Housing-Price-Project

Submitted by:

HARPAL SINGH

ACKNOWLEDGMENT

The project is based on Housing-Price machine building model. Various techniques in the pipeline of Machine Learning model building are used. Various sources of reference include geeksforgeeks, seaborn documentation, Pandas library, NumPy library. Scikit-learn machine learning models have been used for Regression Analysis of target feature. Jupyter notebook has been used throughout the project and its various libraries have been called for various operations.

INTRODUCTION

Business Problem Framing

In this Housing-Price-problem we have to predict the Housing-Price. Housing is a necessary need of each and every individual and therefore housing and real-estate market is one of the major contributors of world economy. DataScience comes as a very important tool to solve the problem in the domain to help companies increase their overall revenue, profits, improving their market strategies and focussing on changing trends in housing and sales and purchases. Predictive modelling, Market-mix modelling and recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies.

Conceptual Background of the Domain Problem

Predictive Modelling and Market-mix modelling is used to find out the prices of houses at a certain location which depends on various factors. And Machine Learning model is able to predict the price of house based on various factors and it is also able to tell which variable is important to predict the price of variable and the description of these variables to predict the price of the house. Various factors like type of zone, type of road access to property, general shape of the property etc decides the housing price.

Review of Literature

Various techniques for data cleaning, EDA analysis, Data Pre-processing, Feature-Engineering and Machine Learning Models selection from sklearn library of machine learning have been used. Insights of the data are found by using various data visualization techniques like countplot and distplot for univariate analysis and scatterplot for bivariate analysis and heatmap for multivariate analysis. geeksforgeeks website, seaborn, pandas and NumPy documentation and sklearn for for the technical reference have been used.

Motivation for the Problem Undertaken

Objective to build this machine learning model is to have a hands on the model building techniques along with new facing new challenges while solving various anomalies in the dataset. And solving issues encountered during the machine learning building model. Since the number of records is high so computation takes a lot of time along with careful selection of features during feature engineering also posed some challenge.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modeling of the Problem

Linear Algebra and Calculus concepts are used in the machine learning models. Since various Regressor Models have been used in the Machine Learning Algorithms, the mathematics working behind them. Calculation of evaluation metrics also involved mathematical concepts like algebraic summation. Use of Linear algebra in calculating the Euclidean distance in So, Log usage in DecisionTreeRegressor for calculating Information gain. Usage of exponential function in adaboost algorithm. In the statistical part descriptive statistics have been used to describe the data and to calculate various statistical parameters like mean, minimum value, maximum value, median value, count, standard deviation etc. And correlation coefficient has been calculated for each feature to analyse the correlation between the columns. And in the analytics modelling various techniques have been used like for analysing the data visualization distplot, countplot and scatterplots have been used.

