HOUSING PRICE PREDICTION PROJECT

Submitted by:

Abhay Surma

ACKNOWLEDGMENT

I express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project on Housing Price prediction using machine learning algorithms. I would also like to thank Flip Robo Technologies for providing me with the requisite datasets to work with.

I acknowledge my indebtedness to the authors of papers titled: “House Price Prediction using a Machine Learning Model: A Survey of Literature” and “The impact of housing quality on house prices in eight capital cities, Australia” for providing me with invaluable insights and knowledge of the dynamic relationships that exist in the economics of real estate and housing markets.

INTRODUCTION

Business Problem Framing

Houses are one of the necessary needs 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. Our problem is related to one such housing company. A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective

properties and decide whether to invest in them or not. For this company wants to know:

Conceptual Background of the Domain Problem

Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Hedonic Characteristics of Housing Price: A Hedonic approach is preferred for predicting the sale prices in the housing market because the market displays resilience, flexibility and spatial fixity. Housing Attributes: Studying the structural, locational, and economic attributes of housing properties is crucial in understanding their mutually inclusive relationships with their pricing.

Review of Literature

2 research papers, namely: "House Price Prediction using a Machine Learning Model: A Survey of Literature" and "The impact of housing quality on house prices in eight capital cities, Australia" were reviewed and evaluated to gain insights into all the attributes that influence the price of house. From studying the papers and analysing the research work it, is learnt that locational attributes and structural attributes are prominent factors in predicting house prices. Studies suggest that there exists a close relationship between House pricing and locational attributes such as distance from the closest shopping center, train station, position offering views of hills or shore, the neighborhood in which the property is situated etc. Structural attributes of the house like lot size, lot shape, quality and condition of the house, garage capacity, rooms, Lot frontage, number of bedrooms, bathrooms, overall finishing of the house etc play a big role in influencing the house price. Neighbourhood qualities can be included in deciding house price. Factors like efficiency of public education, community social status, the socio-cultural demographics

improve the worth of a property. The demand side of the housing market is also a necessary component. Although population growth is widely known as a driver in housing demand, the key issue lies in the proportion of people with abundant financial resources. Variables representing land value such as rents and material costs also demonstrate their influence in explaining house prices, which are positively related to housing prices. Multiple regression analysis models allow to ascertain price predictions by capturing independent and dependent variable data. In Using multiple regression modelling techniques, we can describe changes brought to a dependent variable with changes in the independent variables. In this research, various models were built in which the house Sale Price is projected as separate and dependent variable while locational, structural and various other attributes of housing properties were treated as independent variables. Therefore, the house price is set as a target or dependency variable, while other attributes are set as independent variables to determine the main variables by identifying the correlation coefficient of each attribute.

Motivation for the Problem Undertaken

There is a steady rise in house demand with every passing year, and consequently the house prices are rising every year. The problem arises when there are numerous variables such as location and property demand that influence the pricing. Therefore, buyers, sellers, developers and the real estate industry are keen to know the most important factors influencing the house price to help investors make sound decisions and help house builders set the optimal house price. There are many benefits that home buyers, property investors, and house builders can reap from the house-price model. This model aims to serve as a repository of such information and gainful insights to home buyers, property investors and house builders, that will help them determine best house prices. This model can be useful for potential buyers in deciding the characteristics of a house they want that best fits their budget and will be of tremendous benefit, especially to housing developers and researchers, to ascertain the most significant attributes to determine house prices and to

acknowledge the best machine learning model to be used to conduct a study in this field.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Various Regression analysis techniques were used to build predictive models to understand the relationships that exist between Housing sales prices and various Housing property attributes. The Regression analysis models were used to predict the Sale price value for changes in Housing property attributes. Regression modelling techniques were used in this Problem since Sales Price data distribution is continuous in nature. In order to forecast house price, predictive models such as ridge regression Model, Random Forest Regression model, Decision tree Regression Model, Support Vector Machine Regression model, Extreme Gradient Boost Regression were used to describe how the values of Sale Price depended on the independent variables of various Housing property attributes.

- Data Sources and their formats

  The dataset was compiled by a US-based housing company named Surprise Housing. The company has collected a data set from the sale of houses in Australia. The dataset was made available in .csv file format. There are 2 datasets: One for training the predictive machine learning models and the second one to be used by the models for predicting the Sale Price (target variable)

Importing data

```
data = pd.read_csv('train.csv')
```

```
test_data = pd.read_csv('test.csv')
```

```
data.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Cond
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	

```
data.dtypes
```

```
Id                int64
MSSubClass        int64
MSZoning         object
LotFrontage     float64
LotArea           int64
Street           object
Alley            object
LotShape         object
LandContour      object
Utilities        object
```

First of all, we will load necessary libraries and then will load our HOUSING.csv file

Now we will check the columns and shape of the dataset

Simultaneously with the same process we will check our test data set.

we have 1168 rows and 81 columns in train data set and 292 rows and 80 columns in test data set.

- Data Preprocessing -

  Here we can see that there are many null values present in train dataset. Some columns contains more that 80% of the null values.

