HOUSING: PRICE PREDICTION

Submitted by:

Milind Kuwar

ACKNOWLEDGMENT

The project would not have been built without the constant support from DataTrained and Fliprobo teams.

References:

1. Notes and classes by DataTrained Academy.

INTRODUCTION

- ## Business Problem Framing

Problem Definition:-

To predict the actual value of the prospective properties and decide whether to invest in them or not with the available independent variables.
Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy.
It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies.
A US-based housing company named Surprise Housing has
uses data analytics to purchase houses at a price below their actual values and flip them at a higher price.

- ## Conceptual Background of the Domain Problem

To build model of the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables.
They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

- # Review of Literature

  - **variables are important to predict the price of houses are below**–

    OverallQual, GrLivArea, GarageCars, GarageArea, TotalBsmtSF, 1stFlrSF, FullBath, TotRmsAbvGrd, YearBuilt.

  - **variables are positive correlated with price of houses are below –**

    LotFrontage, LotArea, YearRemodAdd, MasVnrArea, BsmtFinSF1, BsmtUnfSF, 2ndFlrSF, BsmtFullBath, HalfBath, BedroomAbvGr, Fireplaces, GarageYrBlt, WoodDeckSF **,** OpenPorchSF, 3SsnPorch, ScreenPorch, PoolArea, MoSold.

  - **variables are negatively correlated with price of houses are below-**

    ID, MSSubClass, OverallCond, BsmtFinSF2, LowQualFinSF, BsmtHalfBath, KitchenAbvGr, EnclosedPorch, MiscVal, YrSold.

- # Motivation for the Problem Undertaken

The motivation for this project is houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy.

It is a very large market and there are various companies working in the domain. To solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases.

The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  Methodology represents a description about the framework that is undertaken. It consists of various milestones that need to be achieved in order to fulfil the objective. We have undertaken different data mining and machine learning concepts.

  I have check the statistical correlation of dataset features:-

  OverallQual, GrLivArea, GarageCars, GarageArea, TotalBsmtSF, 1stFlrSF, FullBath, TotRmsAbvGrd, YearBuilt, LotFrontage, LotArea, YearRemodAdd, MasVnrArea, BsmtFinSF1, BsmtUnfSF, 2ndFlrSF, BsmtFullBath, HalfBath, BedroomAbvGr, Fireplaces, GarageYrBlt, WoodDeckSF , OpenPorchSF, 3SsnPorch, ScreenPorch, PoolArea, MoSold are positively correlated.

  I have checked the statistical description of dataset: - Outliers presents in dataset and removed by IQR method.

- ## Data Sources and their formats

  Data provided by client in csv format.

  A US-based housing company named Surprise Housing has decided to enter the australian market. The company has collected a data set from the sale of houses in australia.

  We have a Dataset of 1460 entries and it has 81 variables. I have two datasets: train.csv (1168 records) test.csv (292 records). Data Count : 1460 Data Features : 81 Data Types : int64, float64, object

  - **Data variables:**

    MSSubClass: Identifies the type of dwelling involved in the sale.

    MSZoning: Identifies the general zoning classification of the sale.

    LotFrontage: Linear feet of street connected to property.

LotArea: Lot size in square feet.

Street: Type of road access to property.

Alley: Type of alley access to property.

LotShape: General shape of property.

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 various conditions.

Condition2: Proximity to various conditions (if more than one is present).

BldgType: Type of dwelling.

HouseStyle: Style of dwelling.

OverallQual: Rates the overall material and finish of the house.

OverallCond: Rates the overall condition of the house.

YearBuilt: Original construction date.

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions).

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: Evaluates the quality of the material on the exterior.

Foundation: Type of foundation.

BsmtQual: Evaluates the height of the basement.

BsmtCond: Evaluates the general condition of the basement

BsmtExposure: Refers to walkout or garden level walls.

BsmtExposure: Refers to walkout or garden level walls.

BsmtExposure: Refers to walkout or garden level walls.

BsmtFinType2: Rating of basement finished area (if multiple types).

BsmtFinType2: Rating of basement finished area (if multiple types).

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.

BsmtFullBath: Basement full bathrooms.

BsmtHalfBath: Basement half bathrooms.

FullBath: Full bathrooms above grade.

HalfBath: Half baths above grade.

Bedroom: Bedrooms above grade (does NOT include basement bedrooms).

Kitchen: Kitchens above grade.

KitchenQual: Kitchen quality.

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms).