Data Sources and their formats

Data Sources include dataset in the csv file format which contains one train data and one test file. It contains 81 columns and 1460 rows of data.

```
Housing_price=pd.read_csv("Project_housing_train.csv")
Housing_price
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

1168 rows × 81 columns

Data Pre-processing Done

In the data pre-processing and cleaning. Data have been checked for datatype matching values. After that the dataset has been checked for the empty values. Data have been checked for null values. In this dataset several null values have been found which have been imputed using Simple Imputer in python with mean, median and mode Four columns have been dropped since they are having less than 5 percent data. After that data is checked for the type of data like categorical or continuous. Some of the features having categorical and some having continuous data. After that for data visualization countplot is used for the categorical data and distplot have been used for continuous data.

Data Inputs- Logic- Output Relationships

No Inputs-Logic-Output Relationships have been found to exist between the features.

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

No assumption taken during model building.

Hardware and Software Requirements and Tools Used

In the hardware a laptop has been used along with an optical mouse.

In software excel, Python Jupyter notebook, have been used. Here is the table with list of libraries, import method and application of that method/function.

Library	**Import method/function**	**Application**
NumPy	array	Creating an array
pandas	DataFrame	importing DataFrame and other DataFrame related operations
matplolib, seaborn	countplot, distplot, scatterplots, pairplots	for data visualization
sklearn.preprocessing	OrdinalEncoder	used for encoding the categorical and ordinal string data
	power_transform	for removing skewness of both types positive and negative
	StandardScaler	for scaling of normalised data respectively
statsmodels.stats.outliers_influence	variance_inflation_factor	for calculating the variance inflation factor of every feature
sklearn.metrics	r2_score, mean_squred_error, mean_absolute_error	for calculating r2_score various regression model.
	joblib	for saving the final model
sklearn.ensemble	RandomForestRegressor,	For importing various Regressor for machine learning algorithms
	GradientBoostingRegressor	For importing the Regressor
sklearn.neighbors	KNeighborsRegressor	For importing the Regressor.
scipy.stats	zscore	used for outlier removal
sklearn.model_selection	train_test_split	for splitting dataset into training and testing data

	cross_val_score	for importing cross_val_score
	GridSearchCV	used for hyperparameter tuning
sklearn.tree	DecisionTreeRegressor	For importing DecisionTreeRegressor
sklearn.svm	SVR	For importing SVR
sklearn.linear_model	ElasticNet	For importing the elasticnet

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

In the statistical approach various statistical techniques have been used to analyse the data. For descriptive statistics describe() function in python have been used to find out the mean, median, count, and percentiles for various features. And also, corr() function have been used to find out the correlation between the features. And to visualize the correlation coefficient data heatmap have been used. And for skewness detection in the features skew() function have been used and power_transform method has been used to remove the skewness (both positive and negative) from the features. For detecting the multicollinearity between the columns VIF method have been used and features having higher VIF factor have been removed to reduce the multicollinearity.

In the Analytic approach multiple algorithms have been used to check for the r2_score for measuring the performances of the algorithms on the dataset.

Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

Algorithms used in the training and testing are:

RandomForestRegressor

SupportVectorRegressor

KNeighborsRegressor

ElasticNet

DecisionTreeRegressor

GradientBoostingRegressor

Run and Evaluate selected models

RandomForestRegressor is the first algorithm used for the Regression which is a type of ensemble technique which is generated using a random selection of attributes at each node to determine the split. Every DecisionTree has high variance but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn’t depend on one decision tree but multiple decision trees. In Regression the

final output is the mean of all outputs. This part is Aggregation. Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

Evaluation metrics and classification report for the algorithm is shown below.

```
from sklearn.ensemble import RandomForestRegressor
RFReg = RandomForestRegressor(max_depth=2, random_state=0)
RFReg.fit(x_train, y_train)
pred=(RFReg.predict(x_test))
print(RFReg.score(x_train,y_train))
print(r2_score(y_test,pred))
```

```
0.6655966846389958
0.6922817977986266
```

Second Algorithm that have been used for the calculation of r2_score is Support Vector Regressor. This algorithm is used for both categorical and continuous type of data. It constructs a hyperplane in multidimensional manner in an iterative manner, which is used to minimize the error. The core idea of the SVM is to create a hyperplane that best divides the dataset into classes. A hyperplane is a decision plane which separates between a set of objects having different class memberships. The main objective of the SVM is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. SVM searches for the maximum marginal hyperplane in following steps:
Generate hyperplanes which segregates the classes in the best way.
Select the right hyperplane with the maximum segregation from the nearest data points.
SVM is implemented in practice using a kernel. A kernel transforms an input data space into the required form. It converts non-separable problem to separable problems by adding more dimension to either nearest point is more useful in non-linear separation.
Evaluation metrics and classification report for the algorithm is shown below.

```
from sklearn.svm import SVR
SV=SVR(kernel="linear")
SV.fit(x_train,y_train)
print(SV.score(x_train,y_train))
pred=SV.predict(x_test)
print(r2_score(y_test,pred))
```

```
0.040726196433224504
0.05592512987679188
```

The third algorithm used is the KNeighborsRegression, it is very simple to understand versatile and one of the topmost machine learning algorithms. In KNeighborsRegression is the number of nearest neighbors. The number of neighbors is the core deciding factor. It uses ‘feature similarity’ to predict the values of the new data points. This means that the new point is assigned a value based on how closely it

resembles the points in the training set. K is generally an odd number if the number of classes is 2. When k=1 then algorithm is known as the nearest neighbour algorithm. Suppose P1 is the point, for which label needs to predict. First, you find the k-closest point to P1 and then classify points by majority vote of its K-neighbors. Each object votes for their class and the class with the most votes are taken as the prediction. For finding the closest similar points, we need to find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNeighborsRegressor has the following basic steps:
Calculate the distance
Find the closest neighbors
Vote for labels.
KNeighborsRegressor performs best with the lower number of features than large number of features.
Evaluation metrics and classification report for the algorithm is shown below.

```
from sklearn.neighbors import KNeighborsRegressor
neigh = KNeighborsRegressor(n_neighbors=2)
neigh.fit(x_train, y_train)
Neigh=neigh.predict(x_test)
print(neigh.score(x_train,y_train))
print(r2_score(y_test,pred))
```

```
0.8610249982302992
0.6886010635487816
```

The Fourth Algorithm is elasticnet which is the modification of Linear Regression which shares the same hypothetical function for prediction. The Linear Function suffers from overfitting and can't deal with collinear data. This makes the model more complex with inaccurate prediction on the test set. Such a model with high variance does not generalize on the new data. So, to deal with these issues, we include both L-1 and L-2 regularization to get the benefits of both Ridge and Lasso at the same time. It performs feature selection and also makes hypothesis simpler. The modified cost function for ElasticNet Regression is given below:

$$\frac{1}{m}\left[\sum_{l=1}^{m}\left(y^{(i)}-h(x^{i})\right)^{2}+\lambda_1\sum_{j=1}^{n}w_j+\lambda_2\sum_{j=1}^{n}w_j^2\right]$$

Here wj represents the weight for jth feature.
n is the number of features in the dataset.
Lambda1 is the regularization strength for L-1 norm.
Lambda2 is the regularization strength for L-2 norm.

```
from sklearn.linear_model import ElasticNet
enr=ElasticNet(alpha=0.01)
#enr=ElasticNet()
enr.fit(x_train,y_train)
Enrpred=enr.predict(x_test)
print(enr.score(x_train,y_train))
print(r2_score(y_test,pred))
enr.coef_
```

```
0.7934365134146071
0.6886010635487816

array([ 2.72386628e+02,  2.15299827e+03,  1.62964294e+03,  1.00780807e+04,
        7.57632405e+02,  9.81595088e+02,  2.49493819e+03,  0.00000000e+00,
       -5.21654025e+01,  1.67835301e+03,  4.02311125e+03, -1.60034755e+02,
       -4.09105970e+03, -3.49494236e+03, -5.01423040e+03,  2.43098829e+04,
        1.62749251e+03,  9.51069771e+03,  1.03851124e+04, -5.60016975e+02,
       -2.54053943e+03,  4.09464559e+03,  4.48916590e+03, -5.79097054e+03,
       -1.92868579e+01, -6.79879022e+02, -6.33686956e+03,  3.72696062e+02,
       -6.41781086e+03, -1.52537838e+03, -1.01715765e+03, -4.03797261e+03,
       -8.13965836e+02, -2.87168261e+03, -5.33594228e+02,  7.69376295e+03,
       -1.84004493e+02,  1.32746058e+04,  9.33060162e+03,  3.24614532e+03,
       -9.93784802e+02, -6.24016869e+03,  1.68422931e+03,  7.61560267e+03,
       -1.22130815e+03,  2.70857685e+03,  1.05699142e+03, -1.48495607e+03,
       -2.56030656e+03,  2.80315014e+03,  3.08359824e+03,  2.32124988e+03,
        5.04153428e+02, -1.00728597e+03,  1.04408069e+03,  2.13274450e+03,
        4.60890165e+03, -1.13791755e+03,  2.87402440e+02, -7.88782839e+01,
       -2.71308947e+03,  2.83405431e+03])
```

The fifth algorithm is DecisionTreeRegressor which is uses flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility. DecisionTree Algorithm falls under the category of supervised learning algorithms. Its branches/edges represent the truth/falsity of the statement and take makes a decision based on Decision Tree observes the features of an object and trains a model in the structure of tree in order to produce meaningful continuous output. Continuous output means that output/result is not discrete. Evaluation metrics and r2_score is given below:

```
from sklearn.tree import DecisionTreeRegressor
DTR=DecisionTreeRegressor()
DTR.fit(x_train,y_train)
pred=DTR.predict(x_test)
print(DTR.score(x_train,y_train))
print(r2_score(y_test,pred))
```

