HOUSING PRICE PREDICTION PROJECT

Submitted by:

SURAJ CHAKRABORTY

ACKNOWLEDGMENT

First and foremost, praises and thanks to the God, the Almighty, for his Shower of blessings throughout my project work to complete the project successfully.

I would like to express my deep and sincere gratitude to my SME Mr. Sajid Choudhary of Flip Robo technology, for giving me the opportunity to make this project and providing guidance on how to make the project. It was a great privilege to work under his mentorship. I would also like to thank my datatrained institute mentors for giving me all the knowledge in data science, which I am able to implement now for making the project. The projects which I had made while learning with datatrained helped me a lot in providing reference to make this project. The files attached with the data file of this project also helped me to get a better understanding of the project and steps that I needed to follow. Also the internet played a key role in helping me make the project as whenever I got stuck, I looked up on the internet for possible solutions that I could use and also to get rid of errors.

I am extremely grateful to my mother for her love, prayers and care and sacrifices for educating and preparing me for my future.

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.

  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.

  This project will help companies like this to get a better understanding of the business and the factors that may help the company to grow. The companies can then put the knowledge into implementation and put their focus into those factors and save the time and money spend on the non-important factors.

- ## Conceptual Background of the Domain Problem

  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.

  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.

  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?

- ## Review of Literature

  The internet provides a lot of information regarding the topic. There are many articles regarding the topic in websites like towardsdatascience.com, kaggle.com, medium.com etc. These websites gave better understanding of the project to me and how to approach and deal with the problems. Also the data description provided to us gives us a thorough understanding of the dataset.

- ## Motivation for the Problem Undertaken

  The objective is to provide the company with the best model, which can 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. What motivated me to take the project is the dynamic data we need to deal with in the project and the large number of features in this project which will enhance my knowledge on dealing with different types of problems in data science.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  We build a model which can be used to predict the price of the house based on the features provided. Data contains Null values, we treated them accordingly, and also extensive EDA had to be performed to gain relationships of important variable and price. In the Machine Learning models, we had to apply regularization and determine the optimal values of Hyper Parameters and needed to find important features which affect the price positively or negatively.

- Data Sources and their formats

  The company 'Surprise Housing' has collected a data set from the sale of houses in Australia. And we are provided with two datasets, one for the training and building the model, and the other for testing the model by predicting the target values. The data present in the dataset belong to all kinds of categories. There are numeric continuous as well as discrete data, and many categorical object type data are also present in the dataset. All the data given are related to the house in some way or other, but whether they are related to the house price or not, that we needed to find out.

- Data Pre-processing Done

  - First we dropped some unwanted columns from the dataset which had no role to play.

```
# dropping unwanted columns

test.drop(['Id','Utilities'],axis=1, inplace=True)
```

Screenshot of the code

  - Checked for missing values in the dataset. We found some columns containing missing data. In the categorical data, the missing values were present where the feature of the house was not present. So we replaced them with 'Unavailable'. For

the numeric data, we replaced the missing values with the median of the column.

```
# Replacing the missing values

from sklearn.impute import SimpleImputer

imp=SimpleImputer(strategy= 'most_frequent')
df['MasVnrType']= imp.fit_transform(df['MasVnrType'].values.reshape(-1,1))
```

```
df[catg_fetrs]= df[catg_fetrs].fillna('Unavailable')
```

Screenshot of the code for categorical data

```
# Replacing the missing values

df['GarageYrBlt']= imp.fit_transform(df['GarageYrBlt'].values.reshape(-1,1))

df[num_fetrs]=df[num_fetrs].fillna(df[num_fetrs].median())
```

Screenshot of the code for numeric data

- Extracted information from the dates. There were some features containing the year of its built, which was of no use in the model building. Hence we extracted the number of years of built with respect to the year of sale and kept them in the same columns. In further steps we encoded the year of sale column as there were only few categorical years present.

```
df.drop(['GarageYrBlt'],axis=1, inplace=True)
num_fetrs.remove('GarageYrBlt')
```

```
years= ['YearBuilt','YearRemodAdd']

for i in df[years]:
    df[i]=df['YrSold']-df[i]
```

Screenshot of the code

- Visualized the distribution of data in different features of the dataset.

```
# Visualizing the distribution of continious numeric data

df[num_fetrs].plot(kind='density', subplots= True, layout=(4,5), sharex=False, legend=True, figsize=[25,16])
plt.show
```

Screenshot of one of the code

- Encoded the categorical object type data in the dataset.

```
qual_fetrs=['ExterQual','ExterCond','BsmtQual','BsmtCond','HeatingQC','KitchenQual','FireplaceQu','GarageQual','GarageCond','Pool

label={'Ex':0,'Gd':1,'TA':2,'Fa':3,'Po':4,'Unavailable':5}

for i in df[qual_fetrs]:
    df[i]= df[i].map(label)
```

