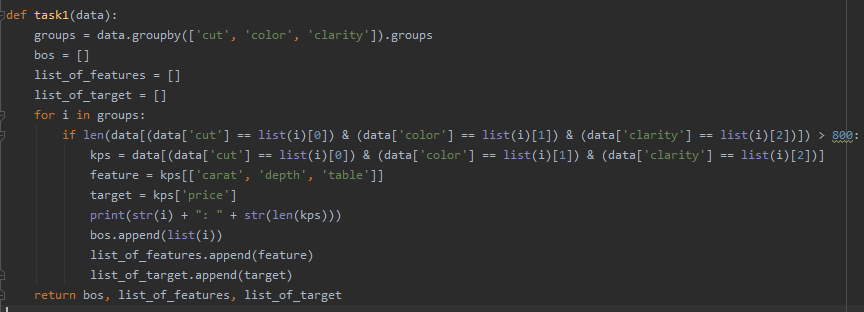
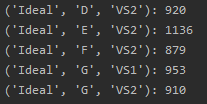
Design Document Assignment 3

**Task 1 – pre-processing**

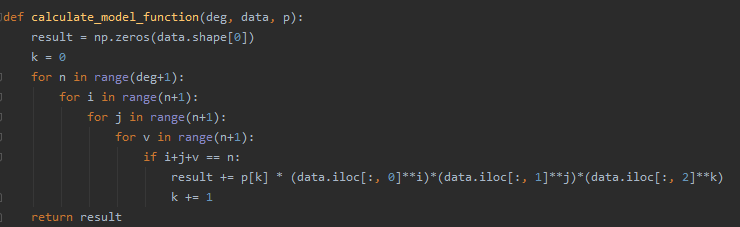


* First, the data is grouped by cut, colour and clarity.
* Then 3 lists are created
  + bos 🡪 contains the list of combinations extracted
  + list\_of\_features 🡪 contains the list of features extracted
  + list\_of\_target 🡪 contains the list of targets extracted
* The function will then iterate through the a list of combinations as declared in
  + groups = data.groupby([‘cut’, ’color’, ‘clarity’]).groups
* it will check if the length of the data combination is greater than 800
* if it is, then the data corresponding to those combinations is extracted (kps)
* the (kps) is split into the feature and targets
  + feature = kps[[‘carat’, ‘depth’, ‘table’]]
  + target = kps[‘price’]
* the 3 lists (bos, list\_of\_features, list\_of\_target) will be filled with the grade combinations, features and targets respectively
* The length of each of the datasets extracted is calculated and printed

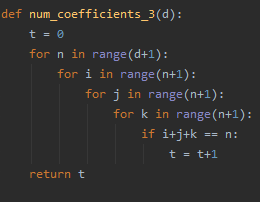


* the function returns bos, list\_of\_features, list\_of\_target

**Task 2 – model function**

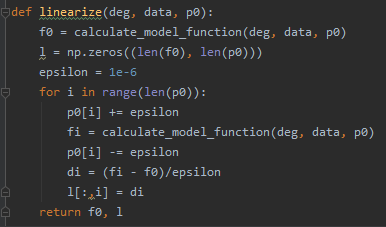


* function takes in the degree, the feature dataset and the parameter vector of coefficients
* a numpy array of zeros is created according to the length of the data
* this step involves processing a trivaritate model function for each of each of the columns in the feature vector since the features have 3 columns.



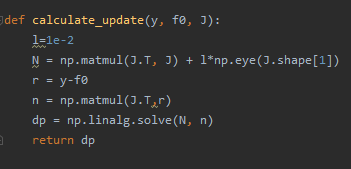
* this function will determine the number of coefficients in a trivariate polynomial and will be used in the regression function

**Task 3 – linearization**



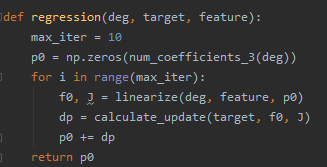
* the function takes in the degree, feature vectors and the linearization point’s coefficients
* the calculate model function is calculated at the linear point (p0)
* all of the components in (p0) is iterated in the for loop
* the model function is calculated while a small petrube is added to the parameter vector’s components
* the jacobian is calculated by dividing the difference between the pertrube