Functional: Home functionality (Assume typical unless deductions are warranted).

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.

3SsnPorch: Three season porch area in square feet.

3SsnPorch: Three season porch area in square feet.

3SsnPorch: Three season porch 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 (MM)

YrSold: Year Sold (YYYY).

SaleType: Type of sale.

SaleCondition: Condition of sale.

```
#Adding source columns to both train and test data with 'train' and 'test' values
train_data['source'] = 'train'
test_data['source'] = 'test'
df = pd.concat([train_data,test_data],ignore_index = True) #Combining both train and test data
df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norn
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norn
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norn
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norn
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norn
5	1197	60	RL	58.0	14054	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norn
6	561	20	RL	NaN	11341	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Sawyer	Norn
7	1041	20	RL	88.0	13125	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	Sawyer	Norn
8	503	20	RL	70.0	9170	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	Edwards	Feed
9	576	50	RL	80.0	8480	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norn
10	449	50	RM	50.0	8600	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	IDOTRR	Norn

Combining both test and train data

```
df.shape
```

```
(1460, 82)
```

- **Target Variable:**

  Selling price is a dependent variable on several other independent variables. After we've trained a model on train.csv data, we'll make predictions using the test.csv data. We'll simply recount our primary observations that will inform our feature engineering. We have 43 categorical attributes which we will have to convert into dummy variables.

- **Data Preprocessing**

  Data pre-processing is a process of transforming the raw, complex data into systematic understandable knowledge. It involves the process of finding out missing and redundant data in the dataset. Entire dataset is checked for Nan and whichever observation consists of Nan was replaced with appropriate values. Thus, this brings uniformity in the dataset. Before applying any model to our dataset, we need to find out characteristics of our dataset. Thus, we need to analyse our dataset and study the different parameters and relationship between these parameters.

  - Here both types of variables present in dataset categorical as well as numerical.
  - In the data set, there are NaN value present in some of columns, so I have imputed that by fillna() method.
  - I have check the correlation between features, plot the graphs between input variables and target variable by using matplotlib library(i.e count plot, line plot, box plot, bar plot) and check the relationship between variables.
  - As per analysis, I have decided to drop some columns like LowQualFinSF, Id, BsmtFinType2, YrSold.
  - Converted categorical features by get_dummies method.
  - Removed outliers by IQR quantile method.
  - StandardScaler is used for scaling data.

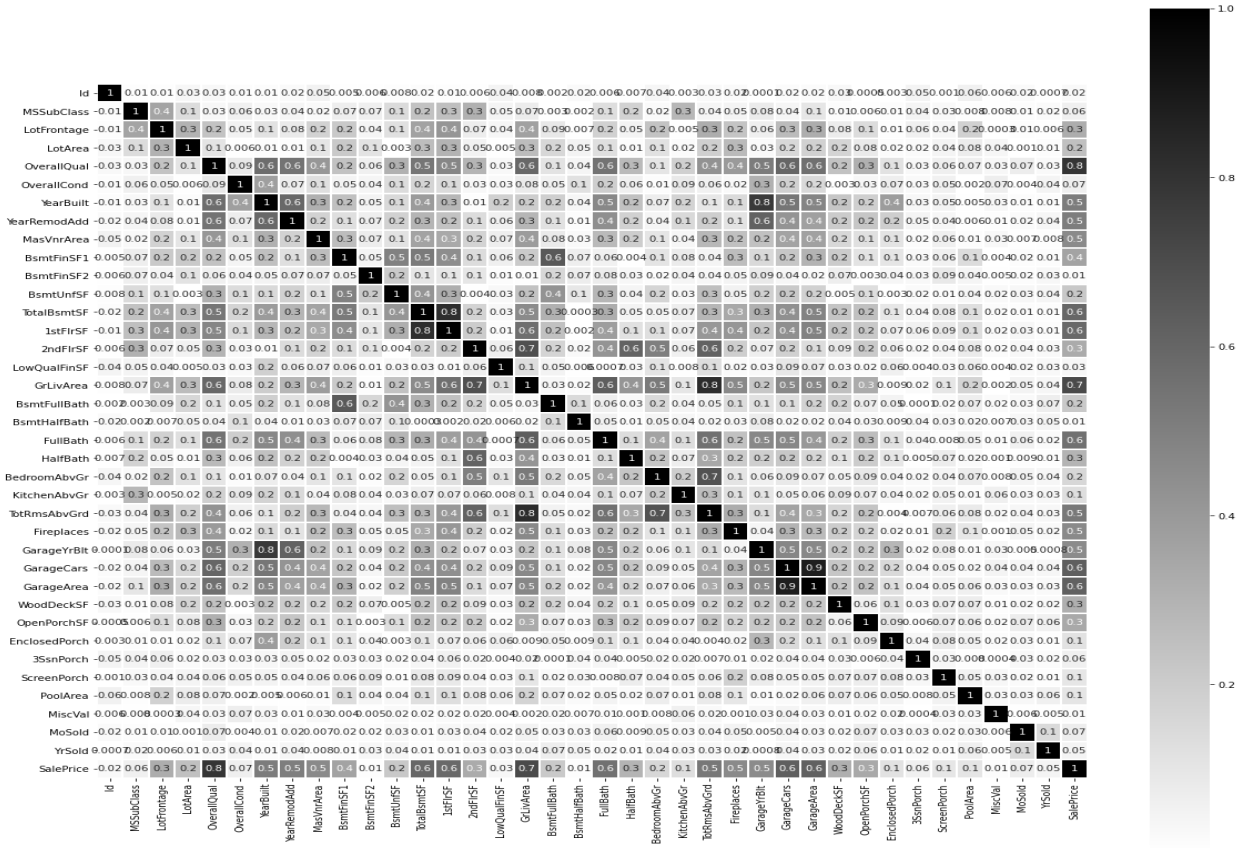
- **Handling Null/Missing Values:**

  Instead of dropping the null values which will result in a data loss, we will impute the null values according to the domain understanding and the data dictionary provided with the data.