In the above code, we are manually encoding the features containing the quality of that feature by mapping the labels to its categories, this is done manually because by using label encoder, the label were not in ordinal manner.

```
# encoding the rest of the object type data.

from sklearn.preprocessing import LabelEncoder

enc= LabelEncoder()
for i in df[catg_fetrs]:
    if df[i].dtypes== 'O':
        df[i]= enc.fit_transform(df[i].values.reshape(-1,1))

# we need to encode the 'year sold' column as well, as it contains five categorical years

df['YrSold']= enc.fit_transform(df['YrSold'].values.reshape(-1,1))
```

Screenshot of the code

- Checked the correlation of the features with the target using different visualization techniques and then removed the features that showed no correlation with the target.

```
plt.figure(figsize=(25,8))
cor['SalePrice'].sort_values(ascending=False).drop(['SalePrice']).plot(kind='bar')
plt.xlabel('Feature')
plt.ylabel('Correlation with target')
plt.title('Correlation')
plt.show()
```

Screenshot of one of the codes

- Checked for outliers in the dataset. We found some outliers present and then removed a considerable amount of outliers using IQR technique.

```
q1= df[num_fetrs].quantile(0.37)
q3= df[num_fetrs].quantile(0.60)
iqr= q3-q1
```

```
new= df[((df<(q1-(1.5*iqr))) | (df>(q3+(1.5*iqr)))).any(axis=1)]
new.shape
```

```
(1115, 73)
```

Screenshot of the code

- Treated skewness in the dataset. We also found some skewness present in the continuous numeric data, we them minimised the skewness using log transformation technique.

```
for i in new[num_fetrs]:
    if 0 in new[i].unique():
        pass;
    else:
        new[i]=np.log(new[i].values.reshape(-1,1))

new[num_fetrs].skew()
```

Screenshot of the code

- Scaled the features of the dataset.

```
# First lets split the data into target and features.

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

```
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()
x=pd.DataFrame(sc.fit_transform(x), columns=x.columns)
x
```

Screenshot of the code

- Performed feature selection on the dataset and kept the best scoring features in the dataset.

```
# importing required libraries

from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel
```

```
fetr_sel= SelectFromModel(Lasso(alpha=0.007, random_state=4))
fetr_sel.fit(x,y)

selected= x.columns[(fetr_sel.get_support())]
len(selected)
```

```
32
```

Here we are using embedded method for the feature selection and we have used Lasso regularization model to select the best performing features, keeping the alpha value as '0.007', which then gave us 32 best scoring features from the dataset.

```
x=x[selected]
x.head()
```

Screenshot of the code

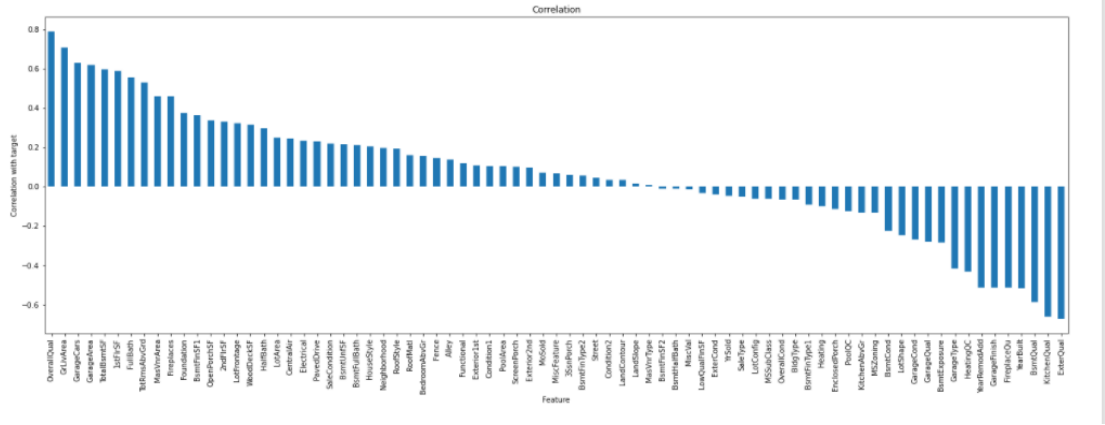
- Data Inputs- Logic- Output Relationships

We checked the distribution of the independent columns in the dataset and found some columns being normally distributed also found many columns having skewness in the data. We then checked the correlation among all the columns of the dataset and then visualized it.

We also visualized the correlation of independent columns with the target separately, where we found the continuous features ‘overall quality’, ‘ground living area’, ’Garage area’, ‘Basement area’, ‘1st floor area’ showing a positive correlation with the target, which meant that as these feature value increases, the sale price also increases. Also the categorical features such as ‘kitchen quality’, ‘year of built’, ‘external quality’, ‘basement quality’ play a positive role in the sale price of the house. Better the quality of these features, higher is the price of the house.

But the features such as ‘Land slope’, ‘masonry veneer type’, ‘basement finished area quality’, ‘basement half bath’, ‘value of miscellaneous feature’ showed no correlation with the target value, a near zero correlation was found between them.

