**Machine Learning A-Z**: <https://radwin.udemy.com/machinelearning/learn/v4/t/lecture/5772258?start=0>

1. **Data preprocessing**
   1. importing libraries
   2. importing the dataset

*#Previous versions : y = dataset.iloc[:, 2].values returns a time series but*

*# y = dataset.iloc[:, [2]].values returns a pandas dataframe*

* 1. splitting the dataset into the Training set and Test set
  2. feature scaling

**General idea:**

From a library import a specific class

*Eg : from sklearn.linea\_model import LinearRegression*

Create an object of that class

*regressor = LinearRegression()*

Use a method of the class (a class contains some specific methods)

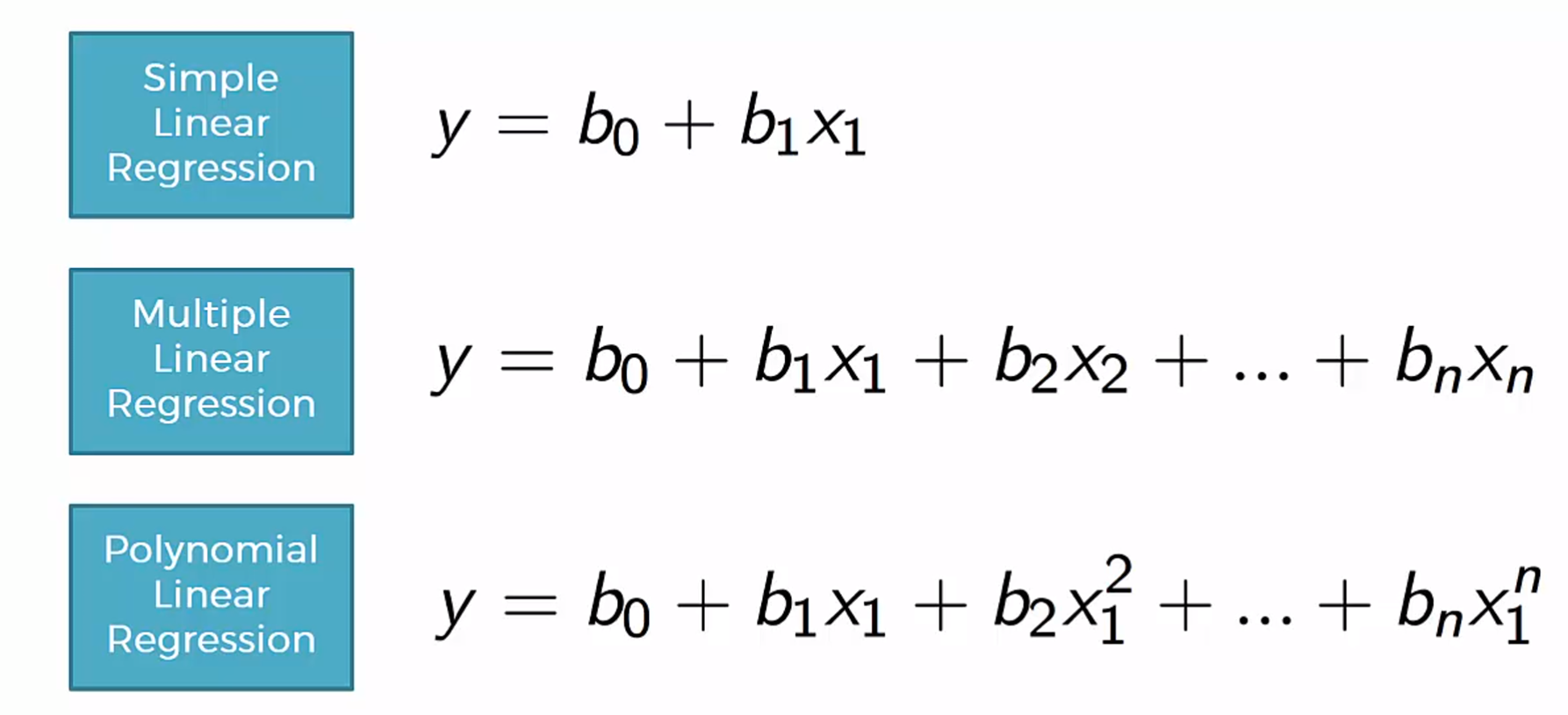
1. **Building a model**
2. All-in : use all variables
3. Backward elimination : Including all variables and afterwards eliminating the independnent variables that are not statistically significant
4. **Simple linear regression**

Least square – SUM(actual\_value – predicted value )^2

1. **Multiple linear regression**

Dummy variable trap = (for categorical variables), the model can’t distinguish between dummyVariable01 and dummyVariable02 – multicoliniarity

Always should omit one dummy variable



1. **Polynomial Linear Regression**

from sklearn.preprocessing import PolynomialFeatures

*#Transforming the original matrix of features X into a polynomial matrix of features x x2*

polyReg = PolynomialFeatures(degree = 4)

X\_poly = polyReg.fit\_transform(X)

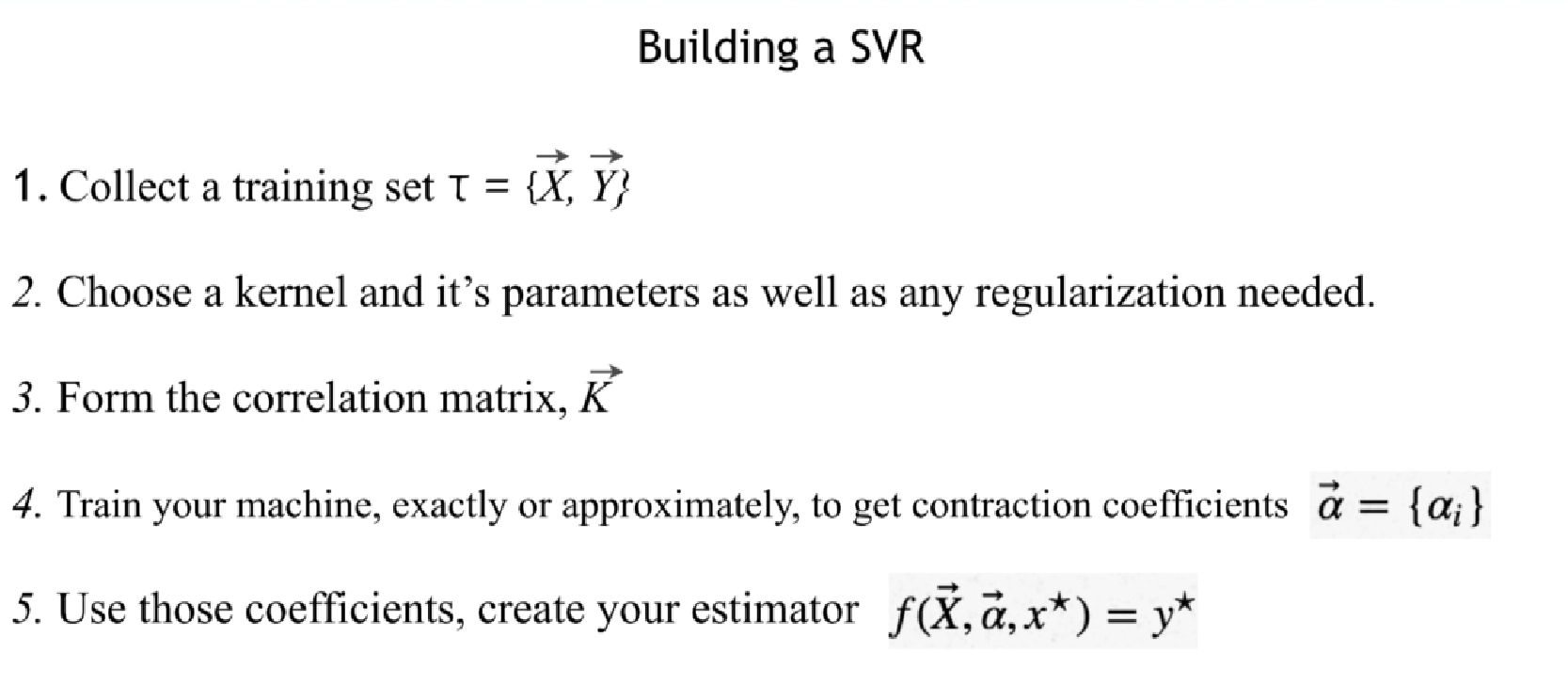
*#Create a new liniar regressiong object that was fitted with the polynomial matrix of features*

lin\_reg\_2 = LinearRegression()

lin\_reg\_2.fit(X\_poly, y)

1. **Support Vector Regression**

SVR = type pf SVM (linear and non-liniar) width is controlled by epsilon



1. **Decision Tree Regression**

CART = Classification Trees

Regression Trees

System entropy = is the split increasing the amount of information

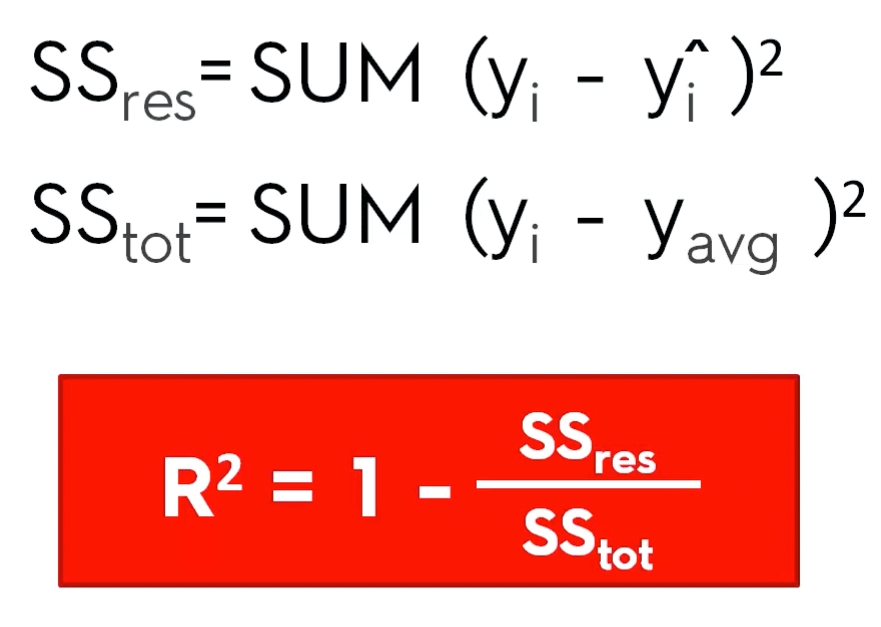
*from sklearn.tree import DecisionTreeRegressor*

*regressor = DecisionTreeRegressor(random\_state = 0)*

*regressor.fit(X, y)*

1. **Random Forest Regression**
2. **Evaluating Regression Models Performance**

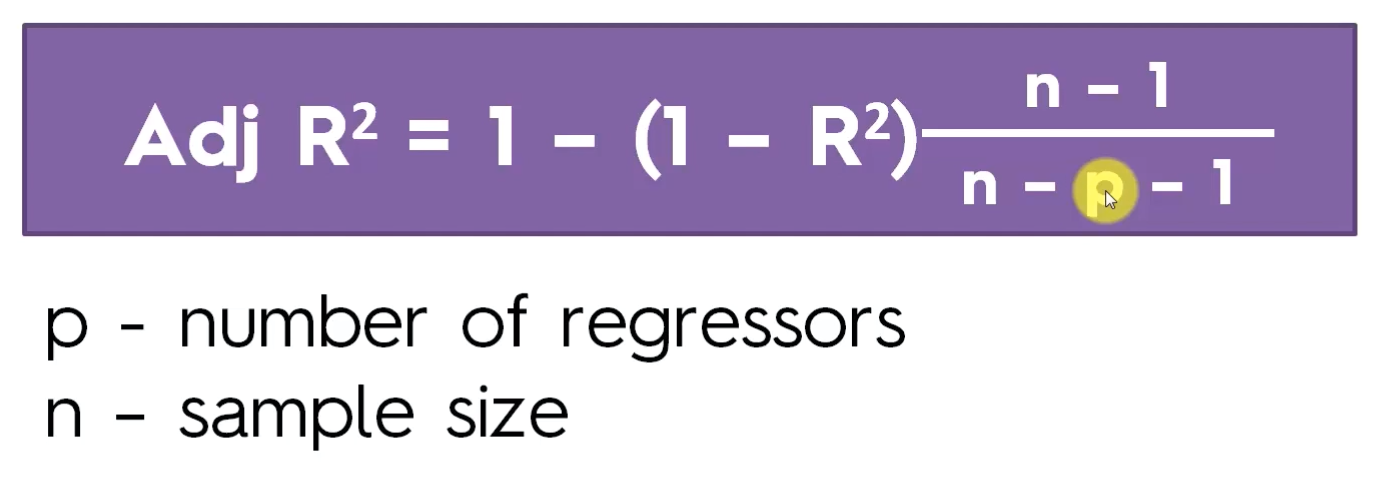
**R squared :**



Goodness of fit (closer to 1, the better) OLS

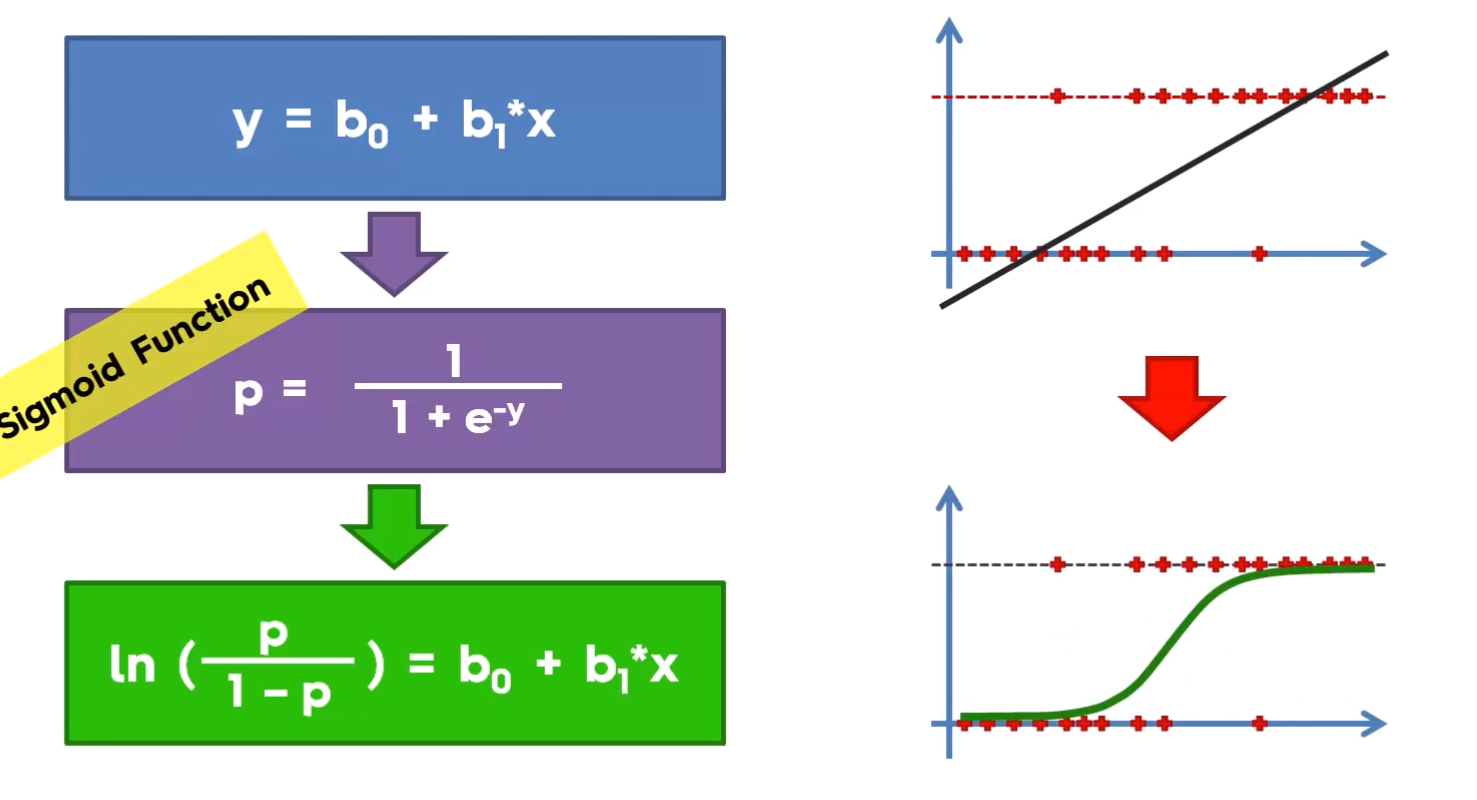
More variables added, r squared will grow

**Adjusted R squared :**



**Ensamble learning = take multiple algorithms**

1. **Logistic Regression**



# Appendix 01

* p-values = p-value is actually the probability of getting a sample like ours, or more extreme than ours IF the null hypothesis is true

how likely is to get a similar result if the null hypothesis is true

null hypothesis = the proposition we are trying to find evidence against (H0) -> we are trying to prove H1

We take a significance level of 0.05. If p value < significance level, we will reject H0

The lower the p value, the most significant the variable would be

* kernel methods are a class of algorithms for [pattern analysis](https://en.wikipedia.org/wiki/Pattern_analysis), whose best known member is the [support vector machine](https://en.wikipedia.org/wiki/Support_vector_machine) (SVM). The general task of pattern analysis is to find and study general types of relations (for example [clusters](https://en.wikipedia.org/wiki/Cluster_analysis), [rankings](https://en.wikipedia.org/wiki/Ranking), [principal components](https://en.wikipedia.org/wiki/Principal_components), [correlations](https://en.wikipedia.org/wiki/Correlation), [classifications](https://en.wikipedia.org/wiki/Statistical_classification)) in datasets
* regularization : major aspects of training your machine learning model is avoiding overfitting. The model will have a low accuracy if it is overfitting. This happens because your model is trying too hard to capture the noise in your training dataset. By noise we mean the data points that don’t really represent the true properties of your data, but random chance.

We are trying to minimize/shrink the coefficients estimates to zero.

We are using a loss function or RSS (residual sum of squares) :

