit(X_train,y_train)

# making predictions
predict_y = lm(X_test)
```

Figure 5: Training multiple linear regression model

dependant variable. I implemented this algorithm with the help of the Linear Regression library of Python. The snapshot code is as below:

We split the dataset and used 70 percent for training and 30 percent for testing the module, performance metrics of the model are presented in next sections.

3.3.2. Random Forest regression model:

Random Forest Regression uses ensemble learning method for regression. A Random Forest operates by constructing several decision trees during training time and outputting the mean value as the prediction of all the trees. The code snippet is below:

```
#Regression Model based on Random Forest Algorithm
ran_forest = RandomForestRegressor(max_depth=2, random_state=0)
scores = cross_validate(ran_forest, X_trainn, y_trainn,
                        scoring=('r2', 'neg_mean_absolute_percentage_error', 'explained_variance'), cv=10)
print("results for ran_forest")
print(mean(scores['test_neg_mean_absolute_percentage_error']))
print(mean(scores['test_r2']))
print(mean(scores['test_explained_variance']))

# fitting model
ran_forest.fit(X_train,y_train)

# making predictions
predict_y = ran_forest(X_test)
```

Figure 6: Random Forest Regression Model

3.3.3. K-Nearest Neighbours based Regression model:

K-Nearest Neighbours is a popular algorithm used for both classification and regression problems. The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the neighbour points in the training set. At first, I set the value of k to 2, just to analyse the results. Of course it was understood already that the model will not give good results but it was done just for experimenting. After that

I set the value of k to 7 and examined the results. Then lastly, the value of k was set to 10 and the results were far better. The code snippet is attached:

```
# Regression based on k-nearest neighbors
neigh = KNeighborsRegressor(n_neighbors=10)
scores = cross_validate(neigh, X_trainn, y_trainn,
                        scoring=('r2', 'neg_mean_absolute_percentage_error', 'explained_variance'), cv=10)
print("results for k-nearest neighbors")
print(mean(scores['test_neg_mean_absolute_percentage_error']))
print(mean(scores['test_r2']))
print(mean(scores['test_explained_variance']))

# fitting model
neigh.fit(X_train,y_train)

# making predictions
predict_y = neigh(X_test)
```

Figure 7: Training Testing of K-nearest Neighbours Regression Model

3.3.4. ElasticNet Regression Model

ElasticNet Regression is an extension to linear regression algorithm. This algorithm adds regularization penalties to the loss function during training. The code snippet is below:

```
# Regression model based on Elastic Net algorithm
en = ElasticNet(alpha=0.01)
scores = cross_validate(en, X_trainn, y_trainn,
                        scoring=('r2', 'neg_mean_absolute_percentage_error', 'explained_variance'), cv=10)
print("results for Elastic Net")
print(mean(scores['test_neg_mean_absolute_percentage_error']))
print(mean(scores['test_r2']))
print(mean(scores['test_explained_variance']))

# fitting model
en.fit(X_train,y_train)

# making predictions
predict_y = en(X_test)
```

Figure 8: Training Testing of ElasticNet Regression Model

3.3.5. Lasso Regression Model

Lasso algorithm is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The code snippet is below:

```
# Linear  Regression model based on Lasso algorithm
lasso = Lasso(alpha=0.01)
scores = cross_validate(lasso, X_trainn, y_trainn,
                        scoring=('r2', 'neg_mean_absolute_percentage_error', 'explained_variance'), cv=10)
print("results for Lasso")
print(mean(scores['test_neg_mean_absolute_percentage_error']))
print(mean(scores['test_r2']))
print(mean(scores['test_explained_variance']))

