House Price Prediction

Submitted by:

PRATIK KUMAR

HOUSING PRICE PREDICTION

Problem Statement:

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. Our problem is related to one such housing company.

A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below.

The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

- Which variables are important to predict the price of variable?
- How do these variables describe the price of the house?

Business Goal:

You 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.

Technical Requirements:

- Data contains 1460 entries each having 81 variables.
- Data contains Null values. You need to treat them using the domain knowledge and your own understanding.
- Extensive EDA has to be performed to gain relationships of important variable and price.
- Data contains numerical as well as categorical variable. You need to handle them accordingly.
- You have to build Machine Learning models, apply regularization and determine the optimal values of Hyper Parameters.
- You need to find important features which affect the price positively or negatively.

- Two datasets are being provided to you (test.csv, train.csv). You will train on train.csv dataset and predict on test.csv file.

The “Data file.csv” and “Data description.txt” are enclosed with this file.

Conceptual Background of the Domain Problem:

Linear Regression:

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable? (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable? These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

Naming the Variables. There are many names for a regression’s dependent variable. It may be called an outcome variable, criterion variable, endogenous variable, or regressand. The independent variables can be called exogenous variables, predictor variables, or regressors.

Three major uses for regression analysis are (1) determining the strength of predictors, (2) forecasting an effect, and (3) trend forecasting.

First, the regression might be used to identify the strength of the effect that the independent variable(s) have on a dependent variable. Typical questions are what is the strength of relationship between dose and effect, sales and marketing spending, or age and income.

Second, it can be used to forecast effects or impact of changes. That is, the regression analysis helps us to understand how much the dependent variable changes with a change in one or more independent variables. A typical question is, “how much additional sales income do I get for each additional $1000 spent on marketing?”

Third, regression analysis predicts trends and future values. The regression analysis can be used to get point estimates. A typical question is, “what will the price of gold be in 6 months?”.

Types of Linear Regression:

Simple Linear Regression: 1 dependent variable (interval or ratio) , 2+ independent variables (interval or ratio or dichotomous).

Logistic Regression: 1 dependent variable (dichotomous), 2+ independent variable(s) (interval or ratio or dichotomous).

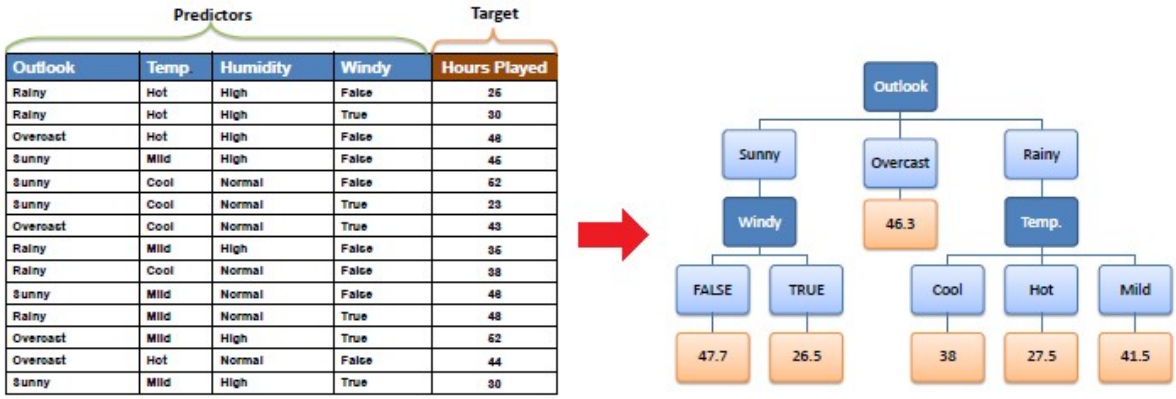
Ordinal Regression: 1 dependent variable (ordinal), 1+ independent variable(s) (nominal or dichotomous).

Multinomial Regression: 1 dependent variable (nominal), 1+ independent variable(s) (interval or ratio or dichotomous).

Discriminant Analysis: 1 dependent variable (nominal), 1+ independent variable(s) (interval or ratio.

Decision Tree Regression:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.



Predictors				Target
Outlook	Temp	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Decision Tree Algorithm:

The core algorithm for building decision trees called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with Standard Deviation Reduction.