```
features_with_na = [features for features in dataset.columns if dataset[features].isnull().sum()>1]
for feature in features_with_na:
  print(feature,np.round(dataset[feature].isnull().mean(),4))
  #printing percentage of missing values
```

```
LotFrontage 0.1832
Alley 0.9341
MasVnrType 0.006
MasVnrArea 0.006
BsmtQual 0.0257
BsmtCond 0.0257
BsmtExposure 0.0265
BsmtFinType1 0.0257
BsmtFinType2 0.0265
FireplaceQu 0.4717
GarageType 0.0548
GarageYrBlt 0.0548
GarageFinish 0.0548
GarageQual 0.0548
GarageCond 0.0548
PoolQC 0.994
Fence 0.7971
MiscFeature 0.9623
```

Observation

i) Nan value can't be imported for columns -

Alley, FireplaceQu, PoolQC, Fence, MiscFeature

ii) other columns which are having less nan value and we can import values for that column by using any imputer technique -

1)LotFrontage

2)MasVnrType

3)MasVnrArea

4)BsmtQual

5)BsmtFinType1

6)BsmtFinType2

7)GarageType

8)GarageYrBlt

9)GarageFinish

10)GarageQual

11)GarageCond

we can impute-

by mean – as it is a continuous data

1) LotFrontage

2) GarageYrBlt

3) MasVnrArea

by mode – as it is discrete data

1) MasVnrType

2) BsmtQual

3) BsmtCond

4) BsmtExposure

5) BsmtFinType1

6) BsmtFinType2

7) GarageType

8) GarageFinish

9) GarageQual

- Data Inputs- Logic- Output Relationships

  Now we will divide the data into input and output, the output will be ‘SalePrice’ and all the other remaining columns will be input.

```
# Splitting Test and Train database.
Train=df1[0:1138]
Test=df1[1138:]
Test=Test.drop(['SalePrice'],axis=1)
```

```
x=Train.drop(['SalePrice'],axis=1)
y=Train["SalePrice"]
```

```
x.tail()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2
1162	30	3	45.00000	8212	1	3	3	0	4	0	7	2	2
1163	20	3	70.98847	9819	1	0	3	0	4	0	19	2	2
1165	160	3	24.00000	2280	1	3	3	0	2	0	13	2	2
1166	70	0	50.00000	8500	1	3	3	0	4	0	9	1	2
1167	60	3	70.98847	7861	1	0	3	0	4	0	8	2	2

```
y.tail()
```

```
1162     58500.0
1163    122000.0
1165    148500.0
1166     40000.0
1167    183200.0
Name: SalePrice, dtype: float64
```

- Hardware and Software Requirements and Tools Used

Hardware: 8GB RAM, 64-bit, 9th gen i7 processor.

Software: Windows 10 Operating System
Anaconda Package and Environment Manager: Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data- science packages suitable for Windows and provides a host of tools and environment for conducting Data Analytical and Scientific works. Anaconda provides all the necessary Python packages and libraries for Machine learning projects. o

Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document. o Python3: It is open source, interpreted, high level language and provides great approach for object-oriented programming. It is one of the best languages used for Data Analytics And Data science projects/application. Python provides numerous libraries to deal with mathematics, statistics and scientific function. o Python Libraries used: o Pandas: For carrying out Data Analysis, Data Manipulation, Data Cleaning etc o Numpy: For performing a variety of operations on the datasets. o matplotlib.pyplot, Seaborn: For visualizing Data and various relationships between Feature and Label Columns o Scipy: For performing operations on the datasets o Statsmodels: For performing statistical analysis o sklearn for Modelling Machine learning algorithms, Data Encoding, Evaluation metrics, Data Transformation,Data Scaling, Component analysis,Feature selection etc