# fitting model
lasso.fit(X_train, y_train)

# making predictions
predict_y = lasso(X_test)
```

Figure 9: Training Testing of Lasso Regression Model

3.4. MODEL EVALUATION AND SELECTION

Now that we have trained five different models, it is time to analyse the results.

The common evaluation metrics used for analysing the results of regression models are, co-efficient of determination, mean absolute error, mean squared error, mean absolute percentage error, and explained variance. The evaluation metrics I used in order to evaluate the models are:

- Co-efficient of Determination (R^2) [2]
- Explained Variance[3]
- Mean Absolute Percentage Error[4]

[2] The coefficient of determination is a statistical measurement that examines how differences in one variable can be explained by the difference in a second variable, when predicting the outcome of a given event. So it measures how well the attributes can predict the target variable. Best possible score is 1.0.

[3] Explained variance (also called explained variation) is used to measure the discrepancy between a model and actual data. Higher percentages of explained variance indicates a stronger strength of association. It also means that the model make better predictions. Best possible score is 1.0, lower values are worse.

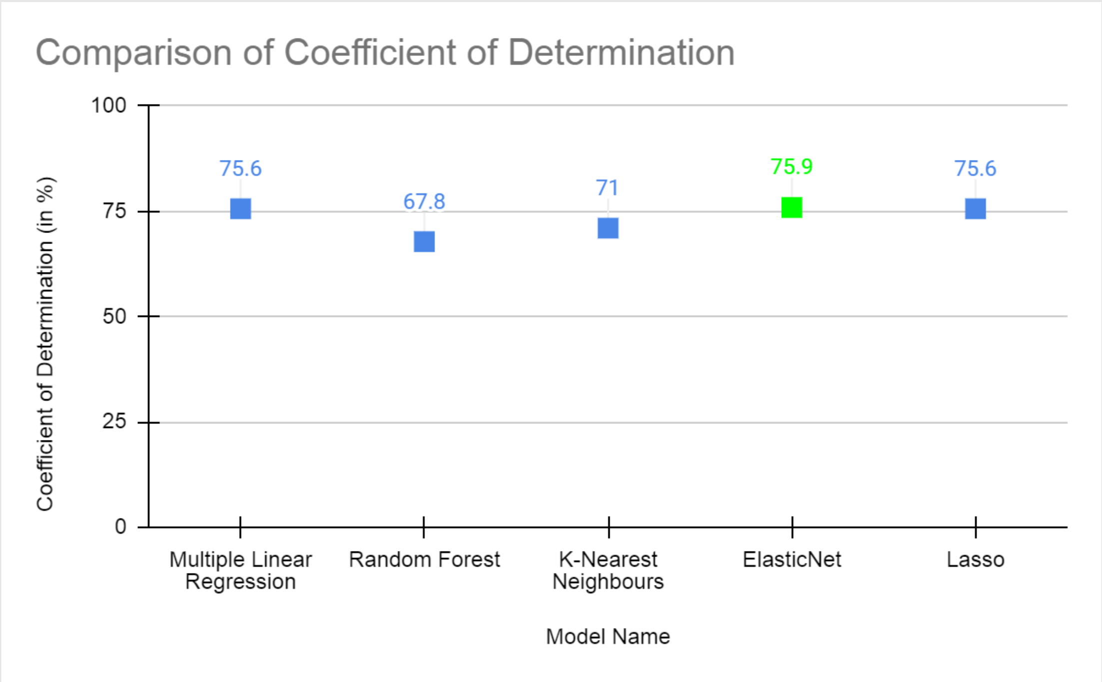
[4] The mean absolute percentage error (MAPE) is the mean of the absolute percentage errors of forecasts. Error is defined as actual or observed value minus the forecasted value. Note here, that we do not represent the output as a percentage in range [0, 100]. Instead, we represent it in range [0, 1]. Best possible value is 0 while worst value is 1.

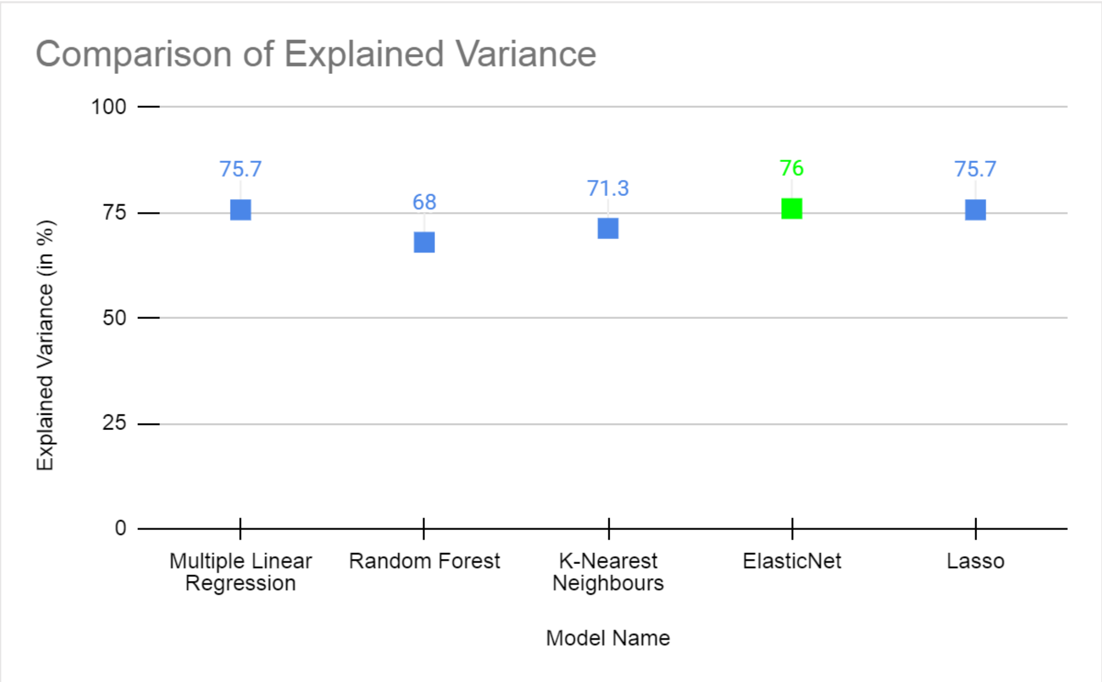
The results are presented as:

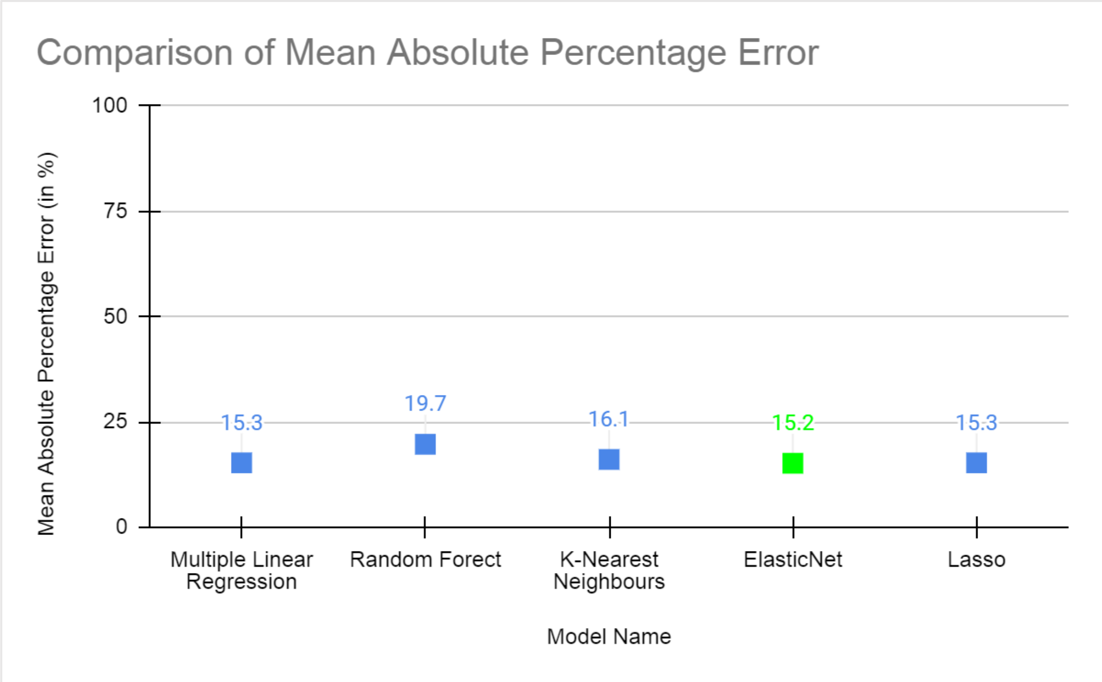
Model	Negative Mean Absolute Percentage Error	Coefficient of Determination	Explained Variance
Multiple Linear Regression Model	-0.153	0.756	0.757
Random Forest Regression Model	-0.197	0.678	0.680
K-nearest Neighbours Regression Model	-0.161	0.710	0.713
