Chapter 2 : Section 7

**Evaluating Regression Models Performance**

**2.7.1 R Squared**

**[***Compare models w.r.t* **]**

We talked about the *simple linear regression* being constructed through the *ordinary least squares method* where we are minimizing

* We were counting that sum and then the *line that has them smallest sum* will be the *best fitting lin*e or will be this *simple* *linear* *regression* model.

|  |  |
| --- | --- |
| * Residual sum of squares: For out fitted-line (model line) the residual sum of squares or sum of squared error (SSE) is : |  |
| * Total sum of squares: Using the average line we get the *total sum of squares:* |  |

* R-square: The following equation is the R-squared
* R-square is says that *"How good the best fitting line (or model)* *is with respect to the* *average line"*. Minimizing is maximize the . is the best scenario (and will never happen yo!!).
* So near 1 is better model and near 0 is the bad model. And also can be negative (model is worse than the average line).

**2.7.2 Adjusted**

**[***Compare models w.r.t and Increase/Decrease Feature variable* **]**

We talked about for a *Simple Linear Regression* while the same concepts apply for a multiple linear regression. We use R-squared as a *goodness of fit parameters*, the bigger it is (close to 1) the better model.

* Impact on by Increase/ Decrease of feature variables: is *biased*. Reason is: when we add *new feature variable* the always *minimize* or stays same. The *does'nt increase* because of the model, if new variable effect the model in negative way, the co-efficient of the new variable just get smaller, so that the effect to be negligible.
* By adding *new feature variable*, may decrease but never increase. Hence is never decrease by adding new feature variable.
* Then how to determine the impact of new feature variable? The solution is to use Adjusted .
* Adjusted : Folowing is the equation for Adjusted .

Where Sample size,

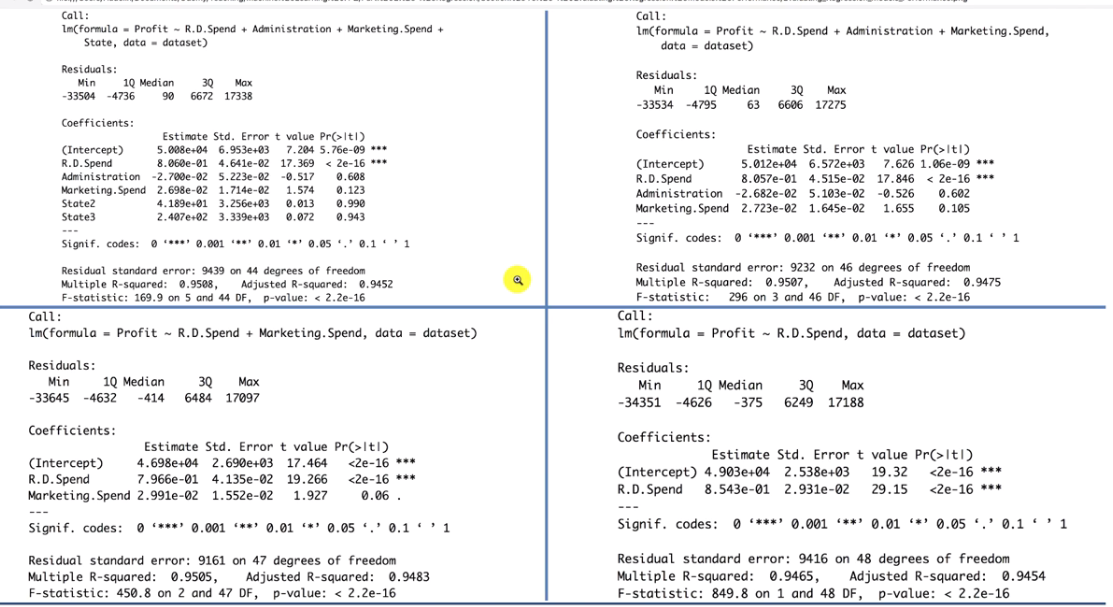
no. of regressor/feature variable.

