HOUSING PROJECT

Submitted by:

HARSH NEMA

ACKNOWLEDGMENT

I would like to express my profound gratitude to **Flip Robo team**, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analytical skills. I would like to express my special thanks to our mentor **Mr Shwetank Mishra Sir** (SME Flip Robo) for their contributions to the completion of my project titled **'Housing Project'**.

I am eternally grateful to **"Datatrained"** for giving me the opportunity for internship at Flip Robo. Last but not the least I have to thank my parents and wife for their love and support during my project.

References:

1. SCIKIT Learn Library Documentation
2. Datatrained recorded videos for Data Science Course
3. Hands on Machine learning with Scikit learn and tensor flow by Aurelien Geron
4. Predicting House Prices with Machine Learning https://www.kaggle.com/code/erick5/predicting-house-prices-with-machine-learning/notebook
5. Prediction and Analysis of Housing Price Based on the Generalized Linear Regression Model by Ahmedin M. Ahmed Volume 2022 | Article ID 3590224 | https://doi.org/10.1155/2022/3590224
6. Konwar, Robart & Kakati, Angshuman & Das, Bhagyashree & Shah, Borah & Muchahari, Monoj. (2021). House Price Prediction Using Machine Learning.

1. 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 are 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.

Regression is a *supervised learning algorithm in machine learning* which is used for prediction by learning and forming a relationship between present statistical data and target value i.e. **Sale Price** in this case.

There are many factors that have an impact on house prices, such as the number of bedrooms and bathrooms, location, accessibility to highways, schools, malls, employment opportunities etc.

We will try to understand how exactly the prices vary with the different variables in order to predict the actual value of the prospective houses.

• Conceptual Background of the Domain Problem

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. We 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:

• Which variables are important to predict the price of variable?

• How do these variables describe the price of the house?

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

The relationship between house prices and the economy is an important motivating factor for predicting house prices. A property's value is important in real estate transactions. Housing price trends are not only the concern of buyers and sellers, but it also **indicates the current economic situation**. Therefore, **it is important to predict housing prices without bias** to **help both the buyers and sellers make their decisions**. [Konwar, Robart & Kakati, Angshuman & Das, Bhagyashree & Shah, Borah & Muchahari, Monoj,(2021), House Price Prediction Using Machine Learning.]

It can be seen that the housing price prediction system based on the **generalized regression model** proposed in one of the article has high housing price prediction accuracy. [Prediction and Analysis of Housing Price Based on the Generalized Linear Regression Model by Ahmedin M. Ahmed Volume 2022]

• Motivation for the Problem Undertaken

The project is provided to me by Flip Robo Technologies as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary motivation.

As far as domain is concerned, the relationship between house prices and the economy is an important motivating factor for predicting house prices. **Rising house prices** generally encourage consumer spending and lead to **higher economic growth** – due to the wealth effect.

2. Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

Our objective is to predict House Sale price which can be done by the use of regression-based machine learning algorithm. In this project, we are going to use different types of algorithms which use their own mathematical equation on background. Two datasets are being provided for this project (test.csv, train.csv). Initially data cleaning & pre-processing was performed over both datasets. Feature engineering was then done to remove unnecessary feature & for dimensionality reduction. In model building, final model was selected based on evaluation benchmark among different models with different algorithms. Further Hyper parameter tuning was performed to build more accurate model.

- ## Data Sources and their formats

Dataset provided by Flip Robo was in the format of CSV (Comma Separated Values). There are 2 data sets that are given. One is training data and one is testing data.

1) **Train file** will be used for training the model, i.e., the model will learn from this file. It contains all the independent variables and the target variable. The dimension of data is 1168 rows and 81 columns.

```
[ ] 1 # Importing housing train dataset csv file
    2 import io
    3 df = pd.read_csv(io.BytesIO(uploaded['train.csv']))

    1 df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	Ho
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	Norm	TwnhsE	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	Norm	1Fam	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	Norm	1Fam	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	Norm	1Fam	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	Norm	1Fam	

```
[ ] 1 print("\033[1m" + 'Number of rows in the given dataset:' + "\033[0m")
    2 print(df.shape[0])
    3
    4 print("\033[1m" + 'Number of columns in the given dataset:'  + "\033[0m")
    5 df.shape[1]

Number of rows in the given dataset:
1168
Number of columns in the given dataset:
81
```

2) **Test file** contains all the independent variables, but not the target variable. We will apply the model to predict the target variable for the test data. The dimension of data is 292 rows and 80 columns

```
[ ] 1 # Importing housing test dataset csv file
    2 import io
    3 df_test = pd.read_csv(io.BytesIO(uploaded['test.csv']))
```

```
1 df_test.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	H
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm	Norm	1Fam	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm	Norm	TwnhsE	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm	Norm	1Fam	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr	Norm	1Fam	

```
[ ] 1 print("Number of rows in test dataset:",df_test.shape[0])
    2 print("Number of columns in test dataset:",df_test.shape[1])
```

```
Number of rows in test dataset: 292
Number of columns in test dataset: 80
```

The data types of different features are shown below:

```
1 # Sorting columns according to datatype
2 df.columns.to_series().groupby(df.dtypes).groups
```

```
{int64: ['Id', 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF',
'2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice'], float64:
['LotFrontage', 'MasVnrArea', 'GarageYrBlt'], object: ['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood',
'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',
'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu',
'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'SaleType', 'SaleCondition']}
```

```
[ ] 1 # Seperating numerical and categorical variables
    2
    3 num = ['Id', 'MSSubClass', 'LotFrontage','LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1', 'BsmtFinSF2',
    4        'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
    5        'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF',
    6        'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold',
    7        'MasVnrArea', 'GarageYrBlt', 'SalePrice']
    8
    9 catg = ['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood',
   10        'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
   11        'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating',
   12        'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish',
   13        'GarageQual', 'GarageCond', 'PavedDrive', 'SaleType', 'SaleCondition']
```

- **<u>Data Preprocessing Done</u>**

The dataset is large and it may contain some data error. In order to reach clean, error free data some data cleaning & data pre-processing is performed on both train and test data.