```
plt.figure(figsize=(25,8))
cor['SalePrice'].sort_values(ascending=False).drop(['SalePrice']).plot(kind='bar')
plt.xlabel('Feature')
plt.ylabel('Correlation with target')
plt.title('Correlation')
plt.show()
```



Screenshot of correlation of target with all features

- ## Hardware and Software Requirements and Tools Used

Hardware required: Minimum Intel i3 processor with 8 GB of RAM.

Operating system: Windows 7, 8, or 10 or LINUX or MacOS

Tools used:

- Jupyter notebook: For the coding and creating the project.
- Microsoft edge: For hosting.
- Microsoft word: For making the project report.
- Microsoft PowerPoint: For making the PPT.

Libraries & Packages used:

- Numpy: Used to perform log transformation on the dataset for treating the skewness. Also used to find the exponent of the final predicted result to inverse the logarithm and round up the values.
- Pandas: Used to load the dataset into the jupyter notebook and controlling the viewing pattern of the data.

- Matplotlib: Uses to visualize and plot the distribution of data in columns as well as the relation between different columns.
- Seaborn: Used to visualize the distribution and correlation between data using different types of figures and graphs.
- Sklearn: Used to perform different operations using different modules in it.
    - SimpleImputer: Used to replace missing values present in the dataset.
    - StandardScaler: Used to scale the dataset.
    - SelectFromModel: Used to perform feature selection on the dataset.
    - Using sklearn we also imported different models for the training and the testing of the data. E.g. LinearRegression, Lasso etc.
- Scipy: Used to get the z-score of the dataset to check for possible outliers.
- Joblib: Used to save the model that we created in .pkl format.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  The approach that we followed is:

  - Analysed the data by looking at the dataset its columns and the data inside it thoroughly.
  - Performed extensive EDA to get better insights of the dataset.
  - Checked for missing values and treated them accordingly.
  - Performed feature engineering to handle different types of data so that they can be used in modelling.
  - Encoded the categorical types of data.
  - Checked the correlation among all the columns in the dataset and performed visualization to see the relationship between the features and the target column.
  - Checked for outliers in the data and treated them.
  - Handled the skewness in the dataset.
  - Performed scaling on the features.
  - Performed feature selection on the dataset.
  - Pre processed the test dataset in the same way.
  - Found the best random state and preformed train test split using it.
  - Trained different models.
  - Checked CV score for all the models.
  - Hyper-parameter tuned the best performing models.
  - Saved the best found model as the final model.
  - Predicted the target for the test set using the final model.

- Testing of Identified Approaches (Algorithms)

  The algorithms that we used are:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor
- Support Vector Regressor
- Ridge regularization

- Run and Evaluate selected models

  - Linear Regression: It measures the relationship between continuous numeric dependent variable and the independent variables by estimating probabilities.

```
lr.fit(x_train,y_train)
predlr= lr.predict(x_test)

print('Score: ',lr.score(x_train,y_train))
print('r2 score: ', r2_score(y_test,predlr))
print('Mean absolute error:', mean_absolute_error(y_test,predlr))
print('Mean squared error:', mean_squared_error(y_test,predlr))
print('Root mean squared error:', np.sqrt(mean_squared_error(y_test,predlr)))
```

```
Score:  0.8751820323324736
r2 score:  0.9257450138280584
Mean absolute error: 0.08573202110240903
Mean squared error: 0.013262535626809431
Root mean squared error: 0.1151630827427324
```

Screenshot of the code

Using linear regression, we found the training score to be 85%, the R2 score to be 92%, MAE of 8%, MSE of 1% and RMSE of 11%.

  - Decision tree regressor: Builds regression models in the form of a tree structure. It breaks down the dataset into smaller subsets.