* Penalization factor: Adjusted R-squared has a penalization factor. It *penalizes you for adding independent variables that don't help your model*.
* As we can see when we increase variable, increase so decreases i.e. increases.
* Now if the new variable doesn't help, then doesn’t decrease so increase and decrease.
* Here actually and balance each other. If the new variable helps, then decrease and will increase.
* In statistics, a *regressor* is the name given to any variable in a *regression* *model* that is used to predict a *response* *variable*. A *regressor* is also referred to as: An *explanatory* *variable*. An *independent* *variable*. A *manipulated* *variable*. Feature, independent variable, explanatory variable, regressor, covariate, or predictor are all names of the variables that are used to predict the target, outcome, dependent variable, regressand, or response.

**2.7.3 Feature selection using Adjusted R-squared**

Recall the Multiple Linear Regression's Backward elimination, where we rejected variables according to p-values. Now at final step we end up with only one variable R&D spend.

* After **4th** and **5th** iteration, we inspected the ***p-values***. At **4th** iteration, we end-up with R&D spend is definitely a very powerful predictor of the profit and definitely has a high statistical effect impact on the dependent variable profit.
* And as the *second independent variable* is with enough *statistical* *effect* *impact* is Marketing Spend, we can see that the p-value is **0.06** that is **6%**.
* Now we decide that we can keep this variable or not according to Adjusted R-squared. If we obseve fropm following diagram, that the Adj. R-square is actually greter with R&D spend and Marketing Spend both. But is get smaller when we reject Marketing Spend. Hence with R&D spend and Marketing Spend both we get the better prediction. And hence we are kipping Marketing Spend even though it has 6% p-value.
* So just inspect the *R-squared* and *Adj. R-square*. Give *Adj. R-square* top priority during feature selection.



**2.7.4 Implement automated Backward Elimination with Adjusted R-Squared In Python:**

Let’s take again the problem of the Multiple Linear Regression, with 5 independent variables. Automated Backward Elimination including Adjusted R Squared can be implemented this way:

#*Fundamental  Libraries*

**import** matplotlib **as** plt

**import** pandas **as** pd

**import** numpy **as** np

#*import dataset*

dataSet = pd**.read\_csv**("50\_Startups.csv")

X = dataSet**.**iloc[:, :-1] #*all rows except last*

y = dataSet**.**iloc[:, 4] #*5th row*

#*categorical to numerical*

**from** sklearn**.**compose **import** ColumnTransformer

**from** sklearn**.**preprocessing **import** OneHotEncoder

colTfrm = **ColumnTransformer**(transformers = [("encoder", **OneHotEncoder**(), [3])], remainder="passthrough" )

X\_encoded = np**.array**(colTfrm**.fit\_transform**(X))

        #*last column is now replaced with dummy colums (1st 3 colmns)*

#*Avoiding dummy-var trap: omit one dummy varable*

X\_go = X\_encoded[:, 1:]         #*select all columns starting from 2nd column*

y\_go = np**.array**(y) #*converting Dataframe to Vector/Array*

#*split dataset to Train and Test*

**from** sklearn**.**model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = **train\_test\_split**(X\_go, y\_go, test\_size = 0.2, random\_state = 0)

#*Fitting multiple linear regression on traing set*

**from** sklearn**.**linear\_model **import** LinearRegression

regResor = **LinearRegression**()

regResor**.fit**(X\_train, y\_train)

#*predict on the test-set X\_test*

y\_pred = regResor**.predict**(X\_test)

#*Checking the score*

**print**(np**.concatenate**((y\_pred**.reshape**(len(y\_pred),1), y\_test**.reshape**(len(y\_test),1)),1))

**print**('\n\n------------ Train Score: ', regResor**.score**(X\_train, y\_train))

**print**('\n\n------------ Test Score: ', regResor**.score**(X\_test, y\_test))

#*================= building the optimal model using Backward elimination ====================*

#*---------- Only P-value implement automatic Backward Elimination: No manual iteration is needed ----------------*

"""

import statsmodels.api as sm

def backwardElimination(x, sl):

    numVars = len(x[0])

    for i in range(0, numVars):

        regressor\_OLS = sm.OLS(endog = y\_go, exog=x).fit()

        # picking the max p-value

        maxPval = max(regressor\_OLS.pvalues).astype(float)

        if maxPval >= sl:

            for j in range(0, numVars - i):

                # deleting the clumn

                if (regressor\_OLS.pvalues[j].astype(float) == maxPval):

                    x = np.delete(x, j, 1) # deletes j-th column. "1" is used for "column". To delet "row" use "0"

    print(regressor\_OLS.summary())

    return x

SL = 0.05

X\_pre\_opt = np.append(arr = np.ones(shape = (50, 1)).astype(int), values =X\_go , axis =1) # interchange the columns

X\_opt = X\_pre\_opt[:, [0, 1, 2, 3, 4, 5]]

X\_Modeled = backwardElimination(X\_opt, SL)

"""

**import** statsmodels**.**api **as** sm

**def** **backwardElimination**(x, SL):

    numVars = **len**(x[0])

    temp = np**.zeros**((50, 6))**.astype**(int)

**for** i **in** **range**(0, numVars):

        regressor\_OLS = sm**.OLS**(endog = y\_go, exog=x)**.fit**()

        maxVar = **max**(regressor\_OLS**.**pvalues)**.astype**(float)

        adjR\_before = regressor\_OLS**.**rsquared\_adj**.astype**(float)

**if** maxVar **>** SL:

**for** j **in** **range**(0, numVars - i):

**if** (regressor\_OLS**.**pvalues[j]**.astype**(float) **==** maxVar):

                    temp[:,j] = x[:, j]

                    x = np**.delete**(x, j, 1)

                    tmp\_regressor = sm**.OLS**(y, x)**.fit**()

                    adjR\_after = tmp\_regressor**.**rsquared\_adj**.astype**(float)

**if** (adjR\_before **>=** adjR\_after):

                        x\_rollback = np**.hstack**((x, temp[:,[0,j]]))

                        x\_rollback = np**.delete**(x\_rollback, j, 1)

**print**(regressor\_OLS**.summary**())

**return** x\_rollback

**else**:

**continue**

    regressor\_OLS**.summary**()

**return** x

SL = 0.05

X\_pre\_opt = np**.append**(arr = np**.ones**(shape = (50, 1))**.astype**(int), values =X\_go , axis =1) #*interchange the columns*

X\_opt = X\_pre\_opt[:, [0, 1, 2, 3, 4, 5]]

X\_Modeled = **backwardElimination**(X\_opt, SL)

#*python prctc\_modl\_eval\_mul\_lin\_regsn.py*

Only p-value

==============================================================================

Dep. Variable: y R-squared: 0.947

Model: OLS Adj. R-squared: 0.945

Method: Least Squares F-statistic: 849.8

Date: Thu, 17 Mar 2022 Prob (F-statistic): 3.50e-32

Time: 13:37:35 Log-Likelihood: -527.44

No. Observations: 50 AIC: 1059.

Df Residuals: 48 BIC: 1063.

Df Model: 1

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------

const 4.903e+04 2537.897 19.320 0.000 4.39e+04 5.41e+04

x1 0.8543 0.029 29.151 0.000 0.795 0.913

==============================================================================

Omnibus: 13.727 Durbin-Watson: 1.116

Prob(Omnibus): 0.001 Jarque-Bera (JB): 18.536

Skew: -0.911 Prob(JB): 9.44e-05

Kurtosis: 5.361 Cond. No. 1.65e+05

==============================================================================

Only p-value and Adjusted R-squared: Notice Adj. R-squared increased

**OLS Regression Results**

**==============================================================================**

**Dep. Variable: y R-squared: 0.950**

**Model: OLS Adj. R-squared: 0.948**

**Method: Least Squares F-statistic: 450.8**

**Date: Thu, 17 Mar 2022 Prob (F-statistic): 2.16e-31**

**Time: 13:39:25 Log-Likelihood: -525.54**

**No. Observations: 50 AIC: 1057.**

**Df Residuals: 47 BIC: 1063.**

**Df Model: 2**

**Covariance Type: nonrobust**

**==============================================================================**

**coef std err t P>|t| [0.025 0.975]**

**------------------------------------------------------------------------------**

**const 4.698e+04 2689.933 17.464 0.000 4.16e+04 5.24e+04**

**x1 0.7966 0.041 19.266 0.000 0.713 0.880**

**x2 0.0299 0.016 1.927 0.060 -0.001 0.061**

**==============================================================================**

**Omnibus: 14.677 Durbin-Watson: 1.257**

**Prob(Omnibus): 0.001 Jarque-Bera (JB): 21.161**

**Skew: -0.939 Prob(JB): 2.54e-05**

**Kurtosis: 5.575 Cond. No. 5.32e+05**

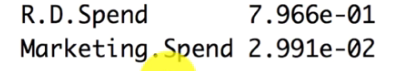
**==============================================================================**

**2.7.5 Interpreting Linear Regression Coefficients**

How do we find/explain/interpret the Coefficient of the final model. Consider the previous Venture Capitalist example,

|  |
| --- |
|  |

* Notice the following:



So these coefficients here these and . What they're telling us is that,

* Sign: First of all you look at the sign. If the sign is positive that means your variable is correlated with your independent variable. Meaning that if you ***change*** the value of your ***independent variable*** then the value then you can see that the ***dependent variable*** will be ***changing*** in the ***same direction***.
* So basically if you'll be increasing spend on R&D then you're profitable be increasing If increasing spend of marking then your profit is also increasing.
* If the sign is negative then it's the opposite effect. So basically increase your independent variable and you depend variable decreases. i.e If any one of them has *–ve* value then increasing the *corresponding feature* will cause decrease the dependent variable (it is the basic math).
* Always remember the magnitude: So you might think that, Magnitude of coefficient for this R&D Spend is bigger than the Market Spend coefficient. So definitely R&D spend has a bigger impact. But that’s not the case, the unit of measurement is important here, to compare something like that we have to measure all the variables in a same measurement scale.
* So, even though coefficient for this R&D Spend is bigger but its magnitude of unit can be lower say 1 Cent i.e ($0.01), and the magnitude of unit for Market Spend say $1 where its coefficient is lower than R&D Spend.
* Use the term "Per Unit Change": R.D. Spend could be million dollar (unit) or it could be in Cents i.e. ($0.01). Or some feature variable can have different units of measure : say Km, or Celsius etc. The magnitude could be different for different feature variables (say R&D spend could be in Billion Dollars and Marketing-spend could be in Million Dollars). In all of those case we should say "Per Unit Change".
* So here we use $1 for measurement unit for R&D Spend and Market Spend . Then we can say, for every dollar (for every unit) of R&D spend that you increase, according to his model your profit will increase by ***79*** ***cents*** or ***$0.79***. For every unit that you decrease in your R&D spend your profit will decrease by 0.79unit of profit.
* Similarly for every dollar (for every unit) of Market Spend that you increase, according to his model your profit will increase by 2.99 cents or $0.0299.

**2.7.6 FAQ**

I understand that in order to *evaluate* *multiple* *linear* *models* I can use the Adjusted Rsquared or the Pearson matrix. But how can I evaluate ***Polynomial Regression*** models or ***Random Forest Regression*** models which are ***not linear***?

* You can evaluate *polynomial* *regression* models and *random* *forest* *regression* models, by computing the *"Mean of Squared Residuals" (the mean of the squared errors)*. You can compute that easily with a ***sum*** function or using a ***for loop***, by computing the ***sum of the squared differences*** between the predicted outcomes and the real outcomes, over all the observations of the test set.
* You couldn’t really apply *Backward* *Elimination* to *Polynomial* *Regression* and *Random* *Forest* *Regression* models because these models don’t have *coefficients* combined in a *linear* *regression* *equation* and therefore don’t have p-values.

However in Chapter 9 - Dimensionality Reduction, we will see some techniques like Backward Elimination, to reduce the number of features, based on Feature Selection & Feature Extraction.

* What are Low/High Bias/Variance in Machine Learning?
* Low Bias is when your model predictions are very *close* to the *real* *values*.
* High Bias is when your model predictions are *far* from the *real* *values*.
* Low Variance: when you *run your model several times*, the *different predictions* of your observation points *won’t vary* much.
* High Variance: when you *run your model several times*, the *different predictions* of your observation points will *vary a lot*.
* What you want to get when you build a model is: Low Bias and Low Variance.