Machine Learning 1

Linear Regression

**Assignment Submitted By:** Janarthanan Balasubramanian (jnvdasa@gmail.com)

## Questions

|  |  |
| --- | --- |
| **1.** | Explain the linear regression algorithm in detail. |
| **2.** | What are the assumptions of linear regression regarding residuals? |
| **3.** | What is the coefficient of correlation and the coefficient of determination? |
| **4.** | Explain the Anscombe’s quartet in detail. |
| **5.** | What is Pearson’s R? |
| **6.** | What is scaling? Why is scaling performed? |
|  | What is the difference between normalized scaling and standardized scaling? |
| **7.** | Sometimes the value of VIF is infinite. Why does this happen? |
| **8.** | What is the Gauss-Markov theorem? |
| **9.** | Explain the gradient descent algorithm in detail. |
| **10.** | What is a Q-Q plot? Explain the use & importance of a Q-Q plot in linear regression. |

## 1. The Linear Regression Algorithm

### Step 1: Exploratory Data Analysis to understand the data

* Prepare the data dictionary (understand the types of the variables: categorical or numeric)
* Identify data quality issues. Are there any null values or missing values? Are there any outliers? How to treat them? Do we need to standardize any of the data values?
* Visualize data to gather insights. Univariate, Segmented Univariate & Bivariate Analysis.
  + Frequency Plot: To understand the levels of categorical variables and the distribution.
  + Box Plot: To study the distribution of a continuous variable and detect outliers.
  + Scatter Plot or Heat Map: To visualize the correlation between two numeric variables.
* Are there any independent variables that have strong correlation with the target variable?
* Check for Multicollinearity: Are there any correlation between the predictor variables? Is there a linear relationship between any of the predictor variables?

### Step 2: Data Preparation

We cannot use categorical variables in model building using linear regression. So we need to treat these categorical variable appropriately to convert them into numeric. If there are binary variables (categorical variables with only 2 levels), map them to 0 and 1. In Python, to map the categorical variables with 2 levels (for instance, yes and no),

df[column] = df[column].map({'yes': 1.0, 'no': 0.0})

If there are more than 2 levels in the categorical variable, we have two options: (a) perform label encoding: convert the labels into machine readable numeric form. (b) create dummies: for a categorical variable with n levels, we need to create n-1 dummy variables. The nth variable is redundant and may create multicollinearity issues. In Python, we can use the pd.get\_dummies() function to create dummy variables from the data frame.

### Step 3: Splitting the data into Test and Train sets

Once we have understood the data, addressed the data quality issues and had sufficiently dealt with the categorical variables, we are ready to perform the linear regression. The first step of performing regression is to split the data set into train data and test data. The algorithm learns from the train data to get deeper understanding on a target variable (the variable to be predicted) and uncovers patterns and relationships between other features in the dataset and the target.

In Python, we use the sci-kit learn library to perform the test-train split of the data set:

from sklearn.model\_selection import train\_test\_split

np.random.seed(0)

df\_train, df\_test = train\_test\_split(dataset, train\_size = 0.7, test\_size = 0.3,

random\_state = 100)

### Step 4: Scaling of the Numeric Variables

Different variables in our data set may be in different scales and this will make it difficult for us to interpret the model because the coefficients in the linear model will all be in different scales. There are two methods for rescaling and fit the data: normalization (min-max scaling) or standardization (mean = 0, sigma = 1). In Python, using the sci-kit learn library, we can perform the scaling as follows:

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df\_train[columns] = scaler.fit\_transform(df\_train[columns])

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df\_train[num\_vars] = scaler.fit\_transform(df\_train[num\_vars])

**Note:** Scaling of the numeric variables should be performed after the test-train split. Scaling does not affect the model. We can plot a heat map to see the correlation between the variables.

### Step 5: Splitting the Train Data set

Split the training data set into X and y sets for model building. y\_train will have the target variable and X\_train will have all the independent variables.

**A note on Model Building and Feature Selection**

Not all the features in our dataset contribute to the model. So, we have to remove the features of low significance to improve the model accuracy, reduce the model complexity and to avoid overfitting the model (the model memorizes the data so well that when presented with fresh data set, it cannot predict the target variable). It is time consuming to try all possible combinations and see which model fits the best. There are two alternative ways to do the feature selection:

**Method 1:** Start with one feature which is highly correlated with the target. Keep adding more features, one at a step, and evaluate the resulting model based on the value of R2. Repeat the steps until you get an R2 value that can explain more than 80-85% of differences in target variable can be explained using the features selected.

**Method 2:** Start with all the features and keep eliminating the insignificant features, one at a step.

### Step 6: Recursive Feature Elimination

If there are 10-15 features, we can manually do the feature elimination. But it becomes impractical to do manual elimination when there are a large number of features. RFE is a utility available in scikit-learn library, which can be used to fit the model and remove the weakest features until a specified number of features is reached. Using this utility select 10-15 features that can further be evaluated manually to build the model step-by-step.

