**Linear Regression**

Contents

[**1.** **Introduction** 3](#_Toc518460139)

[**2.** **Mathematics** 3](#_Toc518460140)

[2.1. Understand the algorithm 4](#_Toc518460141)

[**3.** **Literature Review** 4](#_Toc518460142)

[**4.** **EDA** 4](#_Toc518460143)

[4.1. Assumptions for Linear Regression 5](#_Toc518460144)

[4.2. Unusual and influential data 5](#_Toc518460145)

[**5.** **Data Preparation** 6](#_Toc518460146)

[**6.** **Model Training** 6](#_Toc518460147)

[**7.** **Model Diagnostics** 6](#_Toc518460148)

[7.1. What Is R-squared? 7](#_Toc518460149)

[Are High R-squared Values Inherently Good? 9](#_Toc518460150)

[**8.** **Model Evaluation** 10](#_Toc518460151)

[8.1. RMSE: 10](#_Toc518460152)

[8.2. Feature importance: 10](#_Toc518460153)

[8.3. Plotting Cross-Validated Predictions 10](#_Toc518460154)

[**9.** **Optimization** 12](#_Toc518460155)

[9.1. Gradient boosting: 12](#_Toc518460156)

[**10.** **Deployment** 12](#_Toc518460157)

[**11.** **Case Studies** 13](#_Toc518460158)

[11.1. Boston Housing 13](#_Toc518460159)

[**12.** **Further Studies** 13](#_Toc518460160)

[12.1. P-value for feature importance 13](#_Toc518460161)

[12.2. Gradient dissent for optimization 13](#_Toc518460162)

[12.3. Feature scaling 13](#_Toc518460163)

[**13.** **Appendix** 13](#_Toc518460164)

[13.1. Links reviewed during learning: 13](#_Toc518460165)

# **Introduction**

Objective: We want to learn Linear Regression in detail. While we follow the structure of data science analytical project, we will look into techniques too. Boston housing is our case study for this learning.

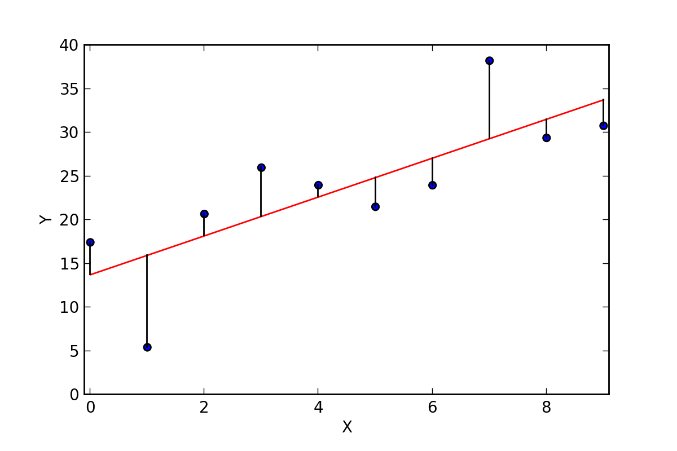
# **Mathematics**

A relationship between variables Y and X is represented by this equation:

**Y = mX + b**

In this equation, Y is the dependent variable — or the variable we are trying to predict or estimate; X is the independent variable — the variable we are using to make predictions; m is the slope of the regression line — it represent the effect X has on Y. In other words, if X increases by 1 unit, Y will increase by exactly m units. (**“Full disclosure”**: this is true only if we know that X and Yhave a linear relationship. In almost all linear regression cases, this will not be true!) b is a constant, also known as the Y-intercept. If X equals 0, Y would be equal to b (**Caveat**: see full disclosure from earlier!). This is not necessarily applicable in real life — we won’t always know the exact relationship between X and Y or have an exact linear relationship.

These caveats lead us to a **Simple Linear Regression** (SLR). In a SLR model, we build a model based on data — the slope and Y-intercept derive from the data; furthermore, we don’t need the relationship between X and Y to be exactly linear. SLR models also include the errors in the data (also known as residuals). I won’t go too much into it now, maybe in a later post, but residuals are basically the differences between the true value of Y and the predicted/estimated value of Y. It is important to note that in a linear regression, we are trying to predict a continuous variable. In a regression model, we are trying to minimize these errors by finding the “line of best fit” — the regression line from the errors would be minimal. We are trying to minimize the length of the black lines (or more accurately, the distance of the blue dots) from the red line — as close to zero as possible. It is related to (or equivalent to) minimizing the [mean squared error (MSE)](https://en.wikipedia.org/wiki/Mean_squared_error) or the sum of [squares of error (SSE)](https://en.wikipedia.org/wiki/Residual_sum_of_squares), also called the “residual sum of squares.” (RSS) but this might be beyond the scope of this blog post :-)



In most cases, we will have more than one independent variable — we’ll have multiple variables; it can be as little as two independent variables and up to hundreds (or theoretically even thousands) of variables. in those cases we will use a Multiple Linear Regression model (MLR). The regression equation is pretty much the same as the simple regression equation, just with more variables:

**Y’i = b0 + b1X1i + b2X2i**

## Understand the algorithm

Most people are familiar with the equation of a line:

y = mx + b

Let's say we own a lemonade stand. We could make a really simple model of our lemonade stand by saying that the lemonade's revenue is equal to the price, sales, and rent. We could simplify this to a linear equation that looks like this

PredictedRevenue = (Sales \* Price) - Rent

But there is more to selling lemonade than just sales and price. What if we could factor in other variables like price changes, sales, temperature, and foot traffic. We could create a more complex model that would look something like this.

PredictedRevenue = (Sales \* Weight) + ( Price \* Weight) + (Temperature \* Weight) + (FootTraffic \* Weight) - Rent

The weights in the example above are known as coefficients. Using [Multiple Linear Regression](https://en.wikipedia.org/wiki/Linear_regression) we can calculate the coefficiens of the different factors given a target.

So what this is saying is that we can string together all of these coefficients and create an equation to predict the price of a house given these 13 variables. It would look like this:

\hat{Y} = \beta\_0 + \beta\_1 x\_1 + \beta\_2 x\_2 + \beta\_3 x\_3 + ... \beta\_n N

# **Literature Review**

# **EDA**

EDA is a practice of iteratively asking a series of questions about data and trying to gain useful insights out of the data to answer the questions and essentially to influence our decision making. And, Linear Regression is one of the most often used algorithm and super useful for EDA.

When run regression models, you need to do regression disgnostics. Without verifying that your data have met the regression assumptions, your results may be misleading. This section will explore how to do regression diagnostics.

## Assumptions for Linear Regression

* Linearity - the relationships between the predictors and the outcome variable should be linear
* Normality - the errors should be normally distributed - technically normality is necessary only for the t-tests to be valid, estimation of the coefficients only requires that the errors be identically and independently distributed
* Homogeneity of variance (homoscedasticity) - the error variance should be constant
* Independence - the errors associated with one observation are not correlated with the errors of any other observation
* Errors in variables - predictor variables are measured without error
* Model specification - the model should be properly specified (including all relevant variables, and excluding irrelevant variables)

Additionally, there are issues that can arise during the analysis that, while strictly speaking, are not assumptions of regression, are none the less, of great concern to regression analysts.

* Influence - individual observations that exert undue influence on the coefficients
* Collinearity - predictors that are highly collinear, i.e. linearly related, can cause problems in estimating the regression coefficients.

## Unusual and influential data

A single observation that is substantially different from all other observations can make a large difference in the results of your regression analysis. If a single observation (or small group of observations) substantially changes your results, you would want to know about this and investigate further. There are three ways that an observation can be unusual.

* Outliers: In linear regression, an outlier is an observation with large residual. In other words, it is an observation whose dependent-variable value is unusual given its values on the predictor variables. An outlier may indicate a sample peculiarity or may indicate a data entry error or other problem.
* Leverage: An observation with an extreme value on a predictor variable is called a point with high leverage. Leverage is a measure of how far an observation deviates from the mean of that variable. These leverage points can have an effect on the estimate of regression coefficients.
* Influence: An observation is said to be influential if removing the observation substantially changes the estimate of coefficients. Influence can be thought of as the product of leverage and outlierness.

<http://songhuiming.github.io/pages/2016/12/31/linear-regression-in-python-chapter-2/>

Great Visualization on EDA: <https://blog.exploratory.io/a-practical-guide-of-exploratory-data-analysis-with-linear-regression-part-1-9f3a182d7a92>

[Verifying the Assumptions of Linear Regression in Python and R](https://towardsdatascience.com/verifying-the-assumptions-of-linear-regression-in-python-and-r-f4cd2907d4c0)

[Assumptions Of Linear Regression Algorithm](https://towardsdatascience.com/assumptions-of-linear-regression-algorithm-ed9ea32224e1)

# **Data Preparation**

The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, record, and attribute selection as well as transformation and cleaning of data for modeling tools.

In this example the data has already mostly been prepared for us. In this step it is common to consider some of the following.

* Do we have any missing fields?
* Do we need to convert any categorical to dummy variables?
* Are there any outliers that need to be examined and/or removed?
* Do we need to create any precomputed fields?
* Do we need to scale our data so that it isn't thrown off by the algorithm we choose?

Imputing the missing values : <http://scikit-learn.org/dev/modules/impute.html>

# **Model Training**

In this phase, various modeling techniques are selected and applied, and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data mining problem type. Some techniques have specific requirements on the form of data. Therefore, stepping back to the data preparation phase is often needed.

we will only be looking at the [LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html) model to understand the concepts behind regression.

X is used as a variable to represent a matrix of known variables

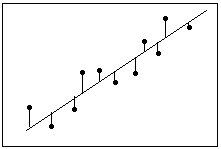
y is used to domonstrate the known values which are the target values (or the labels)

# **Model Diagnostics**

<http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>

After you have fit a linear model using regression analysis, ANOVA, or design of experiments (DOE), you need to determine how well the model fits the data. To help you out, [Minitab statistical software](http://www.minitab.com/en-us/products/minitab/) presents a variety of goodness-of-fit statistics. In this post, we’ll explore the R-squared (R2 ) statistic, some of its limitations, and uncover some surprises along the way. For instance, low R-squared values are not always bad and high R-squared values are not always good!

What Is Goodness-of-Fit for a Linear Model?

Definition: Residual = Observed value - Fitted value

Linear regression calculates an equation that minimizes the distance between the fitted line and all of the data points. Technically, ordinary least squares (OLS) regression minimizes the sum of the squared residuals.

In general, a model fits the data well if the differences between the observed values and the model's predicted values are small and unbiased.

Before you look at the statistical measures for goodness-of-fit, you should [check the residual plots](http://blog.minitab.com/blog/adventures-in-statistics/why-you-need-to-check-your-residual-plots-for-regression-analysis). Residual plots can reveal unwanted residual patterns that indicate biased results more effectively than numbers. When your residual plots pass muster, you can trust your numerical results and check the goodness-of-fit statistics.

## What Is R-squared?

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model. Or:

R-squared = Explained variation / Total variation

R-squared is always between 0 and 100%:

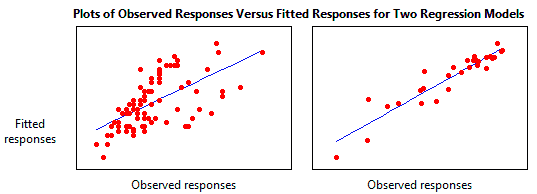
0% indicates that the model explains none of the variability of the response data around its mean.

100% indicates that the model explains all the variability of the response data around its mean.

In general, the higher the R-squared, the better the model fits your data. However, there are important conditions for this guideline that I’ll talk about both in this post and my next post.

Graphical Representation of R-squared

Plotting fitted values by observed values graphically illustrates different R-squared values for regression models.



The regression model on the left accounts for 38.0% of the variance while the one on the right accounts for 87.4%. The more variance that is accounted for by the regression model the closer the data points will fall to the fitted regression line. Theoretically, if a model could explain 100% of the variance, the fitted values would always equal the observed values and, therefore, all the data points would fall on the fitted regression line.

Key Limitations of R-squared

R-squared cannot determine whether the coefficient estimates and predictions are biased, which is why you must assess the residual plots.

R-squared does not indicate whether a regression model is adequate. You can have a low R-squared value for a good model, or a high R-squared value for a model that does not fit the data!

[The R-squared in your output is a biased estimate of the population R-squared](http://blog.minitab.com/blog/adventures-in-statistics/r-squared-shrinkage-and-power-and-sample-size-guidelines-for-regression-analysis).

Are Low R-squared Values Inherently Bad?

No! There are two major reasons why it can be just fine to have low R-squared values.

In some fields, it is entirely expected that your R-squared values will be low. For example, any field that attempts to predict human behavior, such as psychology, typically has R-squared values lower than 50%. Humans are simply harder to predict than, say, physical processes.

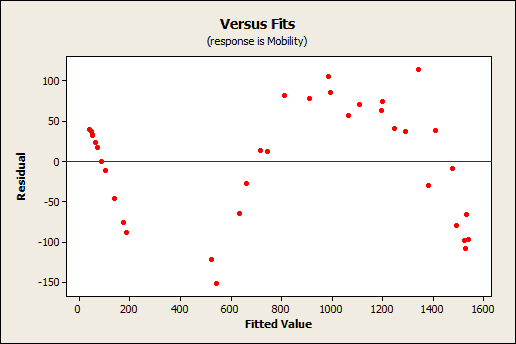
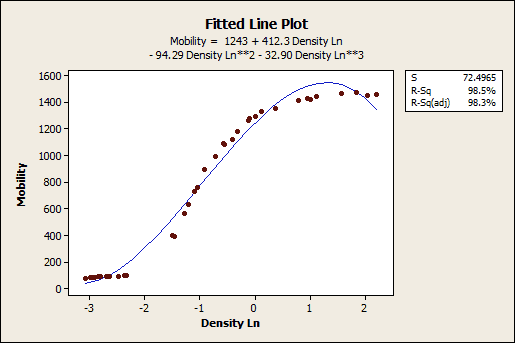
Furthermore, if your R-squared value is low but you have statistically significant predictors, you can still draw important conclusions about how changes in the predictor values are associated with changes in the response value. Regardless of the R-squared, the significant coefficients still represent the mean change in the response for one unit of change in the predictor while holding other predictors in the model constant. Obviously, this type of information can be extremely valuable.

[See a graphical illustration of why a low R-squared doesn't affect the interpretation of significant variables.](http://blog.minitab.com/blog/adventures-in-statistics/how-to-interpret-a-regression-model-with-low-r-squared-and-low-p-values)

A low R-squared is most problematic when you want to produce predictions that are reasonably precise (have a small enough [prediction interval](http://blog.minitab.com/blog/adventures-in-statistics/when-should-i-use-confidence-intervals-prediction-intervals-and-tolerance-intervals)). How high should the R-squared be for prediction? Well, that depends on your requirements for the width of a prediction interval and how much variability is present in your data. While a high R-squared is required for precise predictions, it’s not sufficient by itself, as we shall see.

## Are High R-squared Values Inherently Good?

No! A high R-squared does not necessarily indicate that the model has a good fit. That might be a surprise, but look at the fitted line plot and residual plot below. The fitted line plot displays the relationship between semiconductor electron mobility and the natural log of the density for real experimental data.



The fitted line plot shows that these data follow a nice tight function and the R-squared is 98.5%, which sounds great. However, look closer to see how the regression line systematically over and under-predicts the data (bias) at different points along the curve. You can also see patterns in the Residuals versus Fits plot, rather than the randomness that you want to see. This indicates a bad fit, and serves as a reminder as to why you should always check the residual plots.

This example comes from my post about choosing between [linear and nonlinear regression](http://blog.minitab.com/blog/adventures-in-statistics/linear-or-nonlinear-regression-that-is-the-question). In this case, the answer is to use nonlinear regression because linear models are unable to fit the specific curve that these data follow.

However, similar biases can occur when your linear model is missing important predictors, polynomial terms, and interaction terms. Statisticians call this specification bias, and it is caused by an underspecified model. For this type of bias, you can fix the residuals by adding the proper terms to the model.

For more information about how a high R-squared is not always good a thing, read my post [Five Reasons Why Your R-squared Can Be Too High](http://blog.minitab.com/blog/adventures-in-statistics/five-reasons-why-your-r-squared-can-be-too-high).

Closing Thoughts on R-squared

R-squared is a handy, seemingly intuitive measure of how well your linear model fits a set of observations. However, as we saw, R-squared doesn’t tell us the entire story. You should evaluate R-squared values in conjunction with residual plots, other model statistics, and subject area knowledge in order to round out the picture (pardon the pun).

While R-squared provides an estimate of the strength of the relationship between your model and the response variable, it does not provide a formal hypothesis test for this relationship. The [F-test of overall significance](http://blog.minitab.com/blog/adventures-in-statistics/what-is-the-f-test-of-overall-significance-in-regression-analysis) determines whether this relationship is statistically significant.

In my next blog, we’ll continue with the theme that R-squared by itself is incomplete and look at two other types of R-squared: [adjusted R-squared and predicted R-squared](http://blog.minitab.com/blog/adventures-in-statistics/multiple-regession-analysis-use-adjusted-r-squared-and-predicted-r-squared-to-include-the-correct-number-of-variables). These two measures overcome specific problems in order to provide additional information by which you can evaluate your regression model’s explanatory power.

For more about R-squared, learn the answer to this eternal question: [How high should R-squared be?](http://blog.minitab.com/blog/adventures-in-statistics/how-high-should-r-squared-be-in-regression-analysis)

# **Model Evaluation**

At this stage in the project you have built a model (or models) that appears to have high quality, from a data analysis perspective. Before proceeding to final deployment of the model, it is important to more thoroughly evaluate the model, and review the steps executed to construct the model, to be certain it properly achieves the business objectives. A key objective is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached.

Let's look at the business objectives again:

What is a fair price for a house? - We now have a model that will give us a projected price. It might not be perfectly accurate but it will give us

Identify what makes a property valuable? We know that the the features that minimized the error were LSTAT, RM, PTRATIO

## RMSE:

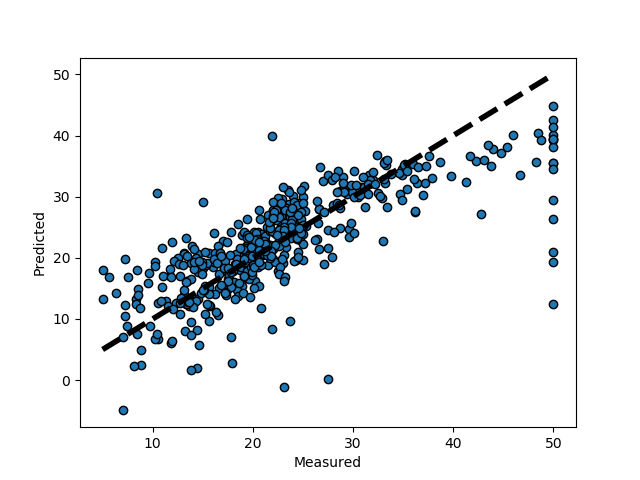
<https://stackoverflow.com/questions/17197492/root-mean-square-error-in-python>

## Feature importance:

<https://datascienceplus.com/linear-regression-in-python-predict-the-bay-areas-home-prices/>

## Plotting Cross-Validated Predictions

This example shows how to use cross\_val\_predict to visualize prediction errors.



**from** **sklearn** **import** datasets

**from** **sklearn.model\_selection** **import** [cross\_val\_predict](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_predict.html#sklearn.model_selection.cross_val_predict)

**from** **sklearn** **import** linear\_model

**import** **matplotlib.pyplot** **as** **plt**

lr = [linear\_model.LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression)()

boston = [datasets.load\_boston](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html#sklearn.datasets.load_boston)()

y = boston.target

*# cross\_val\_predict returns an array of the same size as `y` where each entry*

*# is a prediction obtained by cross validation:*

predicted = [cross\_val\_predict](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_predict.html#sklearn.model_selection.cross_val_predict)(lr, boston.data, y, cv=10)

fig, ax = [plt.subplots](http://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplots.html#matplotlib.pyplot.subplots)()

ax.scatter(y, predicted, edgecolors=(0, 0, 0))

ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)

ax.set\_xlabel('Measured')

ax.set\_ylabel('Predicted')

[plt.show](http://matplotlib.org/api/_as_gen/matplotlib.pyplot.show.html#matplotlib.pyplot.show)()

# **Optimization**

## Gradient boosting:

Prediction with gradient boosted tree

**from** **sklearn.ensemble** **import** GradientBoostingRegressor

clf = GradientBoostingRegressor()

clf.fit(X\_train, y\_train)

predicted = clf.predict(X\_test)

expected = y\_test

plt.figure(figsize=(4, 3))

plt.scatter(expected, predicted)

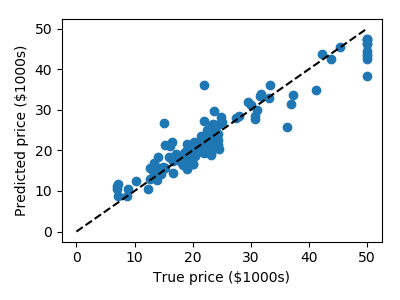
plt.plot([0, 50], [0, 50], '--k')

plt.axis('tight')

plt.xlabel('True price ($1000s)')

plt.ylabel('Predicted price ($1000s)')

plt.tight\_layout()



# **Deployment**

Creation of the model is generally not the end of the project. Even if the purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in a way that is useful to the customer. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable data scoring (e.g. segment allocation) or data mining process. In many cases it will be the customer, not the data analyst, who will carry out the deployment steps. Even if the analyst deploys the model it is important for the customer to understand up front the actions which will need to be carried out in order to actually make use of the created models.

What would deploying this model look like?

Where would we deploy this model? Excel, Website, App?

Should we deploy this model?

What is the cost of inaccuracy?

# **Case Studies**

## Boston Housing

* 1. Git: [link](https://gist.github.com/jpotts18/dac94dc9514172ce020c)

# **Further Studies**

## P-value for feature importance

## Gradient dissent for optimization

## Feature scaling

# **Appendix**

## Links reviewed during learning: