

Predicting Housing Price Using

Machine Learning Model

Submitted by:

Aditya Paruleker

**INTRODUCTION**

* Business Problem Framing

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. Similarly, the goal of this project is to use the data science knowledge to build a model to predict the housing price and also find the solution to following problems

* Which variables are important to predict the price of variable?
* How do these variables describe the price of the house?
* Motivation for the Problem Undertaken

Motivation behind this project is to help the people in the field of real estate management to understand how exactly the prices of the houses vary with the variables. The information obtained from the model can help these individuals to make different strategies by concentrating on the areas that will yield high returns. Further, the model will also be a good way for any real estate management to understand the pricing dynamics of any new market.

**Analytical Problem Framing**

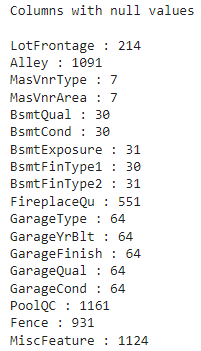
* Data Sources and their formats

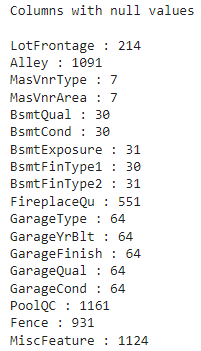
Data set containing sale of houses in Australia is collected by a US-based housing company named Surprise Housing who has decided to enter the Australian market. The data is provided in the CSV file format.

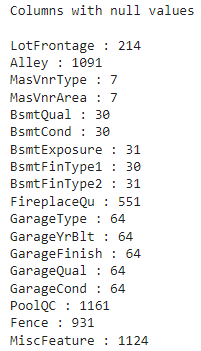
* Data Preprocessing

Data preprocessing is a technique i.e. widely used to transform the raw data into an efficient format that will be suitable for machine learning model.

1. **Null Value Analysis**

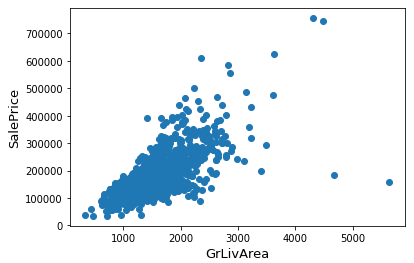






There were 18 features found containing null value, of which 5 features (Alley, FireplaceQual, PoolQC, Fence, MiscFeature) contained more than 30% null value and hence had to be dropped. Remaining features containing null values were replaced using either mean value (if the missing data is numeric) or using mode value (if the missing data is categorical).

1. **Outlier Analysis**



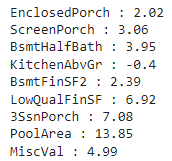
Extreme case of outlier was detected in input feature GirLivArea w.r.t. target feature SalePrice. The two outliers in the can be seen in the above graph indicating that there are two houses with extremely huge area available for very cheap rate, which is technically not possible. Hence we decide to drop these two values.

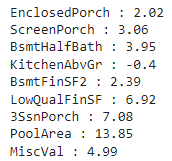
Although there may be many outliers in the training dataset but simply removing all them might affect our models incase there were also outliers in the test dataset. And since removing the outliers from dataset is not always safe, we decided to drop only the extreme outliers that set really bad trend (like outliers that were detected during outlier analysis indicating huge area for low price).

1. **Skew Analysis**

There were total of 21 numeric features that contained skewness. Quantile transformation was applied on these features to remove the skewness. But even after performing several other transformation, there were 8 features (displayed below) whose skew value was not in the range of +/- 0.5.





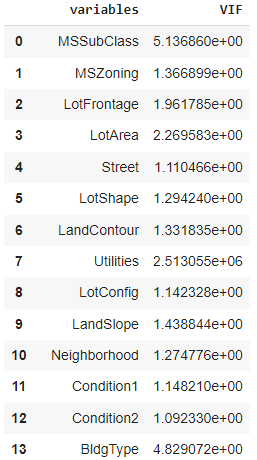
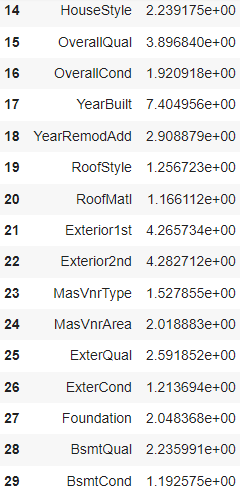
Since these features were highly skewed and also had very weak correlation with the target variable indicating minor contribution to the output; hence we decided to drop these features.