```
median = df[['LotFrontage','GarageYrBlt','MasVnrArea']]

mode = df[['BsmtQual','BsmtCond', 'BsmtExposure', 'BsmtFinType1','BsmtFinType2','GarageType','GarageFinish','GarageQual','Ga

for i in median:
    medianvalue = df[i].median()
    df[i].fillna(medianvalue, inplace = True)

for i in mode:
    modevalue = df[i].mode()[0]
    df[i].fillna(modevalue, inplace = True)
```

- **Feature selection:**

  Used correlation method for feature selection.



Some columns are having high correlation in-between.

- LotFrontage
- Overall Quality
- Year Built
- Year removeadd
- MasVnrArea
- TotalBsmn SF
- 1st Floor SF
- GrLiving Area
- Fullbath
- Fireplaces

➢ **Encoding:**

As the entire data should be in the numerical format for giving in model The Label Encoder method was applied on data to convert the categorical data types to numerical data type.

```
#Encoding the categorical data
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
for i in Categorical_df:
    df[i] = le.fit_transform(df[i])
df.head()

```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2
0	127	120	3	69.0	4928	1	0	3	0	4	0	13	2	2
1	889	20	3	95.0	15865	1	0	3	0	4	1	12	2	2
2	793	60	3	92.0	9920	1	0	3	0	1	0	15	2	2
3	110	20	3	105.0	11751	1	0	3	0	4	0	14	2	2
4	422	20	3	69.0	16635	1	0	3	0	2	0	14	2	2

- ## Data Inputs- Logic- Output Relationships

  Input data feature are both categorical and numerical also and target is in continuous format and hence regression model best suits for this dataset.

- ## State the set of assumptions (if any) related to the problem under consideration

  I have not consider any pre-assumptions, project performance from beginning to end is based on data facts only.

- ## Hardware and Software Requirements and Tools Used

  - **Hardware Requirement-**Laptop with below configurations-

    Windows Edition-Windows 10

    Processor-Intel i3

    Memory-4 GB

    System Type-64 bit OS

  - **Software Requirement-**Anaconda 3.7 & above , Jupiter Notebook 6.

Model/s Development and Evaluation

- **Analytical Approach –** I have check the correlation between features, univariate analysis, bivariate analysis and based on type of data by performing EDA I have decided which model to be used for this data.

- **Statistical Approach–**Data should be in scaled manner, it should not be distorted, for that I have replace all null values using fillna() (mean, mode, median) method for different features. Removed outliers from data by IQR method.

- **Testing of Identified Approaches (Algorithms)**

  - **Supervised Learning methods:-**
    - Linear Regression
    - Decision Tree Regressor
    - KNN Regressor
    - Random Forest Regressor

  - **Regularization techniques:-**
    - Ridge model
    - Lasso Model

  - **Ensemble techniques:-**
    - XGBoost Regressor
    - GradientBoosting Regressor

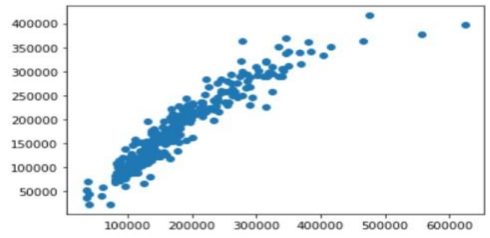
- **Run and Evaluate selected models**

➢ Linear Regression:

```
#Applying linear regression
lr = LinearRegression()
lr.fit(x_train, y_train)
predlr = lr.predict(x_test)
print('R2_Score of Linear regression:',r2_score(y_test,predlr))
print('MAE:', metrics.mean_absolute_error(y_test, predlr))
print('MSE:', metrics.mean_squared_error(y_test, predlr))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predlr)))
plt.scatter(x=y_test,y=predlr)
```

```
R2_Score of Linear regression: 0.8722301402949886
MAE: 20624.76779511629
MSE: 933536546.7952521
RMSE: 30553.830312994345
```

```
<matplotlib.collections.PathCollection at 0x1f5d531f5b0>
```

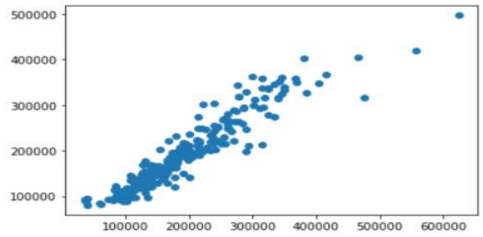


➢ Random Forest Regressor:

```
#Applying RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(x_train,y_train)
predrf = rf.predict(x_test)
RFR = r2_score(y_test,predrf)
print('R2_score of Random Forest:',RFR)
print('MAE:', metrics.mean_absolute_error(y_test, predrf))
print('MSE:', metrics.mean_squared_error(y_test, predrf))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predrf)))
plt.scatter(x=y_test,y=predrf)
```

```
R2_score of Random Forest: 0.8909750092015639
MAE: 18513.232739726027
MSE: 796579206.233285
RMSE: 28223.734803056894
```

```
<matplotlib.collections.PathCollection at 0x1f5d5c793d0>
```

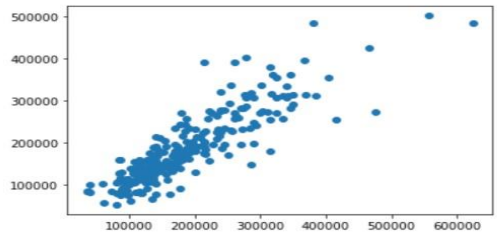


➢ Decision Tree Regressor

```
#Applying DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(x_train,y_train)
preddt = dt.predict(x_test)
DTR = r2_score(y_test,preddt)
print('R2_score of DT:',DTR)
print('MAE:', metrics.mean_absolute_error(y_test, preddt))
print('MSE:', metrics.mean_squared_error(y_test, preddt))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, preddt)))
plt.scatter(x=y_test,y=preddt)
```

```
R2_score of DT: 0.7633512845196915
MAE: 28882.075342465752
MSE: 1729048033.4178083
RMSE: 41581.82335369396
```

```
<matplotlib.collections.PathCollection at 0x1f5daa22c10>
```

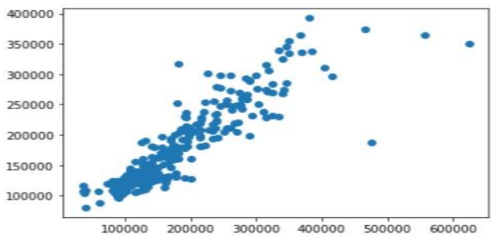


➢ KNN Regressor:

```
#Applying KNeighborsRegressor
KNN = KNeighborsRegressor(n_neighbors = 5)
KNN.fit(x_train,y_train)
predknn = KNN.predict(x_test)
KNNR = r2_score(y_test,predknn)
print('R2_score of KNN :',KNNR)
print('MAE:', metrics.mean_absolute_error(y_test, predknn))
print('MSE:', metrics.mean_squared_error(y_test, predknn))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predknn)))
plt.scatter(x=y_test,y=predknn)
```

```
R2_score of KNN : 0.7738991448203191
MAE: 24741.791780821917
MSE: 1651981242.3619175
RMSE: 40644.57211439084
```