**Task 4 – parameter update**



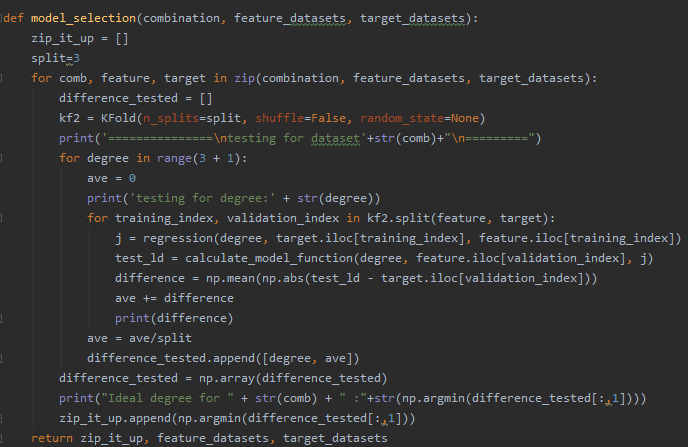
* The function takes in the training target vector, estimated target vector and the jacobian
* N = (multiplication of the Jacobian and its transpose) + regularisation matrix of the same size as the jacobian
* (residual) r = training target vector – estimated target vector (y-f0)
* n = right hand side of the normal equal
* the optimal parameter update is solved with N and n

**Task 5 – regression**



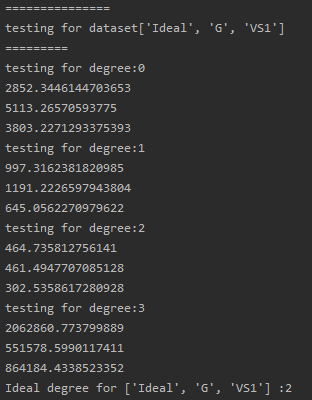
* the parameters include the polynomial degree, targets and features
* p0 is initialized as a parameter vector of trivariate coefficients with zero
* an iterative procedure is produced where the **linearize** function and the **calculate\_update** function is executed in alternating fashion
* the best fitting polynomial coefficient is later calculated and returned

**Task 6 – model selection**

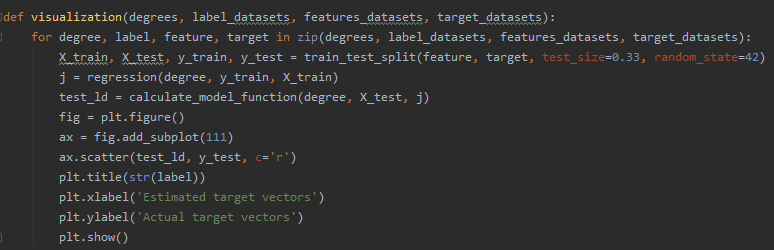


* the function takes in the list of combination names(used to display information), feature datasets(feature\_datasets) and the list of targets(target\_datasets) as input
* a list called zip\_it\_up will contain a list of the best polynomial degree for each dataset
* a for loop is created for zip(feature\_datasets, target\_datasets) which iterates over all of the features and targets simultaneously
* a difference\_tested array list will contain lists with the ideal degree and the average difference calculated
* kf2 🡪 KFold is initiated with the number of splits determined at the beginning of the function(split)
* for each degree tested(0-3),
  + a variable called **ave** is initiated which will calculate the average difference for the KFold splits
  + the KFold split process is carried out with the following procedures
    - regression(degree, targets, features
    - model function is calculated which outputs the estimated feature vector
    - the difference 🡪 mean absolute price difference is calculated
    - the variable **ave** is added with the difference calculated
  + **ave** is divided by the total number of splits
  + difference\_tested is appended with the list containing [degree and the average difference]
* the list difference\_tested is converted into a numpy array
* the zip\_it\_up list is appended with the degree that corresponds to the lowest average difference calculated
* the function returns the list zip\_it\_up along with the feature datasets and the target datasets that was tested on with this function

sample output in task 6

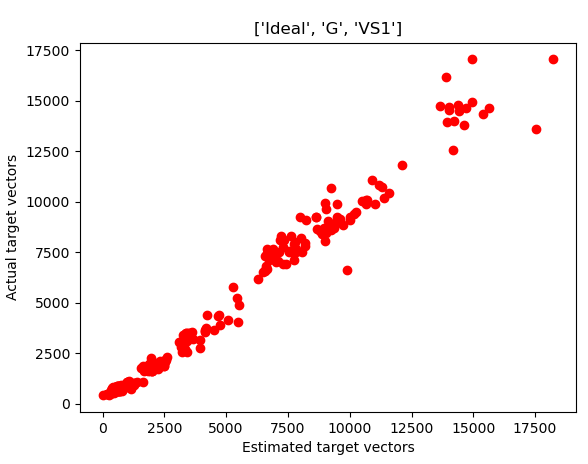


**Task 7 -visualisation of results**



* the function will take in the list of ideal degrees calculated in task 6, combination names(for displaying the title of each plot), features and target dataset
* a for loop is created which iterates through the list of ideal degrees, combination names, features and targets simultaneously
* the features and targets are split into train and test data with a test size of 0.33
* just like in task 6, the regression is calculated along with its model function
* the function then plots the estimated target vectors with the actual test targets.

Fig 2.1 – sample graph plotted with function 7



**Notes**:

* degrees 0-2 can be tested however, a degree above 2 can significantly slow down the code and end up with an abnormally large difference value