Important Libraries

Importing Necessary libraries

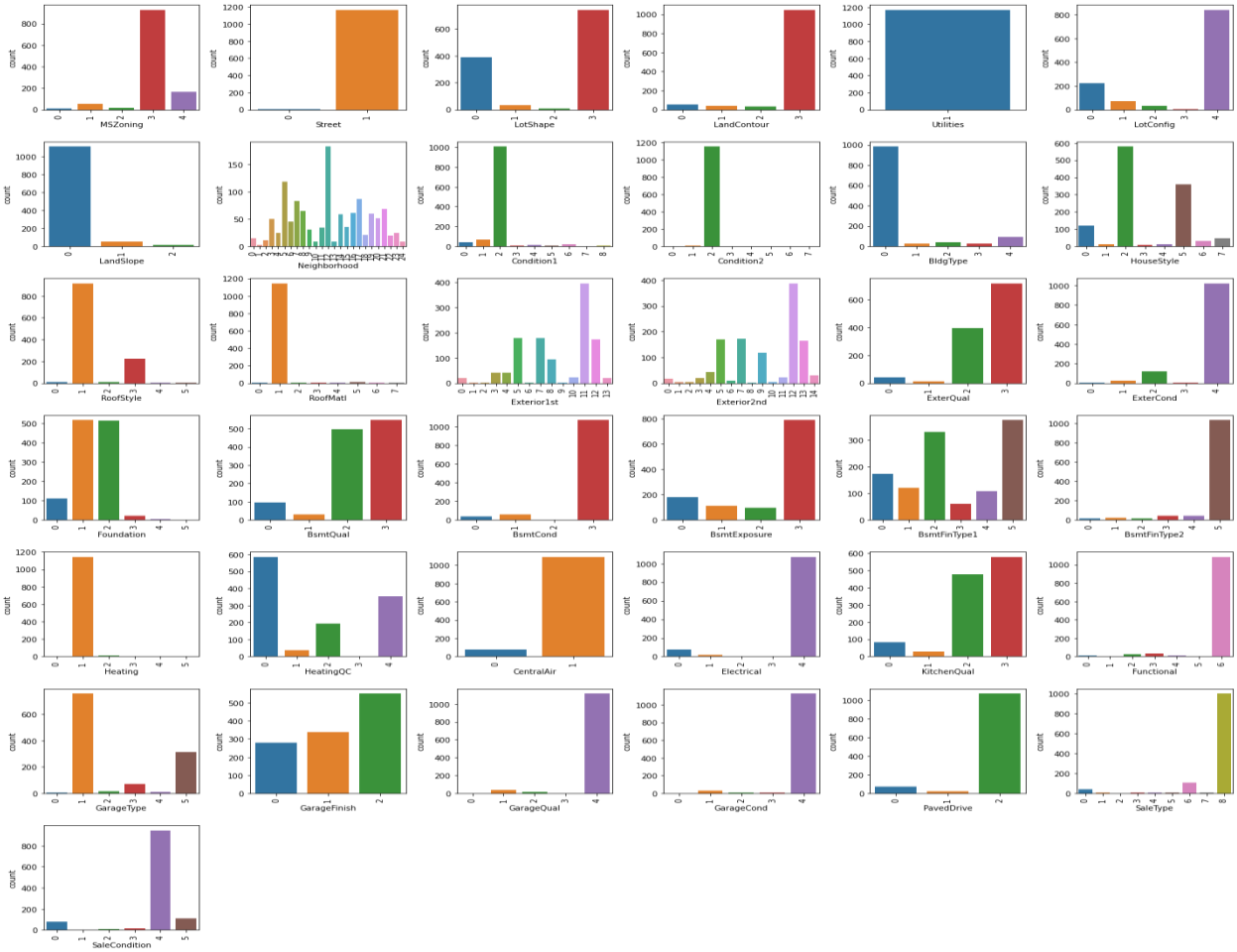
```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.linear_model import LinearRegression
```

Model/s Development and Evaluation

Exploratory Data Analysis:

Visualizations Barplots, Distplots, Boxplots, Countplots, lineplots were used to visualise the data of all the columns and their relationships with Target variable.

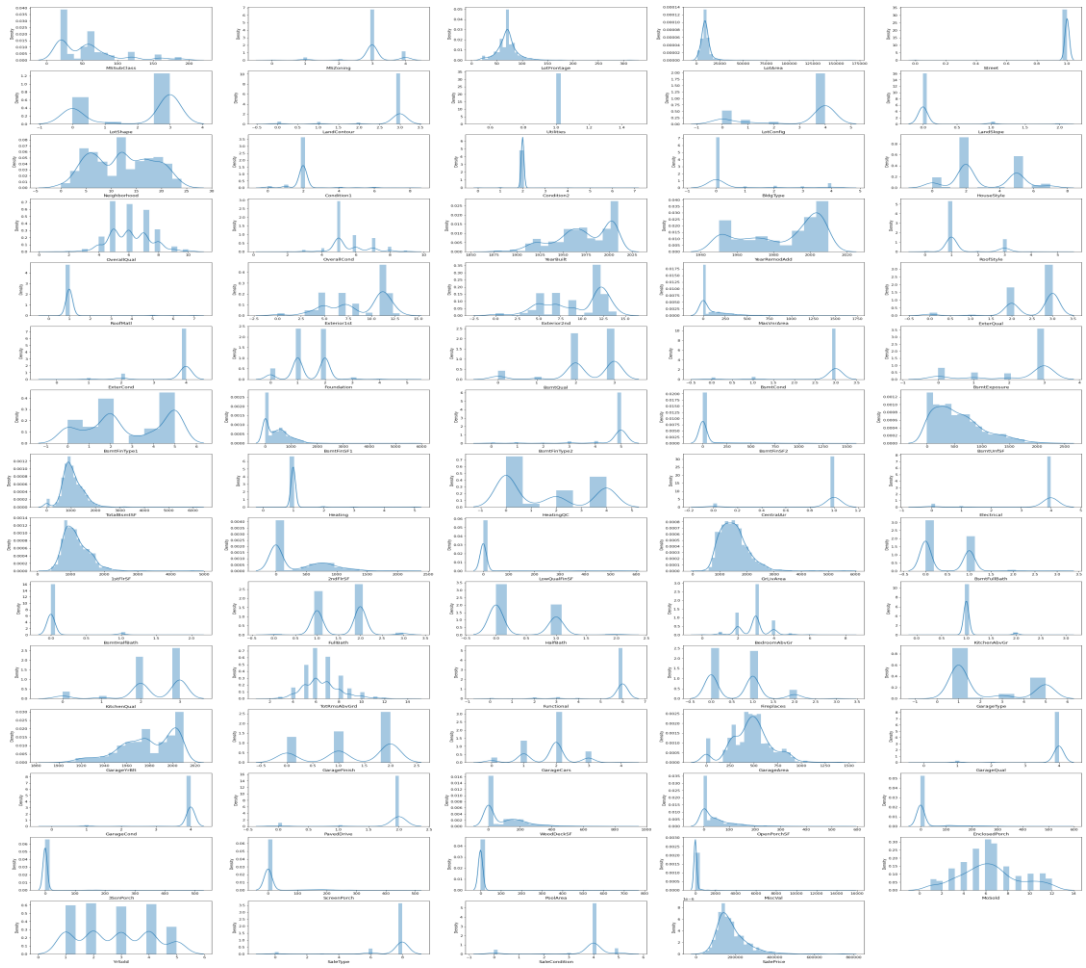
Univariate Analysis:



