NAME OF THE PROJECT

Housing Price Prediction Project

Submitted by:

DOLYPONA DAS

ACKNOWLEDGMENT

I would like to thank our SME(Shubham Yadav) for his expert advice and encouragement throughout this project, and I also took some help from googled documents.

INTRODUCTION

- **Business Problem Framing**
  - 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.
  - We are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.
- **Conceptual Background of the Domain Problem**

  1.Firstly we will check whether the problem is supervised or not, mostly we will have a supervised one where there will be a target variable.

  2. After that we will check whether the project is a regression type or a classification type.

  3.If we have a train and a test dataset ,then we will do all the procedures with the train dataset and the same with the test dataset.

  3. We will also check whether our dataset is balanced or imbalanced. If it is an imbalanced one, we will apply sampling techniques to balance the dataset.

4. Then we will do model building and check its accuracy.
5. Our main motto is to build a model with good accuracy and for that we will also go for hyperparameter tuning.

- **Review of Literature**

I am summarizing my research done on the topic.

- I have imported important libraries for my project.
- I have created the dataframe for the train dataset. I have analysed my data by checking its shape, number of columns, presence of null values if any and checking the datatypes.
- Then I have done some data cleaning steps, e.g. checking the value counts of the target variable, dropping some irrelevant columns from the dataset, checking correlation between the dependant and independent variables, visualizing data using bar-plots, splitting the data into independent and dependant variables and finally scaling the data. Now I have used DecisionTreeRegressor,KNeighborsRegressor,AdaBoostRegressor,LinearRegression,GradientBoostingRegressor,RandomForest Regressor for model building and found **GradientBoostingRegressor** with the highest accuracy. I have seen **91%** accuracy after hyperparameter tuning (RandomizedSearchCV).
- Then I have created the dataframe for the test dataset and follow all the cleaning steps as I have done in the train dataset.
- I have trained my model on my train dataset and predict on my test dataset.

- **Motivation for the Problem Undertaken**

The objective behind to make this project is to contribute to the world's economy. 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 provides motivation as it can 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.

Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem**

If we look at data science, we are actually using mathematical models to model (and hopefully through the model to explain some of the things that we have seen) business circumstances, environment etc and through these models, we can get more insights such as the outcomes of our decision undertaken, what should we do next or how shall we do it to improve the odds. So mathematical models are important, selecting the right one to answer the business question can bring tremendous value to the organization.

Here, I am using six Regressor models, out of which Gradient Boosting Regressor is giving me a good accuracy on my train dataset after hyperparameter tuning.

- **Data Sources and their formats**

  Data Source: The read_csv function of the pandas library is used to read the content of a CSV file into the python environment as a pandas DataFrame. The function can read the files from the OS by using proper path to the file.

  Data description: Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. When this method is applied to a series of string, it returns a different output

- **Data Preprocessing Done**
  - I have dropped the column ‘Id’ as it is of no use, it is only giving some numbers.

- Then I have checked the shape and datatypes of the train dataset.
- Again I have dropped some columns which are not efficient.
- I have checked the value counts of the necessary columns and also checked if any null values are present.
- I have used Label Encoder to encode the object type columns and used mean to replace the null values.
- I have done some visualizations using heatmap,bar-plots,cat-plot and scatter-plots.
- I have checked the correlation between dependant and independent variables using heatmap.
- I have splitted the dependant and independent variables into x and y.
- I have scaled the data using StandardScaler method and made my data ready for model building.

- **Data Inputs- Logic- Output Relationships**

  The logic between the data input and data output is that how the input variables are influencing the output variable. Here the output variable is the 'SalePrice' and there are 80 input variables which are responsible for predicting the output variable in an efficient way.

  We have to look at the input variables and check how much they are correlated with the target variable and accordingly we have to work on it.

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

I have made some assumptions like I have dropped some columns from the dataset as they are irrelevant from my point of view.

I have made this assumption by looking at the input variables which help me predict my target variable in more efficient way.