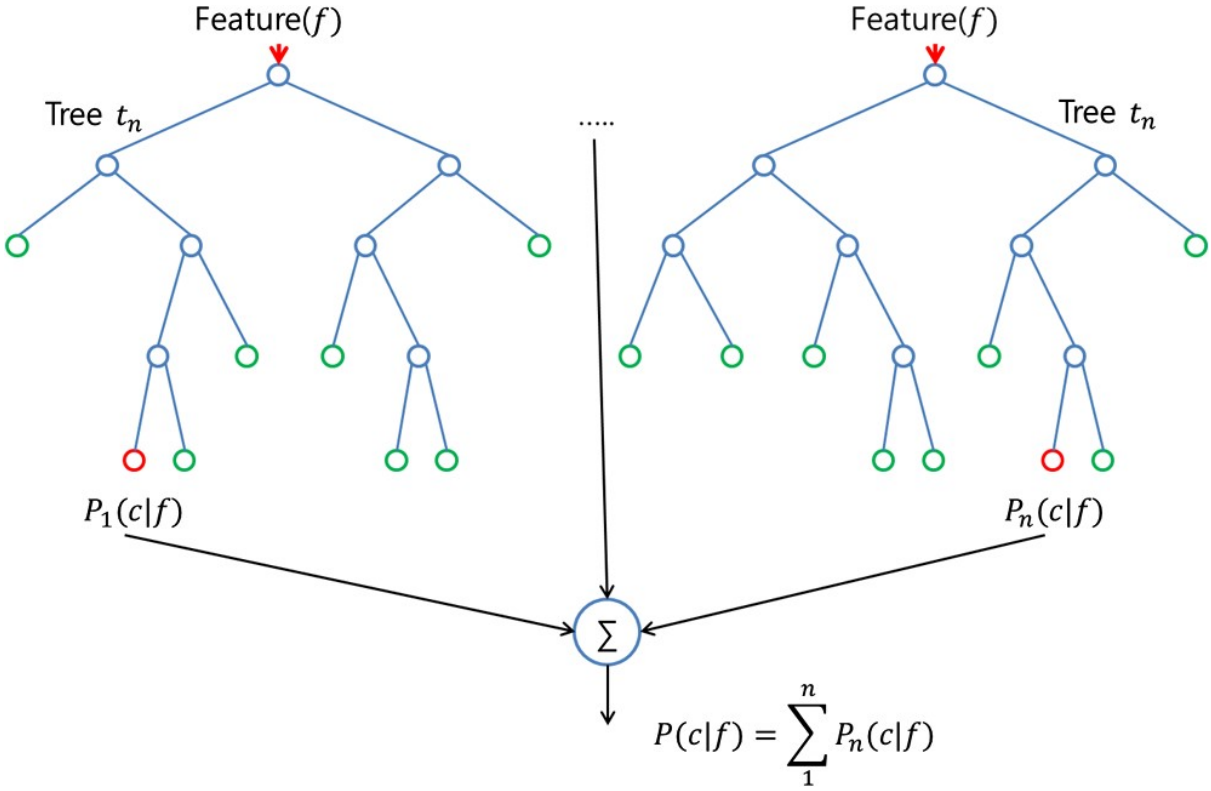
Standard Deviation: A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.

Random Forest Regression: Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees.

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression

tasks by using the algorithm's regressor. Random forest adds additional randomness to the model, while growing the trees.

Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).



Lasso And Ridge Regression:

Ridge and Lasso might appear to work towards a common goal, the inherent properties and practical use cases differ substantially. If you've heard of them before, you must know that they work by penalizing the magnitude of coefficients of features along with minimizing the error between predicted and actual observations. These are called 'regularization' techniques. The key difference is in how they assign penalty to the coefficients:

1. **Ridge Regression:**

   Performs L2 regularization, i.e. adds penalty equivalent to square of the magnitude of coefficients.

   Minimization objective = LS Obj + α * (sum of square of coefficients).

2. **Lasso Regression:**

   Performs L1 regularization, i.e. adds penalty equivalent to absolute value of the magnitude of coefficients.

   Minimization objective = LS Obj + α * (sum of absolute value of coefficients).

Gradient Boosting Regressor:

Gradient boosting is a technique attracting attention for its prediction speed and accuracy, especially with large and complex data. Don't just take my word for it, the chart below shows the rapid growth of Google searches for xgboost (the most popular gradient boosting R package). From data science competitions to machine learning solutions for business, gradient boosting has produced best-in-class results. In this blog post I describe what is gradient boosting and how to use gradient boosting.

Ensembles and boosting Machine learning models can be fitted to data individually, or combined in an ensemble. An ensemble is a combination of simple individual models that together create a more powerful new model.

Machine learning boosting is a method for creating an ensemble. It starts by fitting an initial model (e.g. a tree or linear regression) to the data. Then a second model is built that focuses on accurately predicting the cases where the first model performs poorly. The combination of these two models is expected to be better than either model alone. Then you repeat this process of boosting many times. Each successive model attempts to correct for the shortcomings of the combined boosted ensemble of all previous models.

Gradient boosting explained

Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error. How are the targets calculated? The target outcome for each case in the data depends on how much changing that case's prediction impacts the overall prediction error:

- If a small change in the prediction for a case causes a large drop in error, then next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error.

- If a small change in the prediction for a case causes no change in error, then next target outcome of the case is zero. Changing this prediction does not decrease the error.

The name gradient boosting arises because target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case.

Data Analysis: The Dataset Contains a Data of 1168 entries each having 81 variables, in which some are numerical Data and some are Categorical Data.

As the Data having two datasets:

1. Train Data

2. Test Data

Exploratory Data Analysis:

In EDA we need to Pre-process the Data and Visualization: Steps include in Pre-Processing Data are:

1.Data Cleaning: Removing Outliers, Skewness and imputing Missing Values.

2.Data Transformation: Like Normalization by applying normalization, we can improve the accuracy and efficiency of the models. And also reduce the errors.

