Project Housing Report

Submitted by:

Dattatraya Panda

ACKNOWLEDGMENT

I Would like to thanks my sme Mohd Kashif and Flip Robo Technologies

As they give the poject

I use some website to help me in this project

1) https://www.google.com/
2) https://www.youtube.com/
3) https://github.com/
4) https://scikit-learn.org/stable/

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 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. We are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high

returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

• Conceptual Background of the Domain Problem 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. 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. For this company wants to know:

1. Which variables are important to predict the price variable? 2. How do these variables describe the price of the house?

• Review of Literature Based on the sample data provided to us from our client database where we have understood that the company is looking at prospective properties to buy houses to enter the market. The data set explains it is a regression problem as we need 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. Also, we have other independent features that would help to decide which all variables are important to predict the price of the variable and how do these variables describe the price of the house. • Motivation for the Problem Undertaken Our main objective of doing this project is to build a model to predict the house prices with the help of other supporting features. We are going to predict by using Machine Learning algorithms. The sample data is provided to us from our client database. In order to improve the selection of customers, the client wants some predictions that could help them in further investment and improvement in selection of customers. House Price Index (HPI) is commonly used to estimate the changes in housing price. Since housing

price is strongly correlated to other factors such as location, area, population, it requires other information apart from HPI to predict individual housing price. There has been a considerably large number of papers adopting traditional machine learning approaches to predict housing prices accurately, but they rarely concern themselves with the performance of individual models and neglect the less popular yet complex models.

INTRODUCTION

- Business Problem Framing

Describe the business problem and how this problem can be related to the real world.

- Conceptual Background of the Domain Problem

  Describe the domain related concepts that you think will be useful for better understanding of the project.

- Review of Literature

  This is a comprehensive summary of the research done on the topic. The review should enumerate, describe, summarize, evaluate and clarify the research done.

- Motivation for the Problem Undertaken

  Describe your objective behind to make this project, this domain and what is the motivation behind.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Describe the mathematical, statistical and analytics modelling done during this project along with the proper justification.

- Data Sources and their formats

  What are the data sources, their origins, their formats and other details that you find necessary? They can be described here. Provide a proper data description. You can also add a snapshot of the data.

- Data Preprocessing Done

  What were the steps followed for the cleaning of the data? What were the assumptions done and what were the next actions steps over that?

- Data Inputs- Logic- Output Relationships

  Describe the relationship behind the data input, its format, the logic in between and the output. Describe how the input affects the output.

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

  Here, you can describe any presumptions taken by you.

- Hardware and Software Requirements and Tools Used

  Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Describe the approaches you followed, both statistical and analytical, for solving of this problem.

- Testing of Identified Approaches (Algorithms)

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

- Run and Evaluate selected models

  Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

- Key Metrics for success in solving problem under consideration

  What were the key metrics used along with justification for using it? You may also include statistical metrics used if any.

- Visualizations

  Mention all the plots made along with their pictures and what were the inferences and observations obtained from those. Describe them in detail.

  If different platforms were used, mention that as well.

- Interpretation of the Results

  Give a summary of what results were interpreted from the visualizations, preprocessing and modelling.

CONCLUSION

- Key Findings and Conclusions of the Study

  Describe the key findings, inferences, observations from the whole problem.

- Learning Outcomes of the Study in respect of Data Science

  List down your learnings obtained about the power of visualization, data cleaning and various algorithms used. You can describe which algorithm works best in which situation and what challenges you faced while working on this project and how did you overcome that.

- Limitations of this work and Scope for Future Work

What are the limitations of this solution provided, the future scope? What all steps/techniques can be followed to further extend this study and improve the results.