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

  Hardware requirements:
  Processor: Intel(R) Celeron(R) CPU N3050 @1.60 GHz 1.60 GHz
  RAM: 3.92 GB
  System type: 64-bit operating system, x64-based processor
  Software requirements:
  Python: One of the most used programming languages
  Tools used:
  Jupyter notebook: Jupyter is a free, open-source,interactive web tool known as a computational notebook where I have written my python codes.
  NumPy: NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures.
  Pandas: Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.
  Matplotlib: It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python.
  Seaborn: It is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy.
  Scikit-learn: It is probably the most useful library for machine learning in Python.The sklearn library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction.
  Scipy.stats: This module contains a large number of probability distributions as well as a growing library of statistical functions.
  Imbalanced-learn: Imbalanced-learn (imported as **imblearn**)

is an open source, MIT-licensed library relying on scikit-learn (imported as sklearn) and provides tools when dealing with classification with imbalanced classes.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  1. To check the correlation among the data,I have used **heatmap** to visualize it.
  2. To get a clear view of the input variable columns and the target column visually,I have used **bar plots**.
  3. For scaling the data, I have used **StandardScaler** method.
  4. For training and testing the data, I have imported **train_test_split library** from scikit-learn.
  5. For model building, I have used **GradientBoostingRegressor** on my train dataset .
  6. For better accuracy of the model, I have used **hyperparameter tuning (RandomizedSearchCV).**

- Testing of Identified Approaches (Algorithms)

I have used six algorithms for testing, out of which Gradient Boosting Regressor gives a good accuracy score. So, I have selected this algorithm for my project.

Gradient Boosting Algorithm: Gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As we know that the errors in machine learning algorithms are broadly classified into two categories i.e. Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model.

Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is Mean Square Error (MSE) and when it is used as a classifier then the cost function is Log loss.

- Run and Evaluate selected models

```
In [1020]: model=[DecisionTreeRegressor(),KNeighborsRegressor(),AdaBoostRegressor(),LinearRegression(),GradientBoostingRegressor(),RandomFo
           max_r2_score=0
           for r_state in range(40,90):
               train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=r_state,test_size=0.33)
               for i in model:
                   i.fit(train_x,train_y)
                   pre=i.predict(test_x)
                   r2_sc=r2_score(test_y,pre)
                   print('r2 score correspond to random state',r_state,'is',r2_sc)
                   if r2_sc>max_r2_score:
                       max_r2_score=r2_sc
                       final_state=r_state
                       final_model=i
           print()
           print()
           print()
           print()
           print('max r2 score correspond to random state',final_state,'is',max_r2_score,'and model is',final_model)
```

I have selected six regression models for my train dataset, now I am attaching the model which gives a good accuracy score.

```
r2 score correspond to random state 87 is 0.7808002812118124
r2 score correspond to random state 87 is 0.8261450299300043
r2 score correspond to random state 87 is 0.8418038211950916
r2 score correspond to random state 88 is 0.6661704873676211
r2 score correspond to random state 88 is 0.8320386166561077
r2 score correspond to random state 88 is 0.8276869656914938
r2 score correspond to random state 88 is 0.8477748814130457
r2 score correspond to random state 88 is 0.9004996679469621
r2 score correspond to random state 88 is 0.8847259460946308
r2 score correspond to random state 89 is 0.7225355613709114
r2 score correspond to random state 89 is 0.795174445723673
r2 score correspond to random state 89 is 0.8228472466058746
r2 score correspond to random state 89 is 0.8258541308493501
r2 score correspond to random state 89 is 0.8317022997767474
r2 score correspond to random state 89 is 0.8598294526706354



max r2 score correspond to random state 50 is 0.9024619242067111 and model is GradientBoostingRegressor()
```

Gradient Boosting Regressor gives a good r2 score i.e 90%
The metrics that I have used is given in the screenshot below.

```
In [1029]: from sklearn.metrics import mean_absolute_error
           print('Mean Absolute Error:',mean_absolute_error(test_y,y_pred))
           print('Mean Squared Error:',mean_squared_error(test_y,y_pred))
           print('Root Mean Absolute Error:',np.sqrt(mean_absolute_error(test_y,y_pred)))

           Mean Absolute Error: 16128.679107135193
           Mean Squared Error: 565269816.3852301
           Root Mean Absolute Error: 126.99873663598072
```

Now I will show the result after hyperparameter tuning which is used to increase the accuracy score of the model.

```
n [1040]: gb=GradientBoostingRegressor(n_estimators=300,min_samples_split=4,min_samples_leaf=2,max_features='sqrt',max_depth=5,learning_rat
gb.fit(train_x,train_y)
gb.score(train_x,train_y)
pred=gb.predict(test_x)
gbr=r2_score(test_y,pred)
print('R2 score:',gbr*100)

R2 score: 91.43201385045438
```

The final r2 score that I have achieved is 91%.

