**Name: - UPENDRA DAMA**

**Batch: - 25/01/2020 (Weekend)**

**Module: - 7**

**Prepare a prediction model for sales of the computer**

**Problem Statement: -** Prepare a prediction model for sales of the computer.

**EDA: -** Let’s to Exploratory Data Analysis.

|  |
| --- |
| > library(readr)  > Computer\_Data <- read\_csv("Desktop/Digi 360/Module 7/Computer\_Data.csv")  Parsed with column specification:  cols(  X1 = col\_double(),  price = col\_double(),  speed = col\_double(),  hd = col\_double(),  ram = col\_double(),  screen = col\_double(),  cd = col\_character(),  multi = col\_character(),  premium = col\_character(),  ads = col\_double(),  trend = col\_double()  )  Warning message:  Missing column names filled in: 'X1' [1]  > View(Computer\_Data)  > attach(Computer\_Data)  The following objects are masked from Computer\_Data (pos = 4):  ads, cd, hd, multi, premium, price, ram, screen, speed, trend, X1  The following objects are masked from Computer\_Data (pos = 5):  ads, cd, hd, multi, premium, price, ram, screen, speed, trend, X1  > ##Replace categorical values with dummy values  > library(plyr)  > Computer\_Data$cd <- revalue(Computer\_Data$cd, c("yes"=1, "no"=0))  > Computer\_Data$multi <- revalue(Computer\_Data$multi, c("yes"=1, "no"=0))  > Computer\_Data$premium <- revalue(Computer\_Data$premium, c("yes"=1, "no"=0))  > ###Let's do EDA by looking at summary  > summary(Computer\_Data)  X1 price speed hd ram  Min. : 1 Min. : 949 Min. : 25.00 Min. : 80.0 Min. : 2.000  1st Qu.:1566 1st Qu.:1794 1st Qu.: 33.00 1st Qu.: 214.0 1st Qu.: 4.000  Median :3130 Median :2144 Median : 50.00 Median : 340.0 Median : 8.000  Mean :3130 Mean :2220 Mean : 52.01 Mean : 416.6 Mean : 8.287  3rd Qu.:4694 3rd Qu.:2595 3rd Qu.: 66.00 3rd Qu.: 528.0 3rd Qu.: 8.000  Max. :6259 Max. :5399 Max. :100.00 Max. :2100.0 Max. :32.000  screen cd multi premium  Min. :14.00 Length:6259 Length:6259 Length:6259  1st Qu.:14.00 Class :character Class :character Class :character  Median :14.00 Mode :character Mode :character Mode :character  Mean :14.61  3rd Qu.:15.00  Max. :17.00  ads trend  Min. : 39.0 Min. : 1.00  1st Qu.:162.5 1st Qu.:10.00  Median :246.0 Median :16.00  Mean :221.3 Mean :15.93  3rd Qu.:275.0 3rd Qu.:21.50  Max. :339.0 Max. :35.00 |

Here, the mean is less than the median for ads and trend so the distribution is **left skewed**.

Whereas for remaining variables, mean is greater than the median so the distribution is **right skewed**.

**Relationship: -**

Let’s draw scatter diagram to see the relationship among output variable Price and remaining all input variables.

Here, cd, premium and multi are categorical variables, So, let’s assign the dummy variables for them.

|  |
| --- |
| ##Remove the first column since it is serial number.  > Computer\_Data$X1 <- NULL  > Computer\_Data$cd <- revalue(Computer\_Data$cd, c("yes"=1, "no"=0))  > Computer\_Data$cd <- as.numeric(Computer\_Data$cd)  > Computer\_Data$multi <- revalue(Computer\_Data$multi, c("yes"=1, "no"=0))  > Computer\_Data$multi <- as.numeric(Computer\_Data$multi)  > Computer\_Data$premium <- revalue(Computer\_Data$premium, c("yes"=1, "no"=0))  > Computer\_Data$premium <- as.numeric(Computer\_Data$premium)  ##Draw the scatter plot for entire dataset  > plot(Computer\_Data)    ##Finding correlation coefficient  > cor(Computer\_Data)  price speed hd ram screen cd  price 1.00000000 0.30097646 0.43025779 0.62274824 0.296041474 0.19734334  speed 0.30097646 1.00000000 0.37230410 0.23476050 0.189074122 0.25825980  hd 0.43025779 0.37230410 1.00000000 0.77772630 0.232801530 0.50357041  ram 0.62274824 0.23476050 0.77772630 1.00000000 0.208953740 0.43850441  screen 0.29604147 0.18907412 0.23280153 0.20895374 1.000000000 0.12948766  cd 0.19734334 0.25825980 0.50357041 0.43850441 0.129487662 1.00000000  multi -0.01665139 0.08417193 0.09280483 0.04549689 -0.001740414 0.43217930  premium -0.08069636 0.11420791 0.19692359 0.19714459 0.018745223 0.21607660  ads 0.05454047 -0.21523206 -0.32322200 -0.18166971 -0.093919429 -0.06109108  trend -0.19998694 0.40543833 0.57779013 0.27684384 0.188614445 0.44578018  multi premium ads trend  price -0.016651388 -0.08069636 0.05454047 -0.19998694  speed 0.084171934 0.11420791 -0.21523206 0.40543833  hd 0.092804830 0.19692359 -0.32322200 0.57779013  ram 0.045496894 0.19714459 -0.18166971 0.27684384  screen -0.001740414 0.01874522 -0.09391943 0.18861444  cd 0.432179298 0.21607660 -0.06109108 0.44578018  multi 1.000000000 0.12477474 -0.03039426 0.21090743  premium 0.124774741 1.00000000 -0.15202274 0.04210738  ads -0.030394260 -0.15202274 1.00000000 -0.31855251  trend 0.210907431 0.04210738 -0.31855251 1.00000000 |