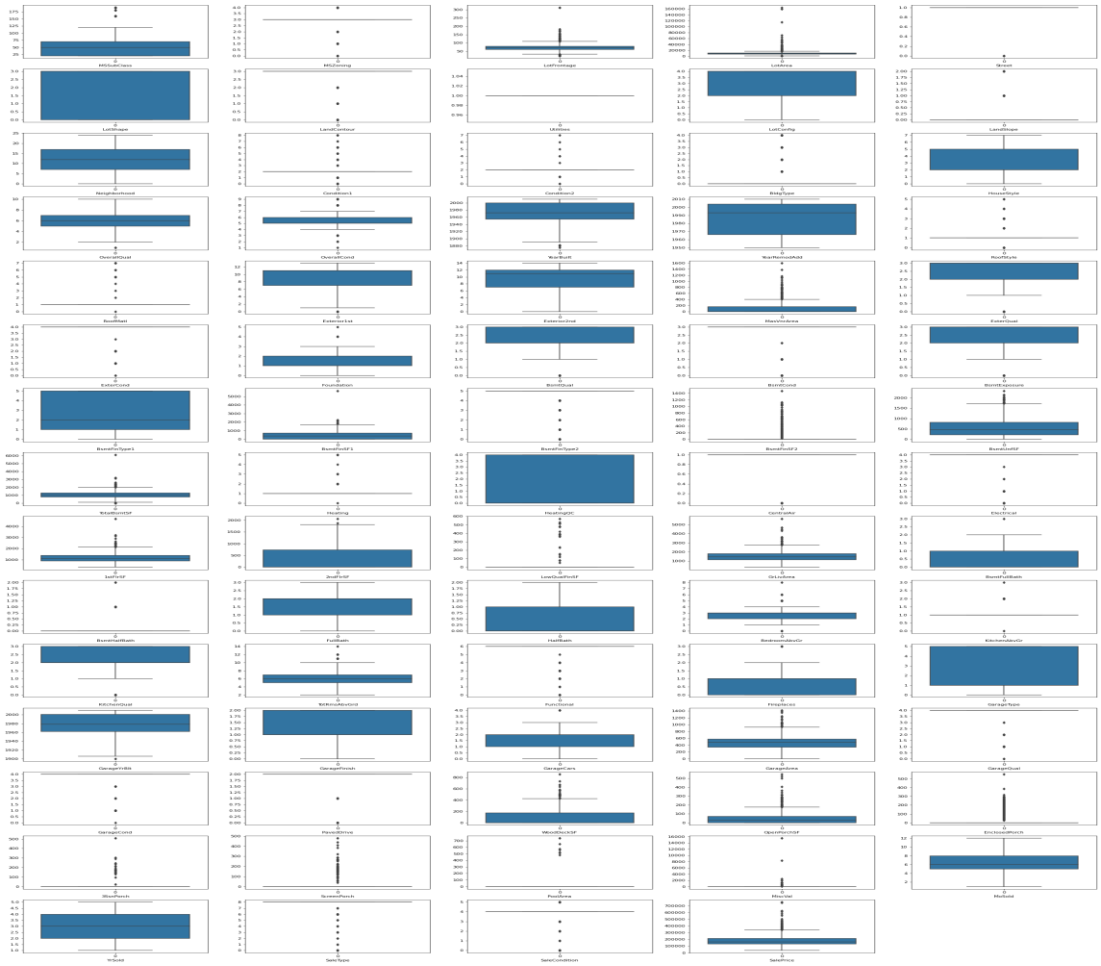
- Density is the most common zoning classification
- Most common Street Type is 'Pave'
- Regular is the most common Lot Shape, followed by Slightly irregular

- Most Properties have Near Flat/Level Land Contour o All public Utilities are available
- Inside lot is the most common Lot configuration o Slope of property land is most commonly gentle
- Most Housing properties are situated in Neighbourhoods of North Ames, followed by College Creek, Edwards and Old Town
- Most Housing properties are in proximity to Normal conditions
- Most Housing properties are of Single-family Detached type
- Most Housing properties 1 storied and 2 storied o Most Houses have Gable roof style