(a) Checking Null values in the dataset

```
1 df.isnull().sum().sort_values(ascending=False)
```

```
PoolQC          1161
MiscFeature     1124
Alley           1091
Fence            931
FireplaceQu      551
LotFrontage      214
GarageYrBlt       64
GarageFinish      64
GarageType        64
GarageQual        64
GarageCond        64
BsmtExposure      31
BsmtFinType2      31
BsmtQual          30
BsmtCond          30
BsmtFinType1      30
MasVnrType         7
MasVnrArea         7
Id                 0
Functional         0
Fireplaces         0
KitchenQual        0
KitchenAbvGr       0
BedroomAbvGr       0
HalfBath           0
FullBath           0
BsmtHalfBath       0
BsmtFullBath       0
TotRmsAbvGrd       0
GarageCars         0
```

- Lot of missing values in the training dataset, need to do the imputation of missing values

```
1 df_test.isnull().sum().sort_values(ascending= False)
```

```
PoolQC          292
MiscFeature     282
Alley           278
Fence           248
FireplaceQu     139
LotFrontage      45
GarageType       17
GarageYrBlt      17
GarageQual       17
GarageCond       17
GarageFinish     17
BsmtCond          7
BsmtFinType2      7
BsmtFinType1      7
BsmtQual          7
BsmtExposure      7
MasVnrArea        1
MasVnrType        1
Electrical        1
HalfBath          0
BsmtFullBath      0
BsmtHalfBath      0
BedroomAbvGr      0
FullBath          0
TotRmsAbvGrd      0
Functional        0
KitchenAbvGr      0
KitchenQual       0
Id                0
Fireplaces        0
LowQualFinSF      0
GarageCars        0
```

- Test dataset has also many null values

(b) Dropping the columns having more than 80% null values

Imputation of Missing Values

1. **PoolQC - 1161 missing values**
2. **MiscFeature - 1124 missing values**
3. **Alley - 1091 missing values**
4. **Fence - 931 missing values**

More than 80% values are missing in the above four columns so it is better to drop these features

```
[ ]  1 df.drop(['PoolQC','MiscFeature','Alley','Fence'],axis=1,inplace=True)
```

(c) Imputation of missing values

Imputation for missing values is done separately for numerical and categorical features. For categorical features, mode strategy is used and for numerical variables mean and median strategy is used.

```
[ ]  1 # Seperating numeerical and categorial variables where missing values are present
     2 catg_missing = ['FireplaceQu','GarageFinish','GarageType','GarageQual','GarageCond','BsmtExposure','BsmtFinType2','BsmtQual',
     3                 'BsmtCond','BsmtFinType1','MasVnrType']
     4 num_missing = ['LotFrontage','GarageYrBlt','MasVnrArea']
```

```
[ ]  1 # Imputation of missing values for categorical variables
     2 for i in df[catg_missing]:
     3   df[i] = df[i].fillna(df[i].mode()[0])
```

- *Outliers present in **LotFrontage** and **MasVnrArea** and hence imputation of mising values will be done using **Median** strategy*
- **GarageYrBlt** *doesn't have outliers nd hence imputation will be done by **Mean** strategy*
- *This is beacause **Mean** is sensitive to* **outliers**

```
1 df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].median())
2 df['MasVnrArea'] = df['MasVnrArea'].fillna(df['MasVnrArea'].median())
3 df['GarageYrBlt'] = df['GarageYrBlt'].fillna(df['GarageYrBlt'].mean())
```

```
[ ]  1 df_test.drop(['PoolQC','MiscFeature','Alley','Fence'],axis=1,inplace=True)

[ ]  1 # Seperating numeerical and categorial variables where missing values are present for test dataset
     2 catg_test_missing = ['FireplaceQu','GarageFinish','GarageType','GarageQual','GarageCond','BsmtExposure','BsmtFinType2','BsmtQual',
     3                      'BsmtCond','BsmtFinType1','MasVnrType','Electrical']
     4 num_test_missing = ['LotFrontage','GarageYrBlt','MasVnrArea']

[ ]  1 # Imputation of missing values for categorical variables for test dataset
     2 for i in df_test[catg_test_missing]:
     3   df_test[i] = df_test[i].fillna(df_test[i].mode()[0])
```

Similar operations are performed for test dataset

(d) Encoding the categorical variables for train and test data using Label Encoder

ENCODING THE CATEGORICAL VARIABLES

```
[ ]  1 from sklearn.preprocessing import LabelEncoder
     2 le = LabelEncoder()
```

1. Training Dataset

```
[ ]  1 for i in df.columns:
     2     if df[i].dtypes == "object":
     3         df[i] = le.fit_transform(df[i].values.reshape(-1,1))
```

2. Test Dataset

```
[ ]  1 for i in df_test.columns:
     2     if df_test[i].dtypes == "object":
     3         df_test[i] = le.fit_transform(df_test[i].values.reshape(-1,1))
```

(e) Selecting best features using Select K Best method

FEATURE SELECTION USING SELECTKBEST

```
[ ]  1 from sklearn.feature_selection import SelectKBest, f_classif

[ ]  1 # Sepearting features and target variable in training dataset
     2 X = df.drop('SalePrice',axis=1)
     3 y = df['SalePrice']

     1 best_features = SelectKBest(score_func = f_classif,k=17)
     2
     3 fit = best_features.fit(X,y)
     4
     5 df_scores = pd.DataFrame(fit.scores_)
     6
     7 df_columns = pd.DataFrame(X.columns)
     8
     9 features_score = pd.concat([df_columns, df_scores], axis=1)
    10
    11 features_score.columns = ['Feature_Name','Score']
    12
    13 features_score.nlargest(17,columns ='Score').Feature_Name

    16     OverallQual
    71         MiscVal
```

- Select K Best use f_classif function to find best features, where f_classif uses ANOVA Test
- Selecting top k=17 (by using correlation chart)features using top score

```
features_score = pd.concat([df_columns, df_scores], axis=1)

features_score.columns = ['Feature_Name','Score']

features_score.nlargest(17,columns ='Score').Feature_Name
```

```
27    concave points_worst
22         perimeter_worst
7       concave points_mean
20            radius_worst
2           perimeter_mean
23              area_worst
0              radius_mean
3                area_mean
6            concavity_mean
26          concavity_worst
5          compactness_mean
25        compactness_worst
10                radius_se
12             perimeter_se
13                  area_se
21           texture_worst
24         smoothness_worst
Name: Feature_Name, dtype: object
```