```
1.0
0.6886010635487816
```

The sixth algorithm that has been used is GradientBoostingRegressor. In gradient boosting, each predictor corrects its predecessor's error. In contrast to adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels. There is a technique called Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees.) The ensemble consists of N trees. Tree1 is trained using the feature matrix X and the labels y. The predictions labelled y1(hat) are used to determine the training set residual errors r1 of Tree1 as labels. The predicted results r1(hat) are then used to determine the residual r2. The process is repeated until all the N trees forming the ensemble are trained. There is an important parameter used in this technique known as

Shrinkage. **Shrinkage** refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate (eta) which ranges between 0 and 1. There is a trade-off between eta and number of estimators, decreasing learning rate needs to be compensated with increasing estimators in order to reach certain model performance. Since all trees are trained now, predictions can be made. Each tree predicts a label and final prediction is given by the formula:

$$y(pred) = y1 + (eta * r1) + (eta * r2) + \cdots \ldots \ldots . + (eta * rN)$$

The evaluation metrics and r2_score for the algorithm is given below:

```
: from sklearn.ensemble import GradientBoostingRegressor
  reg = GradientBoostingRegressor(random_state=2)
  reg.fit(x_train,y_train)
  Gbr=reg.predict(x_test)
  print(reg.score(x_test, y_test))
  print(r2_score(y_test,pred))
```

```
0.8659563089208487
0.6886010635487816
```

Key Metrics for success in solving problem under consideration

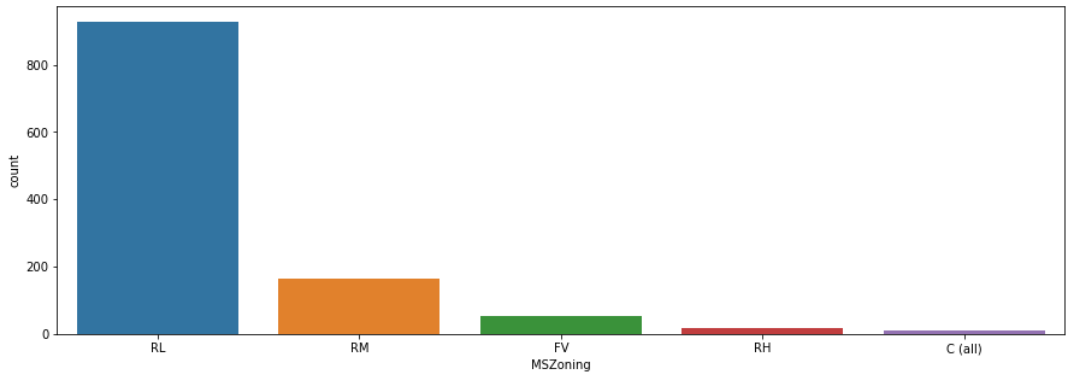
Key Metrics for success in solving problem under consideration is r2_score which is used for all regression algorithms used and are having value between 0.5 to 0.8.

Visualizations

Visualization plots that have been used in the project includes distplots, countplots, scatterplots and boxplots also. Distplots and countplots are used in Univariate analysis whereas Scatterplots are used in Bivariate Analysis. And boxplots are also used in univariate analysis for finding out the outliers. Distplots are used for continuous data whereas countplot is used for nominal data and scatterplots can be used with both continuous as well as categorical data. Below is the countplot shown that are used in the project:

```
countplt,ax=plt.subplots(figsize=(15,5))
ax=sns.countplot(x="MSZoning", data=Housing_price_nominal)
print(Housing_price_nominal["MSZoning"].value_counts())
```

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

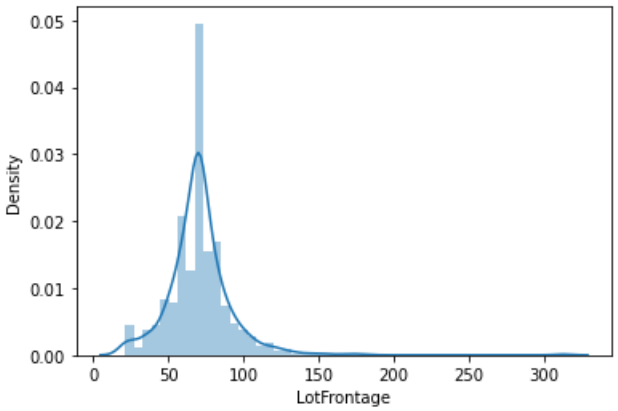


from the plot we can say that 'Residential Low Density zone' is highest followed by 'Residential Medium Density' followed by others.

This countplot shows the relationship between feature MSZoning and its count.