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LinearRegression

lm = LinearRegression() # Create a model

lm.fit(X\_train, y\_train) # Fit the train data

rfe = RFE(lm, 10) # run RFE specifying the number of features

rfe = rfe.fit(X\_train, y\_train)

col = X\_train.columns[rfe.support\_] # gives the list of columns to be used

### Step 7: Build the Model using Ordinary Least Squares Method

The ordinary least squares method fits the model by minimizing the sum of square differences between the observed and predicted values. and determines the β values.

import statsmodels.api as sm

X\_train\_lm = sm.add\_constant(X\_train\_lm) # we have to add a constant beta-0

lr = sm.OLS(y\_train, X\_train\_lm).fit() # using ordinary least squares method

print(lr.summary()) # prints the OLS regression results

OLS Regression Results

==============================================================================

Dep. Variable: price R-squared: 0.505

Model: OLS Adj. R-squared: 0.501

Method: Least Squares F-statistic: 128.2

Date: Tue, 09 Oct 2018 Prob (F-statistic): 3.12e-57

Time: 13:02:47 Log-Likelihood: 297.76

No. Observations: 381 AIC: -587.5

Df Residuals: 377 BIC: -571.7

Df Model: 3

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------

const 0.0414 0.018 2.292 0.022 0.006 0.077

area 0.3922 0.032 12.279 0.000 0.329 0.455

bathrooms 0.2600 0.026 10.033 0.000 0.209 0.311

bedrooms 0.1819 0.041 4.396 0.000 0.101 0.263

==============================================================================

Omnibus: 50.037 Durbin-Watson: 2.136

Prob(Omnibus): 0.000 Jarque-Bera (JB): 124.806

Skew: 0.648 Prob(JB): 7.92e-28

Kurtosis: 5.487 Cond. No. 8.87

==============================================================================

If the **p-value** of the coefficient of the variable (βi) < 0.05 we reject the null hypothesis that the corresponding βi = 0 and hence the variable is significant. If the p-value > 0.05, the feature is insignificant and can be eliminated.

What if the p-value of all the variables are less than 0.05? Does that mean the variables are all signification? No. The variables may be correlated to each other (multicollinearity) which will make the coefficients swing wildly and making the p-value unreliable. So we use a complementing measure: **VIF (Variance Inflation Factor)**, which calculates how well a specific feature is explained by all the other features combined.

In Python, we can compute VIF using the in-build methods.

import statsmodels.api as sm

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

vif = pd.DataFrame() # create a data frame to store VIF values

# the vif data frame should have two columns: Features, VIF

vif['Features'] = X\_train.columns

vif['VIF'] = [variance\_inflation\_factor(X\_train.values, i)

for i in range(X\_train.shape[1])]

# round the VIF value to 2 decimal places, sort them and display

vif['VIF'] = round(vif['VIF'], 2)

vif = vif.sort\_values(by = "VIF", ascending = False)

print(vif)

If the VIF value > 10, we eliminate the feature and build the model again. If VIF > 5, it can be okay, but it is worth inspecting. A value less than 5 indicates that the feature need not be eliminated.

Eliminate one variable at a time (the one with the highest p-value or if the p-values are all less than 0.05, the one with the highest VIF) and then build the model again. The elimination of a variable may significantly impact the p-value and VIF in the next iteration.

**Repeat this step again and again**, by eliminating the features one-by-one, till we have p-value < 0.05 for all the features in the model and their VIF < 5. Check the value of the R2 and Adjusted R2, which are the measures of the percentage of variance that the model can explain.

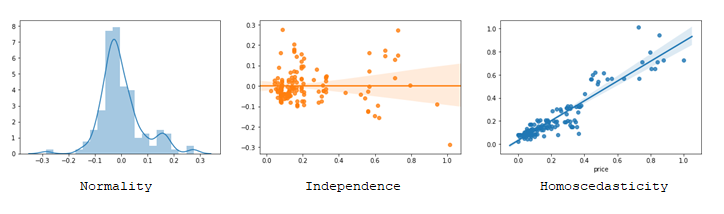
### Step 7: Residual Analysis to Validate the Model

We have to make sure that the model is reliable and can be used for predictions. To do this we perform residual analysis. Residual is the difference between the actual value of y and the predicted value of y. In linear regression we make certain assumptions on these error terms.

**Normality:** Error terms are normally distributed with mean = 0.  
Validation: Create a distribution plot on the error terms. We can also use a Q-Q plot.

**Independence:** Error terms are independent of each other.  
Validation: Create a scatter plot of the error terms and the predicted y values and fitting a regression line. If the error terms do not follow a pattern and are independent of each other the best fit line will be a horizontal line with slope = 0.