- Key Metrics for success in solving problem under consideration

There are three error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are:

- Mean Squared Error (MSE).
- Root Mean Squared Error (RMSE).
- Mean Absolute Error (MAE)

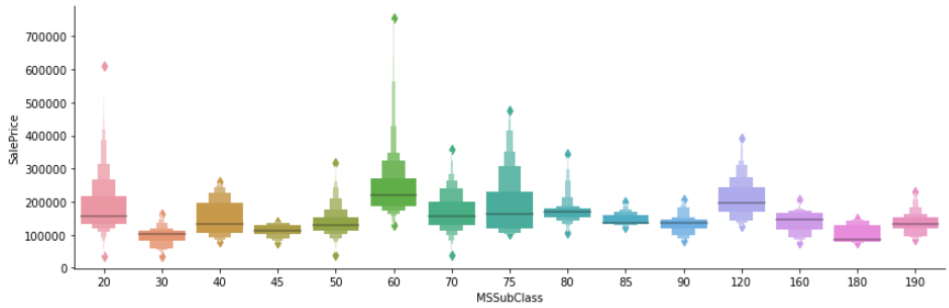
```
In [1029]: from sklearn.metrics import mean_absolute_error
print('Mean Absolute Error:',mean_absolute_error(test_y,y_pred))
print('Mean Squared Error:',mean_squared_error(test_y,y_pred))
print('Root Mean Absolute Error:',np.sqrt(mean_absolute_error(test_y,y_pred)))

Mean Absolute Error: 16128.679107135193
Mean Squared Error: 565269816.3852301
Root Mean Absolute Error: 126.99873663598072
```

- Visualizations

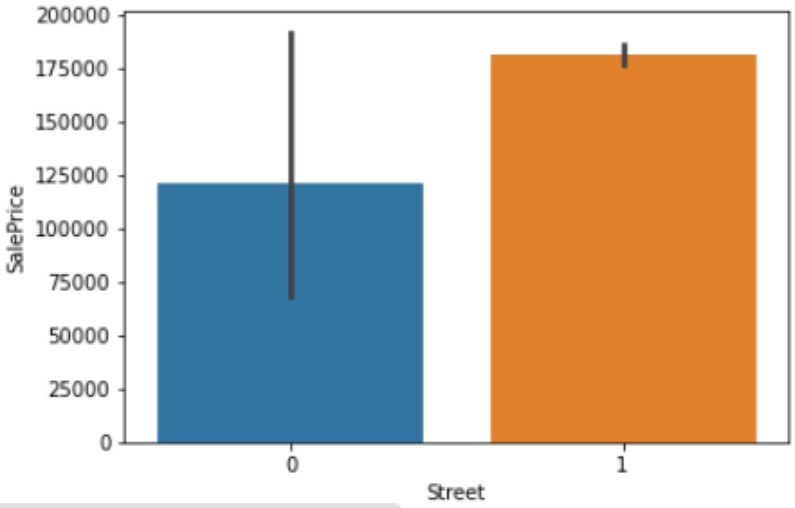
I will put the screenshots of some graphs which will show the relation between the target variable and the input variable.

```
In [825]: #MSSubClass Vs SalePrice
sns.catplot(y = "SalePrice", x = "MSSubClass", data = df.sort_values("SalePrice", ascending = False), kind="boxen", height = 4,
plt.show()
```



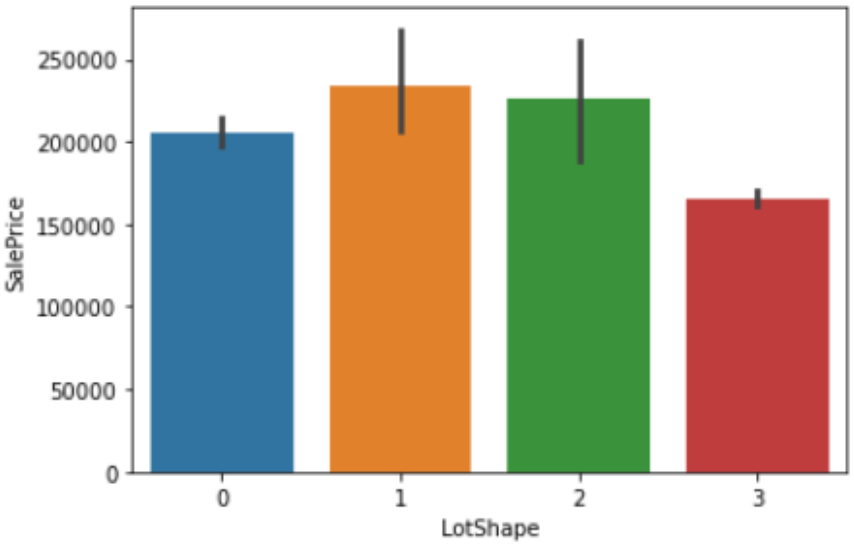
In [839]:
```
sns.barplot(x='Street',y='SalePrice',data=df)
```

Out[839]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x228f3da490>
```



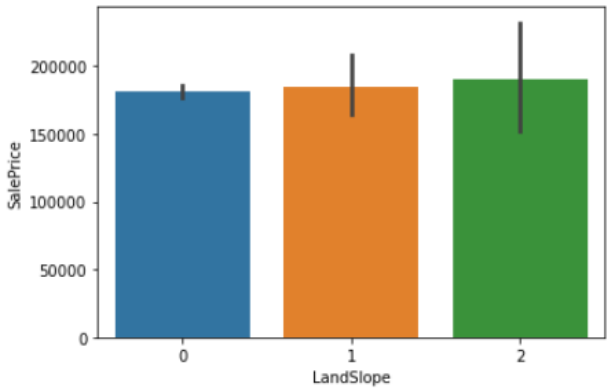
In [845]:
```
sns.barplot(x='LotShape',y='SalePrice',data=df)
```

Out[845]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x228f445970>
```



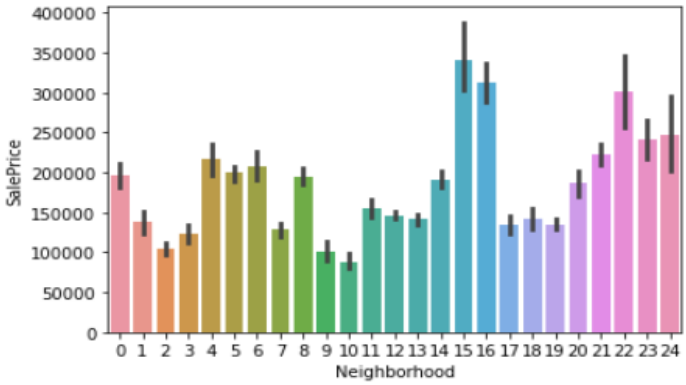
In [861]:
```
#Let us see how this feature is related to the target variable 'SalePrice'
sns.barplot(x='LandSlope',y='SalePrice',data=df)
```

Out[861]: <matplotlib.axes._subplots.AxesSubplot at 0x228f5dce80>



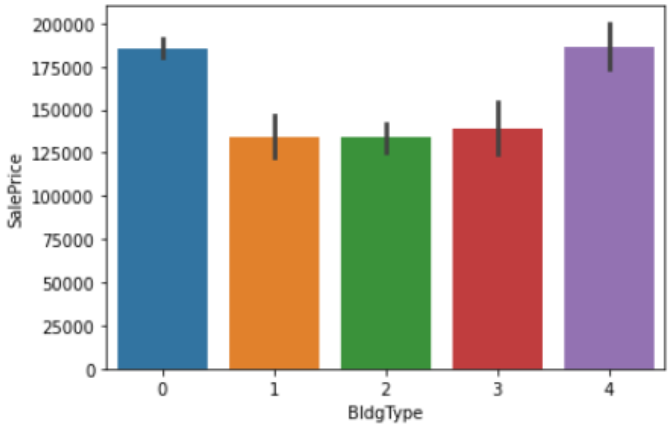
In [866]:
```
#Let us see how this feature is related to the target variable 'SalePrice'
sns.barplot(x='Neighborhood',y='SalePrice',data=df)
```