3.Data Reduction: By Reducing the no of features by Feature Selection Process, PCA And VIF
Data Cleaning: As a Part of EDA we need to do Data cleaning so firstly we need to check any null values in our data, From the below image shows we don't have any null values, so no need to impute any data.

```
# Find columns with missing values and their percent missing
df_train.isnull().sum()
miss_val = df_train.isnull().sum().sort_values(ascending=False)
miss_val = pd.DataFrame(data=df_train.isnull().sum().sort_values(ascending=False), columns=['MissvalCount'])

# Add a new column to the dataframe and fill it with the percentage of missing values
miss_val['Percent'] = miss_val.MissvalCount.apply(lambda x : '{:.2f}'.format(float(x)/df_train.shape[0] * 100))
miss_val = miss_val[miss_val.MissvalCount > 0]
miss_val
```

	MissvalCount	Percent
PoolQC	1161	99.40
MiscFeature	1124	96.23
Alley	1091	93.41
Fence	931	79.71
FireplaceQu	551	47.17
LotFrontage	214	18.32
GarageType	64	5.48
GarageCond	64	5.48
GarageYrBlt	64	5.48
GarageFinish	64	5.48
GarageQual	64	5.48
BsmtExposure	31	2.65
BsmtFinType2	31	2.65
BsmtFinType1	30	2.57

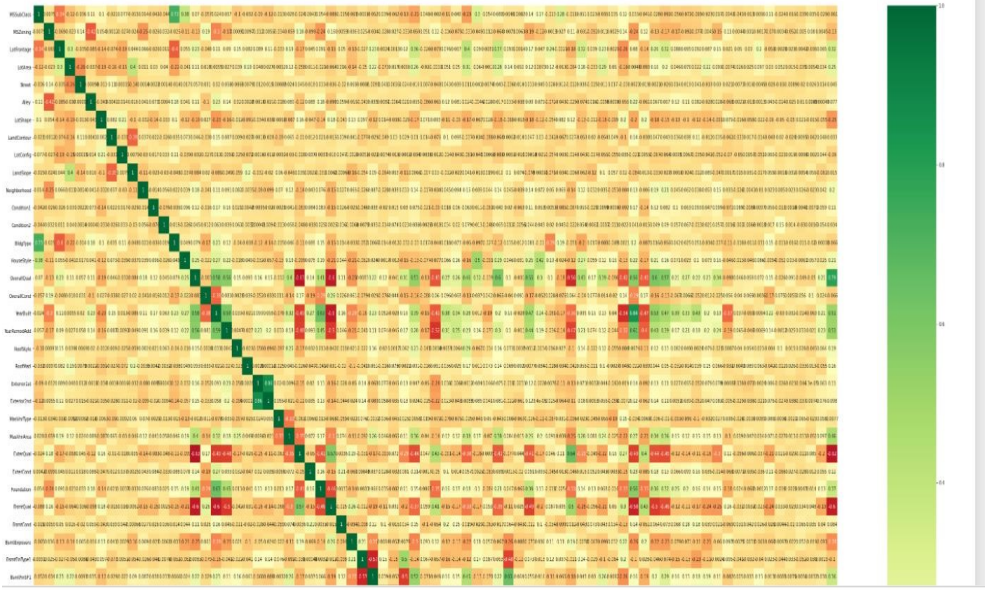
The Dataset Having the null values and I am removing the columns which is having the percentage more than 45%.

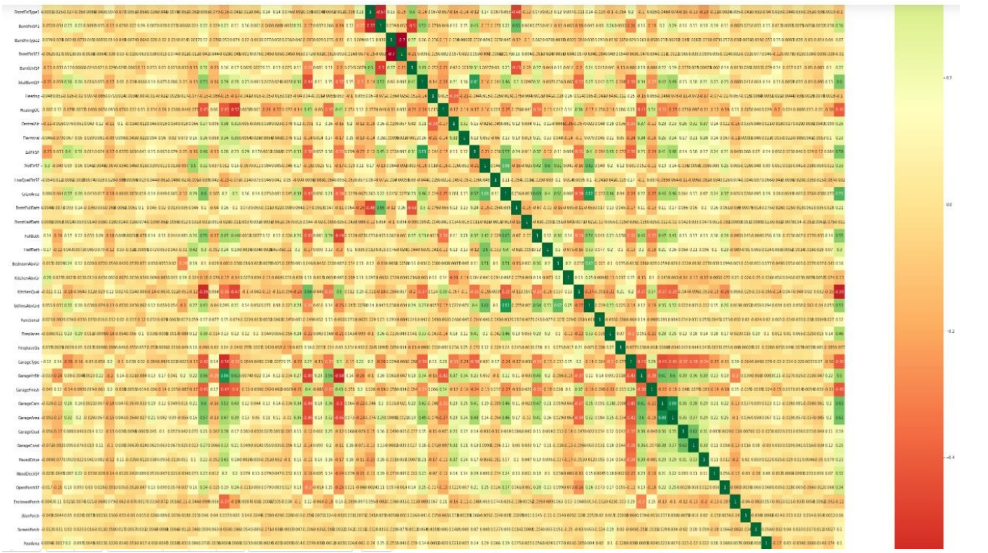
By comparing both train and test data column PoolQC, MiscFeature, Alley, Fence, FireplaceQu having more than 45% data is missing.so removing from data set and remaining all very small percent so fill the null values.