```
from sklearn.tree import DecisionTreeRegressor

dt= DecisionTreeRegressor()
dt.fit(x_train,y_train)
preddt= dt.predict(x_test)

print('Score: ',dt.score(x_train,y_train))
print('r2 score: ', r2_score(y_test,preddt))
print('Mean absolute error:', mean_absolute_error(y_test,preddt))
print('Mean squared error:', mean_squared_error(y_test,preddt))
print('Root mean squared error:', np.sqrt(mean_squared_error(y_test,preddt)))
```

```
Score:  0.9999999969784628
r2 score:  0.688555526834036
Mean absolute error: 0.15236746961568556
Mean squared error: 0.05562647889492546
Root mean squared error: 0.2358526635315477
```

Screenshot of the code

Using Decision tree regressor, we found the training score to be 99%, the R2 score to be 68%, MAE of 15%, MSE of 5% and RMSE of 23%.

- Random forest regressor: It builds multiple decision trees and merges them together to get a more accurate prediction.

```
from sklearn.ensemble import RandomForestRegressor

fr=RandomForestRegressor()
fr.fit(x_train,y_train)
predfr=fr.predict(x_test)

print('Score: ',fr.score(x_train,y_train))
print('r2 score: ', r2_score(y_test,predfr))
print('Mean absolute error:', mean_absolute_error(y_test,predfr))
print('Mean squared error:', mean_squared_error(y_test,predfr))
print('Root mean squared error:', np.sqrt(mean_squared_error(y_test,predfr)))
```

```
Score:  0.9781252968275947
r2 score:  0.9075221282638162
Mean absolute error: 0.08846131437039183
Mean squared error: 0.016517289030970154
Root mean squared error: 0.12851960562875284
```

Screenshot of the code

Using Random forest regressor, we found the training score to be 97%, the R2 score to be 90%, MAE of 8%, MSE of 1% and RMSE of 12%.

- Support vector regressor: It looks at data and sorts it into one of two categories. It helps in determining the closest match between the data points and the function which is used to represent them.

```
from sklearn.svm import SVR

svr= SVR()
svr.fit(x_train,y_train)
preds= svr.predict(x_test)

print('Score: ',svr.score(x_train,y_train))
print('r2 score: ', r2_score(y_test,preds))
print('Mean absolute error:', mean_absolute_error(y_test,preds))
print('Mean squared error:', mean_squared_error(y_test,preds))
print('Root mean squared error:', np.sqrt(mean_squared_error(y_test,preds)))
```

```
Score:  0.9519067108979391
r2 score:  0.849220123791198
Mean absolute error: 0.10271148371732713
Mean squared error: 0.026930494275423874
Root mean squared error: 0.16410513177662628
```

Screenshot of the code

Using support vector regressor, we found the training score to be 95%, the R2 score to be 84%, MAE of 10%, MSE of 2% and RMSE of 16%.

- Ridge regularization: It estimates the coefficients of multiple-regression models in scenarios where independent variables are highly correlated.

```
# Regularization

from sklearn.linear_model import Ridge

rd= Ridge()
rd.fit(x_train,y_train)
predrd= rd.predict(x_test)

print('Score: ',rd.score(x_train,y_train))
print('r2 score: ', r2_score(y_test,predrd))
print('Mean absolute error:', mean_absolute_error(y_test,predrd))
print('Mean squared error:', mean_squared_error(y_test,predrd))
print('Root mean squared error:', np.sqrt(mean_squared_error(y_test,predrd)))
```