- Hardware and Software Requirements and Tools

Used Hardware Used:

- Processor: Core i5 -10300H
- RAM: 8 GB
- Operating System: 64-bit
- ROM/SSD: 512 TB SSD
- Graphics: NVIDIA GeForce GTX 1650 Ti

Software Used:

- Programming language: Python

- Distribution: Anaconda Navigator
- Browser based language shell: Jupyter Notebook

Libraries/Packages Used:

- Pandas
- NumPy
- Matplotlib
- Seaborn
- Scikit-learn
- Pandas_profiling Model/s Development and E

Model/s Development and Evaluation

Use langauge

```
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt


from sklearn import metrics
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression,
Ridge, Lasso, ElasticNet
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import
GradientBoostingRegressor
from sklearn.linear_model import SGDRegressor


from sklearn.metrics import r2_score
```

```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

- Run and evaluate selected models

After choosing the random state amongst 1-1000 number we got the best random state. Then I defined a function for getting the regression model trained and evaluated. The code for the models are listed below.

Random State:

```
lr=LinearRegression()
```

For Test size 0.25

```
maxAccu=0
maxRS=0

for i in range(1,1000):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25, random_state =i)
    lr.fit(x_train, y_train)
    pred = lr.predict(x_test)
    r2 = r2_score(y_test, pred)
    if r2>maxAccu:
        maxAccu=r2
        maxRS=i

print("Best accuracy is ",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is  0.8775578272556023  on Random_state  654
```

For Test size 0.2

```
maxAccu=0
maxRS=0

for i in range(1,1000):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.20, random_state =i)
    lr.fit(x_train, y_train)
    pred = lr.predict(x_test)
    r2 = r2_score(y_test, pred)
    if r2>maxAccu:
        maxAccu=r2
        maxRS=i

print("Best accuracy is ",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is  0.8845504029138069  on Random_state  942
```

We tested using 2 test sizes and we see that out of the random state value ranging from 1 to 1000 the best random state number found is 942 with test size 0.20 and we will use this in our Machine Learning models. I used a total of 12 Regression Models as follows below.

Regression Model Function:

```
# creating a function to run all the regressors

def regressor(model, x, y):
    x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.2, random_state=942)

    # Training the model
    model.fit(x_train, y_train)

    # Predicting y_test
    pred = model.predict(x_test)

    # Root Mean Square Error (RMSE)
    rmse = mean_squared_error(y_test, pred, squared=False)
    print("Root Mean Square Error is:", rmse)

    # R2 score
    r2 = r2_score(y_test, pred, multioutput='variance_weighted')*100
    print("R2 Score is:", r2)

    # Cross Validation Score
    cv_score = (cross_val_score(model, x, y, cv=5).mean())*100
    print("Cross Validation Score is:", cv_score)

    # Result of r2 score - cv score
    result = r2 - cv_score
    print("R2 Score - Cross Validation Score is", result)
```

I have built a regression function that splits the training and testing features and labels, then trains the model, predicts the label, calculates the RMSE score, generates the R2 score, calculates the

Cross Validation score and finally finds the difference between the R2 score and Cross Validation score.

Linear Regression Model

```
model=LinearRegression()
regressor(model, x, y)
```

```
Root Mean Square Error is: 24669.973270297083
R2 Score is: 88.4550402913807
Cross Validation Score is: 76.05393506616835
R2 Score - Cross Validation Score is 12.401105225212348
```

Ridge Regularization Regression Model

```
model=Ridge(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 24669.96635036763
R2 Score is: 88.45504676810401
Cross Validation Score is: 76.05397209064995
R2 Score - Cross Validation Score is 12.401074677454062
```

Lasso Regularization Regression Model

```
model=Lasso(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 24669.97122546663
R2 Score is: 88.45504220524448
Cross Validation Score is: 76.05394024646566
R2 Score - Cross Validation Score is 12.401101958778824
```

Elastic Net Regularization Regression Model

```
model=ElasticNet(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 24666.772658555114
R2 Score is: 88.4580357171353
Cross Validation Score is: 76.0711123437302
R2 Score - Cross Validation Score is 12.386923373405097
```

Support Vector Regression Models

```
model=SVR(kernel='rbf')
regressor(model, x, y)
```

```
Root Mean Square Error is: 72610.53064237421
R2 Score is: -0.01250035464317456
Cross Validation Score is: -6.181094755060825
R2 Score - Cross Validation Score is 6.168594400417651
```

