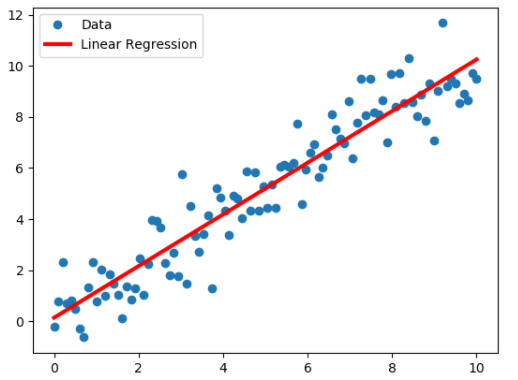
**SUPERVISED LEARNING**

**Linear Regression**

**Linear Regression Theory**

We will encounter two problems when dealing with data: regression and classification. In **classification** problems, we divide our data into several categories (classes). The goal is to predict the classes. The number of classes or categories can be two or more.   
  
But some problems require a different approach - for example, let's consider estimating big companies' gross revenue at the end of a year. In this case, we do not try to classify; rather, we try to quantify using **regression** theory. This means that, in theory, we are interested in an endless number of values. In the below picture, you can see a classical linear regression plot graphic. Blue points represent the real data, and the **red line**is the linear regression line.



Linear regression models the relationship between one or more observed features and the target variable. Based on the data we have, we try to derive a function that will lead us to the desired result. This function creates a line (red line above) where we can search for property values and find predicted target values.   
  
What we need is to find a way to model a model - that is, a mathematical explanation - of the basic relationship between features and target. Conceptually,**we want to draw a line**that we can draw from the scatter chart of our known data, which reasonably **well represents the relationship between features and target**, and at the same time, we can well predict values that are not directly observed in our data.

The following Wikipedia based keywords and definitions we need to know to understand Linear Regression:

* [Simple Linear Regression](https://en.wikipedia.org/wiki/Simple_linear_regression) : a [linear regression](https://en.wikipedia.org/wiki/Linear_regression) model with a single [explanatory variable.](https://en.wikipedia.org/wiki/Covariate)
* [OLS(Ordinary Least Squares)](https://en.wikipedia.org/wiki/Ordinary_least_squares): a type of [linear least squares](https://en.wikipedia.org/wiki/Linear_least_squares) method for estimating the unknown [parameters](https://en.wikipedia.org/wiki/Statistical_parameter) in a linear regression model.
* [Residuals](https://en.wikipedia.org/wiki/Errors_and_residuals) : The difference between the observed value and the *estimated* value of the quantity of interest (for example, a [sample mean](https://en.wikipedia.org/wiki/Sample_mean)).
* [Cost Function](https://en.wikipedia.org/wiki/Loss_function) : In statistics, typically a loss function is used for [parameter estimation](https://en.wikipedia.org/wiki/Parameter_estimation), and the event in question is some function of the difference between estimated and true values for an instance of data.
* [Gradient Descent](https://en.wikipedia.org/wiki/Gradient_descent) : a [first-order](https://en.wikipedia.org/wiki/Category:First_order_methods) iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the [gradient](https://en.wikipedia.org/wiki/Gradient) (or approximate gradient) of the function at the current point, because this is the direction of steepest descent.

We don't need any iterative method to minimize the cost function for simple linear regression, we simply use OLS(Ordinary Least Squares) Method to find the best fit line . However, If you have feature more than one than this close form solution becomes unscalable. In this case we use Gradient Descent Method to minimize the cost function.

Some examples of linear regression problems are:

* Stock Price Prediction
* House/Car Price Prediction etc.

**Q: What Is Linear Regression ?**  
**A:**In simple terms, linear regression is a method of finding the best straight line fitting to the given data and also finding the best linear relationship between the independent and dependent variables. We use Least Squares Method  to decide which line is the best fit the model.

- **Interview Q&A**

UDEMY 2021 Python for Machine Learning & Data Science Masterclass 🡪

"81. Introduction to Linear Regression Section",  
"82. Linear Regression - Algorithm History",  
"83. Linear Regression - Understanding Ordinary Least Squares",   
"84. Linear Regression - Cost Functions" ,  
"85. Linear Regression - Gradient Descent",   
"86. Python coding Simple Linear Regression"

### Linear Regression with Python

In this lesson, you're going to have a brief overview of Sicikit-Learn. Scikit-Learn is a library containing many machine learning algorithms. The most important factor of this library is utilizing what they call a generalized estimate API framework to call and use the various models that you're going to be creating.

You will be working with the Advertising dataset and creating a linear regression model to predict sales.

Since this is your first machine learning algorithm, you will work with some created datasets. Later you're going to progress to use real data sets. Right now, we don't want to focus too much on cleaning data.

**Regression Error Metrics**

In regression problems, we have to use **metrics designed for continuous values**. Regression error metrics inform us about how much the actual values deviate from the regression line, which we estimate.

In this lesson, you will learn some of the **most common evaluation metrics** for regression:

##### [Mean Absolute Error :](https://en.wikipedia.org/wiki/Mean_absolute_error) The mean absolute error (MAE) is a measure of [errors](https://en.wikipedia.org/wiki/Error_(statistics)) between paired observations expressing the same phenomenon. It is thus an arithmetic average of the absolute errors , where  is the prediction and  the true value.

##### [Mean Squared Error :](https://en.wikipedia.org/wiki/Mean_squared_error)  The mean squared error (MSE) of an [estimator](https://en.wikipedia.org/wiki/Estimator) (of a procedure for estimating an unobserved quantity) measures the [average](https://en.wikipedia.org/wiki/Expected_value) of the squares of the [errors](https://en.wikipedia.org/wiki/Error_(statistics))—that is, the average squared difference between the estimated values and the actual value.

##### [Root Mean Square Error :](https://en.wikipedia.org/wiki/Root-mean-square_deviation) The root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an [estimator](https://en.wikipedia.org/wiki/Estimator) and the values observed.

##### 💡Tips:

##### You can easily find the data sets and the notebooks from the Resources folder at the beginning of this session on Udemy.

**In this lesson, you will learn how to:**

##### Import the data,

##### Start to explore the data,

##### Separate the data into X and y (where X is the features and y is what we're trying to predict),

##### Create a linear regression object and fit the model and then evaluate the model by using evaluation metrics.

##### Use a residual plot to interpret the dataset is a valid choice for linear regression.

##### Find out what the coefficients really mean in terms of the linear regression model?

##### Dump or load a model with joblib library.

UDEMY 2021 Python for Machine Learning & Data Science Masterclass 🡪

##### "87. Overview of Scikit-Learn and Python",

##### "88. Linear Regression - Scikit-Learn Train Test Split"

##### "89. Linear Regression - Scikit-Learn Performance Evaluation - Regression"

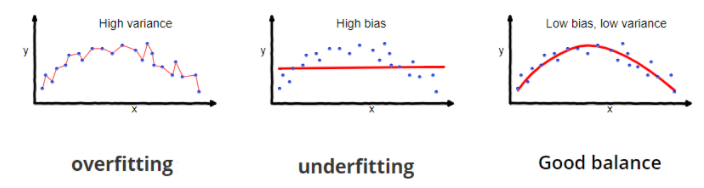
##### "90. Linear Regression - Residual Plots"

##### "91. Linear Regression - Model Deployment and Coefficient Interpretation"

### Bias-Variance Trade-Off - 1

The Bias-variance trade-off and the associated with them, the **underfitting**and **overfitting**concepts will often appear during machine learning lessons. A clear understanding of these concepts will help us grasp the learning and testing process of machine learning and contribute to creating better models. As remembered from the previous section, we create functional structures or curves to achieve the predictions through the features we have. These functional structures give us clues about the success of our model.

The purpose of creating a successful and balanced model is to enable our model to **learn the general structure of the data.** Learning the general structure of the data by our model is very critical. It will enable us to estimate the new data easily and with the minimum error by our model. You can see three types of functional structures (models) in the picture below. It can be seen at first glance that the two on the left side are abnormal.  Let's look at why they are abnormal in the next lesson.



### Bias Variance Trade-Off - 2

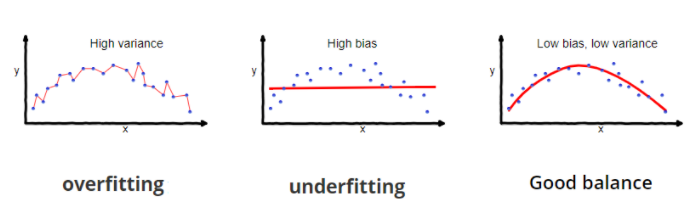
#### Underfitting and Overfitting Problems

**High Bias and Underfitting Problem:**  
  
In basic, the distance between real values and estimated values expresses as a bias. When we focus on the second graph at below, it will be seen that the distances between many real values and our curve are quite high. Also, our curve (red line) is not compatible with the normal distribution of the data. There is a high bias in favor of certain observations.

This means the curve we created with our model does **not represent the data** sufficiently. In this situation, we can state that our model has learned incompletely. We call this situation **underfitting**.

**💡Tips-1:**

In underfitting, the model is not able to learn the data adequately and therefore cannot represent the structure of the data sufficiently.



**High Variance and Overfitting Problem:**

Variance refers to the flexibility and precision of a model. In high variance models, they mapped the dataset exactly, instead of learning the structure of it as you can see in the first graph above. Although this may seem like a good thing at first, but actually, it causes the model to over-learn or memorize.

As we mentioned before, a successful model is to learn the general structure of the data, not to memorize it. The model we will obtain in overfitting is a model-specific to that data. It is not a general model. When the model encounters different data, it will produce erroneous results because it is not designed according to the new data.

**💡Tips-2:**

In Overfitting, the model learns the data in all its details, in other words, it memorizes the data. It's not a good thing, because, in this way, the model is not a general model that represents the data structure, just a special model of that specific data.

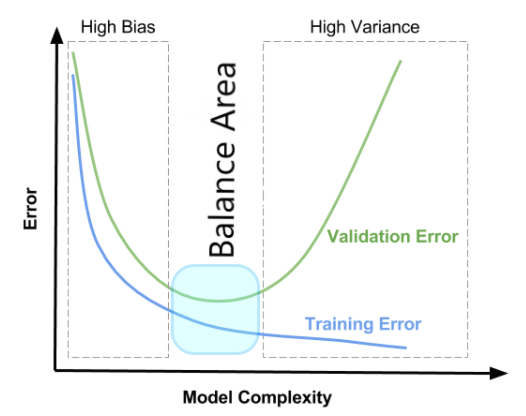
**Example: Underfitting and Overfitting Students**

Let's consider two students who take an exam. The first student works superficially on the exam and does not master the subject. This student will not be able to give clear and correct answers to the questions. We can say that this student could not learn the subject exactly, that is underfitting.  
  
Suppose the second student memorizes all the questions that have arisen before, rather than understanding the subject. This student will also make a mistake in the first question he/she encounters because he/she doesn't learn the general structure of the subject and only take the exam by memorizing the questions. We can also say **overfitting** for this student.

### Bias Variance Trade-off - 3

#### Training Error vs. Validation Error (Test Error)

In the Linear Regression lesson, we explored train and test (validation) errors. Now let us briefly examine the **relationship of these errors with overfitting and underfitting**and how they will help us build a balanced and successful model.



One of the most important criteria for our model success is to decrease error rates to make correct predictions. To reduce the error rates, we can make many changes in the parameters and features of our model that we will use further. While these changes will reduce the error rate and evolve towards a better model, on the other hand,  it will also make our model even **more complex**.     
  
While trying to reduce the error rates, we also make our model more complex, as mentioned above, and put it at the **risk of high variance**, that is, **overfitting**. This paradox is exactly the meaning of bias-variance trade-off.  
  
As can be clearly seen from the picture above, this paradox has a balance area between the high bias and high variance. The model's success is not that the training error is too low, but this is where the training and test errors are in balance. **The point that a successful model should have in this balance area.**

**Q: What’s the trade-off between bias and variance?**  
 **A:** The bias-variance trade-off is the problem of simultaneously minimizing two sources of error that prevent supervised learning algorithms from generalizing beyond their training set. Our goal is to find the optimal middle ground where the errors from both over-fitting and under-fitting are minimal.  We want a model that is highly enough to capture the signals in our data, but not too complex that it can’t be applied to new data.

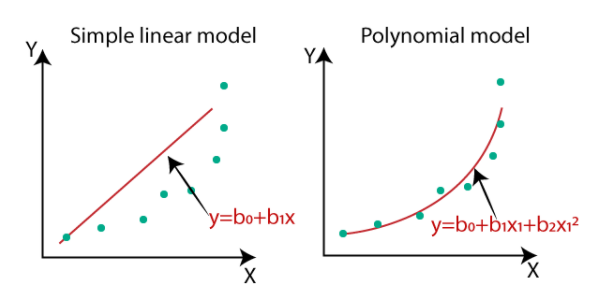
 - **Interwiev Q&A**

UDEMY 2021 Python for Machine Learning & Data Science Masterclass 🡪

"95. Bias Variance Trade-Off"

### Polynomial Regression

[Polynomial Regression](https://en.wikipedia.org/wiki/Polynomial_regression) gives the best estimation of the connection between the dependent and independent variable.

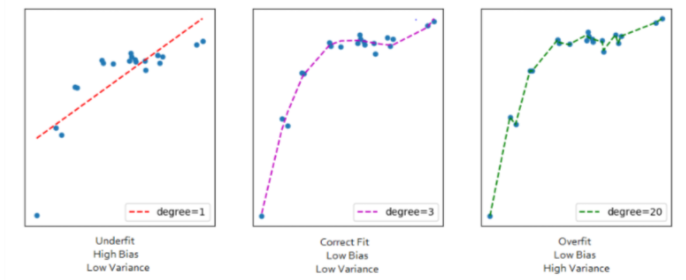


The polynomial models can be utilized in those circumstances where the relationship between dependent and independent variables is curvilinear. Sometimes a nonlinear relationship in a little scope of dependent variables can likewise be demonstrated by polynomials.  
  
Polynomial regression models are simply artificially created estimators based on existing data. We use mathematical tricks to generate new features, which causes our fit line to behave nonlinearly.   
  
In polynomial regression models, the degree of the polynomial must be taken into account. It has a major effect on the model's complexity and, if the degree is high, can lead to overfitting. On the other hand it is equal to the simple linear regression if degree = 1.  
  
UDEMY 2021 Python for Machine Learning & Data Science Masterclass 🡪

" 92. Polynomial Regression - Theory and Motivation"

**Polynomial Regression with Python**

In this section, you will continue on the Advertisement dataset. You will create polynomial features from existing feature columns. For this purpose, you will implement the PolynomialFeatures() method of the Scikit-Learn library.



**💡Tips:**

You can easily find the data sets and the notebooks from the Resources folder at the beginning of this session on Udemy.

In this lesson, you will :

* Convert X features with PolynomialFeatures() method,
* Explore the poly features for the Advertisement dataset,
* Perform Train, Test, Split on polynomial features,
* Create a linear regression model,
* Evaluate the performance metrics and analyze coefficients,
* Differentiate the degree of the polynomial model and examine train test errors.

UDEMY 2021 Python for Machine Learning & Data Science Masterclass 🡪

"93. Polynomial Regression - Creating Polynomial Features"

"94. Polynomial Regression - Training and Evaluation"

"96. Polynomial Regression - Choosing Degree of Polynomial"