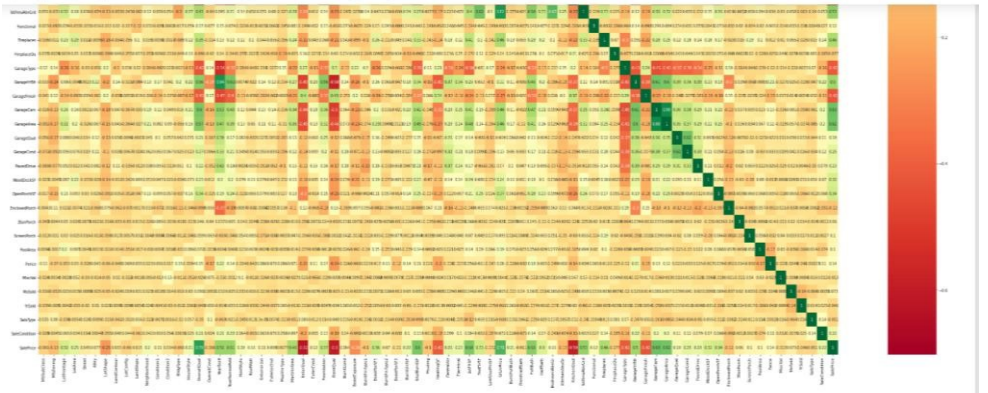
Correlation matrix and its visualization

A correlation matrix is a tabular data representing the 'correlations' between pairs of variables in a given dataset. It is also a very important pre-processing step in Machine Learning pipelines. The Correlation matrix is a data analysis representation that is used to summarize data to understand the relationship between various different variables of the given dataset.

```
plt.figure(figsize = (50,50))
sns.heatmap(df_train.corr(),cmap = 'RdYlGn',annot = True)
plt.show()
```







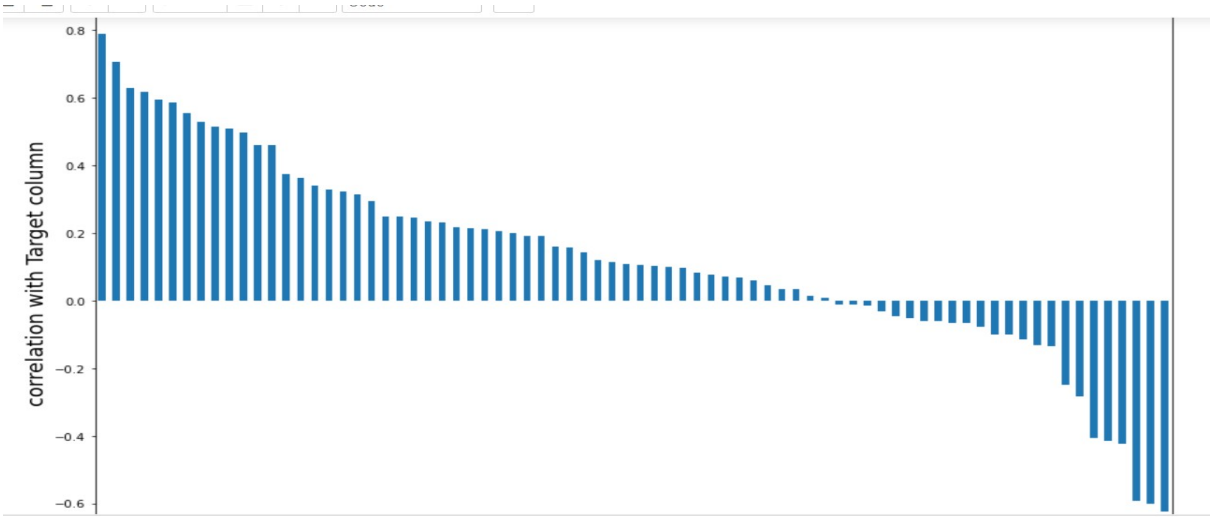
Observations:

-> SalePrice is highly positively correlated with the columns OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea.

-> SalePrice is negatively correlated with OverallCond, KitchenAbvGr, Encloseporch, YrSold.

-> We observe multicollinearity in between columns, so we will be using Principal Component Analysis (PCA).

Correlation with target variable



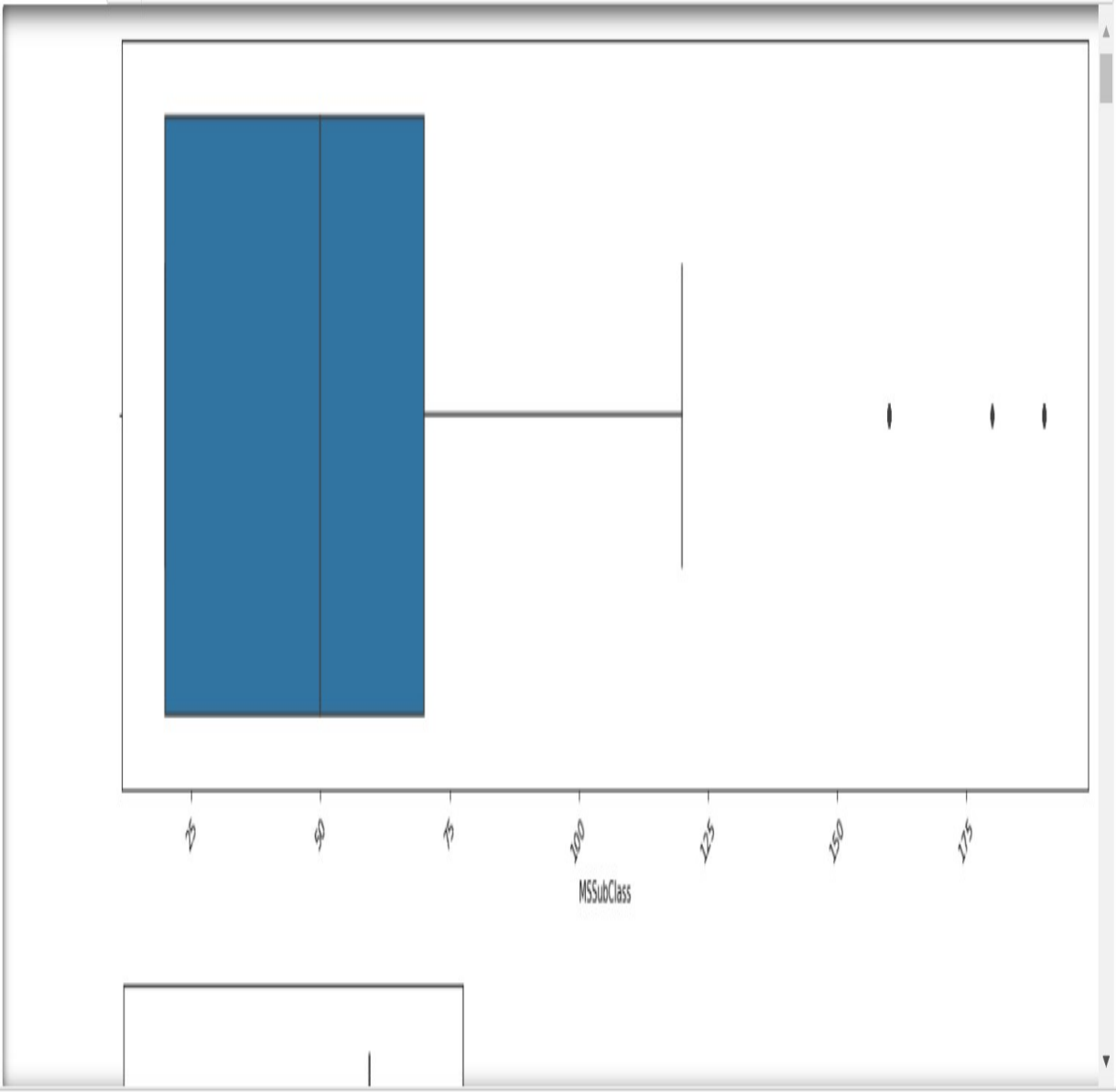
Checking outliers and plotting it

An outlier is a data point in a data set which is distant or far from all other observations available. It is a data point which lies outside the overall distribution which is available in the dataset. In statistics, an outlier is an observation point that is distant from other observations.

A box plot is a method or a process for graphically representing groups of numerical data through their quartiles. Outliers may also be plotted as an individual point. If there is an outlier it will plotted as point in box plot but other numerical data will be grouped together and displayed as boxes in the diagram. In most cases a threshold of 3 or - 3 is used i.e., if the Z-score value is higher than or less than 3 or -3 respectively, that particular data point will be identified as outlier.

Outliers

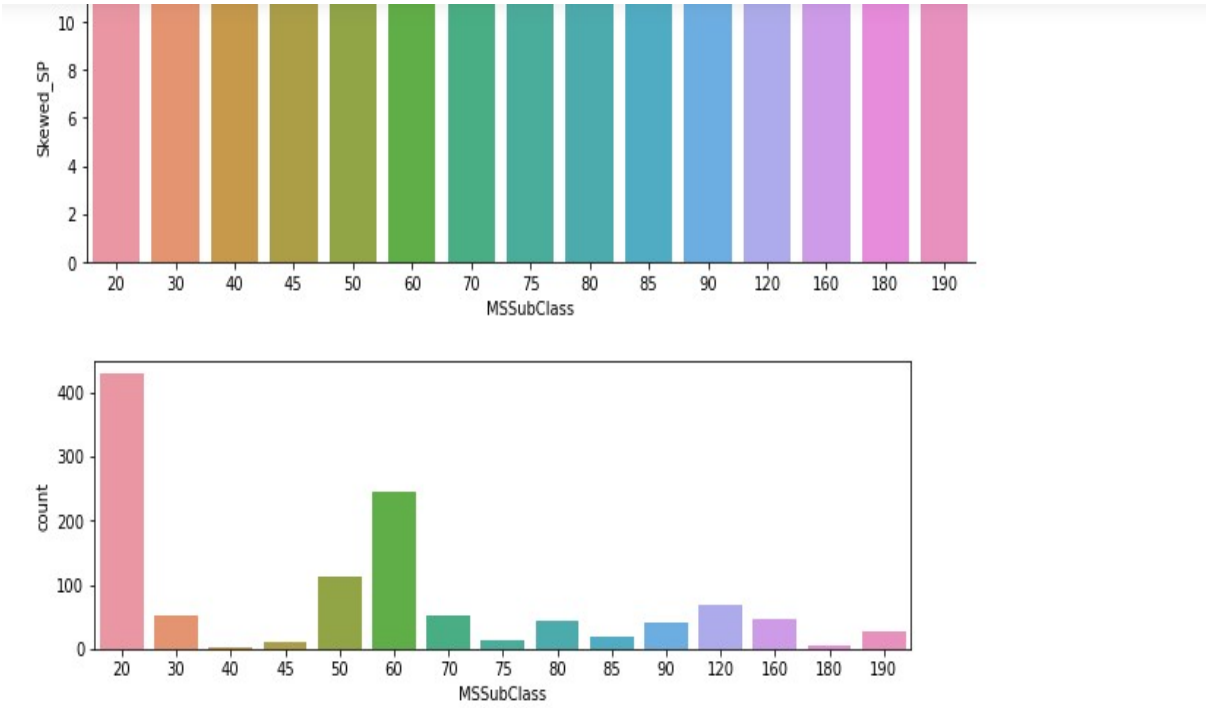
In [16]:
```
## Checking outliers using box plot:
Numerical_cols = df_train._get_numeric_data().columns
plt.figure(figsize=(20,5))
for i in Numerical_cols:
    sns.boxplot(df_train[i])
    plt.xticks(rotation=45)
    plt.show()
```



Visualization:

```
In [19]: sns.factorplot('MSSubClass', 'Skewed_SP', data=df_train,kind='bar',size=3,aspect=3)
         fig, (axis1) = plt.subplots(1,1,figsize=(10,3))
         sns.countplot('MSSubClass', data=df_train)
         df_train['MSSubClass'].value_counts()
```

```
Out[19]: 20     428
         60     244
         50     113
         120     69
         70      53
         30      52
         160     47
         80      43
         90      41
         190     26
         85      19
         75      14
         45      10
         180      6
         40       3
         Name: MSSubClass, dtype: int64
```

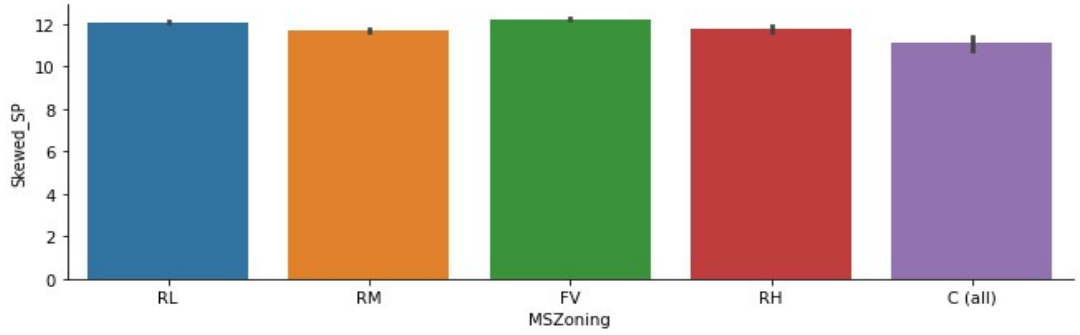


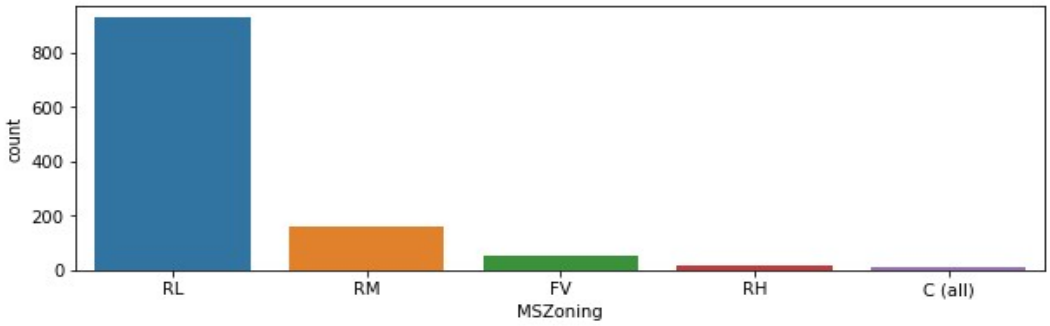
Observations:

MSSubClass = 60 has highest SalePrice, while the sales of houses with MSSubClass = 20 is the highest.

```
sns.factorplot('MSZoning', 'Skewed_SP', data=df_train,kind='bar',size=3,aspect=3)
fig, (axis1) = plt.subplots(1,1,figsize=(10,3))
sns.countplot(x='MSZoning', data=df_train, ax=axis1)
df_train['MSZoning'].value_counts()
```

```
RL         928
RM         163
FV          52
RH          16
C (all)      9
Name: MSZoning, dtype: int64
```

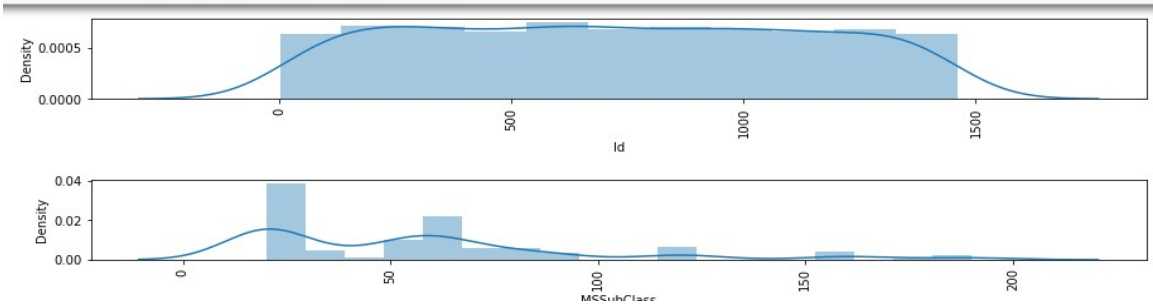


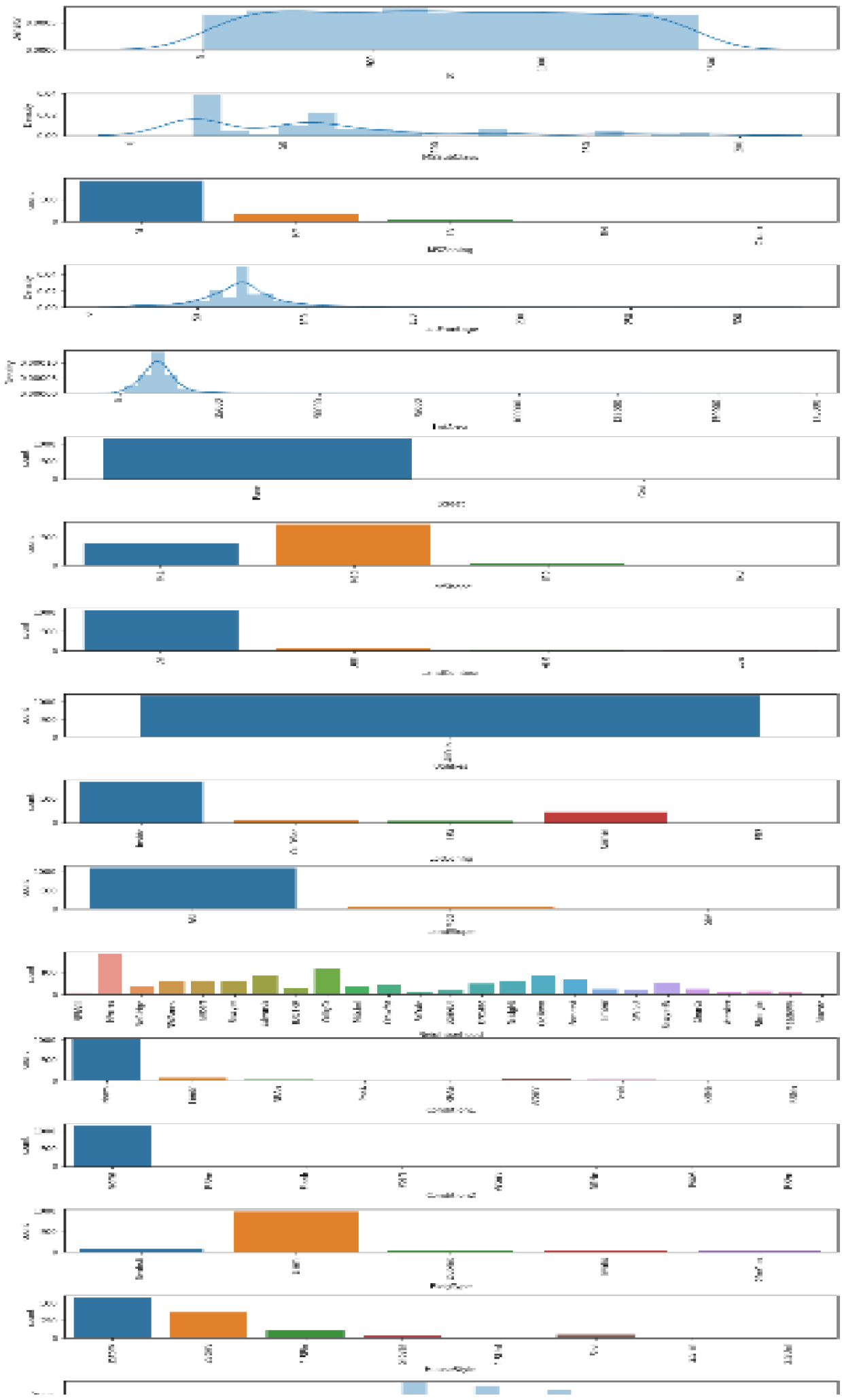


Observations:

For MsZoning RL is the Highest when comparing with Skewed Sales Price FV is Highest.

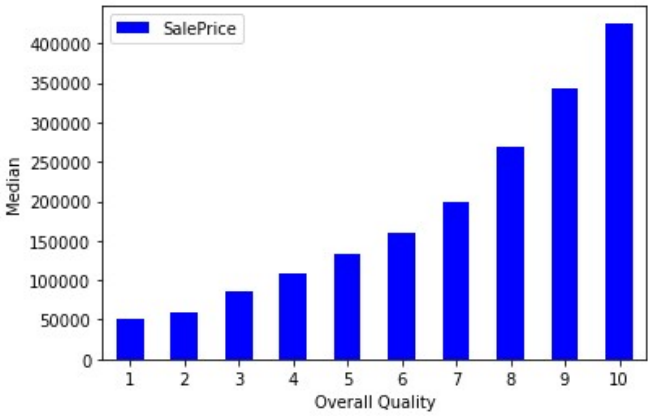
```
plt.figure(figsize=[15,200])
for i, column_data in enumerate(df_train.dtypes.items()):
    column,dtype = column_data
    plt.subplot(80,1,i+1)
    plt.subplots_adjust(hspace=1)

    if dtype == 'object':
        plt.xticks(rotation=90)
        sns.countplot(df_train[column])
    else:
        plt.xticks(rotation=90)
        sns.distplot(df_train[column],kde=True)
```





```
#Creating a pivot table
quality_pivot = df_train.pivot_table(index='OverallQual',values='SalePrice', aggfunc=np.median)
```

```
quality_pivot.plot(kind='bar',color='blue')
plt.xlabel('Overall Quality')
plt.ylabel('Median')
plt.xticks(rotation=0)
plt.show()
```



Label Encoding:

Label Encoding is necessary for the data to process to find any outliers are there as of our data consists of both numerical and categorical need to change categorical into numerical values using the encoding methods.

Checking Any Outliers in our Data and Remove it:

It is defined as the points that are far away from the same points.it can be happen because of the variability of the measurements and may be some error also. If possible, outliers should be removed from the datasets. There are servals methods to remove the outliers. 1)Z score 2) Quantile Method (Capping the data).

Z Score: It can call from the SciPy. Stats library. And for most of the case threshold values should be used 3.