```
model=SVR(kernel='poly')
regressor(model, x, y)
```

```
Root Mean Square Error is: 72624.85470209415
R2 Score is: -0.051963675457589176
Cross Validation Score is: -6.218264793631496
R2 Score - Cross Validation Score is 6.166301118173907
```

```
model=SVR(kernel='linear')
regressor(model, x, y)
```

```
Root Mean Square Error is: 66567.38617347975
R2 Score is: 15.94218971752256
Cross Validation Score is: 8.427824053143972
R2 Score - Cross Validation Score is 7.514365664378587
```

Decision Tree Regression Model

```
model=DecisionTreeRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 35075.41969094207
R2 Score is: 76.66215150359568
Cross Validation Score is: 72.37759971960335
R2 Score - Cross Validation Score is 4.2845517839923275
```

Random Forest Regression Model

```
model=RandomForestRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 22254.926292626802
R2 Score is: 90.60477069415772
Cross Validation Score is: 84.52528600007395
R2 Score - Cross Validation Score is 6.079484694083774
```

K Nearest Neighbours Regression Model

```
model=KNeighborsRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 32349.277026925753
R2 Score is: 80.14891623834271
Cross Validation Score is: 69.82284962070386
R2 Score - Cross Validation Score is 10.326066617638844
```

Stochastic Gradient Descent Regression Model

```
model=SGDRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 25019.905960627424
R2 Score is: 88.12519710241718
Cross Validation Score is: 75.54496036425553
R2 Score - Cross Validation Score is 12.580236738161645
```

Gradient Boosting Regression Model

```
model=GradientBoostingRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 21648.421257063088
R2 Score is: 91.10988202440862
Cross Validation Score is: 85.27201886965973
R2 Score - Cross Validation Score is 5.837863154748888
```

Ada Boost Regression Model

```
model=AdaBoostRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 31995.566329555568
R2 Score is: 80.58065098841882
Cross Validation Score is: 78.71938620111871
R2 Score - Cross Validation Score is 1.8612647873001151
```

Extra Trees Regression Model

```
model=ExtraTreesRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 23591.92755632515
R2 Score is: 89.44199372858674
Cross Validation Score is: 81.77265050173519
R2 Score - Cross Validation Score is 7.669343226851552
```

- Key Metrics for success in solving problem under consideration

  The key metrics used here were r2_score, cross_val_score, MAE, MSE and RMSE. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. Cross Validation:

   Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.
   In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality and their mean are taken for the final CV score.

2. R2 Score:

   It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for R2 Score is 1. The closer the value of R2 Score to 1, the better is the model fitted.

3. Mean Squared Error (MSE):

   MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is the Root Mean Squared Error.

4. Mean Absolute Error (MAE):

   MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

5. Hyperparameter Tuning:

   There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

   We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

Hyper Parameter Tuning

```
# creating parameters list to pass into GridSearchCV

parameters = {'loss' : ['squared_error', 'absolute_error', 'huber', 'quantile'],
              'criterion' : ['friedman_mse', 'squared_error', 'mse', 'mae'],
              'max_features' : ['sqrt', 'log2'],
              'learning_rate' : [0.1, 0.25, 0.5],
              'random_state' : [None, 942],
              'n_estimators': [100, 200, 300]}
```

```
GCV = GridSearchCV(GradientBoostingRegressor(), parameters, cv=5)
```

```
GCV.fit(x_train,y_train)
```

GridSearchCV

```
final_model = GradientBoostingRegressor(criterion = 'mse', learning_rate = 0.1,
```

```
final_fit = final_model.fit(x_train,y_train)   # final fit
```

```
final_pred = final_model.predict(x_test)   # predicting with best parameters
```

```
# final accuracy score
best_r2=r2_score(y_test,final_pred,multioutput='variance_weighted')*100   # chec
print("R2 score for the Best Model is:", best_r2)
```

```
R2 score for the Best Model is: 91.18707842238815
```

```
# final Cross Validation score
final_cv_score = (cross_val_score(final_model, x, y, cv=5).mean())*100
print("Cross Validation Score is:", final_cv_score)
```