```
Score:  0.8751814825355229
r2 score:  0.9257666075065198
Mean absolute error: 0.08572043322021027
Mean squared error: 0.013258678822779528
Root mean squared error: 0.11514633655822284
```

Screenshot of the code

Using Ridge regressor, we found the training score to be 87%, the R2 score to be 92%, MAE of 8%, MSE of 1% and RMSE of 11%.

- Key Metrics for success in solving problem under consideration

Key metrics used are:

- R2 score: R-squared is used to gives the measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination. In general, it is used to know the model performance. As, the higher the R-squared, the better the model fits the data.
- Mean absolute error: This is used because in statistics, the mean absolute error (MAE) is a way to measure the accuracy of a given model. This tells us that the average difference between the actual data value and the value predicted by the model. The lower the MAE for a given model, the more closely the model is able to predict the actual values.
- Mean squared error: The mean squared error (MSE) is used to determine the performance of the algorithm. The mean square error gives measure of the average of error squares

i.e. the average of the square of the difference between the observed and predicted values of a variable.

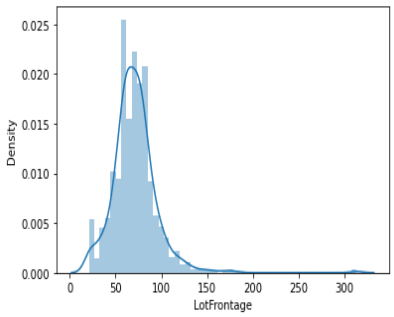
- Root mean squared error: The RMSE is used, as it is a good measure of accuracy. It is the square root of the mean of the square of all of the errors**.** Basically it gives the square root of the mean squared error.

- Visualizations

The plots that are used are:

- Distribution plot:

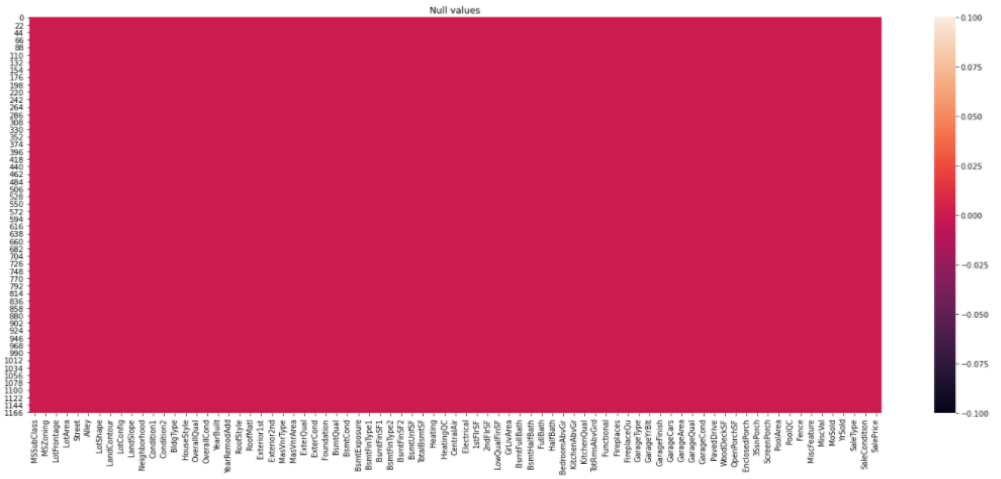
```
sns.distplot(df['LotFrontage']);
```



Screenshot of the plot

The distribution plot was used to check the distribution of data in the columns that contained missing values. We found that the data distribution had some skewness in it and therefore we replaced the missing values with the median of the column data.

- Heatmap:

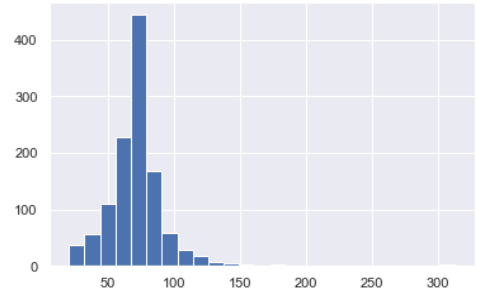


Screenshot of the plot

Heat map was used at two occasions in the project, once for checking any missing values in the dataset and the next time to plot the correlation matrix of the dataset. In the first case, we found some missing values and after treating them we can see no missing values present. And in the second case we found many types of correlation between the columns. There were some positive as well as negative relations in the matrix, and also some columns showed no correlation.

- Histogram:

```
sns.set(style='darkgrid')
plt.hist(df['LotFrontage'], bins=25)
plt.show()
```

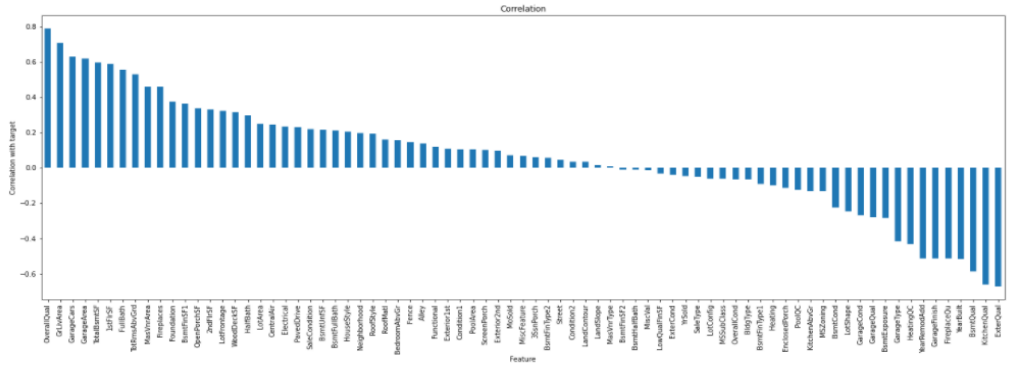


Screenshot of the plot

We used the histogram to see the distribution of data in the continuous numeric columns. We found few columns being close to normal distribution, and the majority of the columns were found to have variance and skewness in the data.

- Bargraph:

```
plt.figure(figsize=(25,8))
cor['SalePrice'].sort_values(ascending=False).drop(['SalePrice']).plot(kind='bar')
plt.xlabel('Feature')
plt.ylabel('Correlation with target')
plt.title('Correlation')
plt.show()
```

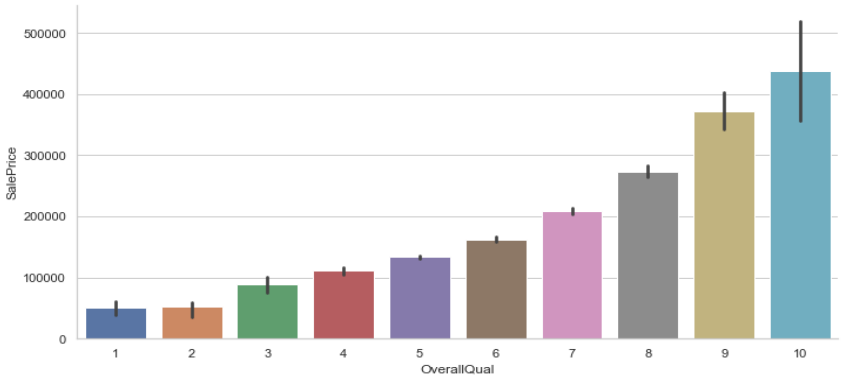


Screenshot of the plot

We plotted this bar graph to visualize the correlation o all the independent columns with the target column. We found mostly positive relations in between the columns. But there were some negative and near zero correlations as well.

- Catplot:

```
sns.set(style='whitegrid')
sns.catplot(x='OverallQual',y='SalePrice', data=df, kind='bar', aspect=2);
```

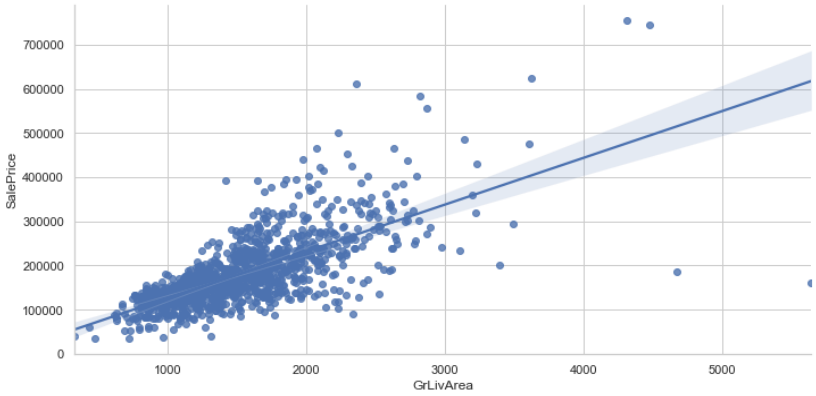


Screenshot of the plot

We used the catplot multiple times to visualize the relationship between an independent columns and the target. Here in this plot we found a strong correlation between the overall quality feature and the sale price. Which meant that as the overall quality increased the sale price also increased.

- Lineplot:

```
sns.lmplot(x ='GrLivArea', y ='SalePrice', data = df, aspect=2);
```

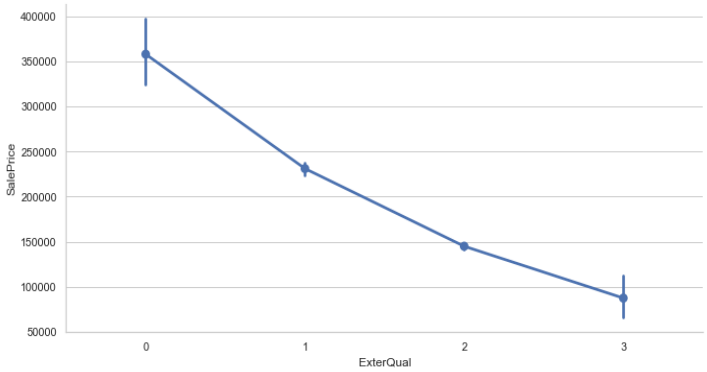


Screenshot of the plot

We used the lineplot multiple times to visualize the relationship between an independent columns and the target. Here in this plot we found a good correlation between the target and the ground living area column, which showed that the sale price increased as the ground living area increased.

- Factorplot:

```
sns.factorplot(x ='ExterQual', y ='SalePrice', data = df, aspect=2);
```

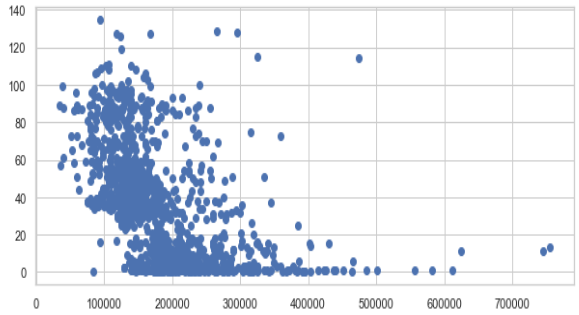


Screenshot of the plot

We used the factorplot multiple times to visualize the relationship between an independent columns and the target. Here in this plot we can see the line of the graph going down, but the external quality in the x axis is in encoded form and the value 0 in it represented an excellent quality and value 4 represented a poor quality, which means that as the external quality improved, the sale price also increased which meant a positive correlation.

- Scatterplot:

```
plt.figure(figsize=(10,4))
plt.scatter(df['SalePrice'],df['YearBuilt'])
plt.show()
```

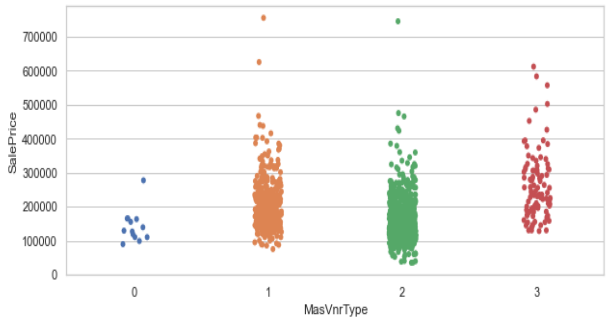


Screenshot of the plot

We used the Scatterplot multiple times to visualize the relationship between an independent columns and the target. The plot here showed a negative correlation between the columns, as the price data can be seen increasing as the years of built decreased. This meant that more new the house, the more costly it was.

- Stripplot:

```
plt.figure(figsize=[10,4])
sns.stripplot(data= df, x='MasVnrType', y='SalePrice')
plt.show()
```

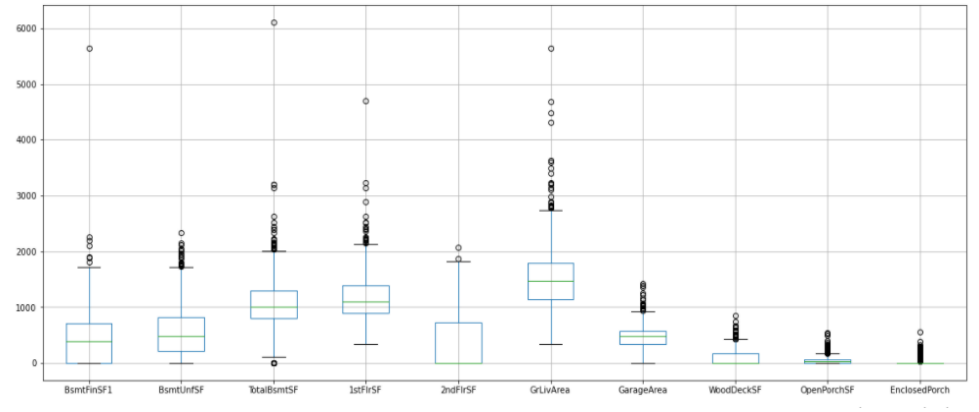


Screenshot of the plot

We used the Stripplot multiple times to visualize the relationship between an independent columns and the target. The plot here showed no correlation between the columns. As, the sale price range remains the same for all types of masonry veneer. This meant that the sale price does not depend on the masonry veneer type in any way.

- Boxplot:

```
df[num_fetrs].iloc[:,5:15].boxplot(figsize=[20,8])
plt.show()
```



Screenshot of the plot

We used the boxplot to check for any outliers in the dataset. In this plot we can see many outliers in the columns, some seem to be very close to the threshold but some also can be seen very far away from the threshold.

- Interpretation of the Results

From the pre-processing we interpreted:

- The dataset was fairly big regarding the number of features in it.
- Some unwanted columns were present in the dataset, which had no role to play in building the model.
- The dataset contained various types of data in it, which included continuous numeric data, discrete numeric data, categorical data and date and time data.
- There were many missing values in certain columns of the dataset, most of which were the non existing features in the particular feature column.
- Using encoder to encode the categorical ordinal data can sometimes give a wrong order.
- Removing too many outliers can lead to data loss, which can affect the model.

- Many features performed poorly in the feature selection process.

From the visualization we interpreted:

- The distribution of data in majority of the columns was not normal. It had some skewness and variance in it.
- The features like ‘overall quality’, ‘ground living area’, ‘garage area’, ‘total basement surface area’, ‘kitchen quality’, etc had a good positive correlation with the target.
- The features like ‘year built’, ‘year remodelled’ had a negative relationship with the target data.
- The features such as ‘masonry veneer type’, ‘land slope’, ‘miscellaneous feature value’, ‘basement half bath’, etc had no correlation with the target data.
- There were some outliers present in majority of the features in the dataset.

From the modelling we interpreted:

- Finding the best performing random state can be very useful in splitting the train and validation data.
- Scaled data give a higher model performance as compared to non-scaled data.
- In this dataset, when we log transformed the features to treat the skewness in them, log transforming the target along with them gave us very high performing models as compared to the target not been transformed.
- All the models performed very well, but the trained models can show a high accuracy due to overfitting.
- Performing cross validation test is very important to detect the overfitted or underfitted models.
- The model that has the least difference between the r2 score and the cv score can be considered the best performing model.
- Hyper parameter tuning plays a crucial role in finding the best parameters for the model.

- In this project the Randomforest regressor performed the best among all the models.
- The predicted target values that we got with the test dataset were in logarithm form, which were then inversed by applying exponential method.

CONCLUSION

- Key Findings and Conclusions of the Study

We found the dataset is big and had a lot of features in it and also had a lot of variance in the type of data in it. There were few columns which did not have any role to play in the modelling. Also many missing values were found in certain columns of the dataset. We found that encoding the categorical data that are ordinal in nature using an encoder did not gave us the result in the right order, for which we had to manually encode them. We also found that removing the outliers in this dataset using z score technique gave us a high data loss, hence inter quartile range was used . We found many features that had no correlation with the target and thus would not effect the model in any way. During the feature selection we found that many features perform very poorly and we had to keep only those features that performed well enough. We also found that scaling the data in an important step when we are dealing with numeric data. The checking of cross validation score and performing the hyper parameter tuning helped us to select the final model for the project.

As the company wanted to know the important features which affect the price positively or negatively, we have the conclusion on it. For the continuous numeric columns, the feature ‘ground living area’ showed a very positive relation with the sale price along with

‘garage area’, ‘total basement surface area’, ‘first floor surface area’. As the value of these above features increases, the sale price also goes up. For the categorical data, the features ‘overall quality’, ‘garage cars’, ‘external quality’, ‘kitchen quality’, ‘basement quality’, ‘full bath’, ‘total rooms above ground’ all showed a very positive relation with the sales price, with the ‘overall quality’ having the highest correlation among all features. All these features refer to the quality of that particular feature, and as the quality becomes better, the sale price also goes up. The features ‘year built’ and ‘year remodelled’ had a negative relationship with the sales price, as the value in these two features increased the sale price went down, which meant that newer the house, the higher its sale price was.

All the above mentioned features are very important in predicting the price of the house. As mentioned above some effect the price positively and some negatively.

- Learning Outcomes of the Study in respect of Data Science

  From this project we learnt that EDA, feature engineering, visualization, feature selection all play a very crucial role in the outcome of the model. The EDA when done thoroughly gives us many insights into the data on the basis of which we can take the approach. After the EDA we can implement the approach in the feature engineering part and clean the data. The visualization of the dataset is very powerful in conveying the importance of different features in the dataset. Through visualization we can identify the important features in the dataset, as well as features that do not have any significant importance, which can help us in feature selection. Also we got the idea that scaling the dataset can be very useful in improving the model performance. We learnt how cross validation and hyper parameter tuning can help us find the best model for out project.

At first we got large amount of errors in MSE, RMSE, MAE in the training of our models, we handled it by log transforming the target data along with the features, which we log transformed to remove the skewness from those features.

After building the model, when we predicted the test dataset for the target using that model, we found all the values were in logarithm form, which we handled by applying exponential method on the predicted data, which inversed the logarithm in the data.

- Limitations of this work and Scope for Future Work

The solution provided can perform weak if large number of outliers and high skewness are present in the given data. Also the size of the dataset used in this project is not very big regarding the number of records in it. It has nearly 1100 records in it, which can be increased during the data collection for the model predictions to be more accurate.

The model created can be used by the company to analyse different new data in this field and to predict the price of the houses. They will also get an understanding on which features are mostly affecting the house prices. This model can be used by the company 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. With the help of it the company can decide whether to invest in a house or not and plan their investment accordingly, increasing their overall revenue, profits and improve their marketing strategies and focus on changing trends in house sales and purchases.

For further improving the results, more and more models can be used and checked which model works best. For tuning the models we can use more numbers of parameters for the score to increase.