Here we observe the correlation coefficient is very far from 1 among all the variables. So, the correlation is not good among the variables.

**Model Building: -**

Now, let’s build the model.

|  |
| --- |
| ##Start building model  > model1 <- lm(price~speed+hd+ram+screen+cd+multi+premium+ads+trend)  > summary(model1)  Call:  lm(formula = price ~ speed + hd + ram + screen + cd + multi +  premium + ads + trend)  Residuals:  Min 1Q Median 3Q Max  -1093.77 -174.24 -11.49 146.49 2001.05  Coefficients:  Estimate Std. Error t value Pr(>|t|)  (Intercept) 307.98798 60.35341 5.103 3.44e-07 \*\*\*  speed 9.32028 0.18506 50.364 < 2e-16 \*\*\*  hd 0.78178 0.02761 28.311 < 2e-16 \*\*\*  ram 48.25596 1.06608 45.265 < 2e-16 \*\*\*  screen 123.08904 3.99950 30.776 < 2e-16 \*\*\*  cdyes 60.91671 9.51559 6.402 1.65e-10 \*\*\*  multiyes 104.32382 11.41268 9.141 < 2e-16 \*\*\*  premiumyes -509.22473 12.34225 -41.259 < 2e-16 \*\*\*  ads 0.65729 0.05132 12.809 < 2e-16 \*\*\*  trend -51.84958 0.62871 -82.470 < 2e-16 \*\*\*  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 275.3 on 6249 degrees of freedom  Multiple R-squared: 0.7756, Adjusted R-squared: 0.7752  F-statistic: 2399 on 9 and 6249 DF, p-value: < 2.2e-16  **> ###Since R^2 value is low, let's do transformations. Based on experience, let's move to 4th degree polynomial directly.**  > model2 <- lm(price~speed+I(speed^2)+I(speed^3)+I(speed^4)+hd+I(hd^2)+I(hd^3)+I(hd^4)+ram+I(ram^2)+I(ram^3)+I(ram^4)+screen+I(screen^2)+I(screen^3)+I(screen^4)+cd+I(cd\*cd)+I(cd\*cd\*cd)+I(cd\*cd\*cd\*cd)+multi+I(multi^2)+I(multi^3)+I(multi^4)+premium+I(premium^2)+I(premium^3)+I(premium^4)+ads+I(ads^2)+I(ads^3)+I(ads^4)+trend+I(trend^2)+I(trend^3)+I(trend^4), Computer\_Data)  >  > summary(model2)  Call:  lm(formula = price ~ speed + I(speed^2) + I(speed^3) + I(speed^4) +  hd + I(hd^2) + I(hd^3) + I(hd^4) + ram + I(ram^2) + I(ram^3) +  I(ram^4) + screen + I(screen^2) + I(screen^3) + I(screen^4) +  cd + I(cd \* cd) + I(cd \* cd \* cd) + I(cd \* cd \* cd \* cd) +  multi + I(multi^2) + I(multi^3) + I(multi^4) + premium +  I(premium^2) + I(premium^3) + I(premium^4) + ads + I(ads^2) +  I(ads^3) + I(ads^4) + trend + I(trend^2) + I(trend^3) + I(trend^4),  data = Computer\_Data)  Residuals:  Min 1Q Median 3Q Max  -948.35 -159.55 -17.92 132.89 1902.52  Coefficients: (11 not defined because of singularities)  Estimate Std. Error t value Pr(>|t|)  (Intercept) 9.478e+03 9.141e+02 10.368 < 2e-16 \*\*\*  speed 1.669e+02 2.074e+01 8.046 1.01e-15 \*\*\*  I(speed^2) -3.875e+00 6.014e-01 -6.444 1.25e-10 \*\*\*  I(speed^3) 4.053e-02 7.268e-03 5.576 2.56e-08 \*\*\*  I(speed^4) -1.539e-04 3.078e-05 -4.999 5.93e-07 \*\*\*  hd 3.654e+00 2.156e-01 16.948 < 2e-16 \*\*\*  I(hd^2) -4.832e-03 4.963e-04 -9.737 < 2e-16 \*\*\*  I(hd^3) 3.120e-06 4.336e-07 7.197 6.89e-13 \*\*\*  I(hd^4) -7.012e-10 1.190e-10 -5.891 4.04e-09 \*\*\*  ram -7.455e+01 1.233e+01 -6.048 1.56e-09 \*\*\*  I(ram^2) 1.491e+01 1.490e+00 10.007 < 2e-16 \*\*\*  