```
Cross Validation Score is: 84.43091783261536
```

```
# final RMSE
final_rmse = mean_squared_error(y_test, final_pred, squared=False)
print("Root Mean Square Error is:", final_rmse)
```

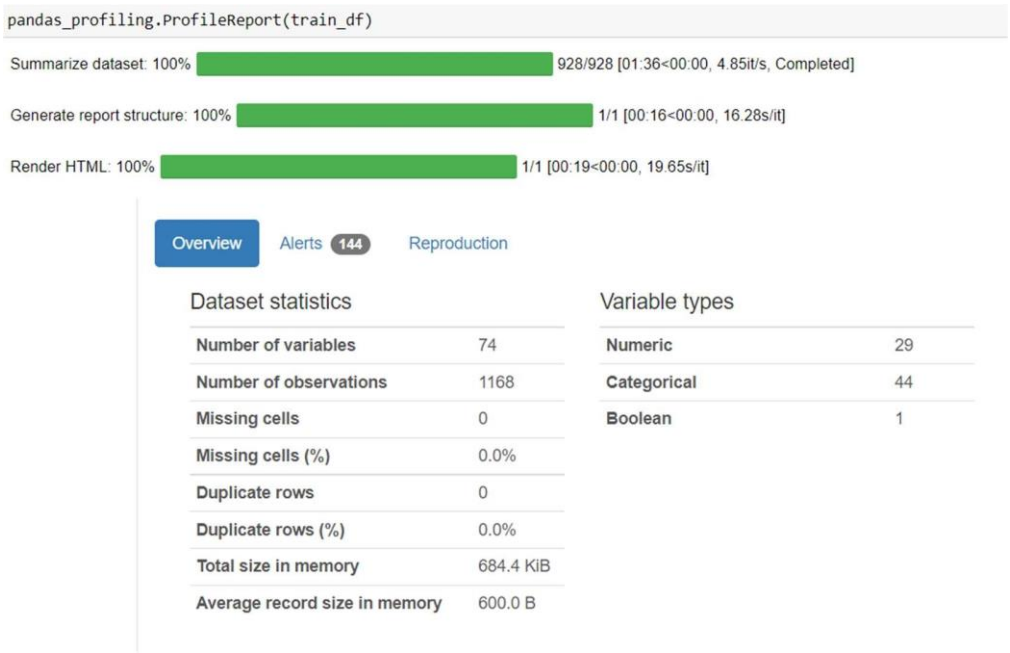
```
Root Mean Square Error is: 21554.225439656977
```

It is possible that there are times when the default parameters perform better than the parameters list obtained from the tuning

and it only indicates that there are more permutations and combinations that one needs to go through for obtaining better results.

- Visualizations

I used pandas profiling to get the over viewed visualization on the pre-processed data. pandas-profiling is an open-source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. It generates interactive reports in web format that can be presented to any person, even if they don't know programming. It also offers report generation for the dataset with lots of features and customizations for the report generated. pandas-profiling does work of visualizing and understanding the distribution of each variable. It generates a report with all the information easily available.

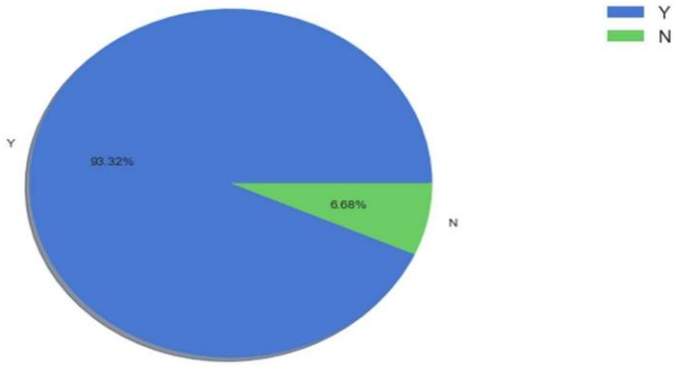


I then created pie plots, count plots, scatter plots and Bivariate Analysis and to get further visual insights on our training dataset feature values.

Pie Plots:

```
plt.style.use('seaborn-muted')
def generate_pie(x):
    plt.style.use('seaborn-white')
    plt.figure(figsize=(10,5))
    plt.pie(x.value_counts(), labels=x.value_counts().index, shadow=True, autopct='%1.2f%%')
    plt.legend(prop={'size':14})
    plt.axis('equal')
    plt.tight_layout()
    return plt.show()

for i in train_df[single]:
    print(f"Single digit category column name:", i)
    generate_pie(train_df[i])
```

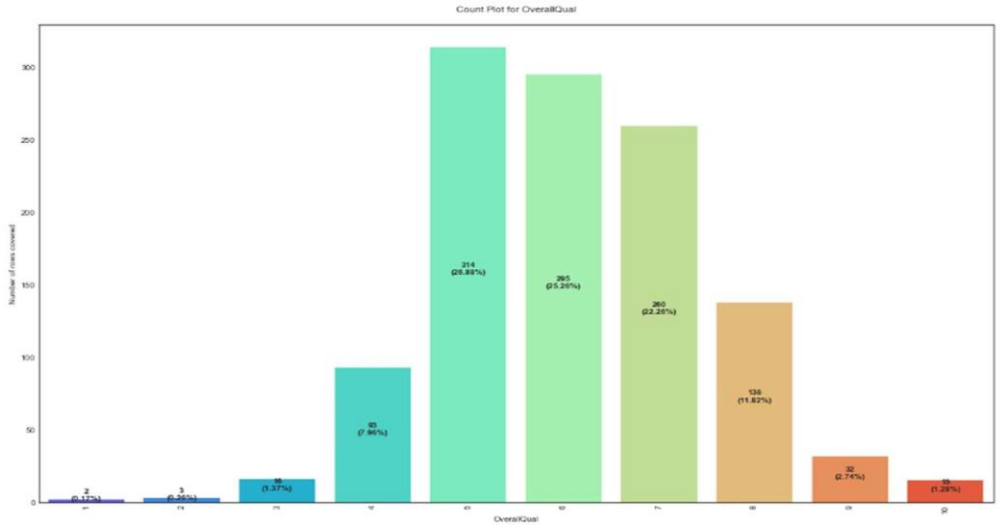
```
Single digit category column name: CentralAir
```




```
Single digit category column name: Street
```

count plots:

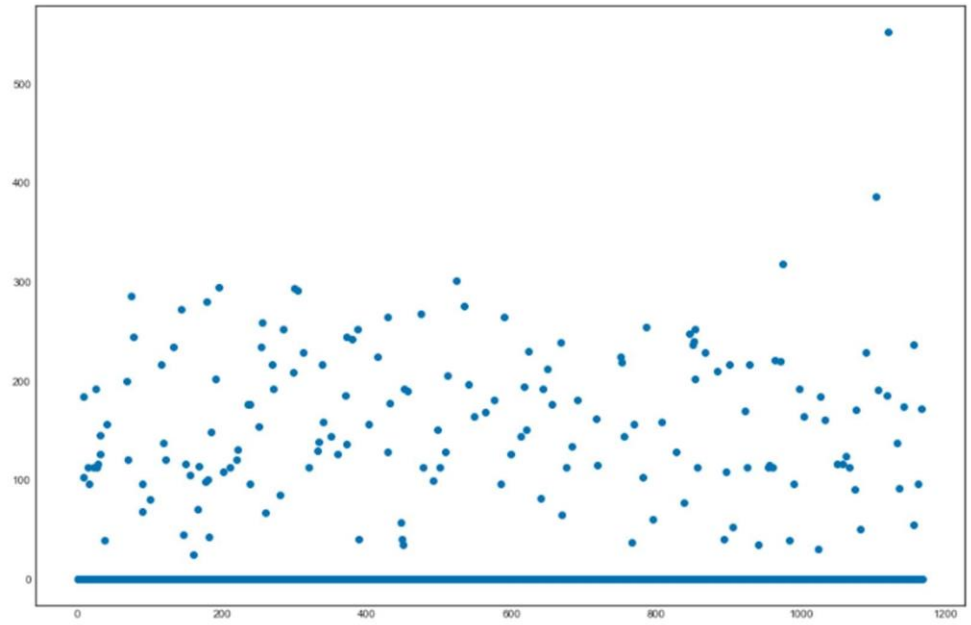
```
for col in train_df[double]:
    plt.figure(figsize=(20,12))
    col_name = col
    values = train_df[col_name].value_counts()
    index = 0
    ax = sns.countplot(train_df[col_name], palette="rainbow")

    for i in ax.patches:
        h = i.get_height() # getting the count of each value
        t = len(train_df[col_name]) # getting the total number of records using length
        s = f"{h}\n({round(h*100/t,2)}%)" # making the string for displaying in count bar
        plt.text(index, h/2, s, ha="center", fontweight="bold")
        index += 1

    plt.title(f"Count Plot for {col_name}\n")
    plt.xlabel(col_name)
    plt.ylabel(f"Number of rows covered")
    plt.xticks(rotation=90)
    plt.show()
```



scatter plots:

```
plt.style.use('seaborn-colorblind')
for j in train_df[triple]:
    plt.figure(figsize=(15,10))
    print(f"Scatter plot for {j} column with respect to the rows covered ->")
    plt.scatter(train_df.index, train_df[j])
    plt.show()
```

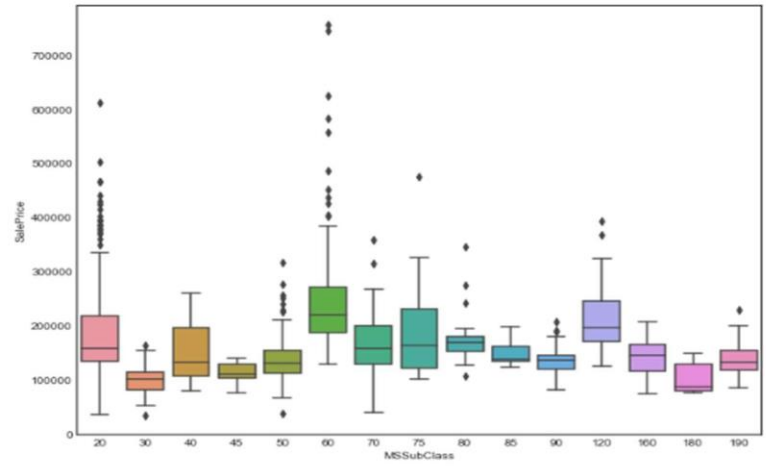
```
Scatter plot for EnclosedPorch column with respect to the rows covered ->
```




```
Scatter plot for LotFrontage column with respect to the rows covered ->
```

Bivariate Analysis (Independent variables vs Target Variable):

```
plt.figure(figsize=(10,8))
sns.boxplot(x='MSSubClass',y='SalePrice',data=train_df.sort_values('SalePrice',ascending=False))
plt.show()
```



MSSubClass vs SalePrice

```
# Checking for the sale price on the basis of road access to the property

plt.figure(figsize=[5,5])
sns.barplot(x='Street', y='SalePrice', data = train_df.sort_values('SalePrice', ascending=False))
plt.show()
```

- Interpretation of the Results

Visualizations: It helped me to understand the correlation between independent and dependent features. Also, helped me with feature importance and to check for multi collinearity issues. Detected outliers/skewness with the help of boxplot and distribution plot. I got to know the count of a particular category for each feature by using count plot and most importantly with predicted target value distribution as well as scatter plot helped me to select the best model. Also checked the feature importance with respect to the target variable.

Pre-processing: Basically, before building the model the dataset should be cleaned and scaled by performing few steps. As I mentioned above in the pre-processing steps where all the important features are present in the dataset and ready for model building.

Model Creation: Now, after performing the train test split, I have x_train, x_test, y_train & y_test, which are required to build Machine learning models. I have built multiple regression models to get the best R2 score, MSE, RMSE & MAE out of all the models.

CONCLUSION

- Key Findings and Conclusions of the Study

  I observed all the encoded dataset information by plotting various graphs and visualised further insights as shown below.

  Histogram:

```
plt.style.use('seaborn-bright')

train_df.hist(figsize=(20,30))
plt.show()
```