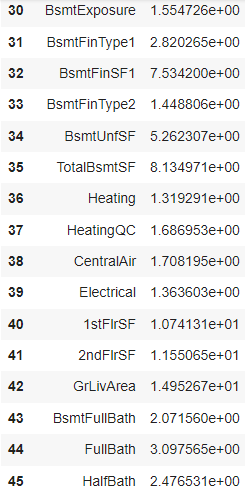
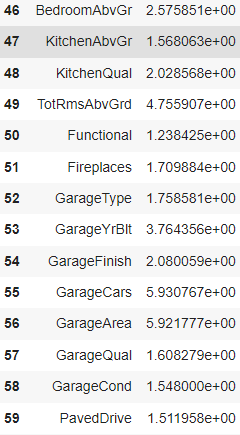
1. **Encoding**

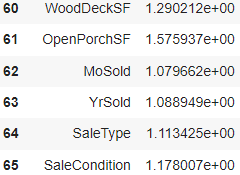
Based on dataset analysis, all the categorical features with hierarchy were encoded using ordinal encoder while remaining categorical features were encoded using label encoder.

1. **Collinearity Analysis**

In order to avoid two or more input features from feeding same information into the model, it is necessary to detect correlation between independent variables using Variance Inflation factor or VIF method. Below is the screenshot of dataset and its VIF value.

** **

** **

****

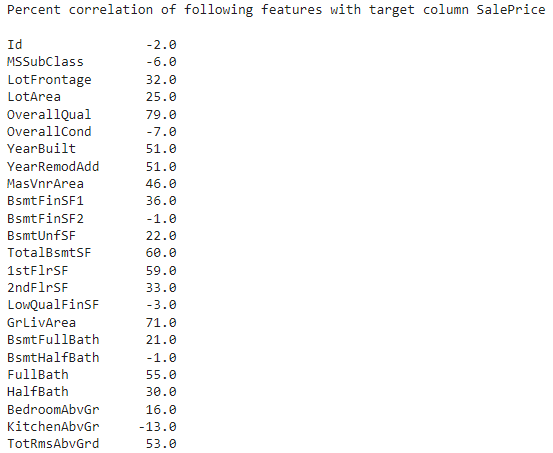
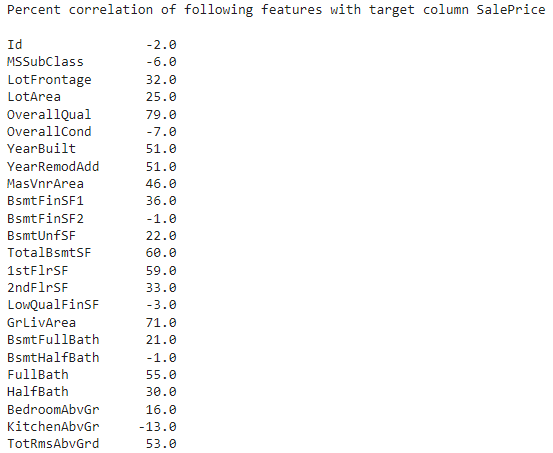
In above case, we decided to drop only those features that are found to have a very high VIF value, indicating severe multicollinearity.

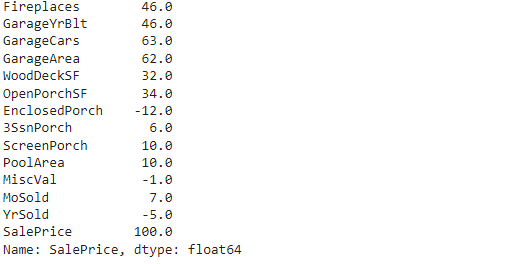
1. **Feature Scaling**

Feature scaling is a technique frequent used in machine learning to generalize the data points so that the distance between them is smaller.The **Min-Max Normalization** uses distribution value between 0 and 1 to re-scales all the feature values.

* Data Inputs- Logic- Output Relationship

Correlation is a measure of how strongly or weakly connected two features are in a dataset. Below table show the percentage of correlation each input feature has with the output column. Higher the percentage stronger is the bond, and vice versa. Positive value indicates that the relationship between the two variables move in the same direction whereas negative value indicates that the relationship between two variables move in opposite direction.

** **

****

* Hardware and Software Requirements and Tools Used

Following libraries were used in this project,

**Pandas**

Created by Wes McKinney in 2008, this python library is widely used for data manipulation, analyzing and cleaning of data. Apart from this, it also helps in finding correlation between columns and in deleting rows.

**Numpy**

Created by Travis Oliphant in 2005, this python library provides an array object called ndarray i.e. up to 50x faster than traditional python lists. It was has functions can be used in linear algebra, matrices etc

**Seaborn**

Seaborn is a high level interface based data visualization library that uses matplotlib library underneath the working.

**Matplotlib**

Unlike saeborn, matplotlib is a low level data visualization python library. Majority of its function lies in the submodule named pyplot

**Scikit-Learn**

It provides tools for classification, regression, clustering and dimensionality reduction through its interface in python.

**Pickle**

Used for serializing (pickling) and de-serializing (unpickling) python object structure so that the data can be easily transferred from system to system and then stored in a file.

**Model/s Development and Evaluation**

* Testing of Identified Approaches (Algorithms)

In this study several machine learning models were tested and their results were compared before selecting the model that gave best result among all.

**Linear Regression**

Using linear approach for modeling, this supervised machine learning model constructs relationship between scalar response and explanatory variables. There are many different types of regression model based on type of relationship between independent and dependent variables.

**XGB Boost**

This Extreme Gradient Boosting supervised machine learning model uses parallel tree boosting method for both regression and classification. It is widely preferred to be used when the size of the dataset is large enough.

**Ensemble Methods**

Ensemble technique uses several base estimators of machine learning model to predict the output. This results are then combined to improve the robustness of model over single estimator.

**Randomforest Regressor**

Using multiple decision trees as base, the model randomly performs the dataset sampling over the rows and features such that it form sample datasets for every model.

**AdaBoost Regressor**

The model first fits on the original dataset and then this Adative Boosting regressor further fits on extra copies of same dataset only with adjusted instances of weights based on the current prediction error.

**GradientBoost Regressor**

The model after calculating the residual i.e. the difference between actual value and predicted target value; trains the weak model for mapping the features to that residual.

**Bagging Regressor**

This ensemble method uses random subsets of the original dataset on which each base regressors are fitted; then final prediction is formed by either by voting or by averaging the aggregate of individual predictions.

**Voting Regressor**

Unlike bagging regressor, voting regressor uses whole dataset on which each base regressors are fitted and then final prediction is formed by averaging the individual predictions.

* Run and Evaluate models

**Linear Regression**

for i in range(100):

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=i)

lr.fit(x,y)

pred\_train=lr.predict(x\_train)

pred\_test=lr.predict(x\_test)

print("At Random State : ",i)

print("Training r2: ",round(r2\_score(y\_train,pred\_train)\*100,2))

print("Testing r2: ",round(r2\_score(y\_test,pred\_test)\*100,2))

print("\n")

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=98)

lr.fit(x,y)

pred\_train=lr.predict(x\_train)

pred\_test=lr.predict(x\_test)

print("Training r2: ",round(r2\_score(y\_train,pred\_train)\*100,2))

print("Testing r2: ",round(r2\_score(y\_test,pred\_test)\*100,2))

print("\n")

print("mean\_absolute\_error: ",mean\_absolute\_error(y\_test,pred\_test))

print("mean\_squared\_error: ",mean\_squared\_error(y\_test,pred\_test))

print("Root mean\_square: ",np.sqrt(mean\_squared\_error(y\_test,pred\_test)))

ls=r2\_score(y\_test,pred\_test)

for j in range(2,20):

cv\_score=cross\_val\_score(lr,x,y,cv=j)

print(cv\_score)

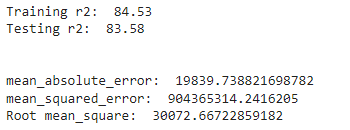
print("At cv: ",j)

print("Cross Validation Score: ",round(cv\_score.mean()\*100,2))

print("r2 Score: ",round(ls\*100,2))

print("\n")

**Output**



**Lasso Regression**

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import Lasso

parameters={'alpha':[.00001,.0001,.001,.01,.1,1,10],

            'random\_state': list(range(0,10)),

            'selection':['cyclic', 'random'],

            'fit\_intercept': ['True','False']}

ls=Lasso()

clf=GridSearchCV(ls,parameters)

clf.fit(x\_train,y\_train)

print(clf.best\_params\_)

from sklearn.linear\_model import Lasso

ls=Lasso(alpha=10,random\_state=6,fit\_intercept=True,selection='random')

ls.fit(x\_train,y\_train)

ls\_score\_train=ls.score(x\_train,y\_train)

pred\_ls=ls.predict(x\_test)

print("Training Accuracy: ",round(ls\_score\_train\*100,2))

lss=r2\_score(y\_test,pred\_ls)

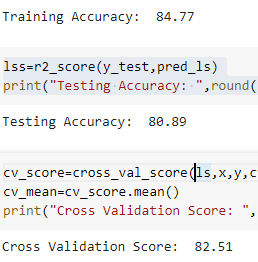
print("Testing Accuracy: ",round(lss\*100,2))

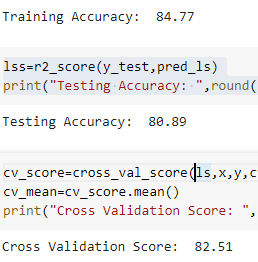
cv\_score=cross\_val\_score(ls,x,y,cv=8)

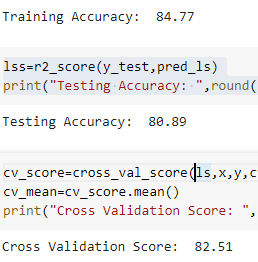
cv\_mean=cv\_score.mean()

print("Cross Validation Score: ",round(cv\_mean\*100,2))

**Output**

****

****

****

**Model Selection User Defined Function**

from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error

def model\_selection(algorithm\_instance,x\_train,y\_train,x\_test,y\_test):

algorithm\_instance.fit(x\_train,y\_train)

model\_pred\_train=algorithm\_instance.predict(x\_train)

model\_pred\_test=algorithm\_instance.predict(x\_test)

print("Accuracy of training model :",round(r2\_score(y\_train,model\_pred\_train)\*100,2))

print("Accuracy of test data :",round(r2\_score(y\_test,model\_pred\_test)\*100,2))

rfscore=cross\_val\_score(algorithm\_instance,x,y,cv=8)

rfc=rfscore.mean()

print('Cross Val Score:',round(rfc\*100,2))

print("\nMean Absolute Error ",mean\_absolute\_error(y\_test,model\_pred\_test))

print("Mean Sq. Error ",mean\_squared\_error(y\_test,model\_pred\_test))print("Root Mean Sq ",np.sqrt(mean\_squared\_error(y\_test,model\_pred\_test)))

print("\n")

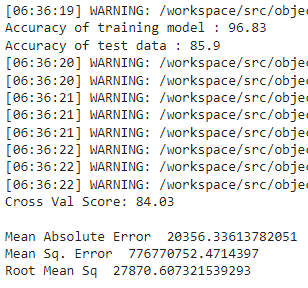
**XGB Boost**

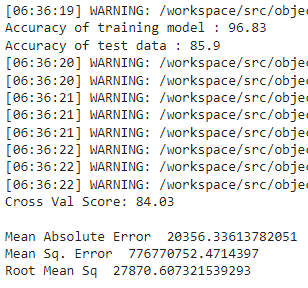
import xgboost as xg

xgb = xg.XGBRegressor()

model\_selection(xgb,x\_train,y\_train,x\_test,y\_test)

**Output**

****

****

**Randomforest Regressor**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestRegressor

parameter={'criterion':['mse','mae'],

          'max\_features' : ["auto","sqrt","log2"],

          'n\_estimators':range(0,100,25)}

rf=RandomForestRegressor()

clf=GridSearchCV(rf,parameter)

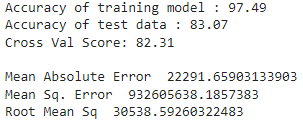
clf.fit(x\_train,y\_train)

print(clf.best\_params\_)

rf=RandomForestRegressor(criterion='mse',max\_features="auto", n\_estimators=75)

model\_selection(rf,x\_train,y\_train,x\_test,y\_test)

**Output**

****

**AdaBoost Regressor**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import AdaBoostRegressor

parameter={'loss':['linear', 'square', 'exponential'],

          'random\_state' : range(0,100,25),

           'learning\_rate':[0,1.0],

           'n\_estimators':range(0,100,25)}

rf2=AdaBoostRegressor()

clf=GridSearchCV(rf2,parameter)

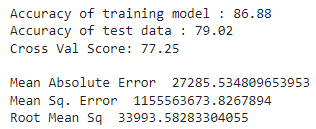
clf.fit(x\_train,y\_train)

print(clf.best\_params\_)

rf2=AdaBoostRegressor(learning\_rate= 1.0, loss= 'exponential', n\_estimators= 50, random\_state= 50)

model\_selection(rf2,x\_train,y\_train,x\_test,y\_test)