**Homoscedasticity:** Error terms have a constant variance.  
Validation: Create a scatter plot of the actual y values against the difference between the actual value and the predicted values.



### Step 8: Using the Model to Predict

Now that we have fitted the model and validated our assumptions on the error terms, we can make predictions using our final model.

**Apply scaling on the test data.** Use the same scaler that was used to fit and transform the train data. When applying the scaling on test data, there is no need to fit, but just transform.

df\_test[columns] = scaler.transform(df\_test[columns])

**Split the test data** into X and y data sets (X\_test with all the features included in the model and y\_test with the target variable). Note that we need to drop all the other features that are not used in the model from X\_test and keep only the features used in building the model.

X\_test = sm.add\_constant(X\_test) # add a constant for beta-0

y\_pred = lm.predict(X\_test) # use the final model to predict y

**Validate the results** by plotting the predicted values against the actuals; and also check whether the assumptions on the error terms still holds good.

The fitness of the model can be evaluated by studying the values of R2 and Adjusted R2.

from sklearn.metrics import r2\_score

r2 = r2\_score(y\_test, y\_pred)

If the value of R2 on test set is close to the value of R2 in the train set, then it means the model is able to generalize well and is not a over fit.

The Adjusted R2 can be computed as follows:

N is the number of samples in the data set and p is the number of predictor variables. If value of Adjusted R2 is close to the R2, it means that there are no redundant features in the model.

## 2. Assumptions in Linear Regression Algorithm

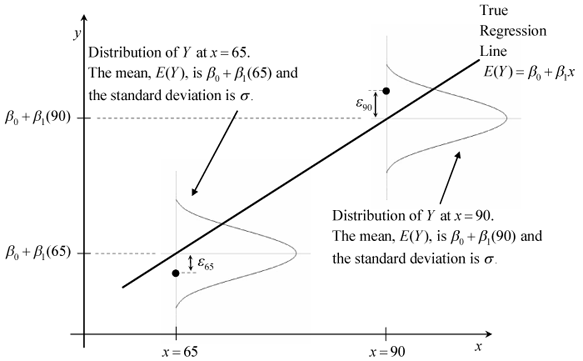
While building the linear model, we assume that there is a linear relationship between the independent variable (output or target variable) and the dependent variables (input variables). But this is not enough to generalize the results we obtain on the sample to the population.

We predict the value of the target variable using the linear equation: where β0 is the y-intercept (the point where the line touches the y-axis) and β1 is the coefficient for the independent variable x and represents the slope of the line. However, the predicted value (ŷ) may be different from the actual value (y). The difference is called an error (ε) or residual.

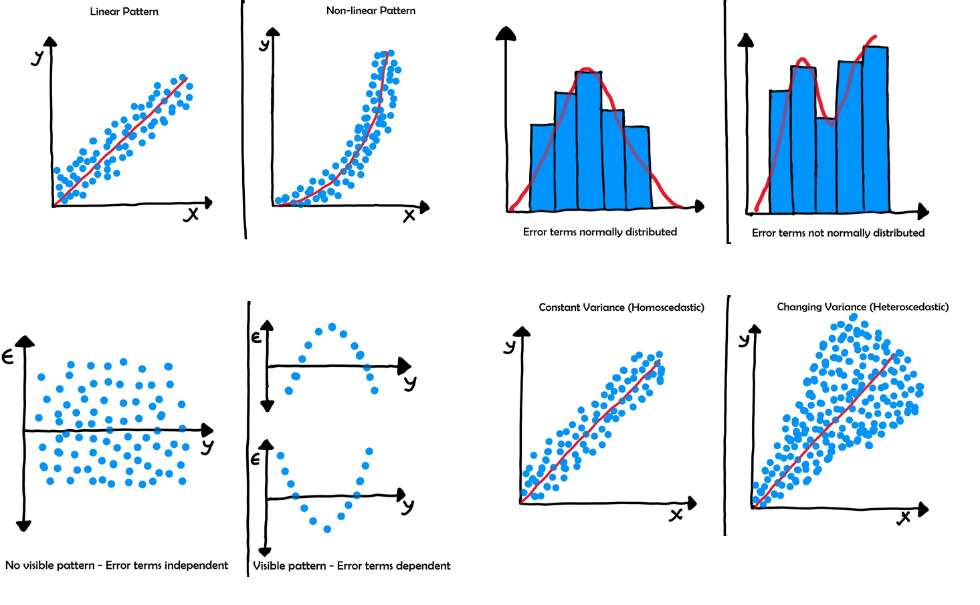
In linear regression, we make certain assumptions w.r.t these error terms (a.k.a. residuals).

1. The error terms are normally distributed with mean = 0 (normality).
2. The error terms are independent of each other (independence).
3. The error terms have a constant variance (homoscedasticity).

These assumptions allow us to make inferences based on the regression. If any of these assumptions are violated, then the forecasts, confidence intervals, and scientific insights yielded by a regression model may be inefficient or seriously biased or misleading.

Image source: <http://reliawiki.org/index.php/Simple_Linear_Regression_Analysis>

**Normality:** At each x, we try to find the best fit value for y. But there are many other possible values for y. The first assumption is that the corresponding error terms (ε) at each x computed as the difference between the actual value and the expected or predicted value ε = y - ŷ are normally distributed with mean = 0 (corresponding to the accurate prediction of y i.e. y = ŷ and ε = 0) and a standard deviation σ. Here we are not talking about how x or y is distributed. It is just the distribution of error terms.



**Independence:** The error terms are independent of each other. If you plot error terms (or residuals) against the predicted values of y, the scatter plot will not show any pattern. The best fit line for this scatter plot will be a horizontal line that indicates no correlation.

**Homoscedasticity:** The error terms have a constant variance (σ2).

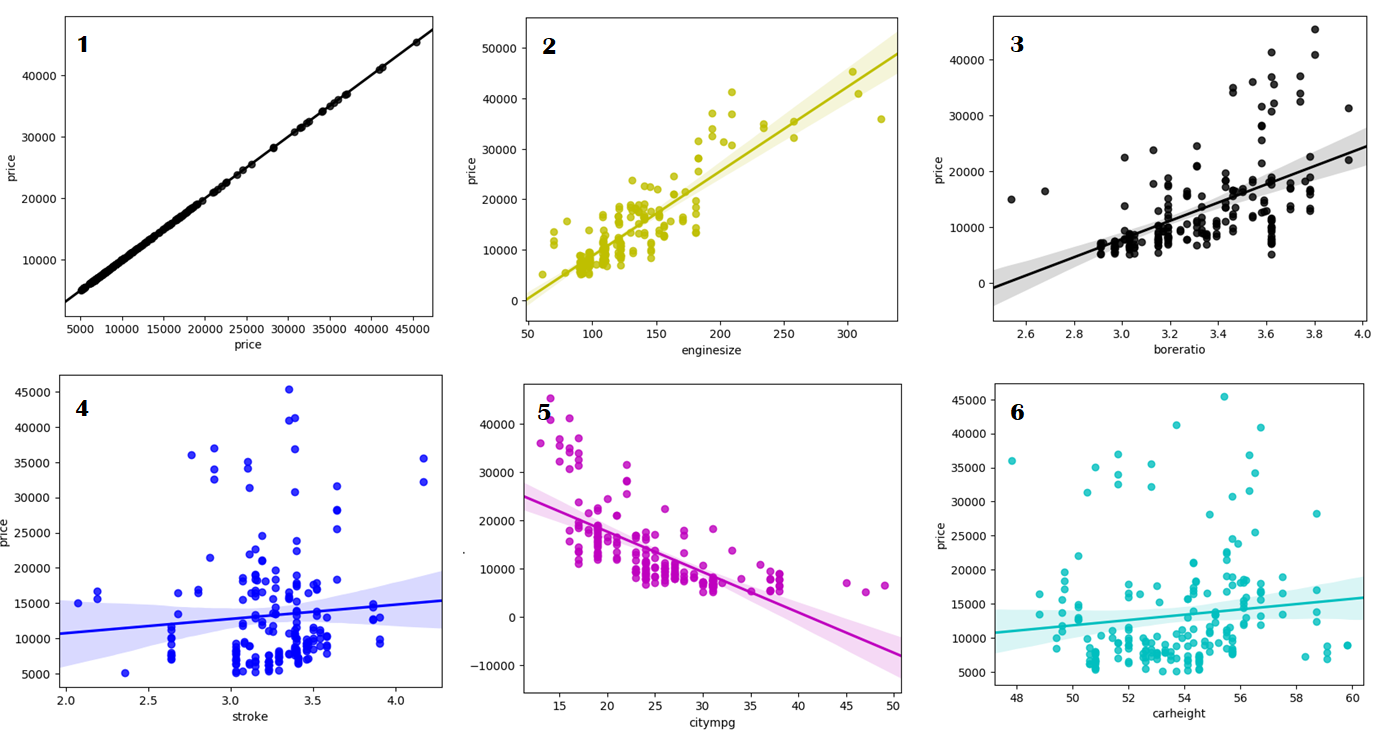
1. The variance should not increase or decrease with changes in error values.
2. The variance should not follow any patterns with changes in error terms.

## 3a. Coefficient of Correlation

Coefficient of Correlation is a measure of the strength and direction of the relationship between two random variables. The value of r is such that **-1 ≤ r ≤ +1**.

* If r = 0, there is no correlation between the two variables.
* If r > 0, there is a positive correlation, which means with every increase in one variable, there is a proportionate increase in the other variable.
* If r < 0, there is a negative correlation, which means with every increase in one variable, there is a proportionate decrease in the other variable.

If the absolute value of the correlation is greater than 0.8 we describe it as a strong relationship. If the absolute value of the correlation is less than 0.5 we call it a weak relationship.



We use scatter plots or heat maps to visualize the correlation between the variables. Plot 1 shows a perfect correlation (the data points lie exactly on a straight line). Plot 2 and 3 show positive correlation; the relationship between the variables is strong in Plot 2 than in Plot 3. Plot 5 shows a negative correlation. Plot 4 and 6 show very weak correlation (almost 0).

The formula for computing coefficient of correlation using Pearson’s Method is given below:

In Python, we use the method pd.DataFrame.corr() to compute the pair-wise correlation of the numeric columns in the data frame. It returns a correlation matrix. By default, the function uses the Pearson’s method for computing the correlation. There are other methods of computing correlation like Kendall rank correlation, Spearman correlation, and the Point-Biserial correlation.

## 3b. Coefficient of Determination

The coefficient of determination, R2, is used to analyze how differences in one variable can be explained by a difference in a second variable. The value of R2 is such that **0 ≤ R2 ≤ 1**.

To find R2, we first compute r, square the resultant value and convert it into a percentage.

Coefficient of Determination = r2 (square of the coefficient of correlation)

For example, if r = 0.9 then the R2 = 0.81 which means that 81% of the variation in the dependent variable can be explained by the independent variable i.e. 81% of the data points are close to the best fit regression line. R2 is also a measure of how well the regression line represents the data and can be used to evaluate the model we build using linear regression.

The value of R2 increases as we add more and more predictor variable to the model. A high R2 value indicates that the model is a good fit for the data. However, we should be careful not to over fit. A model is said to over fit when the training accuracy is high while the test accuracy is very low, i.e. instead of finding the relationship between the variables, the model memorizes the training data so well that when presented with a fresh set of data it cannot predict the target variable.

Adjusted R2 penalizes the model for the number of predictor variables included.

N is the sample size (or the number of observations) and p is the number of predictor variables.

**Finding R2 in Simple Linear Regression**

In simple linear regression, when only one predictor is included in the model, we can compute the coefficient of determination R2 using the following formula.

RSS is the Residual Sum of Squares computed with OLS algorithm and is the cost function that need to be minimized. Error ε = y - ŷ where the predicted value (the linear equation for the best fit regression line to predict the target variable).

TSS is the total sum of squares which is computed as the square of the differences between the actual values and the average of the actual values.

Explained Sum of Squares (ESS), also called the Model Sum of Squares (MSS) and is the sum of the squares of the predicted values minus the average of the actual values. ESS = TSS - RSS.

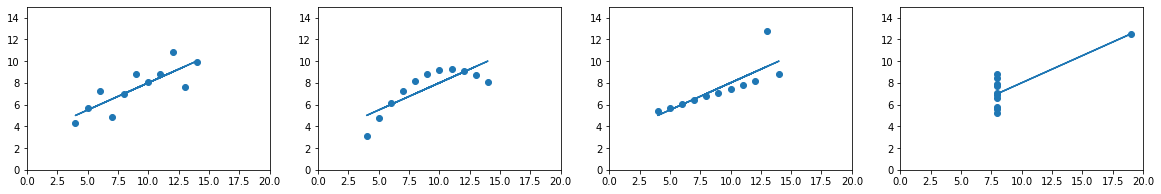
## 4. Anscombe’s Quartet

In 1973, statistician Francis Anscombe published a paper titled *Graphs in Statistical Analysis* in which he demonstrated how the summary statistics may be misleading. He presented four distinct datasets with nearly identical statistical properties, but appear very different when plotted.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | y | x | y | x | y | x | y |
| 10.0 | 8.04 | 10.0 | 9.14 | 10.0 | 7.46 | 8.0 | 6.58 |
| 8.0 | 6.95 | 8.0 | 8.14 | 8.0 | 6.77 | 8.0 | 5.76 |
| 13.0 | 7.58 | 13.0 | 8.74 | 13.0 | 12.74 | 8.0 | 7.71 |
| 9.0 | 8.81 | 9.0 | 8.77 | 9.0 | 7.11 | 8.0 | 8.84 |
| 11.0 | 8.83 | 11.0 | 9.26 | 11.0 | 7.81 | 8.0 | 8.47 |
| 14.0 | 9.96 | 14.0 | 8.10 | 14.0 | 8.84 | 8.0 | 7.04 |
| 6.0 | 7.24 | 6.0 | 6.13 | 6.0 | 6.08 | 8.0 | 5.25 |
| 4.0 | 4.26 | 4.0 | 3.10 | 4.0 | 5.39 | 19.0 | 12.50 |
| 12.0 | 10.84 | 12.0 | 9.13 | 12.0 | 8.15 | 8.0 | 5.56 |
| 7.0 | 4.82 | 7.0 | 7.26 | 7.0 | 6.42 | 8.0 | 7.91 |
| 5.0 | 5.68 | 5.0 | 4.74 | 5.0 | 5.73 | 8.0 | 6.89 |

The four data sets used by Anscombe had 11 observations. The x values are same for the first three datasets. The summary statistics are as follows:

* Mean of the x’s = 9.0 and Mean of the y’s = 7.50.
* Sample Variance of x (σ2) = 11.0 (sum of the squares of )
* Sample Variance of y (σ2) = 4.125 (sum of the squares of )
* Correlation between x and y = 0.816
* If we fit a regression line for these data sets, they have the same equation y = 3 + 0.5 x
* Coefficient of Determination of the linear regression R2 = 0.667

The following graphs are plotted using matplotlib in Python for the above datasets.

*The plot shows a variety of features quickly and vividly* which are not evident from the summary statistics. Anscombe mentions in his paper that if the summary statistics are presented, *figure 1, corresponding to dataset 1, is the kind of thing most people would see in their mind’s eye*. In figure 2, y has a smooth curve relation with x with little residual variability. In figure 3, all but one of the observations lie close to the regression line. In figure 4, we see the effect of a single outlier that plays a critical role in computing the slope. If we delete the outlier, the slope cannot be estimated. Thus, each of these 3 datasets illustrate a peculiar effect in an extreme form. We need to be careful about such effects that may cause the regression relation to be misleading.

## 5. Pearson’s R

Pearson's correlation coefficient (r) is a measure of the strength and direction of the relationship between the two random variables. It is computed as the covariance of the two variables divided by the product of their standard deviations.

The value of R is such that -1 < r < +1. A positive value indicates that with every increase in one variable, there is an increase of a fixed proportion in the other variable. A negative value indicates that with every increase in one variable, there is a decrease of a fixed proportion in the other variable. If r = 0, the two variables are not related to each other and there is correlation.

The covariance is computed as follows:

The standard deviation of x (σx) and y (σy) are computed as follows

|  |  |
| --- | --- |
|  |  |

Applying these values in the formula for Pearson’s Correlation Coefficient, we get

The above equation can be rearranged as follows (refer the computation in the next page):

One of the significant property of Pearson Correlation Coefficient is that it **does not change with a linear transformation of x or y values**, which means that in linear regression, we can scale the variables without changing the r value. Consequently the R-squared and the Adjusted R-squared values remain the same irrespective of the scaling.

Scaling is further explained in detail in the next section.

In Python, when we use pandas.DataFrame.corr() to compute pairwise correlation of columns with continuous numeric values, by default, Pearson method is used for the computation. The function returns a correlation matrix computed using Pearson method as a data frame.

We use scatter plots or heat maps to visualize such correlation between the variables.

## Pearson’s R - Coefficient of Correlation - Formula

**Rearranging the Numerator:**

**Rearranging the Denominator:**

Similarly we can compute:

Applying Equation (1) in the numerator of the formula for r and Equation (2) and (3) in the denominator of the formula for r, we get

## 6. Scaling

In multiple linear regression, as part of data preparation, we normalize the independent variables in the train data set to ensure that all the variables are on the same scale (or a comparable scale).

**Why is scaling performed?** Scaling makes it easy to interpret the model. If we do not normalize the scale, some of the coefficients (β0 and β1) obtained by fitting the regression model might be very large or very small compared to other coefficients. This gets annoying at the time of evaluating our model. Scaling ensures that the units of the coefficients obtained are all on the same scale. Scaling also helps in speeding up the calculations in the algorithm. For example, the convergence will be faster when applying the gradient descent algorithm.

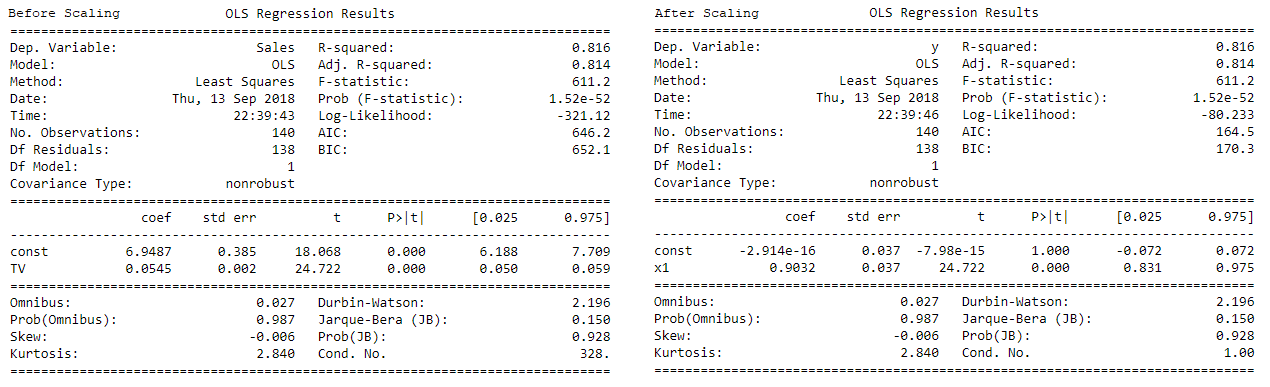
There are **two** **popular methods for scaling the variables**. Min-max Scaling and Standardization. In Python, we use the scikit-learn library for applying scaling.

|  |  |
| --- | --- |
| **Min-max scaling** is the simplest method and uses the following formula:  is the original value and is the normalized value. The scaled value is compressed within the range [-1, 1]. To apply min-max scaling, we have to import the following package from scikit-learn library:  from sklearn.preprocessing  import MinMaxScaler  scaler = MinMaxScaler() | **Standardization** is a method which scales the data in such a way that the mean and the standard deviation . The x value is replaced with the Z-score computed as follows:  Unlike min-max scaling the data is not compressed. In Python, we use the following command to import the standard scaler.  from sklearn.preprocessing  import StandardScaler  scaler = StandardScaler() |

To apply the scaler to the specific columns in the training dataset using the following command.

df\_train[column] = scaler.fit\_transform(df\_train[column])

Note that scaling will not affect the model. The summary statistics and the goodness of fit (the value of R-squared or Adjusted R-squared, F-Statistic) remains unchanged as shown below.



## 7. Variance Inflation Factor

VIF calculates how well a specific feature is explained by all the other features combined. It is a measure of multicollinearity. Suppose there are n independent variables that are used in a model to predict the target variable. VIF computes how a particular variable is correlated with other n - 1 independent variables. It is computed as follows:

If the VIF value is higher, then it means that the variable is highly correlated with other variables. **Sometimes the value of VIF is infinite.** This can happen when the value of R2 = 1, which can happen only when there is a perfect correlation and a linear relationship between the variable and all other variables i.e. the variable can be fully expressed in terms of other and hence redundant.

----------------------------------------------------------------------------------------------------------------------

## 8. Gauss-Markov theorem

Gauss-Markov theorem states that compared to all other linear unbiased estimator, the ordinary least squares (OLS) regression has the lowest sampling variance and gives us the best linear unbiased estimate (BLUE) possible, provided the following conditions are met:

1. Linearity: the parameters estimated using OLS method must be themselves linear.
2. Randomness: there is no recognizable pattern in the sample selected.
3. The expected value of the error term is zero for all observations.
4. The error term is normally distributed (normality).
5. The conditional variance of the error term is constant (homoscedasticity).
6. The error terms are independently distributed and are not correlated (independence).

----------------------------------------------------------------------------------------------------------------------

## 9. Gradient Descent Algorithm

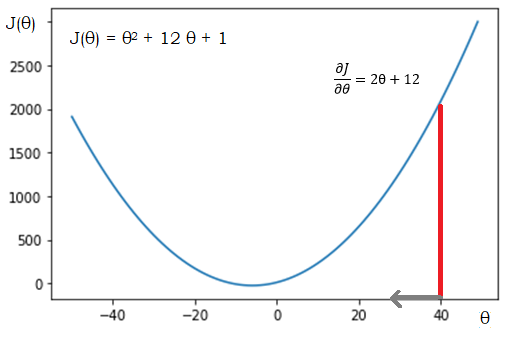
Gradient descent is an optimisation algorithm that optimises the cost function to reach the optimal solution. We will denote the cost function as J(θ). For optimizing the cost function, there are two different methods: closed form solution and iterative form solution. Gradient Descent is a first-order iterative optimization algorithm to find the minimum value of the cost function.

**Step 1:** For the cost function J(θ), start with a specific value for θ0 (the coefficient or gradient).

**Step 2:** Compute the derivate of the cost function at θ0.

**Step 3:**Multiply the derivate of the cost function by a number η called learning rate.

**Step 4:** Repeat the process till we converge at the optimal value for the cost function.

For example, let us define a simple cost function

J(θ) = θ2 + 12 θ + 1; the derivate

If we choose a learning rate η = 0.01, and the starting value for θ as 40,

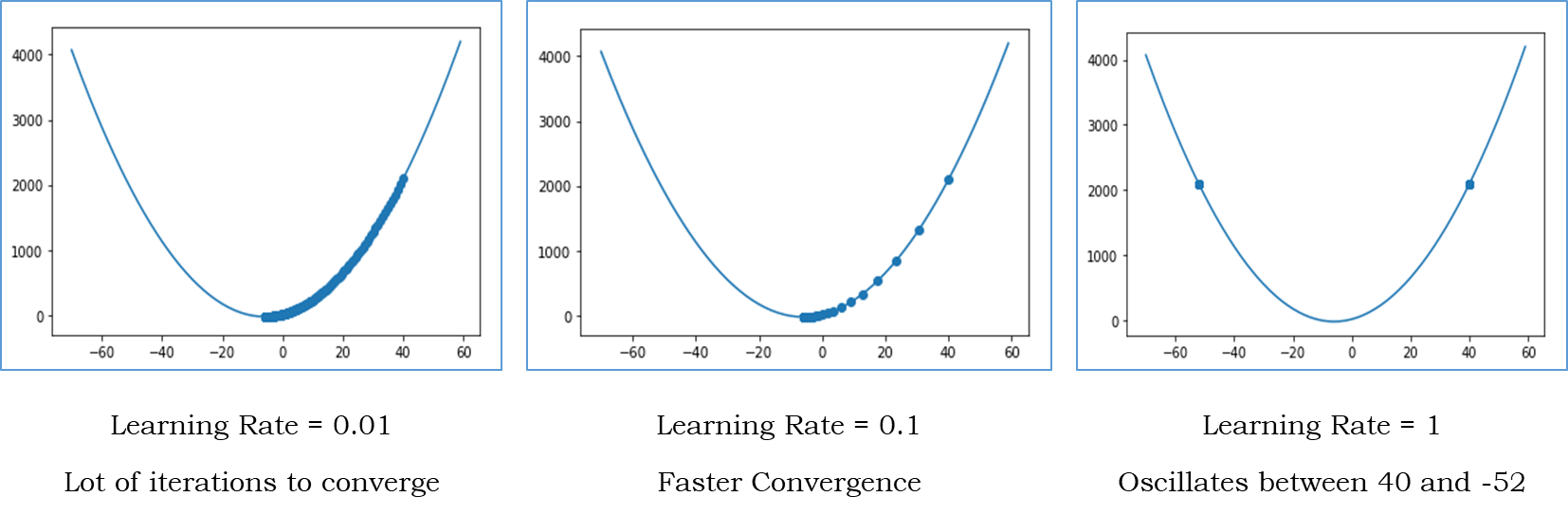
θ1 = 40 - (0.01) (2 x 40 + 12) = 40 - 0.92 = 39.08

θ2 = 39.08 - (0.01) (2 x 39.08 + 12) = 38.18

θ3 = 38.18, θ4 = 37.3, θ5 = 36.43 and so on.

Usually the value for the learning rate is chosen manually. If the learning rate is very small, it will take a lot of iterations to converge at the optimal value. If the learning rate is large, the value may be oscillating between the arcs and may never converge at the optimal point.

In the above case, if we choose η = 1, then θ1 = 40 - 92 = -52 and θ2 = -52 - (2 \* -52 + 12) = 40. The value oscillates between 40 and -52 without reaching convergence. So it is always a good practice to choose a small value for the learning rate and move towards the negative of the gradient.

Plots generated programmatically with Python for different learning rates for the cost function J(θ) = θ2 + 12 θ + 1:

**Applications of Gradient Descent Algorithm**

The Gradient Descent Algorithm is applied not only in linear regression, but also for logistic regression and neural networks. In linear regression, to create the regression line, we use the function: where β0 is the intercept of the fitted line and β1 is the coefficient for the independent variable x. The actual value of where ε is the residual and is the difference between the actual value and the predicted value. .

The sum of the squares of the residuals:

This is our cost function J(β0 , β1). We have to find the values of β0 and β1 such that sum of the squares of the residuals is minimum. We can achieve this using the Gradient Descent Algorithm.

## 10. Q-Q Plot in Linear Regression

Q-Q plot or quantile-to-quantile plot is created with the quantiles of one variable on the x-axis and that of the other variable in the y-axis. In the ordinary least squares method, we assume that the error terms are normally distributed. How do we validate this? One method is to use a Q-Q plot of the error terms against the corresponding values computed (or the predicted y values). If the error terms are normally distributed then the quantiles of error terms are near enough to the quantiles of the corresponding computed values i.e. they fall on a straight line.

In Python, we can create a Q-Q plot as follows for the error terms:

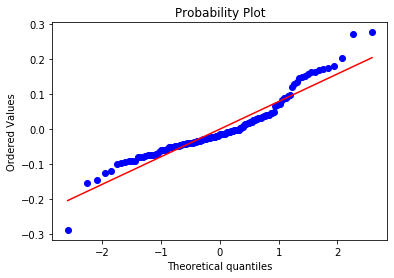
# Creating a Q-Q Plot

import scipy.stats as scs

import matplotlib.pyplot as plt

scs.probplot(y\_train - y\_train\_pred, dist="norm", plot = plt)

The Q-Q plot for the car price prediction model created as part of the assignment is given below:



The probability distribution is on a straight line, indicating the normality of the error terms.