```
<matplotlib.collections.PathCollection at 0x1f5d525cbe0>
```



➢ Boosting Technique:

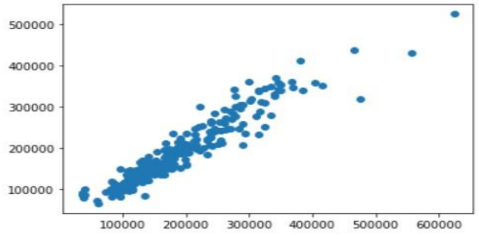
Boosting algorithms represent a different machine learning perspective: turning a weak model to a stronger one to fix its weaknesses. Underlying engine used for boosting algorithms can be anything. It can be decision stamp, margin-maximizing classification algorithm etc. There are many boosting algorithms which use other types of engine such as:

1. Gradient Tree Boosting
2. XGBoost

➢ GradientBoosting Regressor:

```
from sklearn.ensemble import GradientBoostingRegressor
GB=GradientBoostingRegressor()
GB.fit(x_train,y_train)
predgb = GB.predict(x_test)
GBR = r2_score(y_test,predgb)
print('R2_score of GB :',GBR)
print('MAE:', metrics.mean_absolute_error(y_test, predgb))
print('MSE:', metrics.mean_squared_error(y_test, predgb))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predgb)))
plt.scatter(x=y_test,y=predgb)

```

```
R2_score of GB : 0.9026404189248947
MAE: 18362.544103143187
MSE: 711347162.1877451
RMSE: 26671.09225711885
<matplotlib.collections.PathCollection at 0x1f5d5dfc2e0>
```



➢ Regularization:

The alpha parameter in ridge and lasso regularizes the regression model. The regression algorithms with regularization differ from linear regression in that they try to penalize those features that are not significant in our prediction. Ridge will try to reduce their effects (i.e., shrink their coefficients) in order to optimize all the input features. Lasso will try to remove the not-significant features by making their coefficients zero. In short, Lasso (L1 regularization) can eliminate the

not-significant features, thus performing feature selection while Ridge (L2 regularization) cannot.

➢ Ridge model & Lasso Model:

```
x_train,x_test,y_train,y_test = train_test_split(new_x,y,test_size = 0.25,random_state = maxRS)
```

```
from sklearn.linear_model import Ridge

ridge_reg = Ridge(alpha=1,solver='cholesky')

ridge_reg.fit(x_train,y_train)
p_ridge=ridge_reg.predict(x_train)
p_ridge_test=ridge_reg.predict(x_test)
r2_score(y_test,p_ridge_test)
```

```
0.8722386984141501
```

```
from sklearn.linear_model import Lasso

lasso_reg = Lasso(alpha=0.2)

lasso_reg.fit(x_train,y_train)
p_lasso=lasso_reg.predict(x_train)
p_lasso_test=lasso_reg.predict(x_test)
r2_score(y_test,p_lasso_test)
```

```
0.8722311389658923
```

• Metrics used:

The various metrics used to evaluate the results of the prediction are:

➢ Mean Squared Error:

MSE or Mean Squared Error is one of the most preferred metrics for regression tasks. It is simply the average of the squared difference between the target value and the value predicted by the regression model. As it squares the differences, it penalizes even a small error which leads to over-estimation of how bad the model is. It is preferred more than other metrics because it is differentiable and hence can be optimized better.

➢ Root Mean Squared Error:

RMSE is the most widely used metric for regression tasks and is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a

high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

- **Mean Absolute Error:**

  MAE is the absolute difference between the target value and the value predicted by the model. The MAE is more robust to outliers and does not penalize the errors as extremely as mse. MAE is a linear score which means all the individual differences are weighted equally. It is not suitable for applications where you want to pay more attention to the outliers.

- **R Squared (R2):**

  R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform. In contrast, MAE and MSE depend on the context as we have seen whereas the R2 score is independent of context. So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how must regression line is better than a mean line. Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.

CONCLUSION :

In this paper, we have used machine learning algorithms to predict the house prices. We have mentioned the step by step procedure to analyse the dataset and finding the correlation between the parameters. Thus we can select the parameters which are not correlated to each other and are independent in nature. These feature set were then given as an input to nine algorithms and a csv file was generated consisting of predicted house prices. Hence we calculated the performance of each model using R2 metric and compared them. As a recommendation, I advise to use this model (or a version of it trained with more recent data) by people who want to buy a house in the area covered by the dataset to have an idea about the actual price. The model can be used also with datasets that cover different cities and areas provided that they contain the same features. I also suggest that people take into consideration the features that were deemed as most important as seen in the previous section; this might help them estimate the house price better