```
sns.distplot(Housing_price_continuous['LotFrontage'],kde=True)
```

```
<AxesSubplot:xlabel='LotFrontage', ylabel='Density'>
```

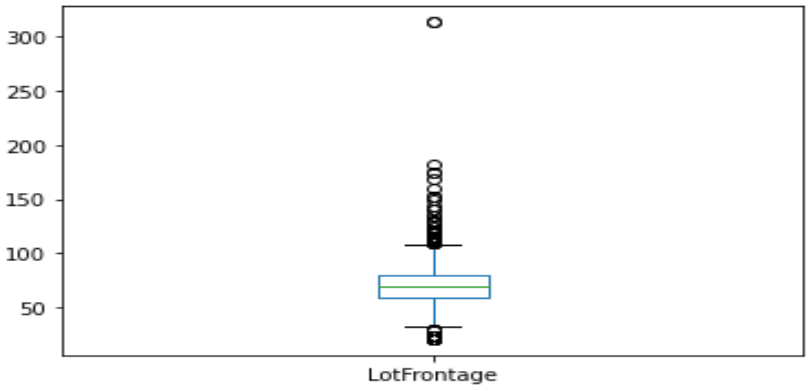


The plot is looking normally distributed with maximum and minimum value 200 and 0 respectively.

This plot shows the relationship between feature LotFrontage and its frequency.

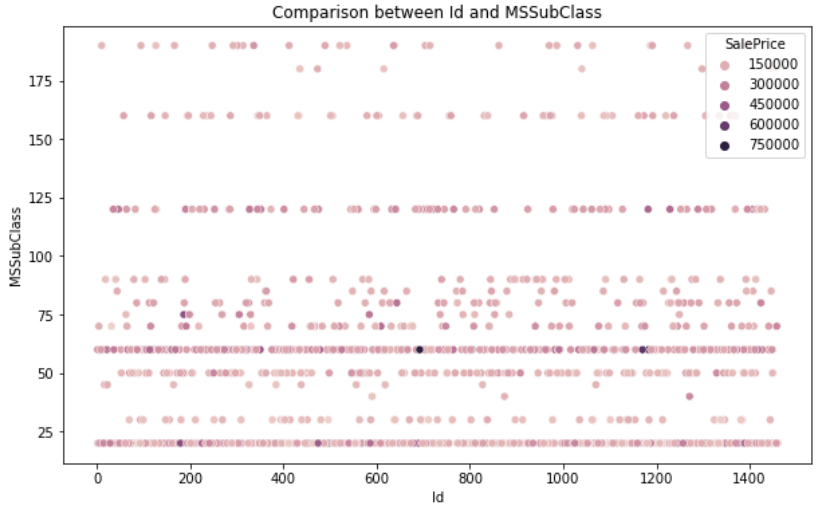
```
Housing_price['LotFrontage'].plot.box()
```

```
<AxesSubplot:>
```



This is boxplot which is showing outlier in feature ‘LotFrontage’. It has some outliers present in it.