Out[866]: <matplotlib.axes._subplots.AxesSubplot at 0x228f630130>



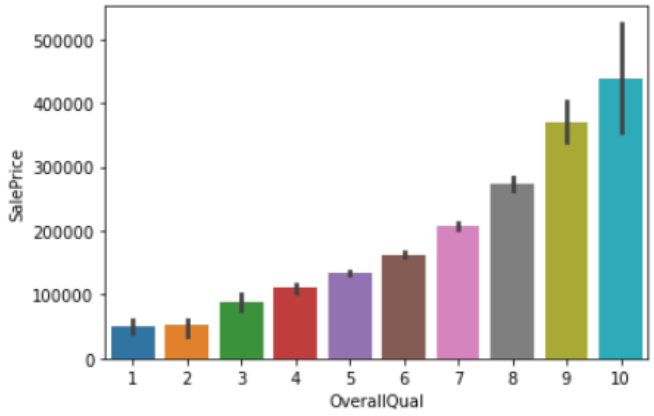
In [876]:
```
#Let us see how this feature is related to the target variable 'SalePrice'
sns.barplot(x='BldgType',y='SalePrice',data=df)
```

Out[876]: <matplotlib.axes._subplots.AxesSubplot at 0x2290f87d30>



```
In [881]: #Let us see how this feature is related to the target variable 'SalePrice'
          sns.barplot(x='OverallQual',y='SalePrice',data=df)
```

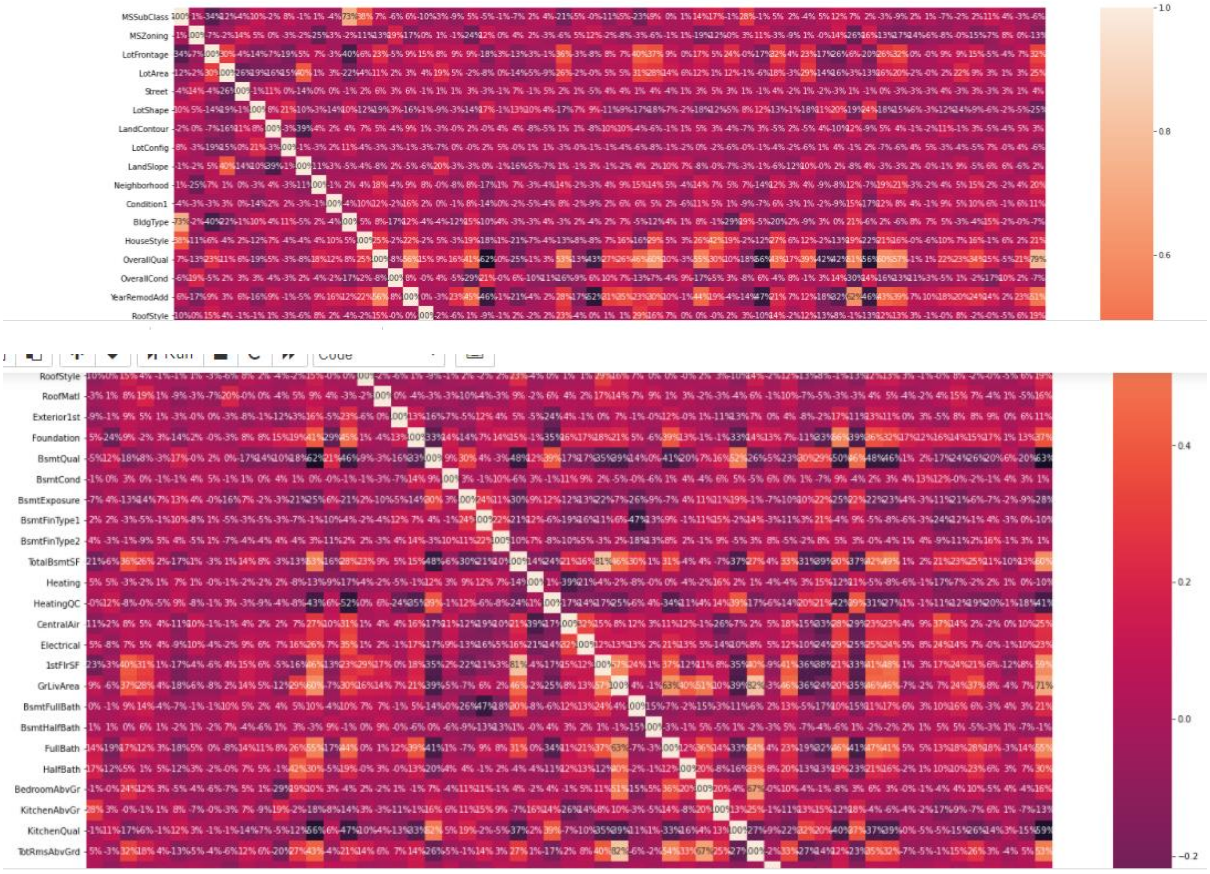
```
Out[881]: <matplotlib.axes._subplots.AxesSubplot at 0x228f5e4310>
```

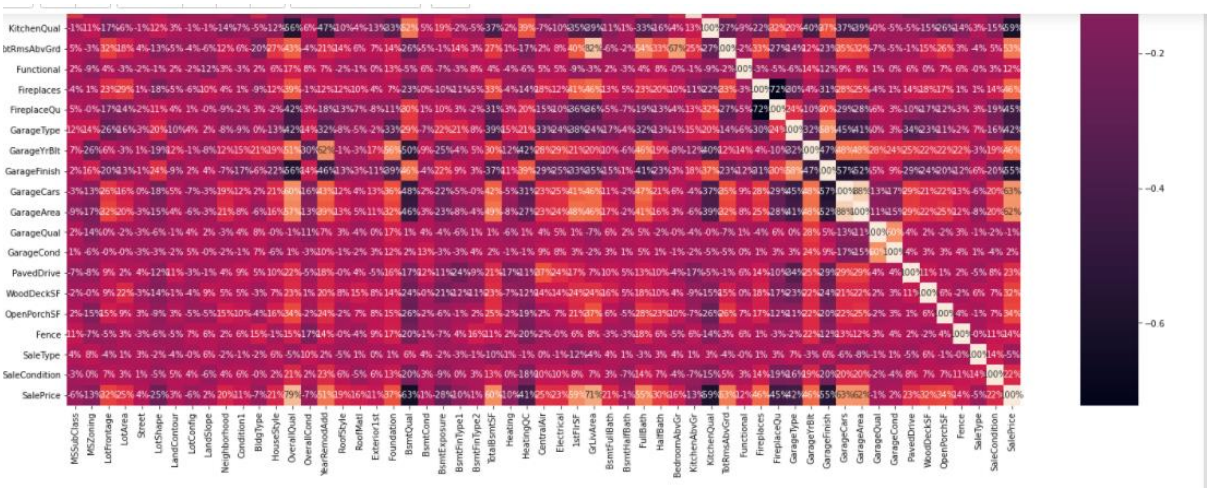


These are some of the barplots and catplot which give us a view how the input variable is related to the output.

```
In [1016]: #Visualizing the correlation
           plt.figure(figsize=(25,25))
           sns.heatmap(df.corr(), annot=True, fmt='.0%')
```

```
Out[1016]: <matplotlib.axes._subplots.AxesSubplot at 0x229196e100>
```





This is the heatmap where I have checked the correlation between all the data.

- Interpretation of the Results

In the visualization part, I have seen how my data is correlated with each other and also checked how my input variables are correlated with the target variable. For this work, I have used heatmap, barplots and catplot.

In the pre-processing part, I have cleaned my data in various ways. Firstly,I have dropped some columns which I feel are not contributing in predicting the target variable. Then I have used LabelEncoder to encode the object type data because a machine can only read numbers. I also replaced the NaN values using mean .

In the modelling part, I have designed our model using GradientBoostingRegressor model, where r2score, mean absolute error, mean squared error and root mean absolute error are calculated for the model.

CONCLUSION

Key Findings and Conclusions of the Study

The key findings are we have to study the data very clearly so that we are able to decide which data are relevant for our findings. The techniques that I have used are heatmap, LabelEncoder,etc.

The conclusion of our study is we have to achieve a model with good accuracy.

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

We will develop relevant programming abilities. We will demonstrate proficiency with statistical analysis of data. We will develop the ability to build and assess data-based models. We will execute statistical analyses with professional statistical software. The best algorithm for this project according to my work is Gradient Boosting Regressor because the accuracy that I have achieved is quite satisfactory than the other models.

- ## Limitations of this work and Scope for Future Work

The results were promising for the public data due to it being rich with features and having strong correlation, whereas the local data gave a worse outcome when the same pre-processing strategy was implemented due to it being in a different shape compared with the public data in terms of the number of features and the correlation strength. Hence, the local data needs more features to be added preferably with a strong correlation with the house price.

Future scope of this work is we can try different algorithms like Lasso Regression, Ridge Regression etc for model building and try to achieve a good accuracy and f1-score.