HOUSE PRICE PREDICTION

PRASANNA SAI KURITI

RA1911026010068

CSE J2 AIML

Steps Involved

1. Importing the required packages into our python environment
2. Importing the house price data and do some EDA on it
3. Data Visualization on the house price data
4. Feature Selection & Data Split
5. Modeling the data using the algorithms
6. Evaluating the built model using the evaluation metrics

# Importing required packages

# The primary packages for this project are going to be pandas for data processing, NumPy to work with arrays, matplotlib & seaborn for data visualizations, and finally scikit-learn for building an evaluating our ML model. We have to import all the required packages into our python environment.

# Importing Data & EDA

# We are going to work with the house price dataset that contains various features and information about the house and its sale price. Using the ‘read\_csv’ function provided by the Pandas package, we can import the data into our python environment. After importing the data, we can use the ‘head’ function to get a glimpse of our dataset.

# We begin our EDA process by removing all the null values that contain in our dataset. We can do this in python using the ‘dropna’ function.

# Then, using the ‘describe’ function we can get a statistical view of the data like mean, median, standard deviation, and so on.

# Our final step in the EDA process is to check the data types of the variables that are present in our variables. In case if there is any float or object type variable, we have to convert them into integer type. We can have a look at the data types of the variables present in our dataset using the *‘dtypes’* function in python.

# We can notice that the variable *‘MasVnrArea’* is in the form of a float data type. It is essential to change float types to integer types because linear regression is supported only on integer type variables. It can be converted using the *‘astype’* function in python.

# Data Visualization In this process, we are going to produce three different types of charts including heatmap, scatter plot, and a distribution plot.

# (i) Heatmap:

# Heatmaps are very useful to find relations between two variables in a dataset. Heatmap can be easily produced using the *‘*heatmap*’* function provided by the seaborn package in python.

# (ii) Scatter plot

# Like heatmap, a scatter plot is also used to observe linear relations between two variables in a dataset. In a scatter plot, the dependent variable is marked on the x-axis and the independent variable is marked on the y-axis. In our case, the ‘SalePrice’ attribute is the dependent variable, and every other are the independent variables. It would be difficult to produce a plot for each variable, so we can define a function that takes only the dependent variable and returns a scatter plot for every independent variable present in a dataset.

# (iii) Distribution Plot

# Distribution plots are very useful to check how well a variable is distributed in the dataset. Let’s now produce a distribution plot using the *‘*distplot*’* function to check the distribution of the *‘*SalePrice*’* variable in the dataset.

# Feature Selection & Data Split

# In this process we are going to define the ‘X’ variable (independent variable) and the ‘Y’ variable (dependent variable). After defining the variables, we will use them to split the data into a train set and test set. Splitting the data can be done using the ‘train\_test\_split’ function provided by scikit-learn in python.

# Modeling

# In this process, we are going to build and train five different types of linear regression models which are the OLS model, Ridge regression model, Lasso regression model, Bayesian regression model, Elastic Net regression model. For all the models, we are going to use the pre-built algorithms provided by the scikit-learn package in python. And the process for all the models are the same, first, we define a variable to store the model algorithm, next, we fit the train set variables into the model, and finally make some predictions in the test set.

# Now, to know which model is more appropriate for our data, we can evaluate each of the models using the evaluation metrics and come to a conclusion.

# Model Evaluation

# To evaluate our model we are going to use the *‘explained\_variance\_score’*metric and the *‘r2\_score’* metric functions which are provided by the scikit-learn package in python.

# When it comes to the *‘explained\_variance\_score’*metric, the score should not be below 0.60 or 60%. If it is the case, then our built model is not sufficient for our data to solve the given case. So, the ideal score of the *‘explained\_variance\_score’* should be between 0.60 and 1.0.

# Our next evaluation metric is the *‘r2\_score’*(R-squared) metric. What is R-squared? R-squared is a measurement of how well the dependent variable explains the variance of the independent variable. It is the most popular evaluation metric for regression models. The ideal ‘r2\_score’ of a build should be more than 0.70 (at least > 0.60).

# EXPLAINED VARIANCE SCORE

# We can see that, every model while rounding the output values will result in a score of 0.77 (77%) or 0.78 (78%) which means our model performs well on our dataset and can be used to solve real-world problems. Coming to the case of choosing the best model, the Elastic Net regression model takes the place of being more accurate while comparing the other models (on the basis of Explained Variance Score). It is followed by the Lasso regression model. The worst performer among the models is the Bayesian regression model.

# R-SQUARED

# When analyzing the report, it is noted that the R-squared of the Lasso regression model is seemed to be the highest which means, it takes the place of being the most suitable model for our dataset (on the basis of R-squared). It is followed by the Elastic Net regression model. The worst performer among the models is again the Bayesian regression model so, it is more ideal to neglect the Bayesian regression model for our dataset.

# THANK YOU!