ElasticNet Regression Model	-0.152	0.759	0.760
Lasso Linear Regression Model	-0.153	0.756	0.757

Since we performed a 10-fold cross validation on each of the model, so the above values are mean values of all the 10-fold results. For example, at each fold, the model computed explained variance of let us say Lasso model, so we computed the mean of all the 10 explained variance values and then wrote the mean value in the table above. Also notice that, the mean absolute percentage error is negative. This is because the Sklearn library in Python, abides by the convention of "the greater the better", although the metrics depicting error values should be less in order for the model to be better. So the Sklearn library just multiplies the error value by -1 in order to negate the value. In this way, we can just select the model with the highest value in error. For example, -0.153 is greater than -0.197. So, -0.153 depicts 15.3% mean absolute regression error.

For simplicity, let us show the plots with the comparison of different models. In these plots, we will plot all the values on a scale of [0,100], unlike in the table where the values are presented on the scale [0,1]. Also, we will show positive values of the error. In this way we will have to choose the model, with the lowest mean absolute percentage error.


Comparison of Coefficient of Determination
Coefficient of Determination (in %)
100
75
50
25
0
75.6
67.8
71
75.9
75.6
Multiple Linear Regression
Random Forest
K-Nearest Neighbours
ElasticNet
Lasso
Model Name

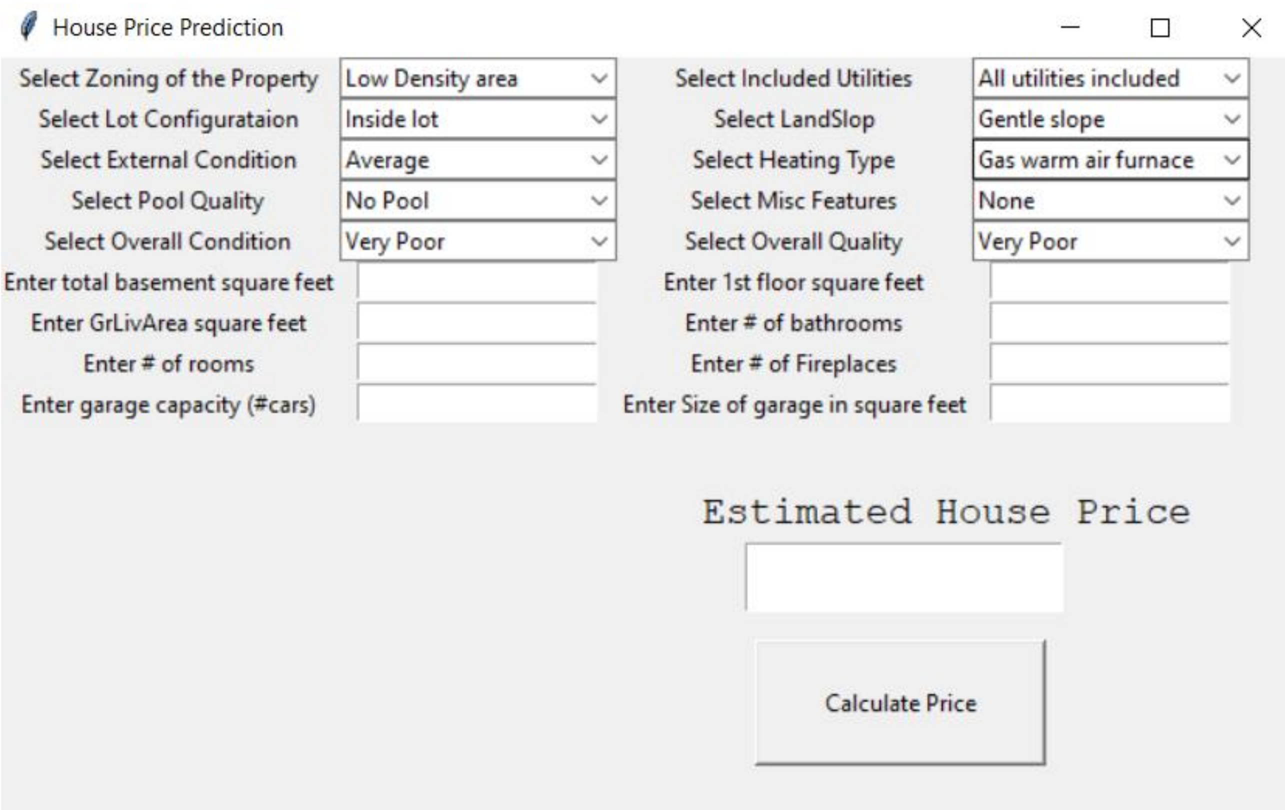

Comparison of Explained Variance
Explained Variance (in %)
100
75
50
25
0
75.7
68
71.3
76
75.7
Multiple Linear Regression
Random Forest
K-Nearest Neighbours
ElasticNet
Lasso
Model Name



By analysing the above charts and table, it is obvious that ElasticNet regression model shows the best results out of all the models. So, I chose this model for my application.

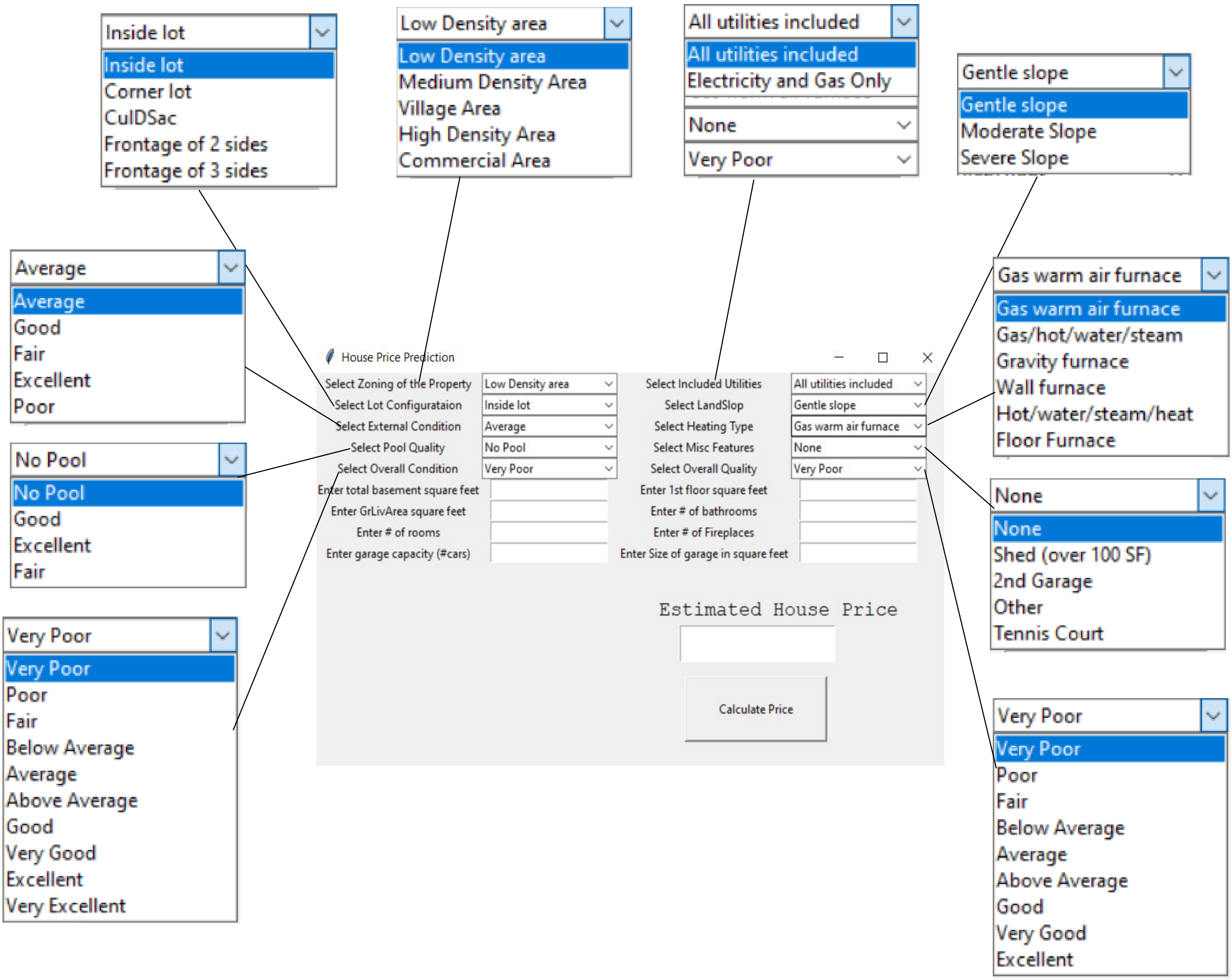
4. USAGE OF APPLICATION