Data distribution to check Skewness,

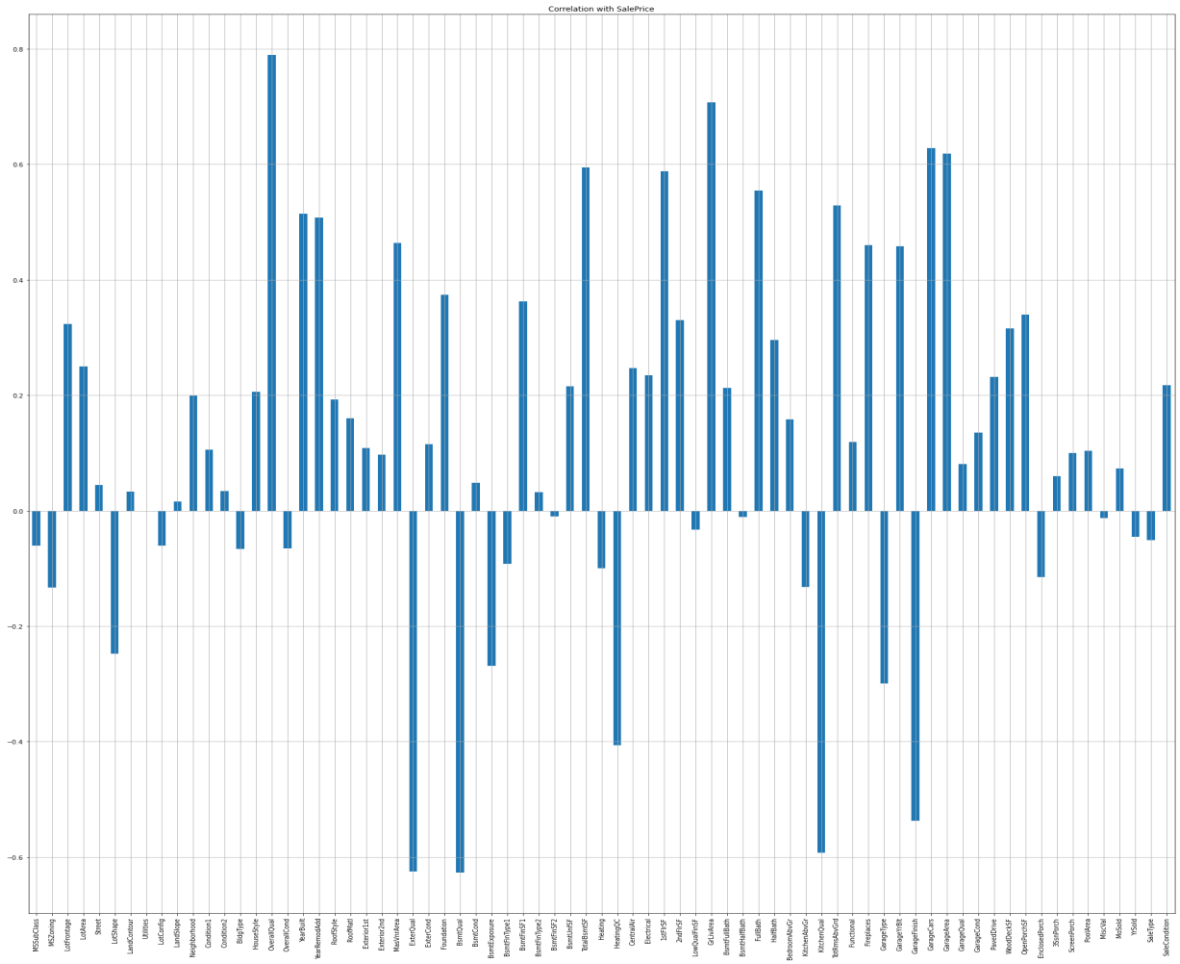


There is so much of skewness its data so we will apply power transformer to correct its skewness.



LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFin SF2, Bs mtUnFSF,TotalBsmtSF,1stFlrSF,2ndFlrSF,GrLivArea,WoodDeck SF, OpenPorchSF, EnclosedPorch are skewed and contain outliers

Correlation plot -



There is a considerable number of outliers in the columns. However, they will not be removed, since we have a very small dataset to work with and removing outliers results in 13.86% of loss in data. Normalizing Data Distribution using PowerTransformer The skewness in Data Distributions of the feature columns was reduced using the Yeo-Johnson Power transformer method

Observation by graphs -

o Saleprice is highest for Floating Village and Low density Residen tial zones

o Saleprice is highest for housing properties near paved streets o Saleprice is highest for irregular lot shapes

o Hill side properties sell for the highest amount o Utilities & Landslope columns don'tshow a strong relationship with Sales Price

o Housing Properties in Northridge, Stone Brook,Northridge Heig hts,Timberland, Somerset,Veenker fetch the highest Sales amount

o Cul-de-sac and 3 sided frontage lot configurations fetch the high est Sales amount

o Proximity to Railroads, Off-site features like parks etc fetch the highest Sales amount

o Townhouse and Single-family Detached are the most valued

o Two story and Two and one-half story: 2nd level finished sell for the highest amount

o Houses with Wood Shingle Roofs sell for the highest amount

o Houses with Exterior covering of Cement Board,Stone,Imitation Stucco sell for the highest amount

o Houses with Stone Mason veneer type sell for the highest amount

o Houses with Excellent exterior material quality sell for the highest amount

o Houses with Excellent exterior material condition sells for the highest amount

o Houses with Poured Concrete and stone foundation types sell f or the highest amount

o Houses with Excellent (100+ inches) height of the basement sell for the highest amount

o Houses with Excellent Basement Condition sell for the highest a mount

o Houses with Good Basement Exposure sell for the highest amount

o Houses with Good and Average Living Quarters in Basement sell for the highest amount

o Houses with Gas forced warm air furnace and Gas hot water he ating systems sell for the highest amount

o Houses with Excellent Heating quality and condition sell for the highest amount

o Houses with Central Air Conditioning sell for the highest amount

o Houses with Standard Circuit Breakers & Romex sell for the highest amount

o Houses with Excellent Kitchen Quality sell for the highest amount

o Houses with Built in Garages sell for the highest amount

o Houses with Good / Typical Garage condition sell for the highest amount

o Houses with Finished Garage sell for the highest amount

o Houses with excellent Garage Quality sell for the highest amount

o Houses with Paved Driveway sell for the highest amount

o Homes just constructed and sold, Contract 15% Down payment regular terms sell for the highest amount

o New Homes (not completed when last assessed) sell for the highest amount

Feature Engineering:

