Housing Price Prediction Case Study

Multiple Linear Regression ### Problem Statement: Consider a real estate company that has a dataset containing the prices of properties in the Delhi region. It wishes to use the data to optimise the sale prices of the properties based on important factors such as area, bedrooms, parking, etc. Essentially, the company wants — - To identify the variables affecting house prices, e.g. area, number of rooms, bathrooms, etc. - To create a linear model that quantitatively relates house prices with variables such as number of rooms, area, number of bathrooms, etc. - To know the accuracy of the model, i.e. how well these variables can predict house prices. ### Data Use housing dataset.

Reading and Understanding the Data

Data Inspection

Data Cleaning

Exploratory Data Analytics Let's now spend some time doing what is arguably the most important step - **understanding the data**. - If there is some obvious multicollinearity going on, this is the first place to catch it - Here's where you'll also identify if some predictors directly have a strong association with the outcome variable

Visualising Numeric Variables Let's make a pairplot of all the numeric variables

Visualising Categorical Variables As you might have noticed, there are a few categorical variables as well. Let's make a boxplot for some of these variables.

We can also visualise some of these categorical features parallely by using the `hue` argument. Below is the plot for `furnishingstatus` with `airconditioning` as the hue.

Data Preparation

- You can see that your dataset has many columns with values as 'Yes' or 'No'. - But in order to fit a regression line, we would need numerical values and not string. Hence, we need to convert them to 1s and 0s, where 1 is a 'Yes' and 0 is a 'No'.

Dummy Variables

The variable `furnishingstatus` has three levels. We need to convert these levels into integer as well. For this, we will use something called `dummy variables`.

Now, you don't need three columns. You can drop the `furnished` column, as the type of furnishing can be identified with just the last two columns where — - `00` will correspond to `furnished` - `01` will correspond to `unfurnished` - `10` will correspond to `semi-furnished`

Splitting the Data into Training and Testing Sets

Rescaling the Features As you saw in the demonstration for Simple Linear Regression, scaling doesn't impact your model. Here we can see that except for `area`, all the columns have small integer values. So it is extremely important to rescale the variables so that they have a comparable scale. If we don't have comparable scales, then some of the coefficients as obtained by fitting the regression model might be very large or very small as compared to the other coefficients. This might become very annoying at the time of model evaluation. So it is advised to use standardization or normalization so that the units of the coefficients obtained are all on the same scale. As you know, there are two common ways of rescaling: 1. Min-Max scaling 2. Standardisation (mean-0, sigma-1) This time, we will use MinMax scaling.

As you might have noticed, `area` seems to the correlated to `price` the most. Let's see a pairplot for `area` vs `price`.

Dividing into X and Y sets for the model building

Model Building

This time, we will be using the **LinearRegression function from SciKit Learn** for its compatibility with RFE (which is a utility from sklearn)

RFE

Recursive feature elimination

Building model using statsmodel, for the detailed statistics

Residual Analysis of the train data

So, now to check if the error terms are also normally distributed (which is infact, one of the major assumptions of linear regression), let us plot the histogram of the error terms and see what it looks like.

Model Evaluation

Applying the scaling on the test sets

Dividing into X_test and y_test

We can see that the equation of our best fitted line is: $ price = 0.35 \times area + 0.20 \times bathrooms + 0.19 \times stories+ 0.10 \times airconditioning + 0.10 \times parking + 0.11 \times prefarea $