```
plt.figure(figsize=[10,6])
plt.title('Comparison between Id and MSSubClass')
sns.scatterplot(Housing_price['Id'], Housing_price['MSSubClass'], hue=Housing_price["SalePrice"]);
```



This plot shows relationship between MSSubClass and Id and we can see that there exist is no correlation between them.

Interpretation of the Results

From the Data Visualization part, we can say that some of the features having skewed data that is negatively skewed and positively skewed. Correlation between the features is negligible some distributions are also multimodal. The outliers can be found in some of the plots like 'SalePrice','SaleCondition','SaleType' etc. From the scatterplot data we can say that the correlation can not be found among features. From the correlation heatmap we can say that feature ‘OverallQual’ is affecting ‘SalePrice’ highest since they are having correlation coefficient of 0.789185 which is highest among other feature followed by ‘GrLivArea’ followed by ‘GarageCars’ with value 0.707300 and 0.628329 and other features also exist which impact the ‘SalePrice’

positively. In Negative Correlation we have ‘BsmtQual’ with maximum negative impact on the ‘SalePrice’ with value -0.626850 followed by ‘ExterQual’ with value=-0.624820 followed by others.

From the preprocessing part we found that the dataset has some missing, null and empty values. And all the datatype are matching with the datatypes of the features in the dataset. Some datatype in the dataset were having object datatype which needs to be converted into datetime datatype, after the countplot they have been converted back to numeric for machine learning. Label Encoding was also done to features. From the feature scaling we found that some of the features 'MSSubClass','OverallCond','BsmtFinType1','BsmtFinSF1','BsmtFinType2’ that are having high VIF so they have been removed from the dataset. In the train_test_split method DecisionTreeRegressor have been used for finding out the best random state to do further r2_score checking using other algorithms. Using cross-validation score best ML model has been selected for the Hyperparameter tuning. After Hyperparameter tuning was done, best model was selected and it was called as Final Model which is further used for prediction of test data.

CONCLUSION

Key Findings and Conclusions of the Study

From the complete project we came to know that starting from importing the dataset and to the end of the model building and checking its r2_score there come a lot of challenges. Challenges include starting from data cleaning which includes checking for null and empty values and then passing appropriate values to the missing data using Imputation techniques. Then other pre-processing techniques like EDA analysis and Data Visualization. In data visualization we have to find out whether the data is normally distributed or skewed (positively or negatively). In the project the positive skewness is found in lot of features like ‘BsmtUnfSF’,’ 2ndFlrSF’, ‘OpenPorchSF’ etc. when the data is positively skewed mean is greater than median and mode. From the boxplot visualization we can find the presence of outliers in the dataset. In our dataset there are some outliers present in various features like ‘PoolArea’,’ScreenPorch’,’GarageArea’,’GarageCars’,’LotArea’ etc. Outliers presence makes the model learn dataset with decrease in accuracy and all other metrics, which will decrease the overall performance of the model. Then comes the scatterplots which shows the relationship between two feature variables whether they are positively correlated or negatively correlated. In our project the data is found to be not correlated in most of the times. Then from the descriptive statistics point of view, the various statistical finding can be analysed like mean, median, mode, count, various quartiles and maximum value for each feature. In the multivariate analysis correlation table and heatmap gives us an idea about the correlation coefficient of various features. Then the outlier removal is done using z-score to find out the dataset without outliers. After that skewness detection and skewness removal using the power_transform function from sklearn.preprocessing library was done to make the data normal and then the StandardScaler was used to make the data scaled, so it can be fed to the machine learning models for training and testing. Then VIF checking was done for checking the multicollinearity issue in the dataset. Some of the features have found to be having high VIF so they were removed from the dataset to make it

more easily used by the machine learning model. Now in the train_test_split we used DecisionTreeRegressor for checking the best random state for other algorithms to work upon. After finding the best random state various Regressor algorithms were used to find out the evaluation metrics for different algorithms. And findings in the model selection was DecisionTreeRegressor performance was best for the dataset. And then Hyperparameter tuning was used to find out the for tuning the best parameters in this algorithm and then the r2_score was found out again. And the increase in r2_score was seen in the final model.

Learning Outcomes of the Study in respect of Data Science

Learnings includes deep analysis of various data visualization techniques to get insights of the data and the variation of data in features with respect to label. In univariate analysis we have found that countplots that are used which helps in visualizing the data clearly. From the statistical point of view, various descriptive parameters like mean, median, mode, standard deviation, minimum and maximum value are observed to find out the mean, median, minimum and maximum value of the dataset which can be further used to predict the skewness and outliers in the dataset. StandardScaler technique used in the dataset for the scaling of the data which converts the data to scaled form so that it can be read effectively and easily by the machine learning model. Using various Machine Learning models for checking the r2_score of models to find the best model for Hyperparameter tuning. Hyperparameter tuning is done to choose the best parameters from the list of various parameters of the model. Challenges in this project include feature selection, Exhaustive computation for Hyperparameter tuning since number of rows are large. Selecting the best algorithm for the Final Model.

Limitations of this work and Scope for Future Work

Limitations of this work is that more robust model can be build using some more advanced techniques in machine learning model building and Exhaustive EDA can also improve the accuracy of the model. Using some more techniques for the EDA analysis can solve the problem of less r2_score. Future work can be done in the pre-processing and data visualization part for analysing data more effectively.