In order to better understand the relationships between age of a housing property and SalePrice, following columns were created based on data of existing columns: House Age data was extracted from YearBuilt, and was placed in new column titled:

House_Age, House Remodeling Age was extracted from YearRemodAdd, and was placed in new column titled: Remod_Age and age of Garage was extracted from GarageYrBuilt nd placed in a new column titled Garage_Age. Encoding the Categorical Columns Before Proceeding with finding the correlations of the columns, The data of the categorical columns needs to be encoded using LabelEncoder

```
features = ['MSZoning', 'Street',
       'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle','RoofStyle',
       'RoofMatl', 'Exterior1st', 'Exterior2nd', 'ExterQual',
       'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure',
       'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical',
       'KitchenQual', 'Functional', 'GarageType', 'GarageFinish', 'GarageQual',
       'GarageCond', 'PavedDrive','SaleType', 'SaleCondition']
```

```
from sklearn.preprocessing import LabelEncoder
labenc = LabelEncoder()
for col in data[data.columns[data.dtypes == 'object']]:
    data[col] = labenc.fit_transform(data[col])
data['YrSold'] = data.YrSold.map({2007:2,2009:4,2006:1, 2008: 3, 2010: 5}) # encoding years in YrSold Column
data['Utilities'] =data.Utilities.map({0:1})
data.dtypes[data.dtypes != 'object']
```

```
MSSubClass         int64
MSZoning           int32
LotFrontage      float64
LotArea            int64
Street             int32
```

Applying Imputing technique – Knn imputer(mean) and simple imputer(mode):

```
from sklearn.impute import KNNImputer
imp = KNNImputer(n_neighbors=3)
data[['LotFrontage']] = imp.fit_transform(data[['LotFrontage']])
data[['LotFrontage']].isnull().sum()
data[['GarageYrBlt']] = imp.fit_transform(data[['GarageYrBlt']])
data[['MasVnrArea']] = imp.fit_transform(data[['MasVnrArea']])
```

```
imp = KNNImputer(n_neighbors=3)
test_data[['LotFrontage']] = imp.fit_transform(test_data[['LotFrontage']])
test_data[['LotFrontage']].isnull().sum()
test_data[['GarageYrBlt']] = imp.fit_transform(test_data[['GarageYrBlt']])
test_data[['MasVnrArea']] = imp.fit_transform(test_data[['MasVnrArea']])
```

imputing mode by simple imputer

```
from sklearn.impute import SimpleImputer
si = SimpleImputer(missing_values = np.nan,strategy = 'most_frequent',verbose = 0 )
si = si.fit(data[['MasVnrType','Electrical','MasVnrArea','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2',
data[['MasVnrType','Electrical','MasVnrArea','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2','GarageType'
data.isnull().sum()
```

```
MSSubClass       0
```

Power Transformer to remove skewness and scaling of data:

```
features=['LotFrontage','LotArea','MasVnrArea','BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','1stFlrSF','2ndFlrSF','Gr
```

```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
data[features]  = pt.fit_transform(data[features].values)
data.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	Bld
0	120	3	0.093658	-1.213954	1	0	3	1	4	0	13	2	2	
1	20	3	1.117135	1.100521	1	0	3	1	4	1	12	2	2	
2	60	3	0.998803	0.158048	1	0	3	1	1	0	15	2	2	
3	20	3	1.495566	0.496002	1	0	3	1	4	0	14	2	2	
4	20	3	0.093658	1.196626	1	0	3	1	2	0	14	2	2	

```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
test_data[features]  = pt.fit_transform(test_data[features].values)
test_data.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	Bld
0	20	2	0.982253	0.842656	1	0	1	1.0	0	0	21	2	0	
1	120	2	0.035790	-0.739104	1	0	3	1.0	1	0	21	2	0	
2	20	2	0.035790	0.524304	1	3	3	1.0	4	0	4	2	0	
3	70	2	0.457389	0.548484	1	3	0	1.0	4	0	5	2	0	

CHECKING MULTI COLINEARITY,

Data scaling and checking multicolinearity value

```
X = data.drop(columns = ['SalePrice'],axis=1)
y = data['SalePrice']
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
X=pd.DataFrame(X_scaled,columns=X.columns)
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
test_scaled=scaler.fit_transform(test_data)
test_data=pd.DataFrame(test_scaled,columns=test_data.columns)
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif"]=[variance_inflation_factor(X_scaled,i) for i in range(X_scaled.shape[1])]
vif["Features"]=X.columns

# lets check the vif
print(vif)
```

```
        vif       Features
0    5.471893     MSSubClass
```

SELECTING K BEST FEATURES:

Using SelectKBest and f_classif for measuring the respective ANOVA f-score values of the columns, the best 70 features were selected. Using

StandardScaler, the features were scaled by resizing the distribution values so that mean of the observed values in each feature column is 0 and standard deviation is 1. From sklearn.model_selection's train_test_split, the data was divided into train and test data. Training data comprised 90% of total data whereas test data comprised 10% based on the best random state that would result in best model accuracy. Inorder to find the best random state for the train and test split, the following code was used:

```
### Selecting K best features
```

```
from sklearn.feature_selection import SelectKBest, f_classif
```

```
bestfeat = SelectKBest(score_func = f_classif, k = 'all')
fit = bestfeat.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

```
fit = bestfeat.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
dfcolumns.head()
featureScores = pd.concat([dfcolumns,dfscores],axis = 1)
featureScores.columns = ['Feature', 'Score']
print(featureScores.nlargest(75,'Score'))
```

```
         Feature     Score
13   OverallQual  5.303071
62       MiscVal  3.564855
22     ExterQual  3.514221
25      BsmtQual  2.876879
45   KitchenQual  2.617125
51    GarageCars  2.578547
41      FullBath  2.435854
52    GarageArea  2.242545
50  GarageFinish  2.187163
15     YearBuilt  2.133300
33   TotalBsmtSF  2.070692
2         Street  1.835751
```

- Testing of Identified Approaches (Algorithms)

  The model algorithms used were as follows:

  - Linear regression (Ridge: Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. Since the features have multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values. Ridge shrinks the parameters. Therefore, it is used to prevent multicollinearity)

● DecisionTreeRegressor: Decision Tree solves the problem of machine learning by transforming the data into a tree representation. Each internal node of the tree representation denotes an attribute and each leaf node denotes a class label.A decision tree does not require normalization of data. A decision tree does not require normalization of data.

● XGBRegressor: XGBoost uses decision trees as base learners; combining many weak learners to make a strong learner. As a result it is referred to as an ensemble learning method since it uses the output of many models in the final prediction. It uses the power of parallel processing, supports regularization, and works well in small to medium dataset.

● RandomForestRegressor: A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A random forest produces good predictions that can be understood easily. It reduces overfitting and can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

● Support Vector Regressor: SVR works on the principle of SVM with few minor differences. Given data points, it tries to find the curve. But since it is a regression algorithm instead of using the curve as a decision boundary it uses the curve to find the match between the vector and position of the curve. Support Vectors helps in determining the closest match between the data points and the function which is used to represent them. SVR is robust to the outliers. SVR performs lower computation compared to other regression technique

Random state selection by Linear regression,

Model Training

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error
import scikitplot as skplt
```

```
for i in range(0,1000):
    x_train,x_test,y_train,y_test = train_test_split(scaled_x_best,y,test_size=0.10,random_state = i)
    lr.fit(x_train,y_train)
    pred_train = lr.predict(x_train)
    pred_test = lr.predict(x_test)
```

- Models -

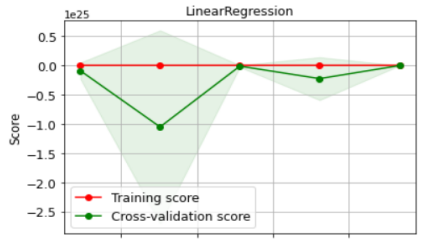
Linear regression -

Random_state =522

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.10,random_state=522)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
lr = LinearRegression()
lr.fit(x_train,y_train)
pred = lr.predict(x_test)
print(r2_score(y_test,pred)*100)
print("mean_absolute_error",mean_absolute_error(y_test,pred))
mse = mean_squared_error(y_test,pred)
print('RMSE :', np.sqrt(mse))
skplt.estimators.plot_learning_curve(lr,X,y,cv=5,scoring='r2',text_fontsize='large',title='LinearRegression')
print(plt.show())
```

```
82.22988770136858
mean_absolute_error 24800.1660667033
RMSE : 39938.05691815641
```



Random forest:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(x_train,y_train)
pred = rf.predict(x_test)
print(r2_score(y_test,pred)*100)
print(mean_absolute_error(y_test,pred))
mse = mean_squared_error(y_test,pred)
print('RMSE :', np.sqrt(mse))
skplt.estimators.plot_learning_curve(rf,X,y,cv=5,scoring='r2',text_fontsize='large',title='Randomforest')
print(plt.show())
```

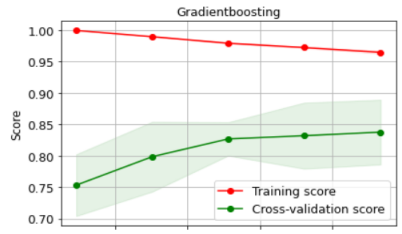
```
87.50108664651577
20549.342905982907
RMSE : 33494.84513194701
```

None

Gradient boosting:

```
from sklearn.ensemble import GradientBoostingRegressor
gb = GradientBoostingRegressor()
gb.fit(x_train,y_train)
pred = gb.predict(x_test)
print(r2_score(y_test,pred)*100)
print(mean_absolute_error(y_test,pred))
mse = mean_squared_error(y_test,pred)
print('RMSE :', np.sqrt(mse))
skplt.estimators.plot_learning_curve(gb,X,y,cv=5,scoring='r2',text_fontsize='large',title='Gradientboosting')
print(plt.show())
```

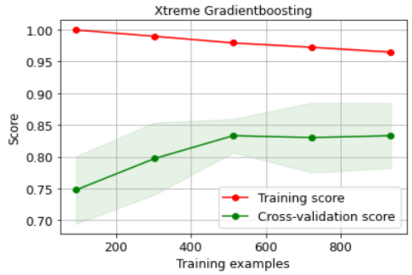
```
88.59447423453717
20395.02989545487
RMSE : 31996.28086397267
```



Xgboost:

```
from xgboost import XGBRegressor
xgbt = GradientBoostingRegressor()
xgbt.fit(x_train,y_train)
pred = xgbt.predict(x_test)
print(r2_score(y_test,pred)*100)
print(mean_absolute_error(y_test,pred))
mse = mean_squared_error(y_test,pred)
print('RMSE :', np.sqrt(mse))
skplt.estimators.plot_learning_curve(xgbt,X,y,cv=5,scoring='r2',text_fontsize='large',title='Xtreme Gradientboosting')
print(plt.show())
```

```
88.50431244179333
20389.553714993755
RMSE : 32122.498773053012
```



- Results- After Hypertuning we get Gradient boosting as Best model,

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
gbr=GradientBoostingRegressor()
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=.10,random_state=522)
params = {'n_estimators':[100,150],
        'min_samples_split':[2,6],
        'min_samples_leaf':[1,5],
          'learning_rate': np.arange(0.1,0.5,0.1)
        }
grd = GridSearchCV(gbr,param_grid = params)
grd.fit(x_train,y_train)
print('best_pram', grd.best_params_)

rf=grd.best_estimator_ #reinstantiating with best params

rf.fit(x_train,y_train)
pred = rf.predict(x_test)
print(r2_score(y_test,pred)*100)
print(mean_absolute_error(y_test,pred))
mse = mean_squared_error(y_test,pred)
print('RMSE :', np.sqrt(mse))
```

```
best_pram {'learning_rate': 0.1, 'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 150}
87.99887426171998
20277.413048933617
RMSE : 32821.07778455571
```

We increase the accuarcy after this by changing parameter by our own,

```
from sklearn.ensemble import GradientBoostingRegressor
gb = GradientBoostingRegressor(learning_rate=0.15,min_samples_leaf=1,min_samples_split=2,n_estimators=150,max_depth=5,max_fe
gb.fit(x_train,y_train)
pred = gb.predict(x_test)
print(r2_score(y_test,pred)*100)
print(mean_absolute_error(y_test,pred))
mse = mean_squared_error(y_test,pred)
print('RMSE :', np.sqrt(mse))
```

```
91.37720676472144
18535.35613679279
RMSE : 27820.557689028996
```

Based on comparing Accuracy Score results with Cross Validation results, it is determined that Gradient boosting Regressor is the best model. It also has the lowest Root Mean Squared Error score.

Based on the input parameter values and after fitting the train datasets The Gradient boosting Regressor model was further tuned based on the parameter values yielded from GridsearchCV. The Gradient boosting Regressor model displayed an accuracy of 91.37%.

This model was then tested using a scaled Test Dataset comprising of 292 entries for 80 features. The model performed with good amount of accuracy.

Summary, Based on the visualizations of the feature-column relationships, it is determined that, Features like OverallQual,GrLiveArea,MiscVal,ExterQual,KitchenQual,Gara geCars,
GarageArea,TotalBsmtSF,1stFlrSF,FullBath,TotRmsAbvGrd,

MasVnrArea, FirePlaces have the strongest positive correlation with SalePrice and are some of the most important features to predict the label values.

Gradient boosting Regressor Performed the best out of all the models that were tested. It also worked well with the outlier handling

CONCLUSION

Key Findings and Conclusions of the Study and Learning Outcomes with respect to Data Science

Based on the in-depth analysis of the Housing Project, The Exploratory analysis of the datasets, and the analysis of the Outputs of the models the following observations are made:

● Structural attributes of the house Structural attributes of the house like lot size, lot shape, quality and condition of the house, garage capacity, rooms, Lot frontage, number of bedrooms, bathrooms, overall finishing of the house etc play a big role in influencing the house price.

● Neighbourhood qualities can be included in deciding house price.

● Various plots like Barplots, Countplots and Lineplots helped in visualising the Feature-label relationships which corroborated the importance of structural and locational attributes for estimating Sale Prices.

● Due to the Training dataset being very small, the outliers had to be retained for proper training of the models.

● Therefore, Gradient boosting Regressor, being robust to outliers and being indifferent to non-linear features, performed well despite having to work on small dataset.

Limitations of this work and Scope,

for Future Work While features that focus on structural and locational attributes of housing properties are crucial for estimating the Sale Price of Housing properties, they aren't the only factors that influence the value in the housing market.

Data on Demographics(Age,Income,Regional preferences of buyers, purpose of buying a property) is very important for understanding the Housing market. Interest Rates too impact the price and demand of houses. Economic cycles also influence Real Estate prices. Government Policies, Regulations, Legalizations are also important factors that may influence the sales of houses. The availability of data on above features would help build a predictive model that would more accurately understand the relationship between the features and target variable and yield more accurate predictions