(f) Scaling of both train and test data using Standard Scaler

Scaling using Standard Scaler

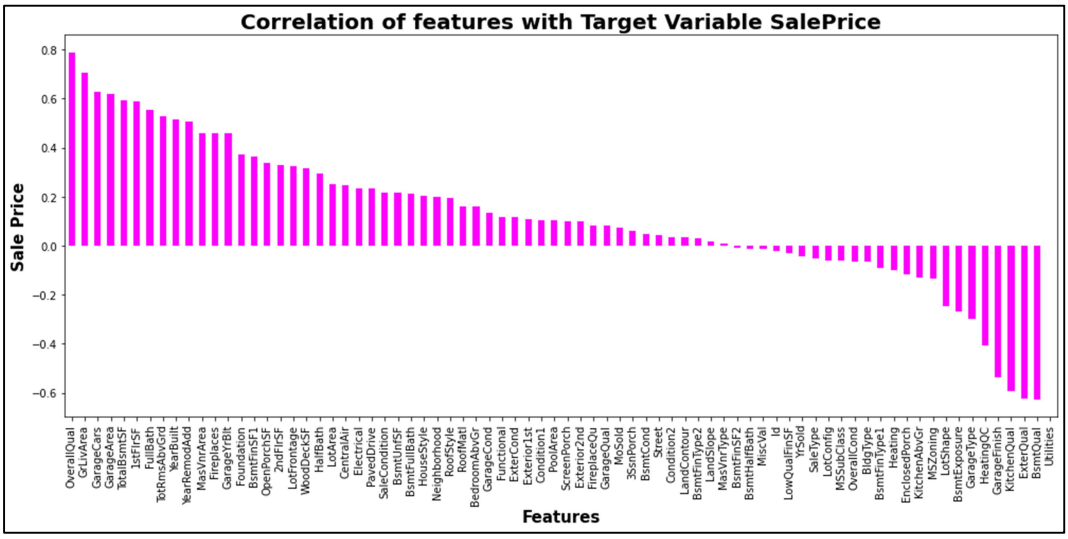
1.Training dataset

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaler = scaler.fit_transform(X_new)
```

2. Test Dataset

```
X_scaler_t = scaler.fit_transform(X_new_t)
```

- **<u>Data Inputs- Logic- Output Relationships</u>**



Correlation heat map is plotted to gain understanding of relationship between target features & independent features.

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

➢ Taking assumptions that features with at least Correlation +/- 0.4 or greater should be taken for further investigation

➢ Selecting k = 17 for Select K Best method, features having permissible correlation

- **Hardware and Software Requirements and Tools Used**

Hardware Used -

1. Processor — Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz 3.60 GHz

2. RAM — 16.0 GB

3. GPU — 2GB AMD Radeon Graphics card

Software utilised -

1. Google Colab Notebook - to write and execute arbitrary python code through the browser

2. Microsoft office – for making project report and ppt

Libraries Used – General libraries used for data wrangling

```
[1]  1 import pandas as pd
     2 import numpy as np

[2]  1 import seaborn as sns # For Purpose of Visualization
     2 import matplotlib.pyplot as plt # Plotting Package
     3 import warnings
     4 warnings.filterwarnings('ignore') # Filtering warnings
     5 %matplotlib inline
```

```
[ ]  1 # Importing required libraries
     2
     3 from sklearn.model_selection import train_test_split,GridSearchCV
     4 from sklearn.linear_model import LinearRegression
     5 from sklearn.linear_model import Ridge, Lasso, RidgeCV, LassoCV
     6 from sklearn.tree import DecisionTreeRegressor
     7 from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
     8 from sklearn.ensemble import RandomForestRegressor
     9 from sklearn.neighbors import KNeighborsRegressor
    10 from sklearn.svm import SVR
    11 from sklearn.ensemble import AdaBoostRegressor, GradientBoostingRegressor
    12 import pickle
    13 from sklearn import metrics
    14 from sklearn.model_selection import cross_val_score
    15 from xgboost import XGBRegressor
```

3. Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

  Our objective is to **predict house price** and analyse feature impacting Sale price. This problem can be solved using **regression-based machine learning algorithm** like linear regression. For that purpose, first task is to convert categorical variable into numerical features. Once data encoding is done then data is scaled using **standard scalar**. Final model is built over this scaled data. For building ML model before implementing regression algorithm, data is **split in training & test data** using train & test split from model selection module of sklearn library.

  Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. After that model is train with various regression algorithm and 5-fold cross validation is performed. Further **hyper parameter tuning** performed to build more accurate model out of best model.

- ## Testing of Identified Approaches (Algorithms)

The different regression algorithm used in this project to build ML model are as below:

- ✓ Linear Regression
- ✓ KNeighbors Regressor
- ✓ Random Forest Regressor
- ✓ Ada Boost Regressor
- ✓ Support Vector Regressor
- ✓ Extreme Gradient Boosting Regressor (XGB)

- ## Run and Evaluate selected models

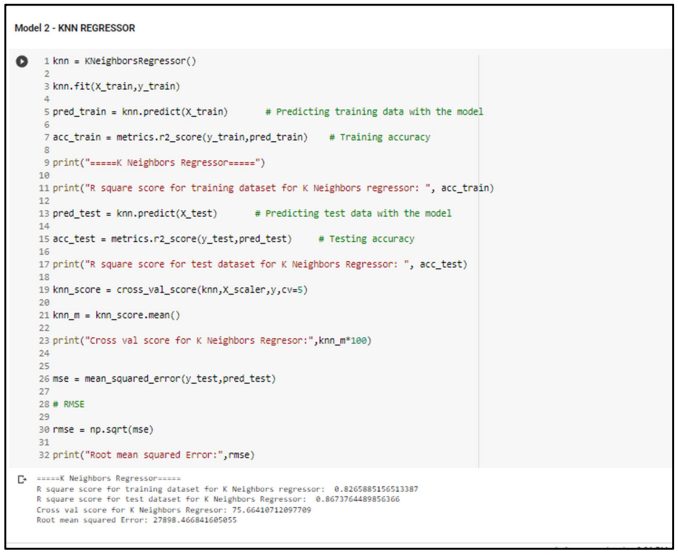
1. Linear Regression Model

Model 1 . LINEAR REGRESSION

```
lr = LinearRegression()

lr.fit(X_train,y_train)

pred_train = lr.predict(X_train)

pred_test = lr.predict(X_test)

score_train = r2_score(y_train,pred_train)

score_test = r2_score(y_test,pred_test)

print("Training accuracy:",score_train*100)

print("Testing accuracy:",score_test*100)

cv_score = cross_val_score(lr,X_scaler,y,cv=5)

cv_mean = cv_score.mean()

print(f"At cross fold 5, the cv score is {cv_mean} and accuracy score for training is {score_train} and accuracy for testing is {score_test}")

# MSE - ignoring some outliers means larger errors are punished. Hard to interpretate but most popular

mse = mean_squared_error(y_test,pred_test)

# RMSE

rmse = np.sqrt(mse)

print("Root mean squared Error:",rmse)
```

```
Training accuracy: 77.00248136219118
Testing accuracy: 89.30220963701831
At cross fold 5, the cv score is 0.7623186441964647 and accuracy score for training is 0.7700248136219118 and accuracy for testing is 0.8930220963701831
Root mean squared Error: 25056.306496302997
```

2. KNN Regressor



Model 2 - KNN REGRESSOR

```
knn = KNeighborsRegressor()

knn.fit(X_train,y_train)

pred_train = knn.predict(X_train)       # Predicting training data with the model

acc_train = metrics.r2_score(y_train,pred_train)    # Training accuracy

print("=====K Neighbors Regressor=====")

print("R square score for training dataset for K Neighbors regressor: ", acc_train)

pred_test = knn.predict(X_test)       # Predicting test data with the model

acc_test = metrics.r2_score(y_test,pred_test)     # Testing accuracy

print("R square score for test dataset for K Neighbors Regressor: ", acc_test)

knn_score = cross_val_score(knn,X_scaler,y,cv=5)

knn_m = knn_score.mean()

print("Cross val score for K Neighbors Regresor:",knn_m*100)


mse = mean_squared_error(y_test,pred_test)

# RMSE

rmse = np.sqrt(mse)

print("Root mean squared Error:",rmse)
```

```
=====K Neighbors Regressor=====
R square score for training dataset for K Neighbors regressor:  0.8265885156513387
R square score for test dataset for K Neighbors Regressor:  0.8673764489856366
Cross val score for K Neighbors Regresor: 75.66410712097709
Root mean squared Error: 27898.466841605055
```

3. Random Forest Regressor

Model 3 - RANDOM FOREST REGRESSOR

```
rf = RandomForestRegressor()

rf.fit(X_train,y_train)

rf.score(X_train,y_train)

pred_train = rf.predict(X_train)

pred_test = rf.predict(X_test)

score_train = r2_score(y_train,pred_train)

score_test = r2_score(y_test,pred_test)

print("=====RANDOM FOREST REGRESSOR=====")

print("Training accuracy for Random Forest model:",score_train*100)

print("Testing accuracy for Random Forest model:",score_test*100)

cv_score = cross_val_score(rf,X_scaler,y,cv=5)

cv_mean = cv_score.mean()

print(f"At cross fold 5, the cv score is {cv_mean*100} ")

# MSE - ignoring some outliers means larger errors are punished. Hard to interpretate but most popular

mse = mean_squared_error(y_test,pred_test)

# RMSE

rmse = np.sqrt(mse)

print("Root mean squared Error for Random Forest Regressor:",rmse)

```

```
=====RANDOM FOREST REGRESSOR=====
Training accuracy for Random Forest model: 97.06715526733156
Testing accuracy for Random Forest model: 90.3525523252906
At cross fold 5, the cv score is 82.2502546924130 4
Root mean squared Error for Random Forest Regressor: 23794.480488415582
```

4. Ada Boost Regressor

Model 4 - ADA BOOST REGRESSOR

```
ada = AdaBoostRegressor()

ada.fit(X_train,y_train)

pred_train = ada.predict(X_train)       # Predicting training data with the model

acc_train = metrics.r2_score(y_train,pred_train)    # Training accuracy

print("=====ADA BOOST REGRESSOR=====")

print("R square score for training dataset for Ada Boost regressor: ", acc_train)

pred_test = ada.predict(X_test)       # Predicting test data with the model

acc_test = metrics.r2_score(y_test,pred_test)

print("R square score for test dataset for Ada Boost Regressor: ", acc_test)

ada_score = cross_val_score(ada,X_scaler,y,cv=5)

ada_m = ada_score.mean()

print("Cross val score for Ada Boost Regresor:",ada_m*100)


mse = mean_squared_error(y_test,pred_test)

# RMSE

rmse = np.sqrt(mse)

print("Root mean squared Error for ada boost Regressor:",rmse)

```

```
=====ADA BOOST REGRESSOR=====
R square score for training dataset for Ada Boost regressor:  0.8495414783778362
R square score for test dataset for Ada Boost Regressor:  0.840759786545021
Cross val score for Ada Boost Regresor: 75.12024724636987
Root mean squared Error for ada boost Regressor: 30570.066736720488
```

5. Support Vector Regressor

Model 5 - SUPPORT VECTOR REGRESSOR

```
svr = SVR()

svr.fit(X_train,y_train)

pred_train = svr.predict(X_train)       # Predicting training data with the model

acc_train = metrics.r2_score(y_train,pred_train)    # Training accuracy

print("R square score for training dataset for SVR: ", acc_train)

pred_test = svr.predict(X_test)       # Predicting test data with the model

acc_test = metrics.r2_score(y_test,pred_test)

print("R square score for test dataset for SVR: ", acc_test)

svr_score = cross_val_score(svr,X_scaler,y,cv=5)

svr_m = svr_score.mean()

print("Cross val score for SVR:",svr_m*100)


mse = mean_squared_error(y_test,pred_test)

# RMSE

rmse = np.sqrt(mse)

print("Root mean squared Error for SVR:",rmse)
```

```
R square score for training dataset for SVR:  -0.050931417533669476
R square score for test dataset for SVR:  -0.051828843869095254
Cross val score for SVR: -6.063551569263259
Root mean squared Error for SVR: 78567.43037521666
```

6. XGradient Boosting Regressor

Model 6 - Extreme Gradient Boosting Regressor

```
xgb = XGBRegressor()

xgb.fit(X_train,y_train)

xgb.score(X_train,y_train)

pred_train = xgb.predict(X_train)

pred_test = xgb.predict(X_test)

score_train = r2_score(y_train,pred_train)

score_test = r2_score(y_test,pred_test)

print("=====XGBoosting REGRESSOR=====")

print("Training accuracy for XGradient Boost model:",score_train*100)

print("Testing accuracy for XGradient Boost model:",score_test*100)

cv_score = cross_val_score(xgb,X_scaler,y,cv=5)

cv_mean = cv_score.mean()

print(f"At cross fold 5, the cv score is {cv_mean*100} ")

# MSE - ignoring some outliers means larger errors are punished. Hard to interpretate but most popular

mse = mean_squared_error(y_test,pred_test)

# RMSE

rmse = np.sqrt(mse)

print("Root mean squared Error for XGradient Boosting model:",rmse)
```

```
[08:51:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
=====XGBoosting REGRESSOR=====
Training accuracy for XGradient Boost model: 94.48432637782345
Testing accuracy for XGradient Boost model: 91.23401084499494
[08:51:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
[08:51:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
[08:51:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
[08:51:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
```

- Key Metrics for success in solving problem under consideration

Following metrics used for evaluation:

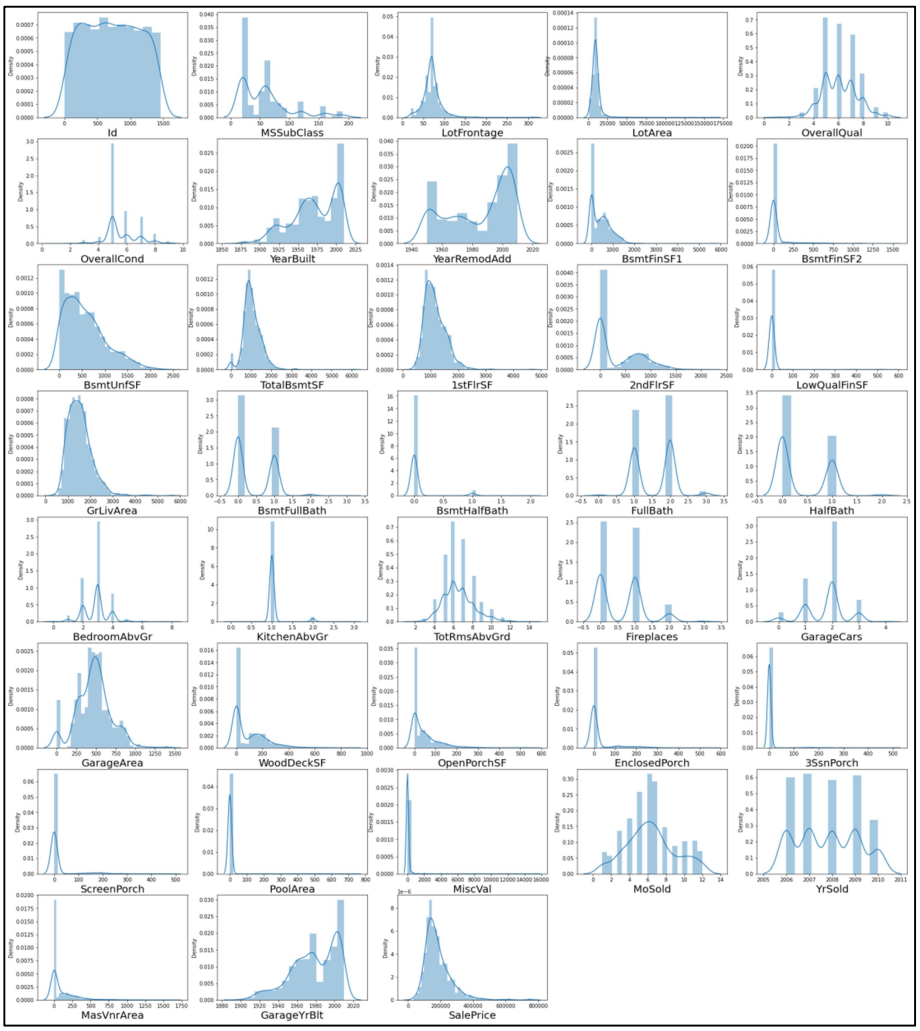
1. Root mean square error is one of the most commonly used measures for evaluating the quality of predictions.

2. R2 score which tells us how accurate our model predict result, is going to be an important evaluation criterion

3. Cross Validation Score

- Visualizations

- ✓ Almost 80% of the houses belongs to Residential Low Density as far as zoning classification of the sale is considered
- ✓ 99.6% of streets belongs to Pavel category (Type of road access to property)
- ✓ Utilities feature has a single unique value for all the observation and hence this feature can be dropped for further analysis
- ✓ More than 70% of Lot configuration belongs to Inside Lot
- ✓ Almost 95% property has Gentle slope
- ✓ Around 85% of Building type belongs to Single-family Detached
- ✓ Gas Forced air heating is a way a cooling or heating system distributes air throughout a home or a structure and majority of the houses in Australia has this heating system

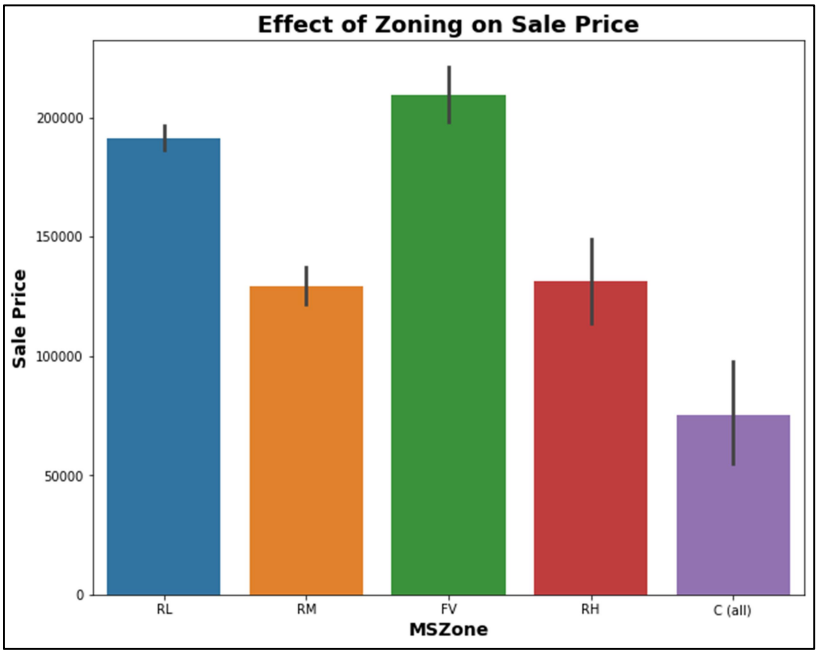


- ✓ Outliers present in some of the features and it is evident by seeing 75% and max values
- ✓ The oldest house was built in 1875 and the latest built was in 2010
- ✓ Min Sale Price is 34900 and maximum Sale price is 755000 in Australian currency
- ✓ Average sale price is 181477 and median is 163995 in Australian currency
- ✓ Skewness can be seen in most of the variables either positive skewed or negatively skewed
- ✓ Target variable - Sale Price has some skewness too on the right hand side

❖ **How Quality & Area of house affect Pricing?**

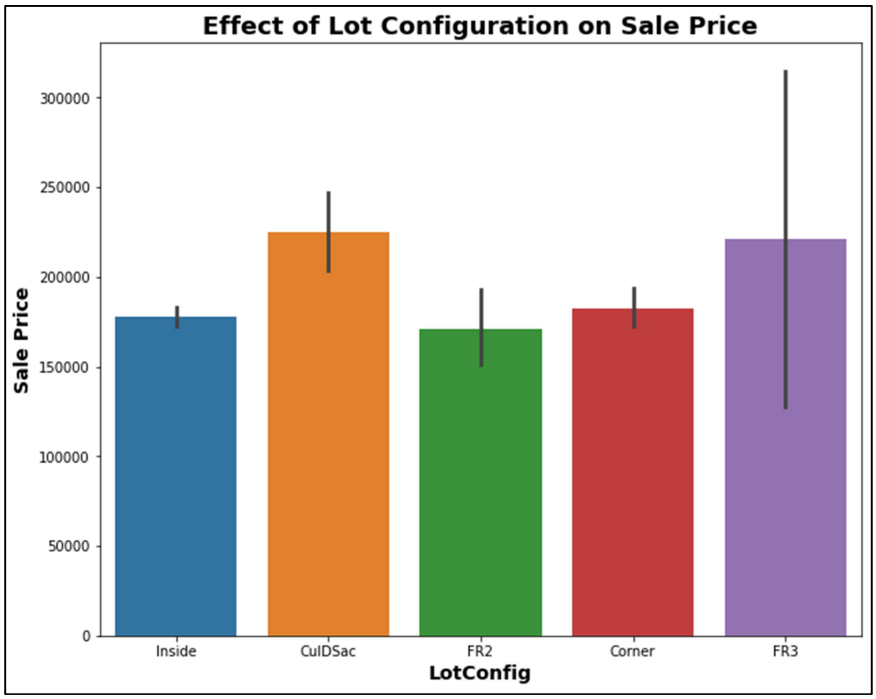
- ✓ Effect of Lot Area looks less significant on the Sale price of the houses
- ✓ As the overall quality of house increases, sale price of house also increases

❖ **Effect of Zone on Pricing**



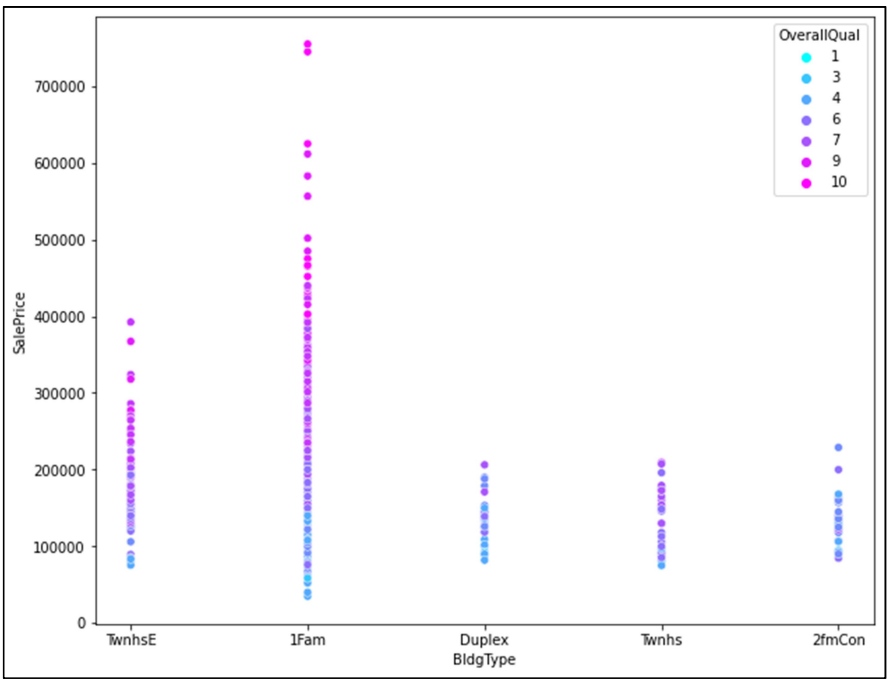
- ✓ We can clearly see the effect of zones on sale price
- ✓ Floating Village Residential are less in number but having more pricing than any other zones, means FV zone are costlier than RL which are in majority

❖ **Effect of Lot configuration on Pricing**



- ✓ Instead of presence of 70% Inside lot, total sale price of inside lot houses is least in the lot
- ✓ It implies that Inside lot houses are the cheapest among all other Lot configuration
- ✓ FR3 - Frontage on 3 sides of property are the costliest among all the Lot configuration

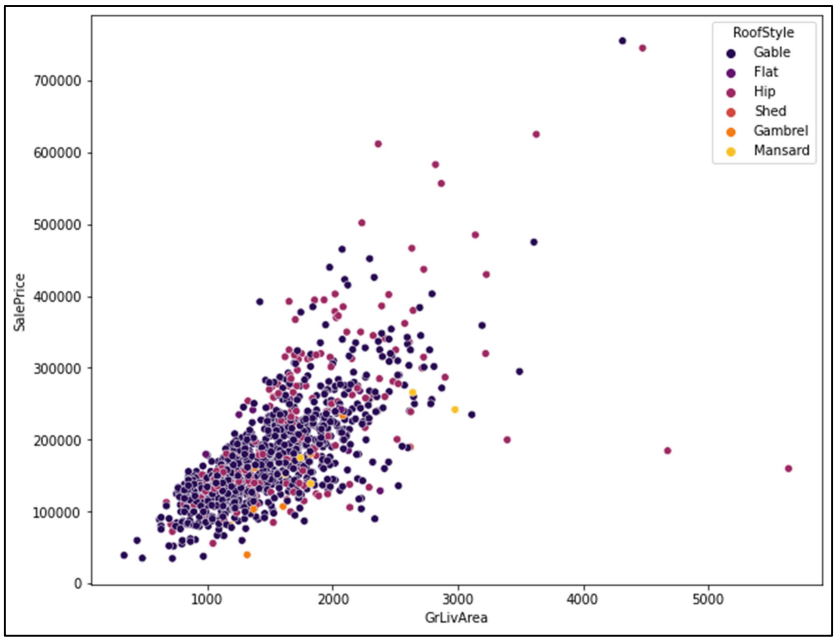
❖ **Effect of Building type and Overall quality on Pricing**



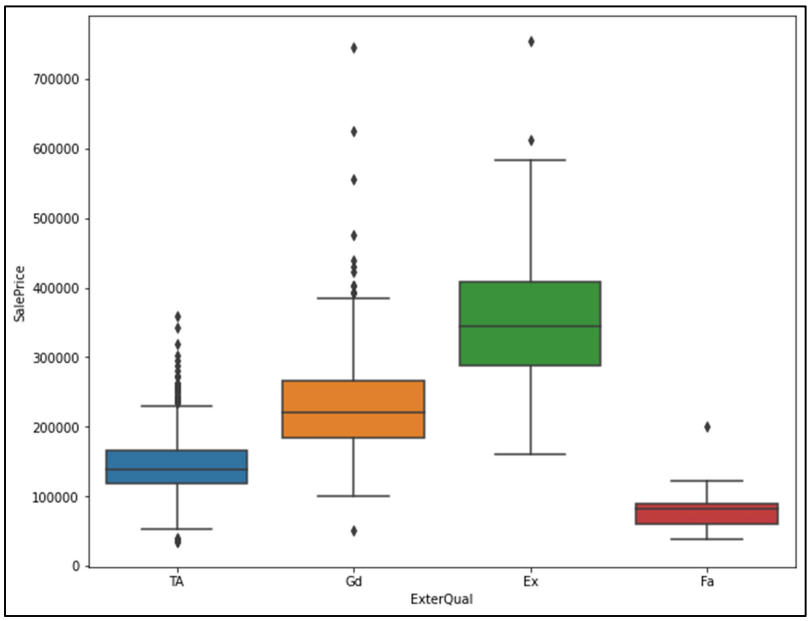
- ✓ As seen above, around 85% building in our dataset is 1Fam(Single Family Detached)
- ✓ It can be seen increase in overall quality of house increase the sale price of the house
- ✓ The overall quality of house is higher in 1Fam building type

❖ **Effect of Floor Area and Roof Style on Pricing**

- ✓ Clearly, a positive linear relationship can be seen between floor area and sale price of the house
- ✓ As total floor area increases the sale price also get increases
- ✓ For large floor area construction, Hip style Roof is used and invariably high cost properties mostly comes up with Hip Style Roof



❖ **Effect of Exterior Quality on Pricing**

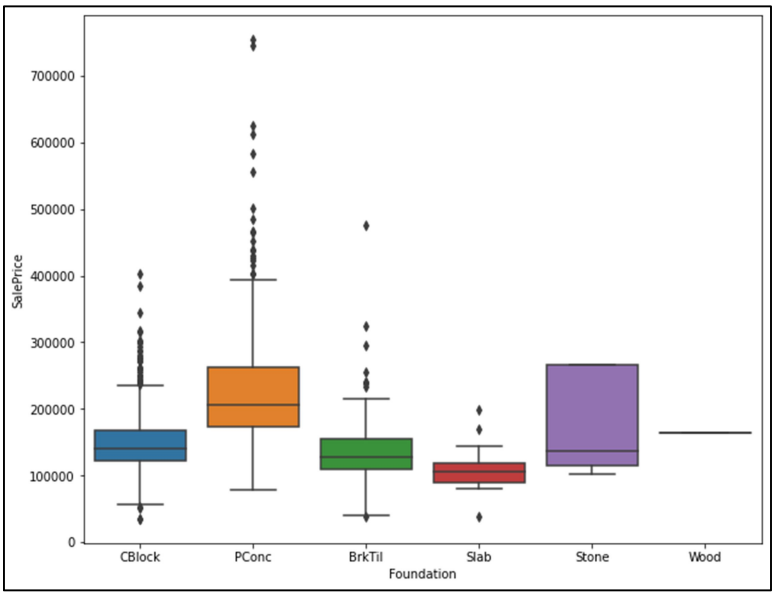


✓ As seen above in the count plot, around 60% of house properties come with Average Exterior quality and all of them below 400000

✓ Very few House Properties comes with Excellent Exterior Quality

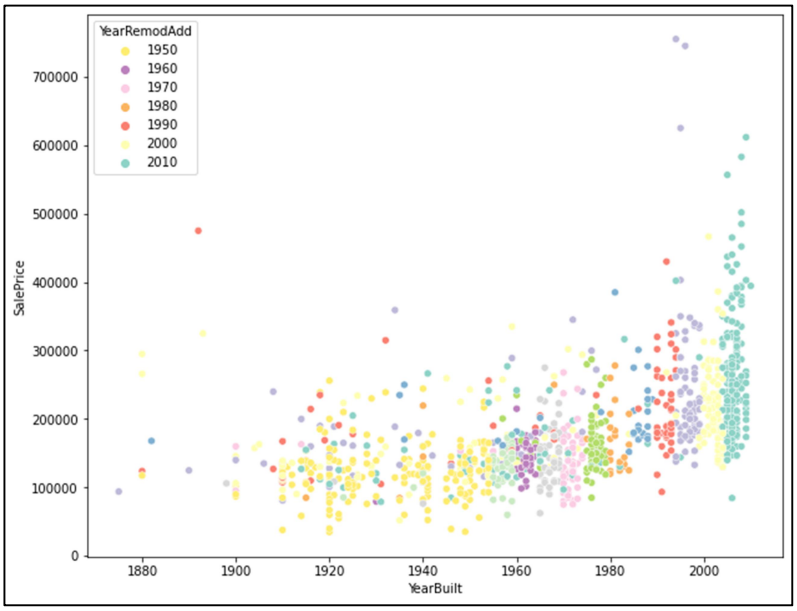
✓ Costlier house properties come with Good & Excellent exterior quality

❖ **Effect of Foundation on Pricing**



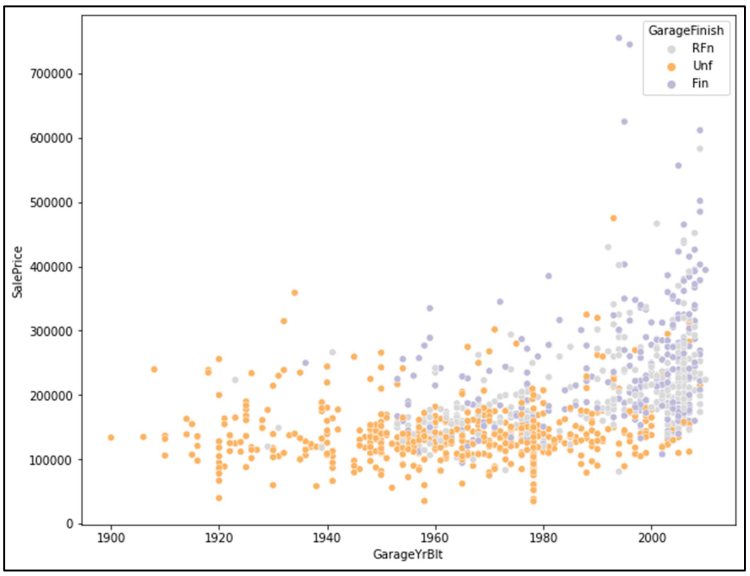
✓ Poured Concrete houses are costlier than any other type of house

❖ **Effect of Property Age on Pricing**



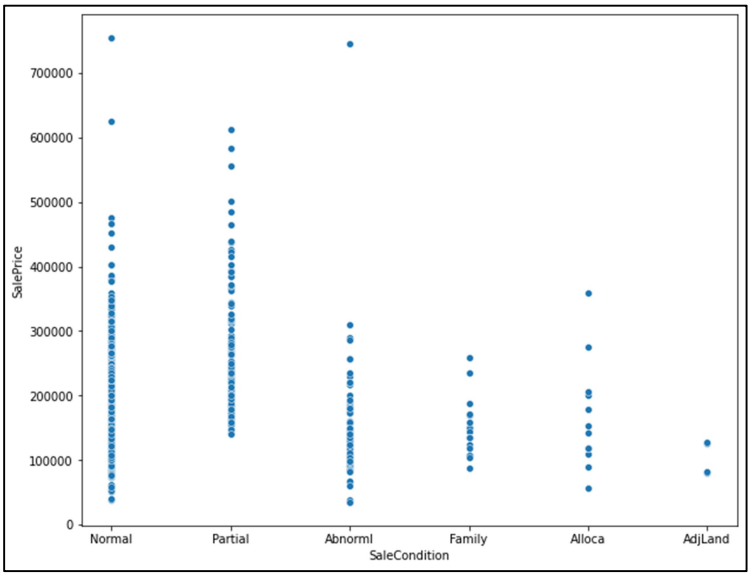
✓ Clearly seen that as the property gets older its Sales Price depreciates

❖ **Effect of Garage age and Garage Finish on Pricing**



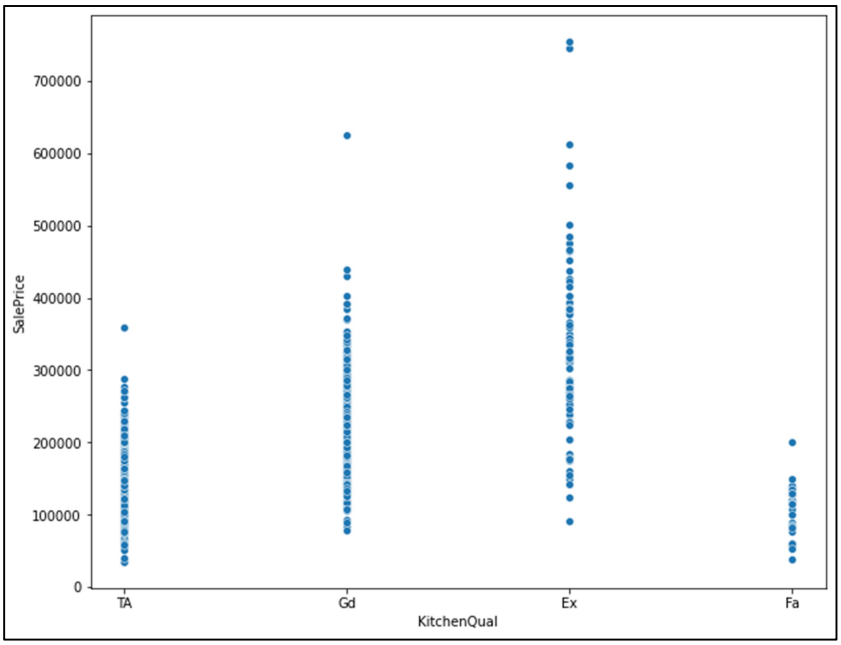
- ✓ Older the Garage age, less is the Price of property
- ✓ Finished Garage are costlier than Unfinished Garage

❖ Effect of Sale Condition on Pricing



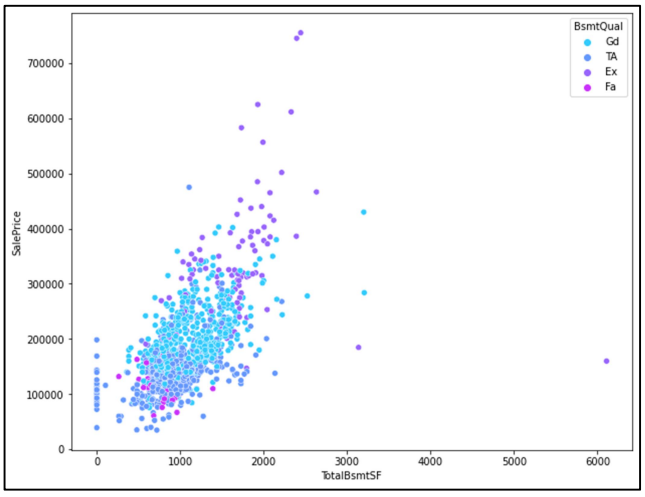
- ✓ Sale condition like Abnormal, Family, Alloca and AdjLand has sale price below 300000
- ✓ Normal and Partial sales condition were mostly preferred and have better pricing than others
- ✓ We can observe maximum Sale Price with Normal Sales condition

❖ **Effect of Kitchen Quality on Pricing**



✓ ***Excellent*** quality kitchens are costliest and Good and Excellent kitchens are mostly preferred ***one***

❖ **Effect of Basement on Pricing**



✓ We can observe Positive relationship between basement area and Sale price

✓ Increase in quality always increases the Price of the house

✓ Both Basement area and Basement quality increases the price of any house

4. CONCLUSION

- **Key Findings and Conclusions of the Study**

ML Algorithm	R2 Score		CV Score
	Train	Test	
Linear Regression	77%	89.30%	76.23%
K Neighbors Regressor	82.60%	86.70%	75.66%
Random Forest Regressor	97.06%	90.35%	82.25%
Ada Boost Regressor	84.95%	84.07%	75.12%
Support Vector Regressor	5.09%	5.18%	-6.06
Extreme Gradient Boost Regressor(XGB)	94.48%	91.23%	82.70%
Hyper Parameter Tuning for XGB	94.48%	91.23%	82.70%

- ✓ Extreme Gradient Boosting(XGB) model gives the best result for the given dataset
- ✓ As we get highest test accuracy as 91.23% and training accuracy as 94.48%
- ✓ Cross validation scores is also looking best among other models as 82.7%

- **Learning Outcomes of the Study in respect of Data Science**

- ✓ XGB works well compared to others machine learning algorithms
- ✓ XGBoost consists of objective function and base learners
- ✓ Loss function is present in objective function
- ✓ Loss function shows the difference between actual values and predicted values
- ✓ Regularisation term is used for showing how far is actual value away from predicted value
- ✓ XGBoost considers many models which are known as base learners for predicting a single value

- ✓ Not all base learners are expected to have bad prediction so that after summing up all of them bad prediction cancelled out by good prediction
- ✓ The XGBoost regression algorithm helps to satisfy the needs of customers by increasing accuracy of the choice of estates and decreasing the risk for customers to invest in real estate

- **<u>Limitations of this work and Scope for Future Work</u>**

  Deep learning algorithms may enhance the prediction of price of house and decrease test error rate percentage.