In order to run the application, the admin has to run the preprocessing.py file located in the Preprocessing folder. This file includes all the methods performing data pre-processing, attribute selection, model training and model validation. After the model is trained and saved, the user can use the application. In order to do that, we have to run application.py file present in the Frontend folder. The user will see this screen:

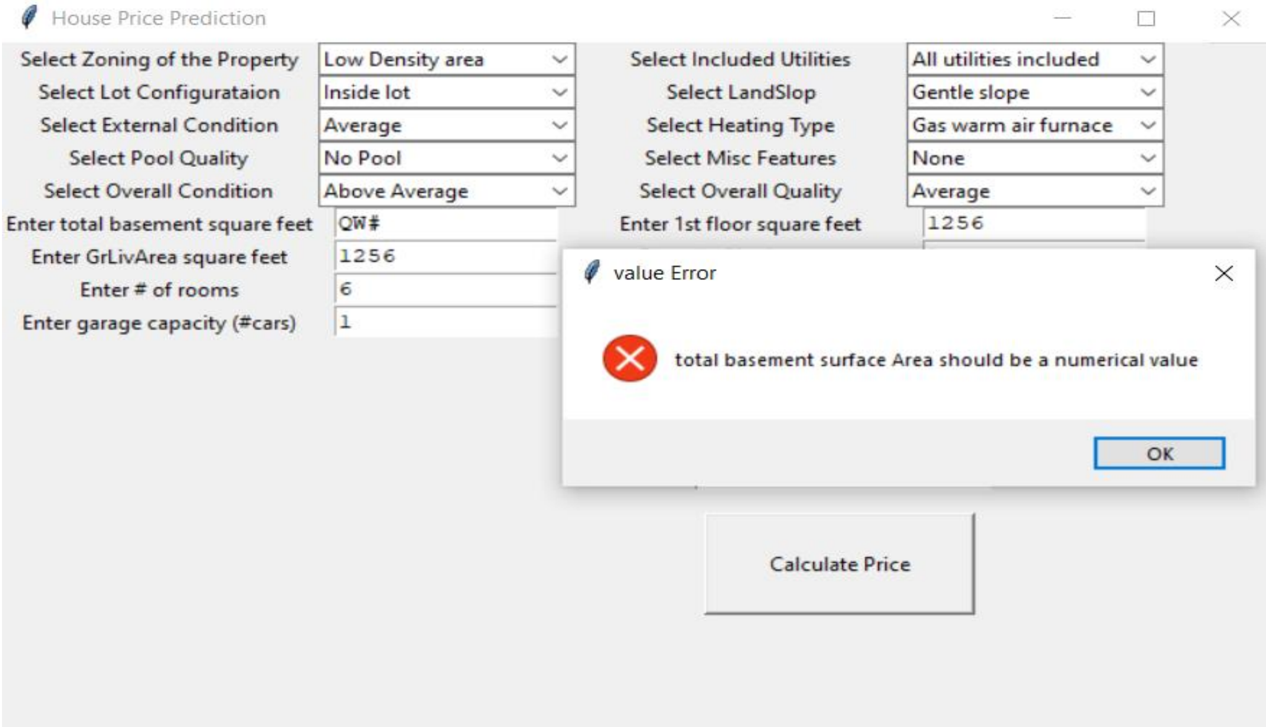


For categorical attributes, I have used drop-down list containing all the possible values for each variable. And for numerical attributes, an input text box which accepts integers only. The user has to enter the values of all the attributes in order to calculate the final price of the house.

The next image shows the values inside the drop down.



The application throws an error when a user enters strings in numerical attributes:



Now let us predict a price value by providing inputs to the application. The output is: