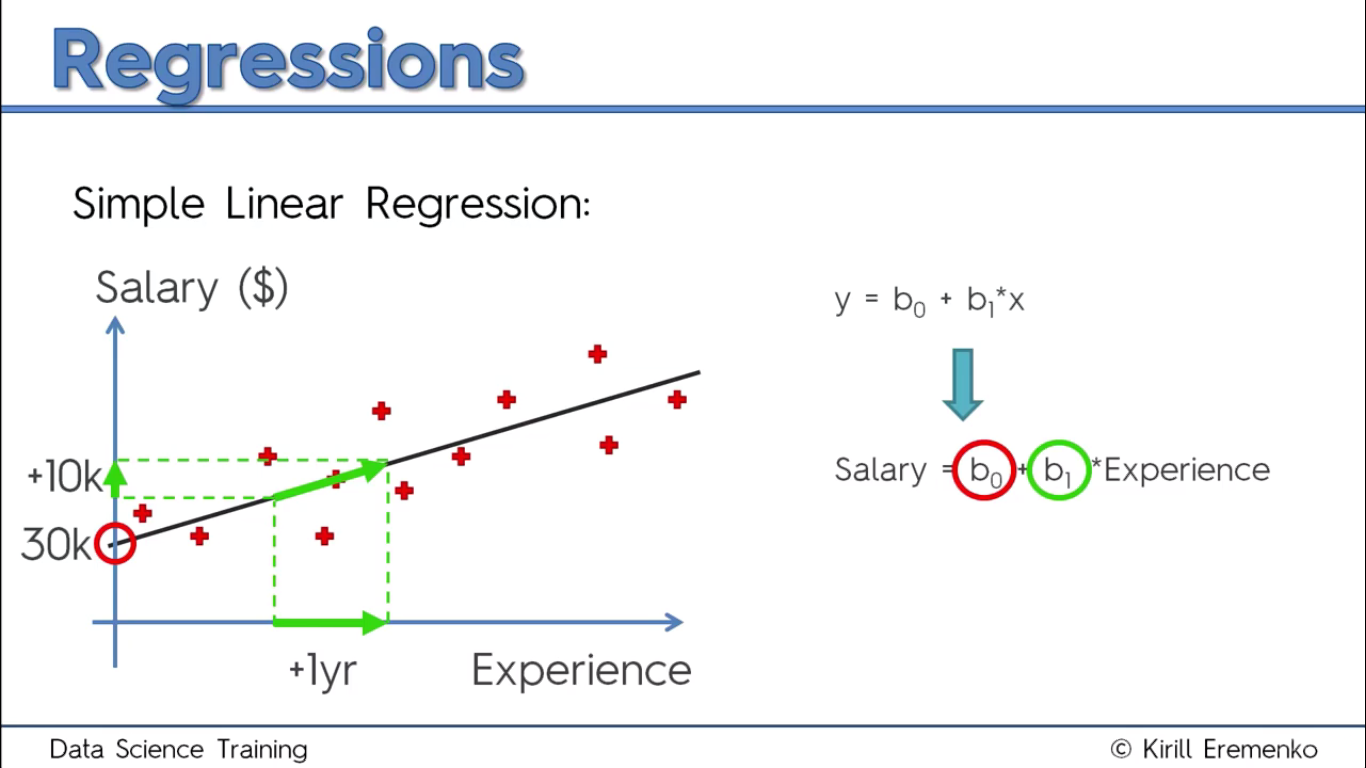
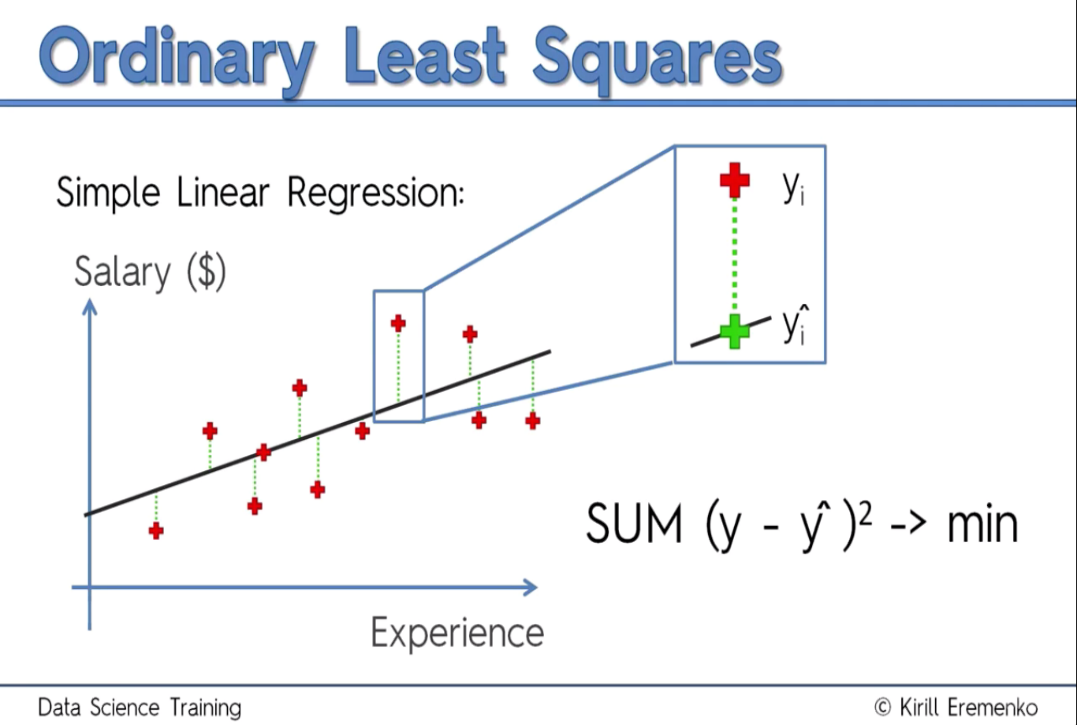
Simple Linear Regression

y is the dependant variable.(Something you are trying to understand how it depends on other things.) is the independent variable which is only one. is a coefficient, can be for unit change, its like the proportional changer of y. is a constant



In the above graph, from the real world values, a line of best fit has been created. This line is represented by the equation. The steeper the slope(), the higher the salary per xp increase. Its essentially like y = mx+c.

Ordinary Least Squares Method

What happens is that the model tries a random line, draws green lines, finds the difference, squares that difference, then finds the sum of those squares, then stores this sum. It repeats this for many lines and the one with the min sum is the line of best fit.

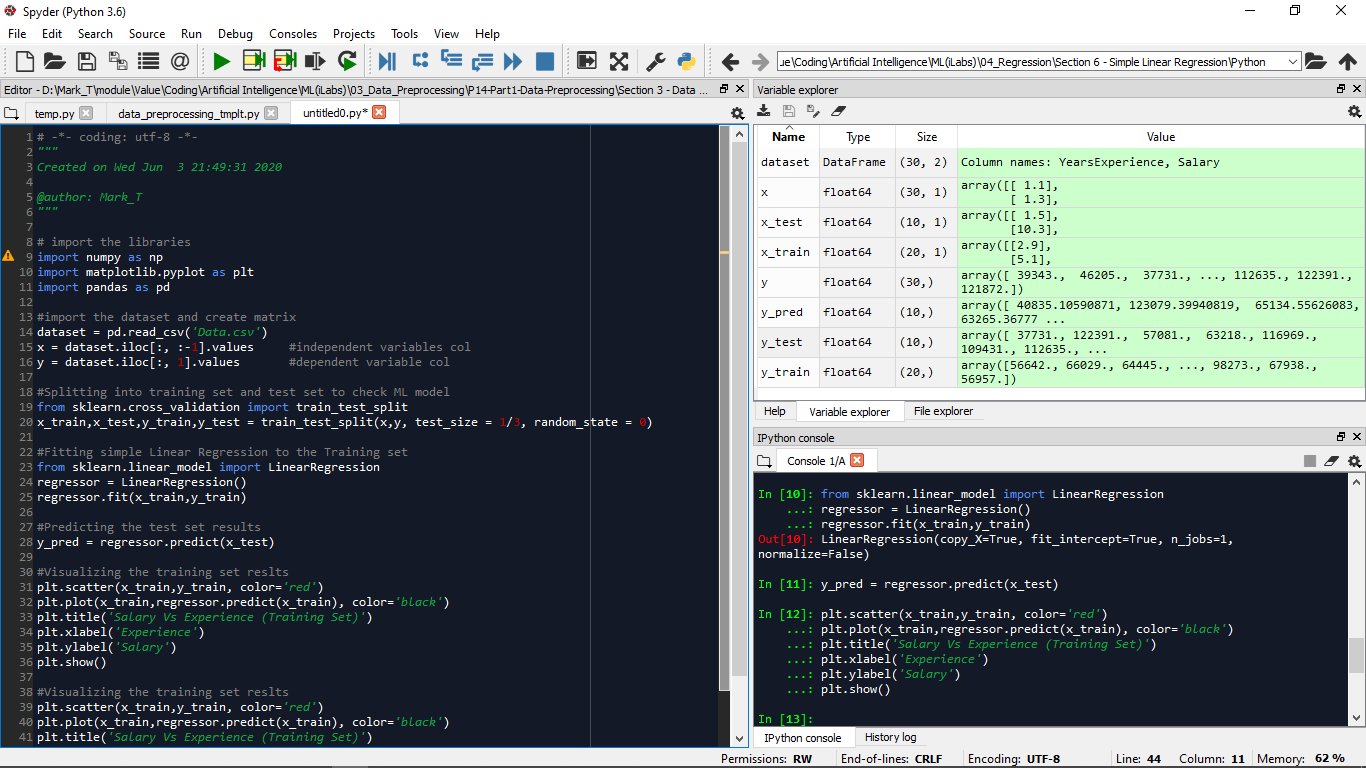
Simple Linear Regression in Python.

Step 1: Editing the Data Preprocessing template

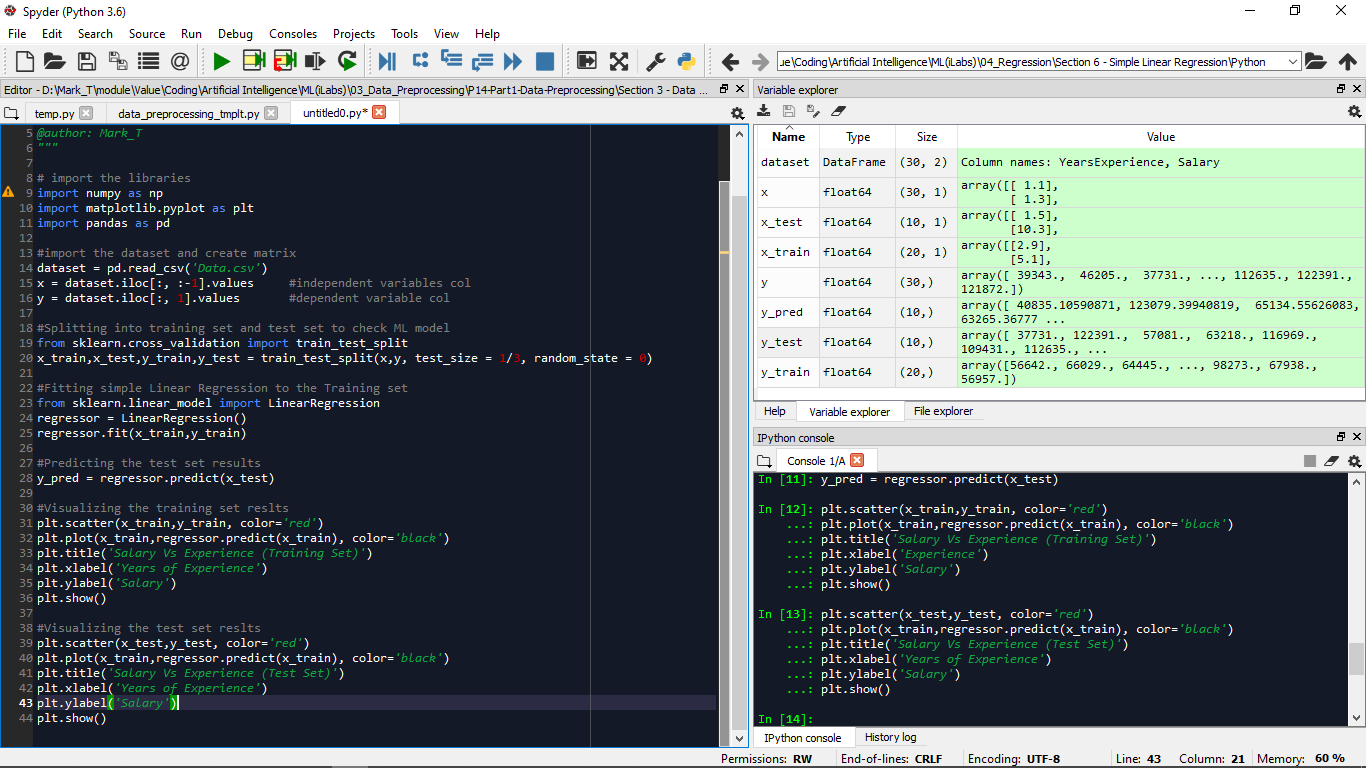
Usually we want to split 80% into a training set and 20% into a test set. We shall compare the test set values with the model’s predictions. Note that y is a vector while x is a matrix.

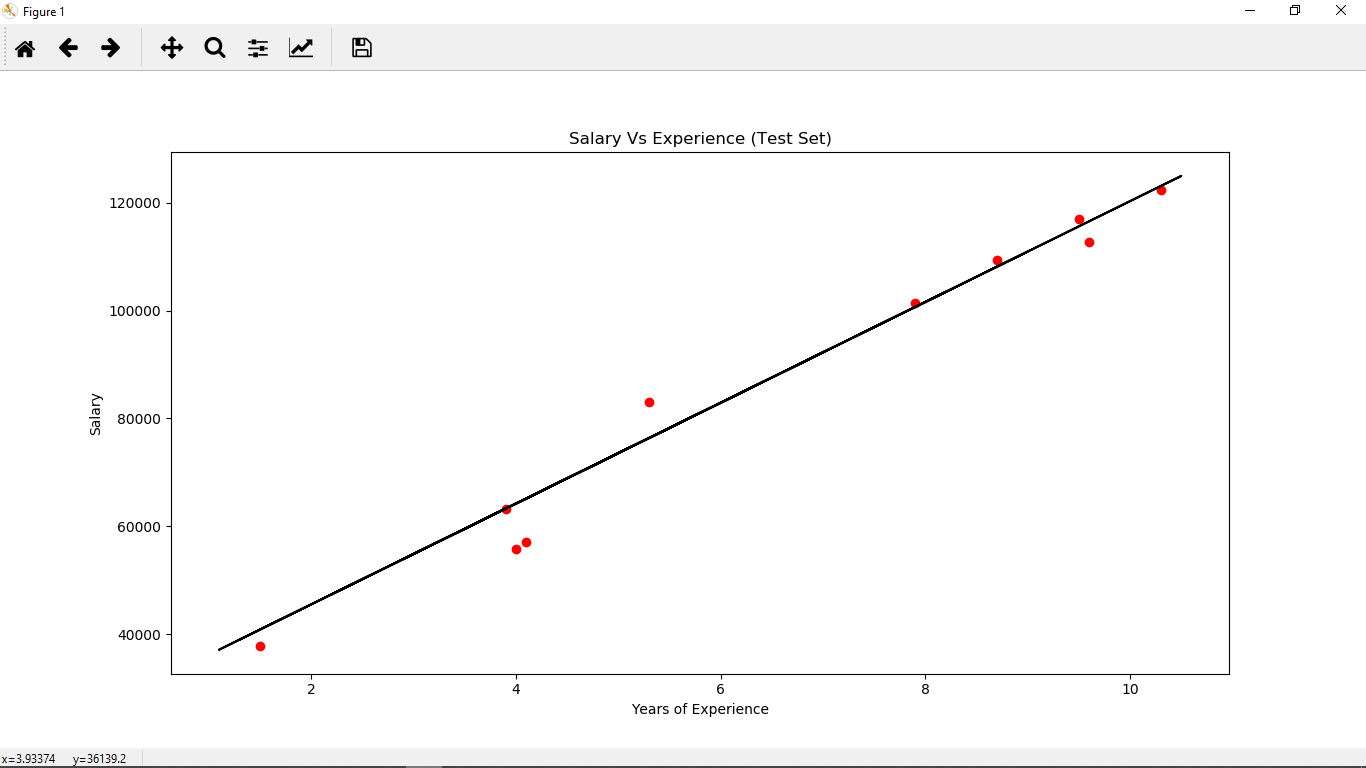
Step 2: Fitting the Simple Linear Regresion into the training set

We create an object of the type LinearRegression from the sklearn.linear\_model. Through the fit method, the model learns the correlations between x\_train and y\_train

Step 3: Predicting from the test set using LinearRegression.predict

Visualizing





The line of best fit plotted is not affected whether we now use the test set or training set(via plt.plot()) coz its formula has been developed in the LinearRegression object.

Qns:

What are the effects of increasing the training set on the accuracy of the line of best fit

Real life Applications of Simple Linear Regression

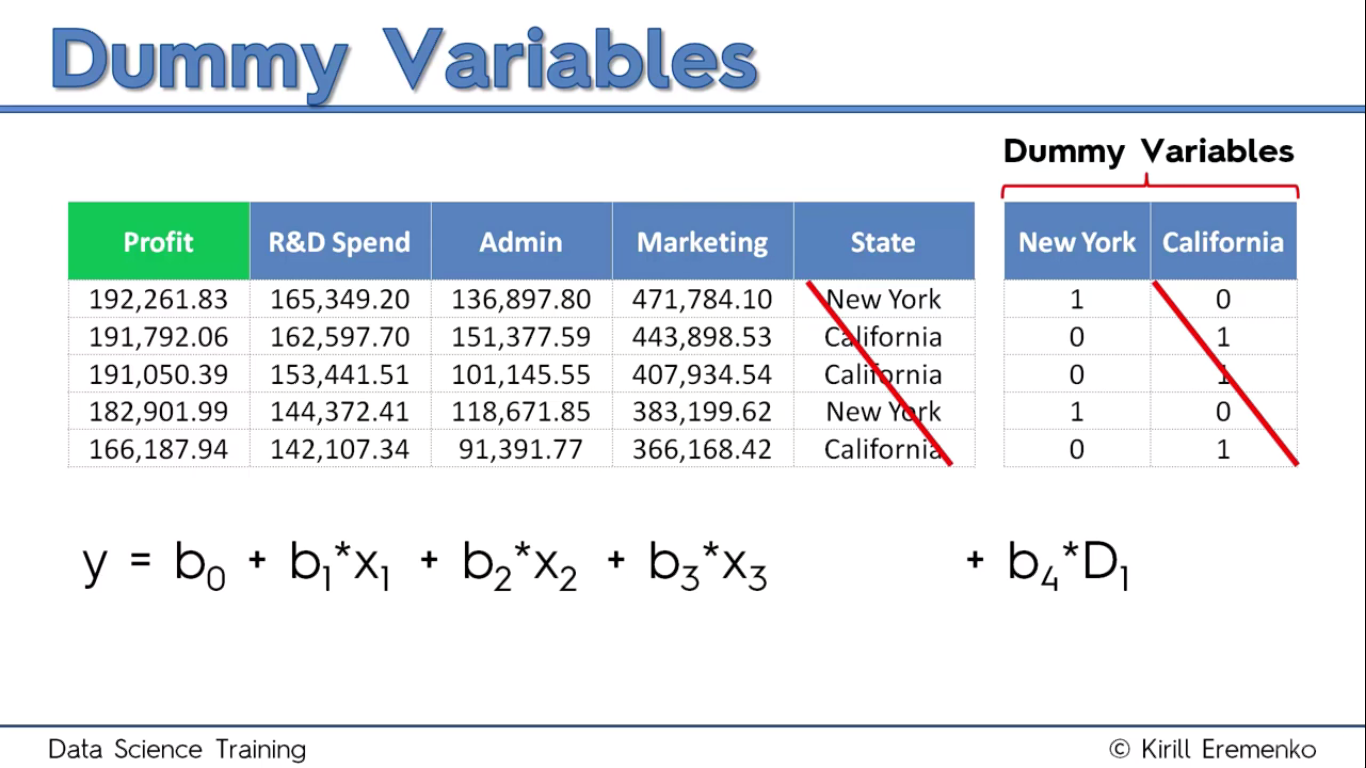
Multiple Linear Regression

Same as Simple, but more variables

Problem: A Venture Capitalist firm seeks to know if there are any correlations between the expenditures in R&D, Admin, Marketing; the state in which the start-up is operating, and the Profit of the startup.

Soln:

We can use Linear Regression



This ignoring of California coefficient seems biased towards NewYork but actually the algorithm considers by default as the values of Carlifonia. So the equation alternates; = 1 for Calif and 0 for NewYork but when the side is not 0.

State variables in the pic are categorical variables and we can use Dummy variables(as shown.) but w e choose 1 not both. This is coz you r basically repeating the dummy variable. The phenomenon in which one or several independent variables predict another is called ***Multiple Linearity***. So the model doesnot fail to distinguish effects of D1 from those of D2. This is the Dummy Variable Trap. So Always emit one dummy variable. Even when the sets of categorical variables increase, still for each, emit one.

Building A Model

Initially, we had simply one independent variable. When many independent variables come in we sometimes choose some to leave out. Some reasons include:

Garbage In Garbage Out: Throwing a lot of stuff at uo model, expect it to be affected.

Understandability: The model should be explainable by you.

Resource Consuming.

5 Methods of Building A Model:

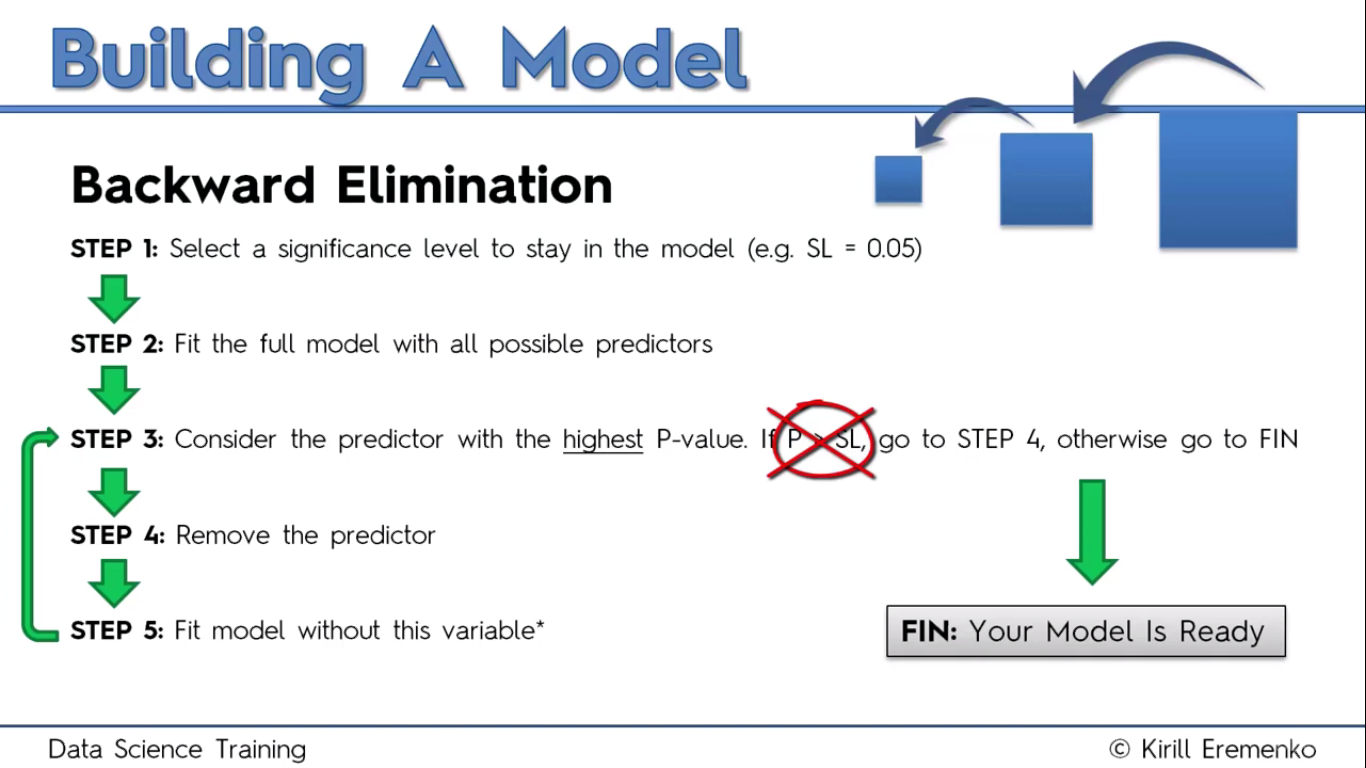
All - in

Stepwise Regression

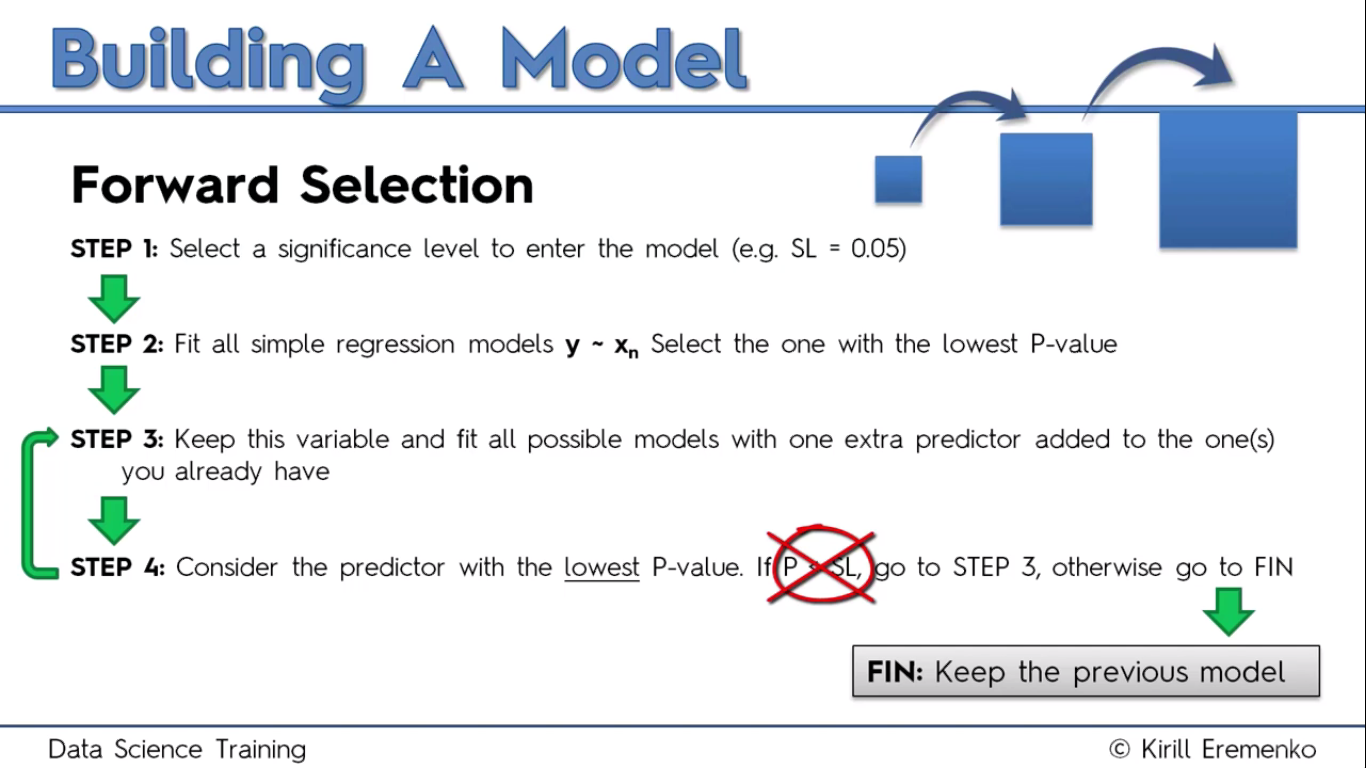
Score Comparison

All – in: Used when you’re sure that they are required. When you have to, or when prep for Backward Elimination.

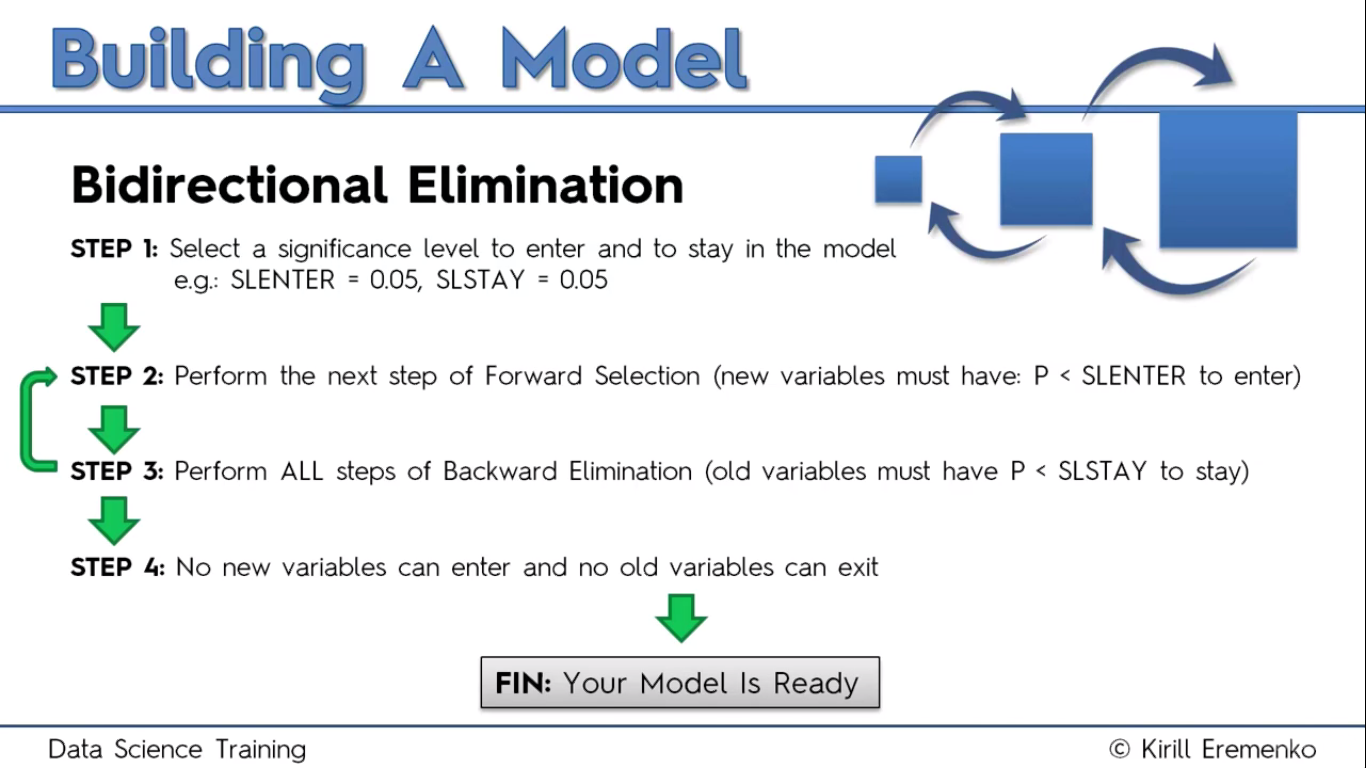
BackWard Elimination



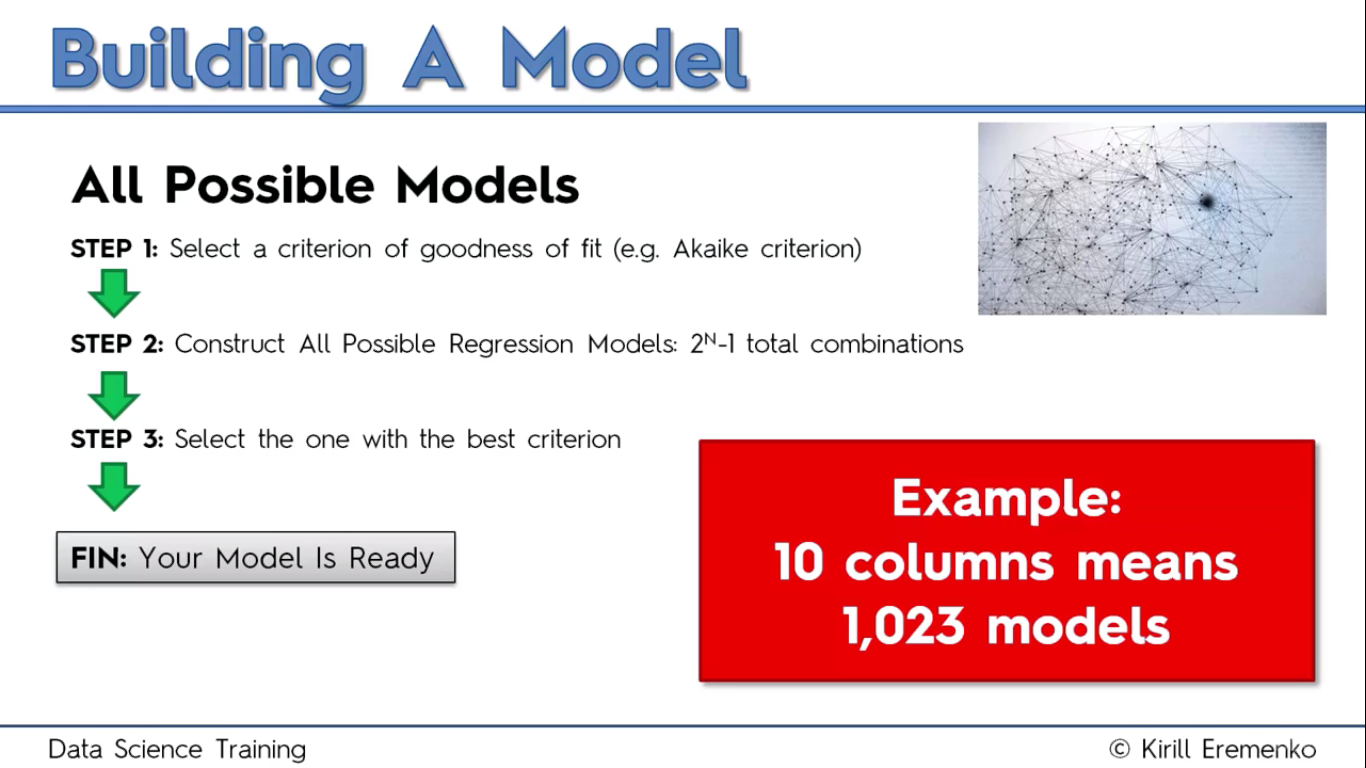
Forward Selection



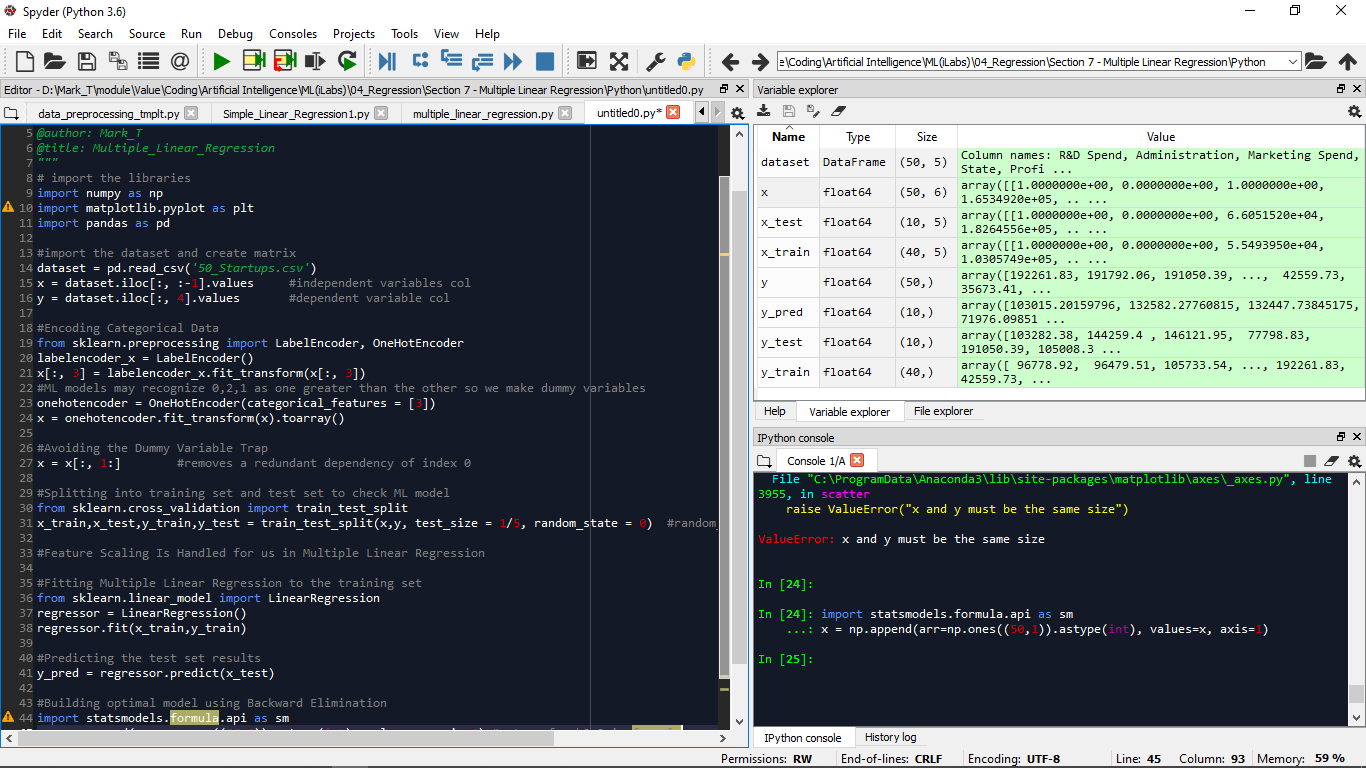
Bidirectional Elimination



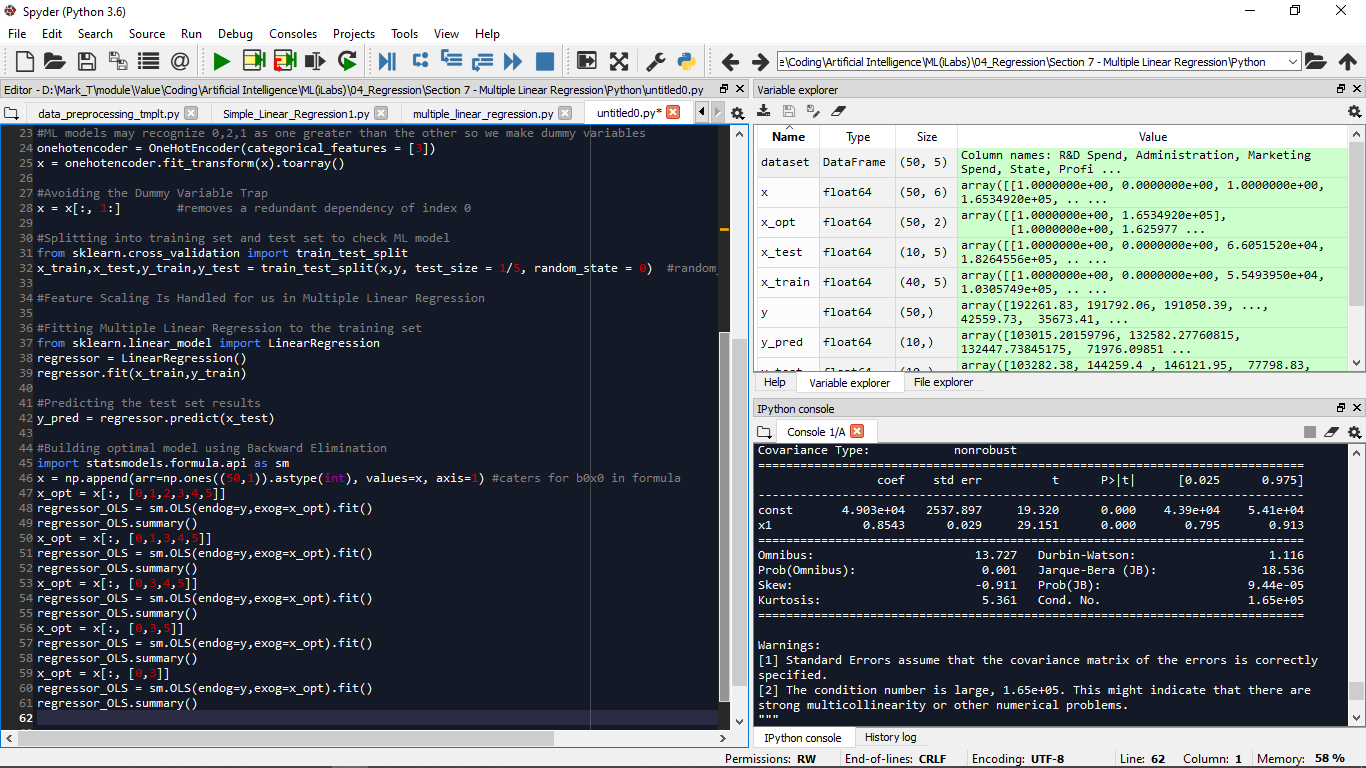
All Possible Models



Multiple Linear Regression in Python



Building an optimal model with Backward Elimination

Mosttimes we need to have an IV matrix that contains variables that are most statistically significant for our DV - profit. (Seems these algorithms use matrices math.) Hence we do Backward Elimination.

Zoom to 200% to see very well

So to do this, we begin by considering all possible predictors. We have initialised x\_opt coz its gonna keep changing but the changes should not affect IV matrix x. Then we create an object of the sm.OLS class(Odinary Least Squares) initialized with the DV matrix – y, and the gonna-be changing IV matrix – x\_opt. Note after tapping Ctrl+I that an intercept is not added by default and shd be added by the user.We then fit the OLS algorithm which is Multiple Linear Regression itself, to x\_opt and y.

Next to find the predictor with the highest p-value, we use the summary function that returns a table of statistical metrics that can be used to make the model more robust, like R-squared, Adj. R-squared…. In the table, A p-value is a probability and the lower the probability, the more significant its IV is w.r.t the DV.

If the p-value is greater than the chosen Significance Level(SL) – 0.05, we remove the predictor. So from the table, we remove x2. Note that we cant have a 0 p-value. So in the end, R&D is the most significant in determining profit

Qns

How to visualize the regressor.predict()

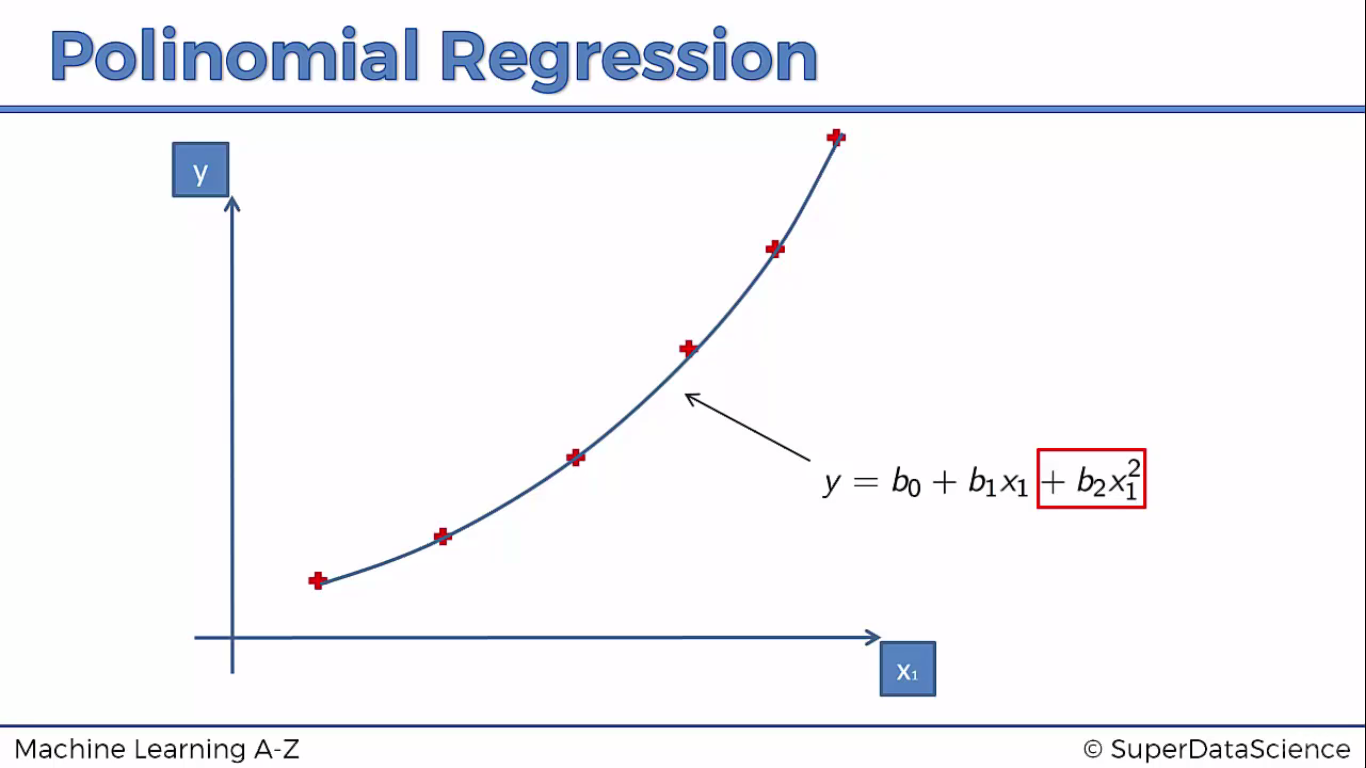
What are p-values and how do they come up?

How to do the process of backward elimination with a loop.

Polynomial Linear Regression

(n = 0,1,2,3, …)

Sometimes the dataset takes the form below and clearly this regression handles the data better.

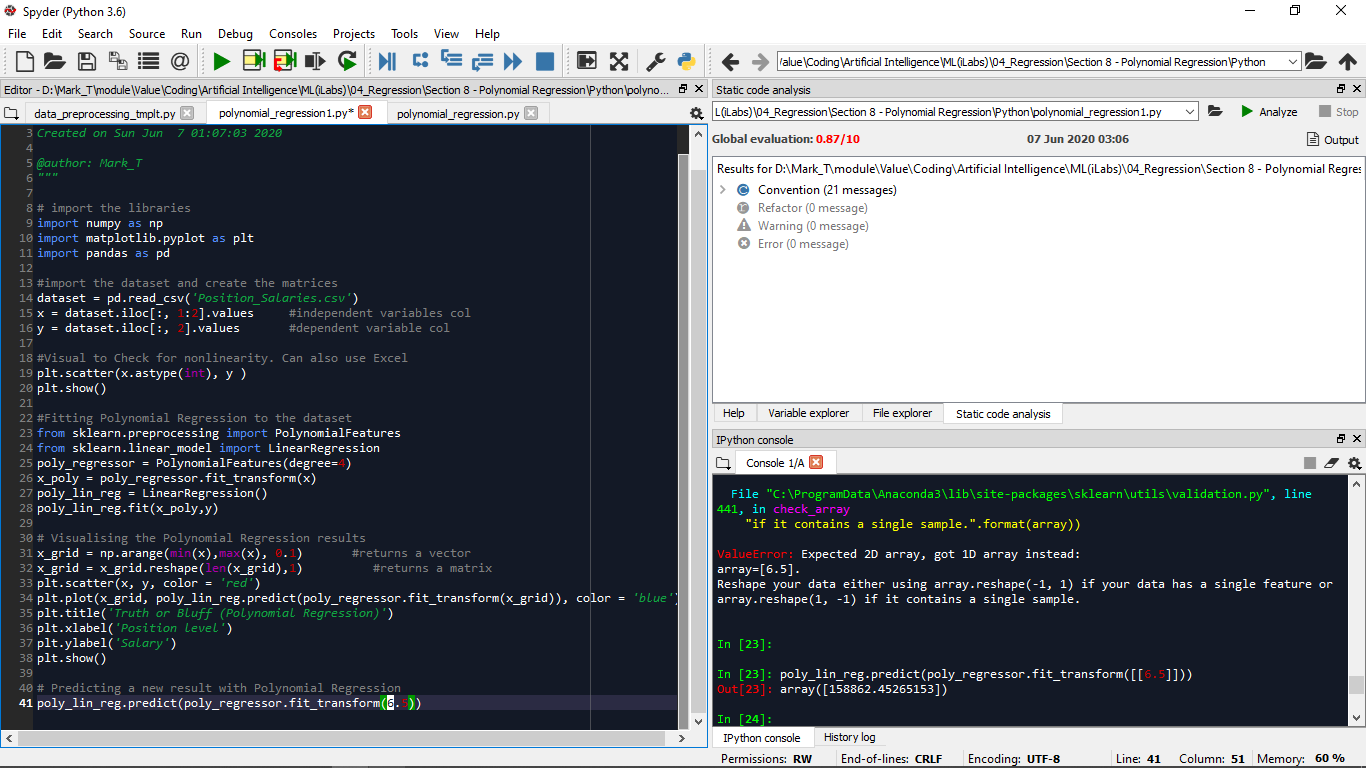


This regression is called linear because of the coefficients’ nature not x.

Polynomial Regression in Python

Problem: We are a HR team in a big company, about to hire a new employee who says he used to earn 160k previously, so expects to earn not less than that during negotiations. Data acquired from the previous company needs to be used to build a bluff detector.

Soln



In the data preprocessing template, we need to ensure that x is a matrix not a vector hence the list notation shown. We have only ten observations which are few so its best not to split coz the accuracy of prediction is dependent on the training basically. Feature scaling is not applied here coz we are gonna use a library that caters for this already.

To do the Polynomial Regression, We create an object of the PolynomialFeatures class, initialized with the value of n(degree) in the equation. We then fit matrix x to the object and transform it to the polynomial values up to n.(check the x\_poly and realize that the object automatically adds a column of 1s ). This x\_poly and y r fitted to a LinearRegression Object ( I think to apply MLR)

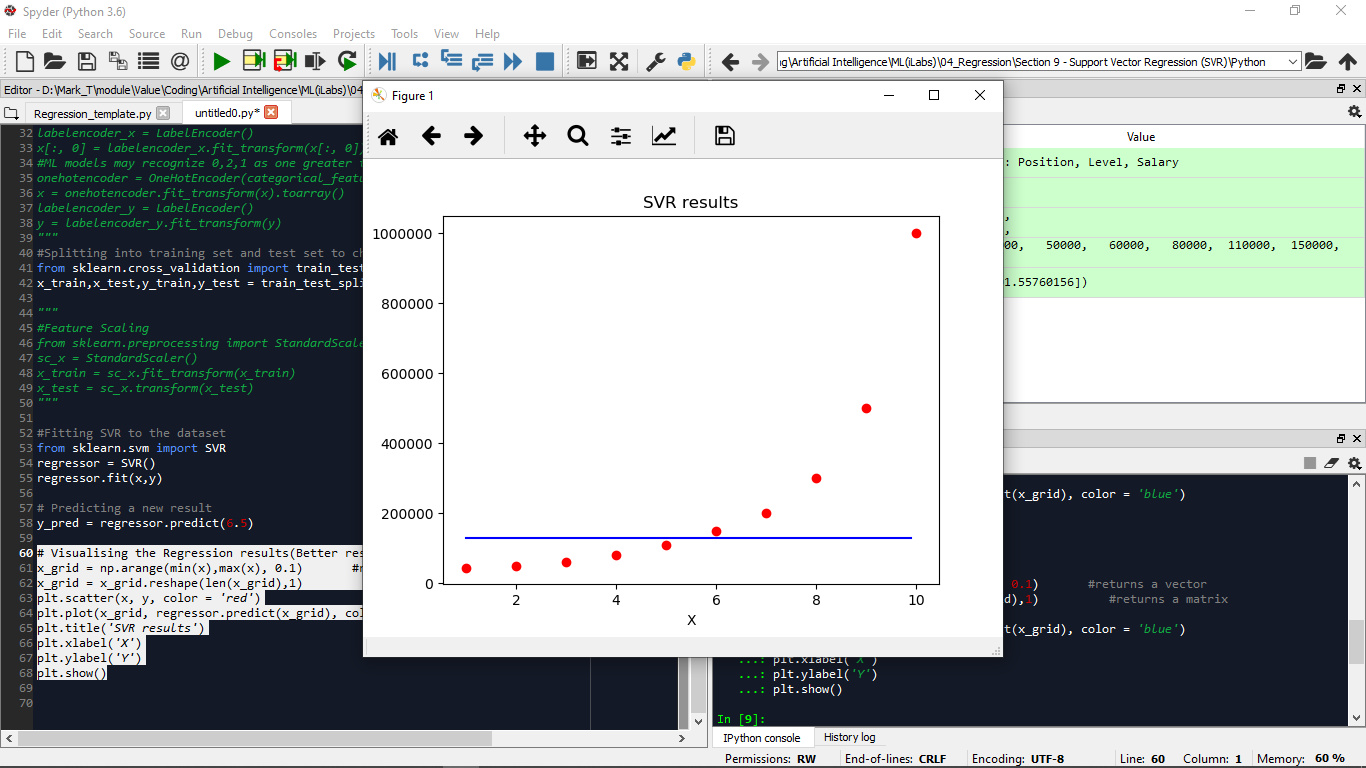
Support Vector Regression

More Research needs to be made. Made some in phone

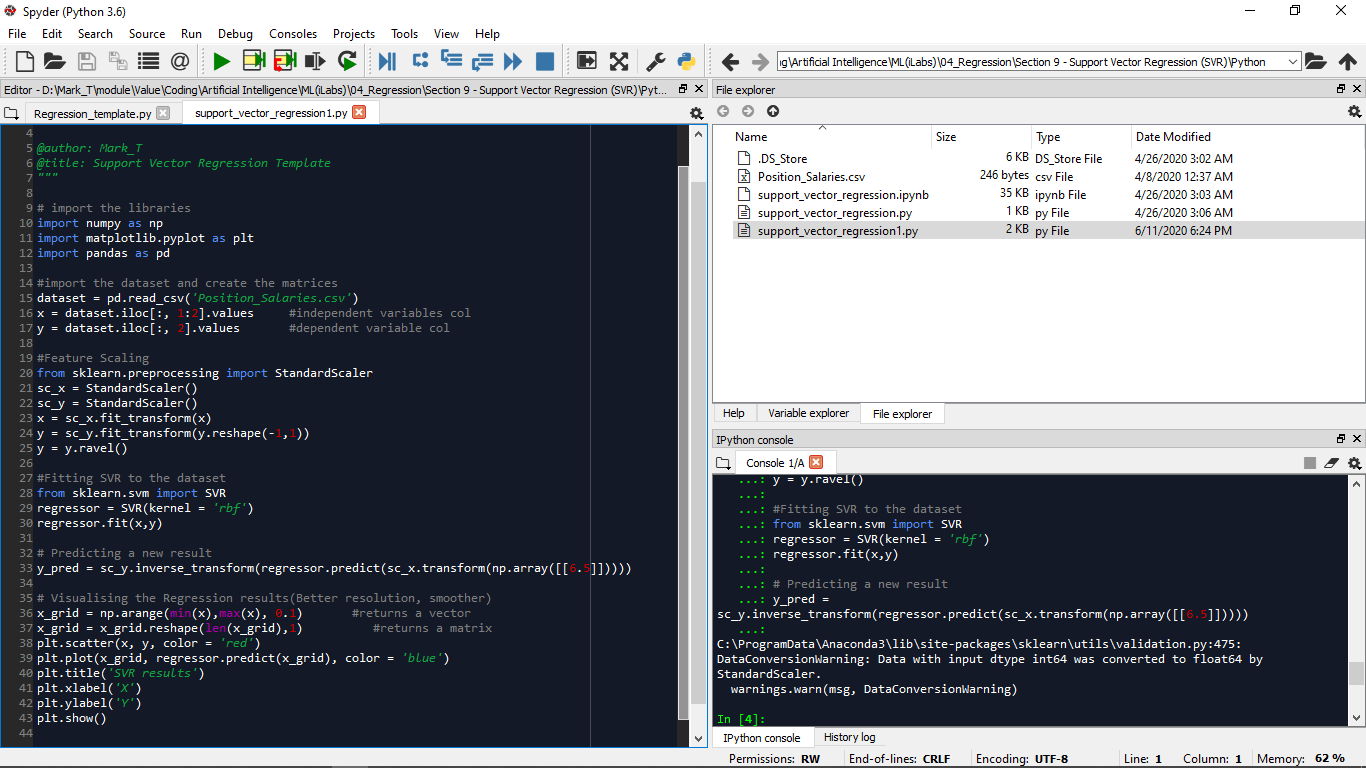
SVR in Python

An SVR object is created and can be initialized with stuff. For this, we are interested in kernel parameters. You can use rbf (Gaussian), sigmoid, polynomial, linear, depending on the training data. Our problem is non-linear so we can use poly or rbf but rbf is mor common so…

Without feature scaling, the results are misguiding and the predicted value can be wrong. This is seen in the graph below.

So we apply feature scaling.

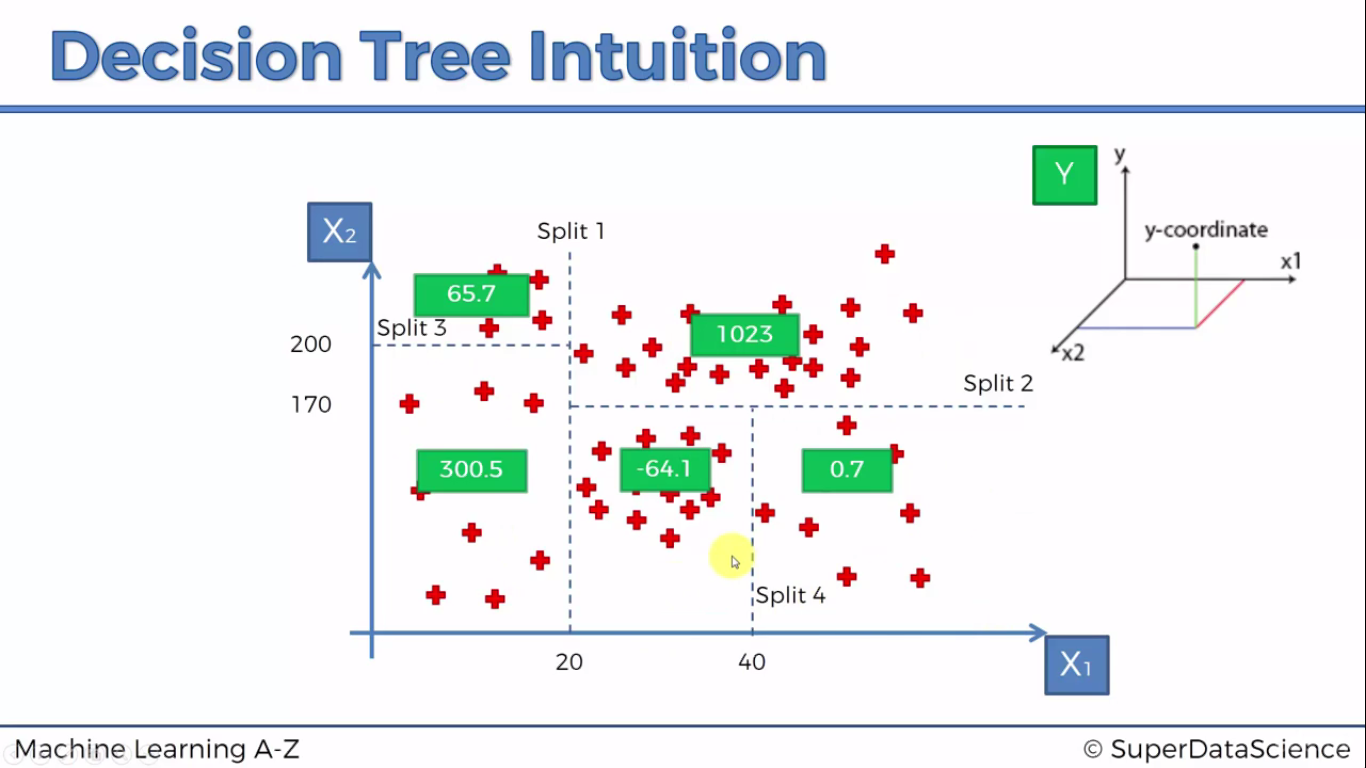
Using functions, its good to check for type issues in the expected parameters by checking the function definition. Via np.array(), using [ ], creates a vector, [ [ ] ] creates a matrix.



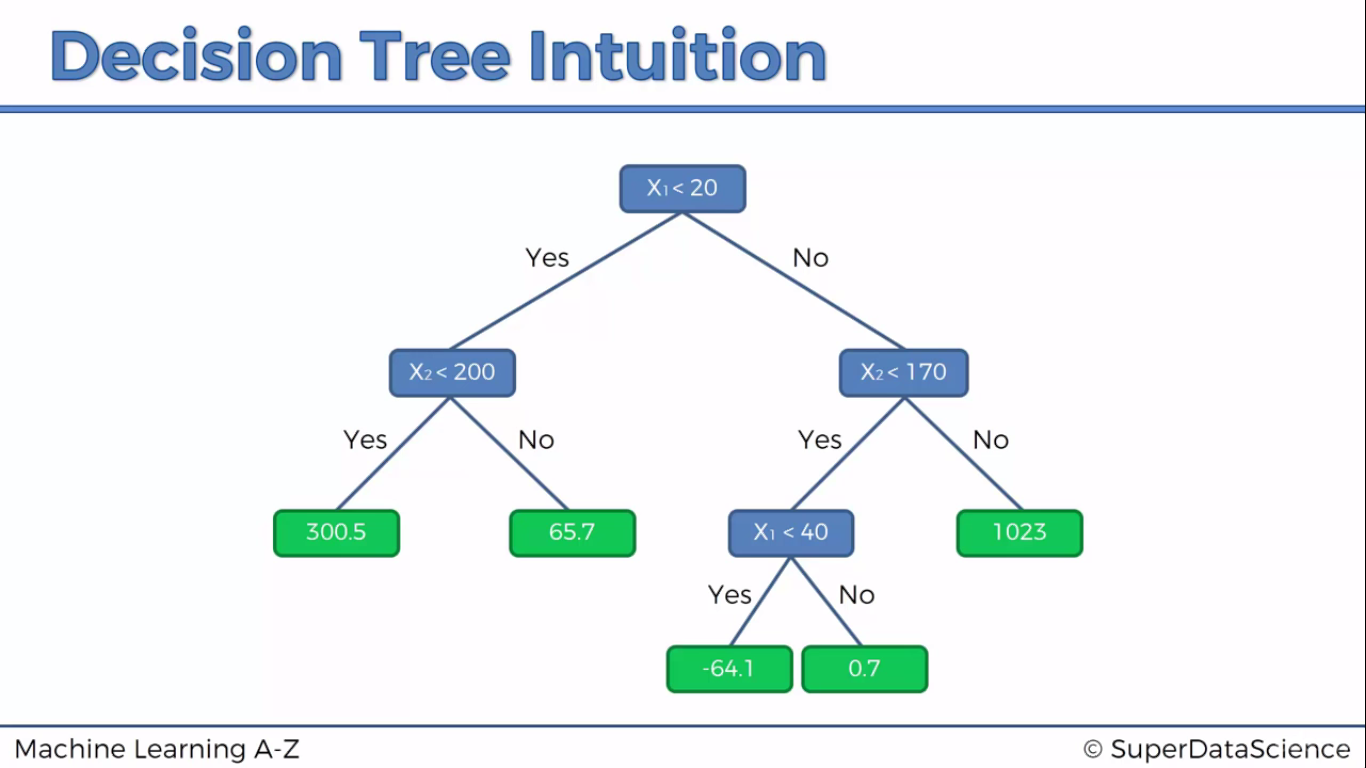
Classification and Regression Trees(CART)

Regression Trees

Consider the data scatter plot below:



The X1 and X2 are two IVs and we shall come up with Y later, as a third dimension. When applying a Regression tree in a regreession sense, the algorithm applies splits(forms leaves) as shown above. How these are done is related to Information Entropy. A mathematical concept . Basically, when a split (eg split 4) is performed, is it increasing the amount of information we have about our points? The algorithm knows when to stop is when is there a certain minimum for which info is to be added.

We understand the Decision tree now.

Beginning with X1 blue box, wn split 1 is made, it becomes X1<20 with Yes or No. For Yes, split 3 is made and X2<200 is put. For No, X2<170 is put. For split 4, X1< 20 No, X2<170 yes are considered; to form x1<40.

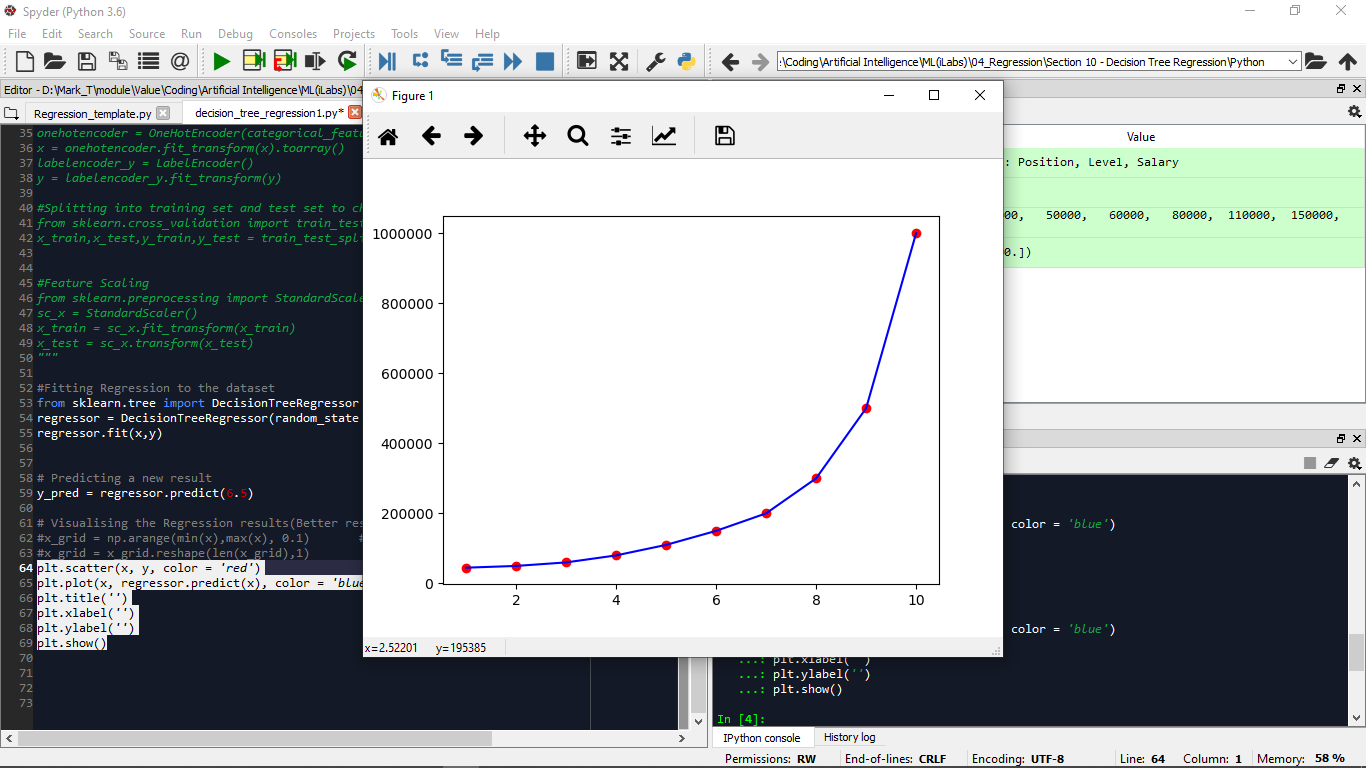
The point of this is, remember we want to predict a y value for any new added scatter plot. So what happens is that the average for each terminal leaf is taken and so for any new point eg (30,50) we expect a y within -64.1

The more the values, the more accurate the mean value.

Decision Tree Regression Template(Python)

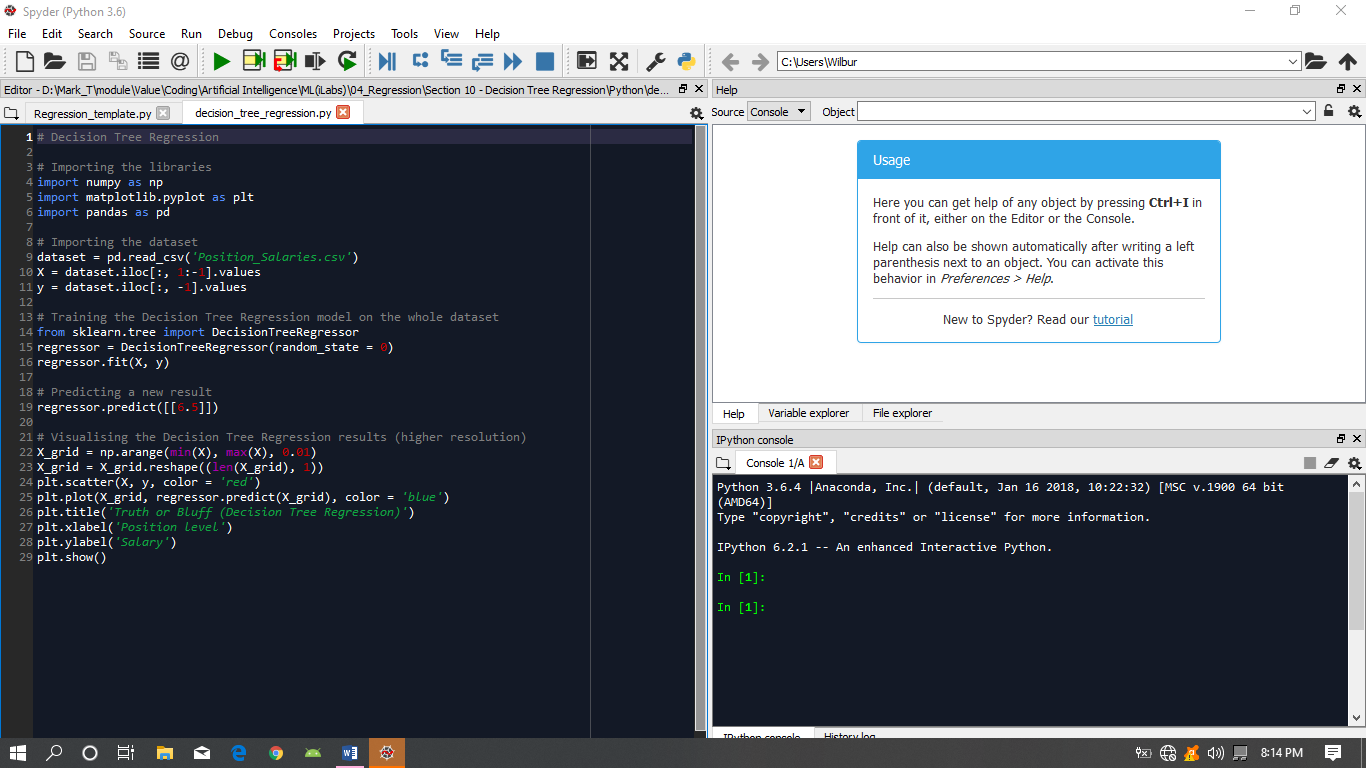
An Object of the DecisionTreeRegressor class is instantiated with initial value (random state = 0.) Some Object parameters include criterion. Its default value is mse(mean square error) The difference btwn square of predicted value and actual result, and we take the sum of the differences to measure the error.

From the graph below, remembering that we are using DTR, this raises a red flag.

This is bcoz the algorithm forms terminal leaves along the x axis(in this case), and we expect that atleast in each leaf, the point is the mean value and the blue line should intercept each point but showing constant lines not changing ones.

What has happened is that the algorithm formed 9 leaves and in each, there were no values inside to find a mean, so it just connects the points.

DTR is a non continuous model. What we can employ is the higher resolution code to add points



Random Forest Regression (Basic theory)

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Random Forest is a version of Ensemble Learning where one uses multiple algorithms or the same algorithm multiple times and you put them together to come up with something much more powerful than the original. So lets see how this works.



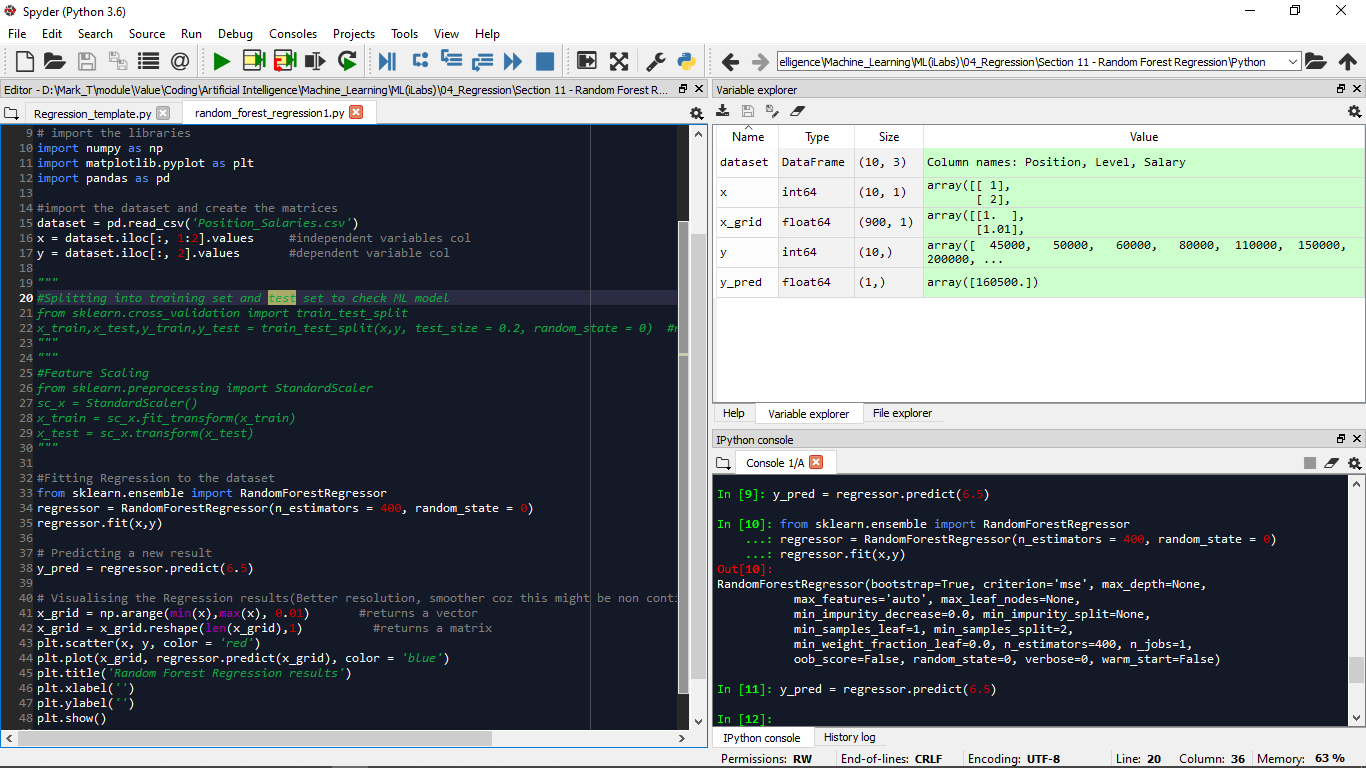
This can give you a more accurate result than for one regression tree because:

The data for one tree might cause your algorithm to predict a wrong value than when you have like 500 trees where the average can weaken the error.

Also the algorithm is more stable coz a few mistakes or changes in one tree can affect results compared to those of a forest.

I also believe with a forest, consistencies can be found with trees than with a tree hence can be applicable in for example classification.

Code Execution:



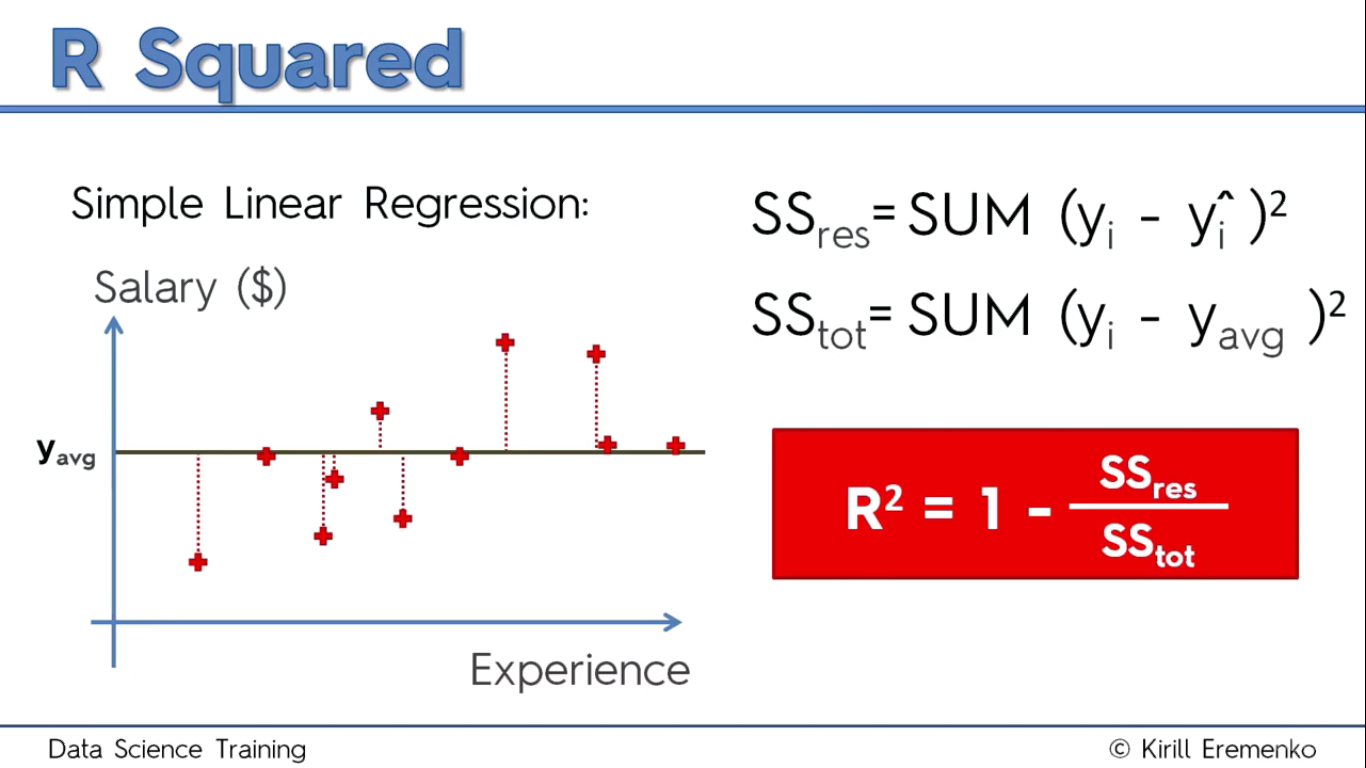
When we try to visualize the code segment, we observe:



There are extra steps btwn intervals than compared to one regression tree. This is because, we have more regression trees. Therefore each split is one interval. Each line is an average from ten regression tree predictions. We also have a lot more steps coz the forest is calculating many different averages for the intervals. Increasing the number of trees doesn’t mean the stps increase. Because more trees contribute to accuracy, convergence, the shape just changes.(Whatif we use too many trees?)

Evaluating Regression Models Performance

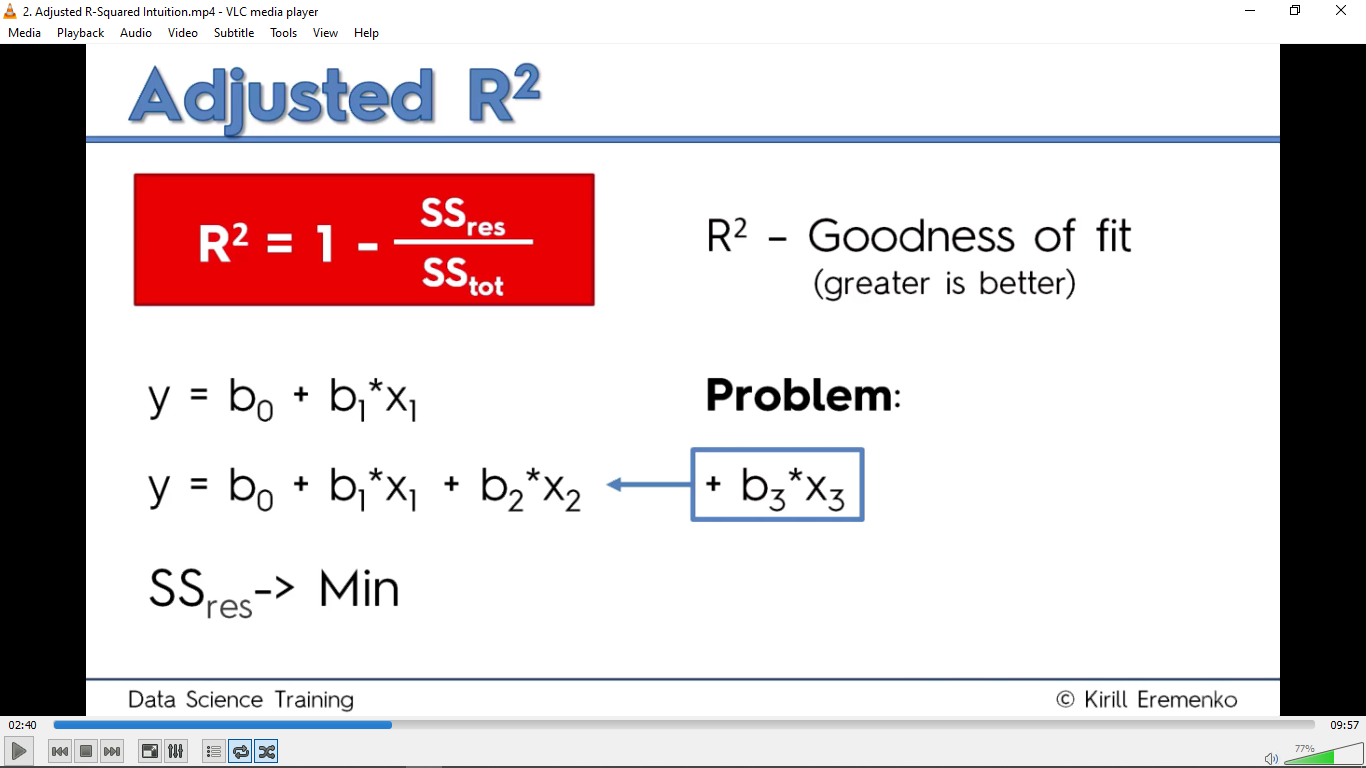
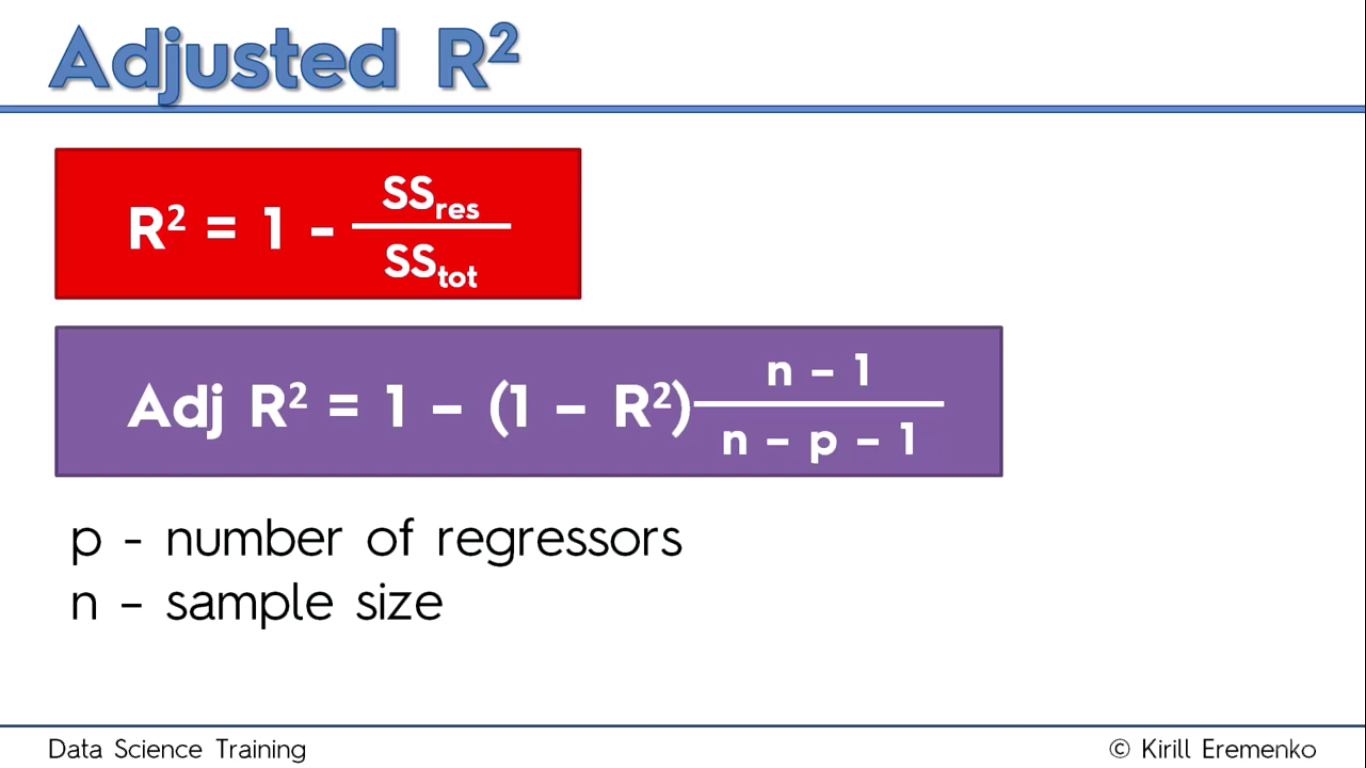
R-Squared

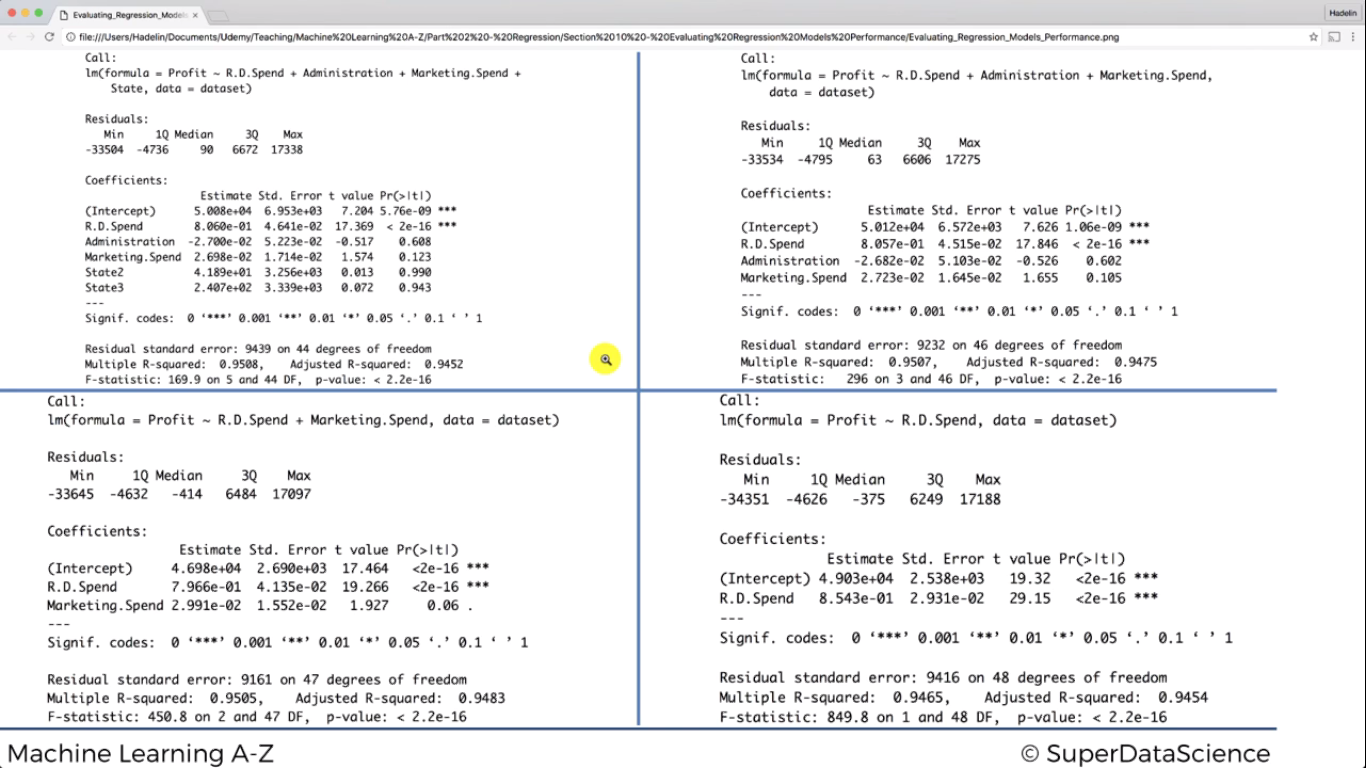


With Regression, What u want is to minimize the sum of residual squares (SSres). Ideally when SSres = 0(trend line goes thru all data points), R\*\*2 = 1. So from the formula above, the closer R\*\*2 is to 1 the better. When negative, means the model could be broken. It tells us how well your model is fitted to the data

Adjusted R-Squared.

Basically, it helps you know whether you are adding significant IVs to your model or not. With R\*\*2, a problem comes in when more variables are added to the model.

Usually we add variables to help reduce SSres and also regression algorithm tries to reduce SSres. So either X3 will help minimize it or the model will some how find a way and the correlation coefficient is very small. Hence R\*\*2 will never decrease. This is bad because we cannot tell whether the IVs added are helping improve the model. That’s where Adj. R\*\*2 comes in.

Adjusted R\*\*2 has a penalization factor. It penalizes u for adding more regressors(IVs). When the p increases, Adj. R\*\*2 becomes smaller. At the same time, when normal R\*\*2 is increasing, Adj.R\*\*2 increases. So u have a ‘fair’ battle here. If the added regressor is not helping the model alot, the small change in R\*\*2 is outweighed by the change in p. So we see a less Adj. R\*\*2. That’s how it helps build a more robust model.

From this realize that despite following the chosen signficancy level, the Adj.R\*\*2 is better on the third quadrant than the fourth.

Interpreting Coefficients:

When we choose the third quadrant,

Estimate is the value of b1, b2. When the sign is positive, the DV inc with the IV. If negstive, then it dec with inc in the IV.

For the Magnitude, first take note of whether the scales are the same before saying wc is bigger. In most cases, consider this value as per unit. For example, For each Unit(dollar) inc in R&D, profit increases by 0.79 per unit(79 cents).

The Estimate values change with the number of regressors coz each coefficient talks about the additional effect an IV brings into the model relative to other IVs already in play.