Quantile Methods: Inter Quantile Range is used to detect or cap the outliers. Calculate the IQR by scipy.stats.iqr Multiply Interquartile range by 1.5 Add 1.5 x interquartile range to the third quartile. Any number greater than this is a suspected outlier. Subtract 1.5 x interquartile range from the first quartile. Any number lesser than this is a suspected outlier. Now our Data is ready for modelling as our data is clean so only thing need to do is normalizing the data before need to check our dependent variable.

Skewness of Data: As of our numeric data is skewed, we need to do normalization before go for training and testing for that need to check the skewness of data if our data is Greater than 0.5% in both positive and negative sides, then need to do power transformation and do scaling.

Scaling are of two types:

Standard Scaler: Standard scalar standardizes features of the data set by scaling to unit variance and removing the mean (optionally) using column summary statistics on the samples in the training set.

MIN-MAX Scaler: MinMaxScaler. For each value in a feature, MinMaxScaler subtracts the minimum value in the feature and then divides by the range. The range is the difference between the original maximum and original minimum. MinMaxScaler preserves the shape of the original distribution.

Splitting Data Into train_test_split: - This function is in sklearn. Model selection splitting the data array into two arrays. Train and Test with this function we don't need to splitting train and test manually.by default it make random partition and we can also set the random state.it gives four o/p like x_train, x_test, y_train, y_test.

As we do splitting the data for training and testing the data then now, we need to modelling Try Different Models.

Linear Regression: Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable? (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable? These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

Ridge Regression: Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

Lasso Regression: Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

Lasso Regression uses L1 regularization technique (will be discussed later in this article). It is used when we have more number of features because it automatically performs feature selection.

Decision Tree Regression: Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

KNN Regressor: KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighbourhood. The size of the neighbourhood needs to be

set by the analyst or can be chosen using cross-validation (we will see this later) to select the size that minimizes the mean-squared error.

SVR Regressor: Support Vector Regression (SVR) is a regression algorithm and it applies a similar technique of Support Vector Machines (SVM) for regression analysis. As we know, regression data contains continuous real numbers. To fit such type of data, the SVR model approximates the best values with a given margin called ε-tube (epsilon-tube, ε identifies a tube width) with considering the model complexity and error rate.

Random Forest Regression: Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifierclass of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Ada Boost Regression: An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

Gradient Boost Regressor: Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error.

Stochastic Gradient Regressor: Stochastic Gradient Descent (SGD) regressor basically implements a plain SGD learning routine supporting various loss functions and penalties to fit linear regression models. Scikit-learn provides SGDRegressor module to implement SGD regression.

Now Lets Discuss About each variable in output:

Mean Absolute Error: In statistics, mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon. This is known as a scale-dependent accuracy measure and therefore cannot be used to make comparisons between series using different scales.

Mean Squared Error: Mean Squared Error (MSD) of an estimator measures the average of error squares i.e. the average squared difference between the estimated values and true value. It is a risk function, corresponding to the expected value of the squared error loss. It is always non – negative and values close to zero are better. The MSE is the second moment of the error (about the origin) and thus incorporates both the variance of the estimator and its bias.

R2_score: Coefficient of determination also called as R2 score is used to evaluate the performance of a linear regression model. It is the amount of the variation in the output dependent attribute which is predictable from the input independent variable(s). It is used to check how well-observed results are reproduced by the model, depending on the ratio of total deviation of results described by the model. Cross Validation: - This technique is used to check weather out data set is over fitting or under fitting. If model score is high and cv score is less it means model perform well in train dataset but did not perform well in unseen or test dataset.

CONCLUSION Key Findings and Conclusions of the Study

-> After getting an insight of this dataset, we were able to understand that the Housing prices are done on basis of different features.

-> First, I loaded the train dataset and did the EDA process and other pre-processing techniques like skewness check and removal, handling the outliers present, filling the missing data, visualizing the distribution of data, etc.

-> Then I did the model training, building the model and finding out the best model on the basis of different metrices scores I got like Mean Absolute Error, Mean squared Error, Root Mean Squared Error, etc.

-> I got Lasso and Ridge Regressor as the best algorithm among all as it gave more r2_score and cross_val_score. Then for finding out the best parameter and improving the scores, we performed Hyperparameter Tuning.

-> As the scores were not increased, we also tried using Ensemble Techniques like RandomForestRegressor, AdaBoostRegressor and GradientBoostingRegressor algorithms for boosting up our scores. Finally, we concluded that RandomForestRegressor was the best performing algorithm, although there were more errors in it and it had less RMSE compared to other algorithms. It gave an r2_score of 89.47 and cross_val_score of 84.37 which is the highest scores among all.

-> I saved the model in a pickle with a filename in order to use whenever we require.

-> I predicted the values obtained and saved it.

-> Then we used the test dataset and performed all the pre-processing pipeline methods to it.

-> After treating skewness, I loaded the saved model that I obtained and did the predictions over the test data .

-> From this project, I learnt that how to handle train and test data separately and how to predict the values from them. This will be useful while we are working in a real-time case study as we can get any new data from the client we work on and we can proceed our analysis by loading the best model we obtained and start working on the analysis of the new data we have.

-> The final result will be the predictions we get from the new data and saving it separately.

-> Overall, we can say that this dataset is good for predicting the Housing prices using regression analysis and RandomForestRegressor is the best working algorithm model we obtained.

-> I can improve the data by adding more features that are positively correlated with the target variable, having less outliers, normally distributed values, etc.