**Output**

****

**GradientBoost Regressor**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import GradientBoostingRegressor

parameter={'loss':['squared\_error', 'absolute\_error', 'huber','quantile'],

          'criterion':['friedman\_mse', 'squared\_error', 'mse'],

           'learning\_rate':[0,1.0],

           'n\_estimators':range(0,100,25)}

rf3=GradientBoostingRegressor()

clf=GridSearchCV(rf3,parameter)

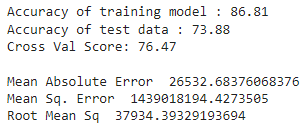
clf.fit(x\_train,y\_train)

print(clf.best\_params\_)

rf3=GradientBoostingRegressor(learning\_rate= 1.0, loss= 'absolute\_error', n\_estimators= 25, criterion= 'squared\_error')

model\_selection(rf3,x\_train,y\_train,x\_test,y\_test)

**Output**



**Bagging Regressor**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import BaggingRegressor

parameter={'oob\_score':['True','False'],

          'n\_jobs':range(0,10,2),

           'random\_state':range(0,100,25),

           'n\_estimators':range(0,100,25)}

rf4=BaggingRegressor()

clf=GridSearchCV(rf4,parameter)

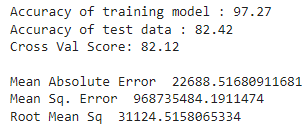
clf.fit(x\_train,y\_train)

print(clf.best\_params\_)

rf4=BaggingRegressor(oob\_score= True, n\_jobs= 2, n\_estimators= 75, random\_state= 75)

model\_selection(rf4,x\_train,y\_train,x\_test,y\_test)

**Output**

****

**Voting Regressor**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import VotingRegressor

from sklearn.svm import SVR

from sklearn.neighbors import KNeighborsRegressor

estimators = [ ('knc', KNeighborsRegressor()), ('svr',SVR()) ]

parameter={

          'n\_jobs':range(0,10,2),

           'verbose':[True,False]

          }

rf5=VotingRegressor(estimators)

clf=GridSearchCV(rf5,parameter)

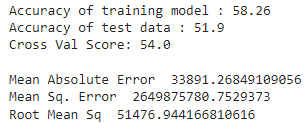
clf.fit(x\_train,y\_train)

print(clf.best\_params\_)

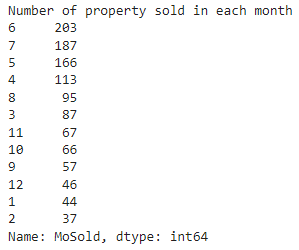
rf5=VotingRegressor(estimators, n\_jobs= 2, verbose= True)

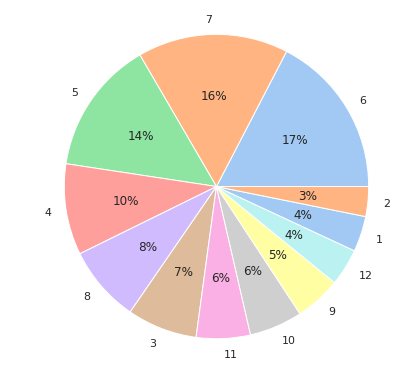
model\_selection(rf5,x\_train,y\_train,x\_test,y\_test)

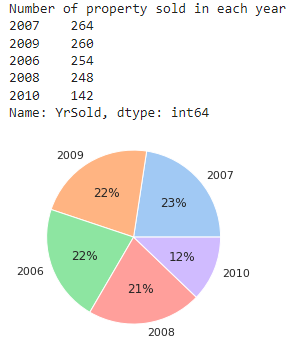
**Output**

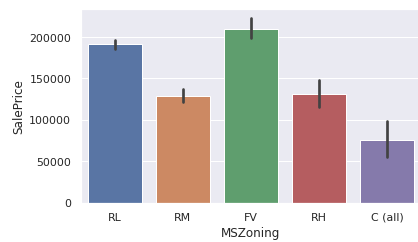


* Visualizations





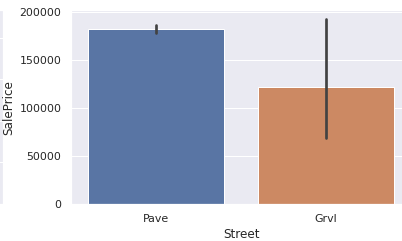




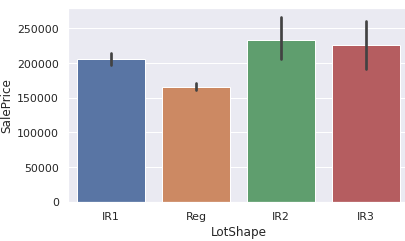
MSZoning: Identifies the general zoning classification of the sale.

The above graph shows that housing in Floating Village Residential

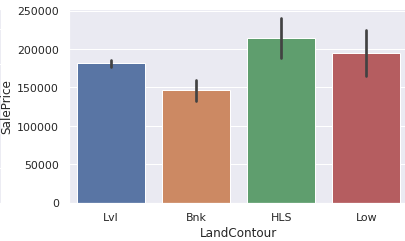
has the highest sales price among all the zoning followed of RL, RM, RH and C



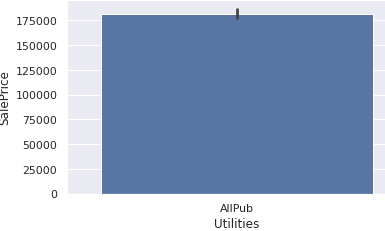
Housing with paved street are comparatively more expensive than the Gravel street.



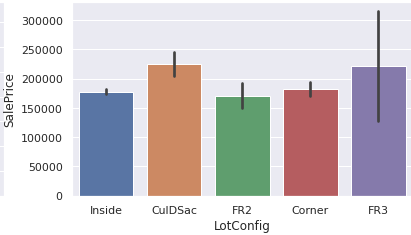
General shape of property -Housing with IR2 lot shape has the highest sales prices followed by IR3, IR1 and Reg.



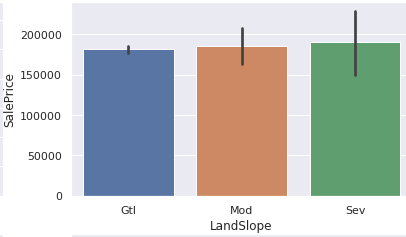
Flatness of the property - Housing with HLS land contour has the highest sales prices followed by IR3, IR1 and Reg.



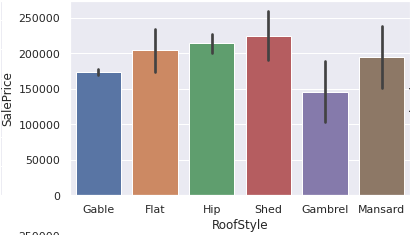
Utilities Available – All the housing available for sale have pub.



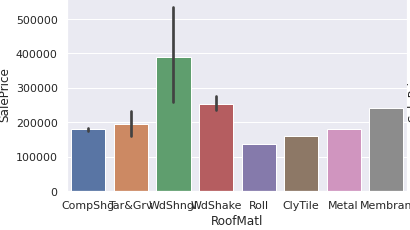
The FR3 Lot Configuration housing has the highest sale price followed by CulDSac, Corner, Inside, FR2.



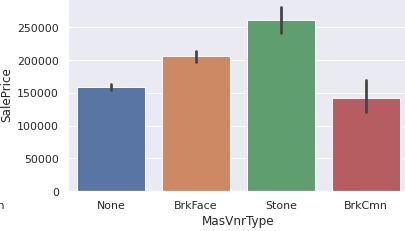
Housing with Severe slope are slightly more expensive than Moderate and Gentle slope.



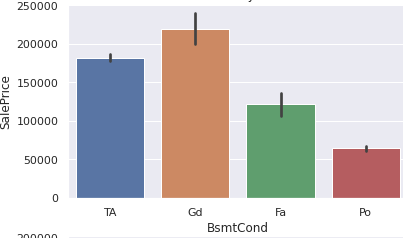
Housing with shed roofstyle has highest sales price followed by Hip, Flat, Mansard, Gable and Gambrel



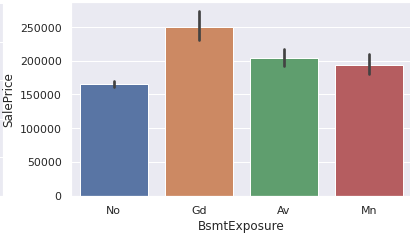
Housing that used Wood Shingles as roof material has highest sales price followed by WoodShake, Membran, Tar & Gravel, Composite Shingles, Metal, Clay tile and roll.



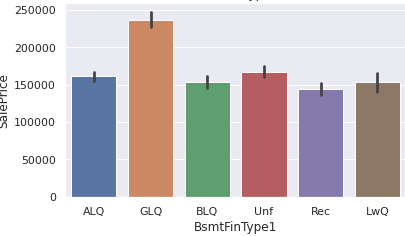
Masonry veneer type – Stone MasVnrType has highest housing sales price followed by brick face, brick common and housing with no masvnrtype



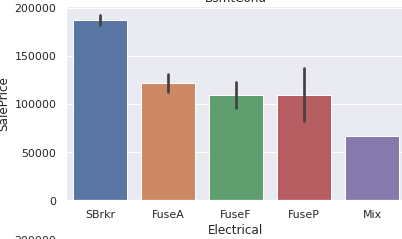
Basement with good condition is highly valued over typical, fair and poor basement condition.



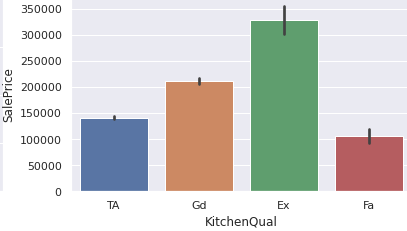
Similar to basement condition, basement with good exposure is highly valued over average, minimum or no exposure.



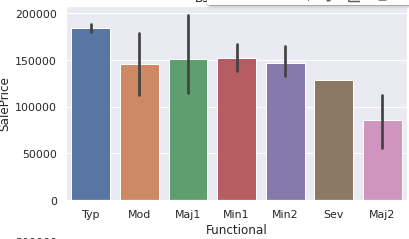
Basement with GLQ (good living quarters) is more expensive than the rest.



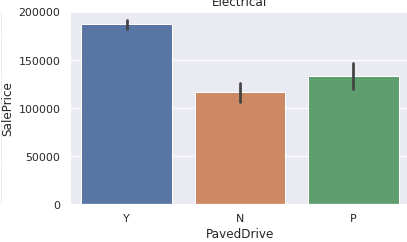
Housing with standard circuit breaker has highest sale price followed by fuseA, fusef, fuseP and mix.



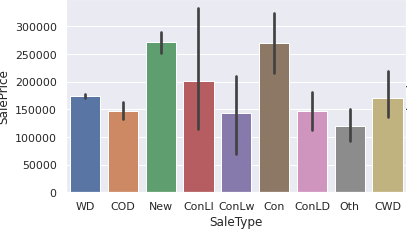
Excellent kitchen quality housing is more expensive than the other.



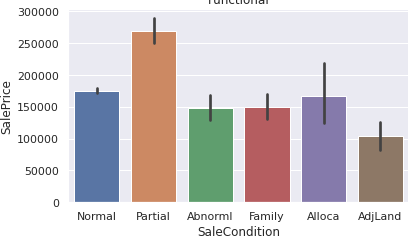
Typical home functionality has highest sales price whereas home functionality with major deduction 2 has lowest sales price.



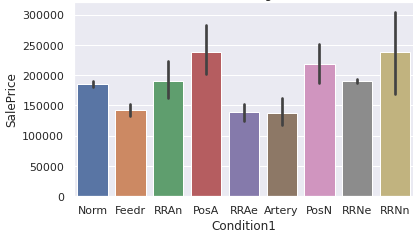
Housing with the paved driveway are more expensive than housing with no or partial driveway.

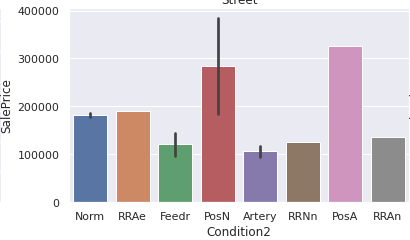


Newly built housing has highest sales price followed by housing with contract of 15% Down payment.

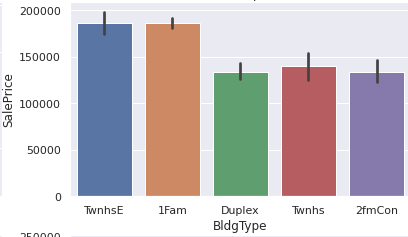


Partial sale condition i.e. Home that was not completed when last assessed (associated with New Homes) has highest sales price whereas housing with AdjLand i.e. Adjoining Land Purchase sale condition has lowest sale price.

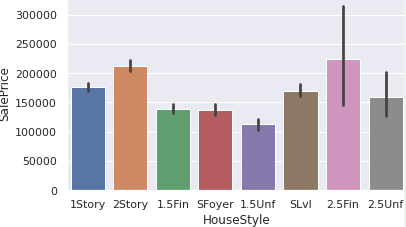




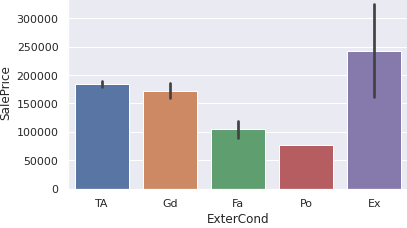
Proximity to various conditions - Housing with PosA(Adjacent to postive off-site feature) condition 1 and 2 has highest sales price.



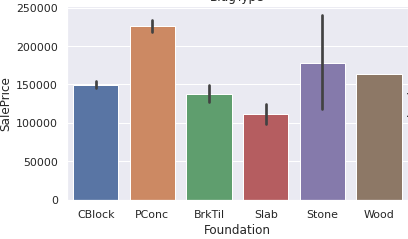
Type of dwelling - Housing with TwnhsE(Townhouse End Unit) and 1Fam (Single-family Detached)



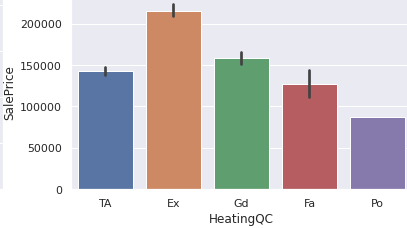
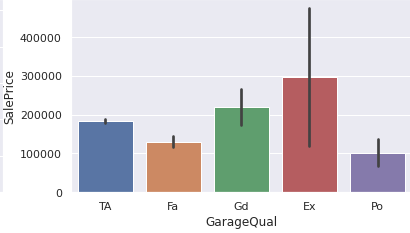
Housing with 2.5Fin(Finished) housing style has highest sales price whereas 1.5Unf(Unfinished) has lowest sale price.

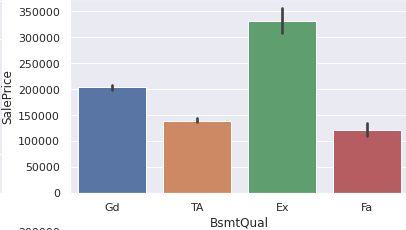
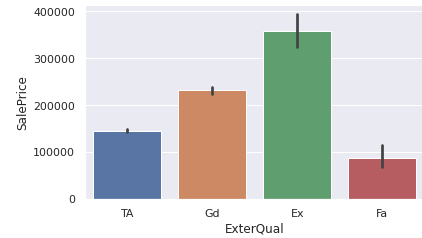


Excellent exterior condition house has highest sale price among all.

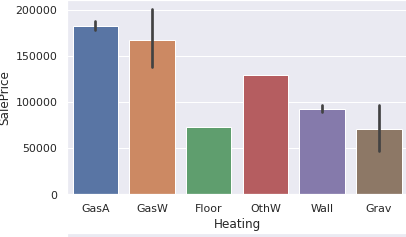


Foundation with PConc (Poured Concrete) has highest sale price where housing with slab has lowest sale price

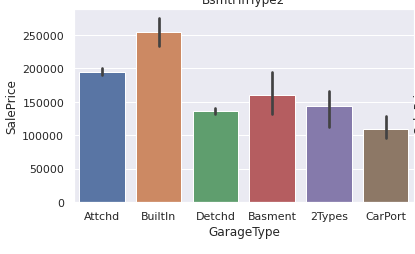
 

Housing with excellent garage, basement, exterior and heating quality has highest sale price among all.



Type of heating – Housing with Gas A i.e. Gas forced warm air furnace has highest sales price.



Housing with Builtin garage is more expensive than the rest garage type.

* Interpretation of the Results
* Housing with Excellent Exterior Quality, Exterior condition, Garage condition, Heating quality, kitchen quality are comparatively more expensive than the rest housing.
* The total area of the house, availability of garage area and basement, total number of rooms and the construction year also impact the overall sale price.
* Around 41% of the houses i.e. 482 houses were sold in 2nd quarter of the financial year.
* June was the best month interms of property sale, with total of 203 houses being sold followed by July (187).

**CONCLUSION**

* Key Findings and Conclusions of the Study
* In this study we found that linear regression algorithm performs slightly better than random regression and rest of the algorithm tested.
* Overall quality, above ground living area, total basement area, garage area/garage cars, 1st Floor area, total rooms above ground, year built, remodel date, Masonry veneer area are the top features that highly impact the sale price among all the features in the dataset.
* It is observed that around 41% of the houses (i.e. 482) were sold in 2nd quarter of the financial year.
* June was the best month interms of property sale, with total of 203 houses being sold followed by July (187).
* Limitations of this work

Although we were successfully able to find the top features that contributed to the sale price, our best saved model has mean absolute error of 19839.73, thus indicating that the margin of difference between actual price and the predicted price is around 19839.73 Australian dollars.