I(ram^3) -6.856e-01 7.238e-02 -9.473 < 2e-16 \*\*\*  I(ram^4) 1.033e-02 1.189e-03 8.691 < 2e-16 \*\*\*  screen -1.347e+03 1.136e+02 -11.851 < 2e-16 \*\*\*  I(screen^2) 4.739e+01 3.692e+00 12.836 < 2e-16 \*\*\*  I(screen^3) NA NA NA NA  I(screen^4) NA NA NA NA  cd 5.036e+01 8.932e+00 5.638 1.80e-08 \*\*\*  I(cd \* cd) NA NA NA NA  I(cd \* cd \* cd) NA NA NA NA  I(cd \* cd \* cd \* cd) NA NA NA NA  multi 9.701e+01 1.037e+01 9.356 < 2e-16 \*\*\*  I(multi^2) NA NA NA NA  I(multi^3) NA NA NA NA  I(multi^4) NA NA NA NA  premium -5.621e+02 1.154e+01 -48.702 < 2e-16 \*\*\*  I(premium^2) NA NA NA NA  I(premium^3) NA NA NA NA  I(premium^4) NA NA NA NA  ads -2.831e+00 3.065e+00 -0.924 0.355741  I(ads^2) 3.215e-02 2.478e-02 1.297 0.194572  I(ads^3) -1.359e-04 8.369e-05 -1.624 0.104472  I(ads^4) 1.850e-07 1.005e-07 1.840 0.065765 .  trend -6.958e+01 1.257e+01 -5.534 3.26e-08 \*\*\*  I(trend^2) 4.650e+00 1.263e+00 3.683 0.000233 \*\*\*  I(trend^3) -2.613e-01 5.608e-02 -4.659 3.24e-06 \*\*\*  I(trend^4) 4.040e-03 8.772e-04 4.606 4.19e-06 \*\*\*  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 249 on 6233 degrees of freedom  Multiple R-squared: 0.8169, Adjusted R-squared: 0.8161  F-statistic: 1112 on 25 and 6233 DF, p-value: < 2.2e-16  **##Since R^2 is moderate let’s do further transformations**  **##5th degree polynomial**  > model3 <- lm(price~speed+I(speed^2)+I(speed^3)+I(speed^4)+I(speed^5)+hd+I(hd^2)+I(hd^3)+I(hd^4)+I(hd^5)+ram+I(ram^2)+I(ram^3)+I(ram^4)+I(ram^5)+screen+I(screen^2)+I(screen^3)+I(screen^4)+I(screen^5)+cd+I(cd^2)+I(cd^3)+I(cd^4)+I(cd^5)+multi+I(multi^2)+I(multi^3)+I(multi^4)+I(multi^5)+premium+I(premium^2)+I(premium^3)+I(premium^4)+I(premium^5)+ads+I(ads^2)+I(ads^3)+I(ads^4)+I(ads^5)+trend+I(trend^2)+I(trend^3)+I(trend^4)+I(trend^5), Computer\_Data)  > summary(model3)  Call:  lm(formula = price ~ speed + I(speed^2) + I(speed^3) + I(speed^4) +  I(speed^5) + hd + I(hd^2) + I(hd^3) + I(hd^4) + I(hd^5) +  ram + I(ram^2) + I(ram^3) + I(ram^4) + I(ram^5) + screen +  I(screen^2) + I(screen^3) + I(screen^4) + I(screen^5) + cd +  I(cd^2) + I(cd^3) + I(cd^4) + I(cd^5) + multi + I(multi^2) +  I(multi^3) + I(multi^4) + I(multi^5) + premium + I(premium^2) +  I(premium^3) + I(premium^4) + I(premium^5) + ads + I(ads^2) +  I(ads^3) + I(ads^4) + I(ads^5) + trend + I(trend^2) + I(trend^3) +  I(trend^4) + I(trend^5), data = Computer\_Data)  Residuals:  Min 1Q Median 3Q Max  -960.1 -158.4 -21.9 128.0 1916.3  Coefficients: (15 not defined because of singularities)  Estimate Std. Error t value Pr(>|t|)  (Intercept) 8.881e+03 1.210e+03 7.341 2.39e-13 \*\*\*  speed 3.827e+02 8.849e+01 4.325 1.55e-05 \*\*\*  I(speed^2) -1.321e+01 3.674e+00 -3.597 0.000325 \*\*\*  I(speed^3) 2.292e-01 7.244e-02 3.164 0.001566 \*\*  I(speed^4) -1.943e-03 6.781e-04 -2.866 0.004169 \*\*  I(speed^5) 6.396e-06 2.407e-06 2.657 0.007908 \*\*  hd 7.583e+00 4.406e-01 17.212 < 2e-16 \*\*\*  I(hd^2) -1.721e-02 1.382e-03 -12.455 < 2e-16 \*\*\*  I(hd^3) 1.950e-05 1.868e-06 10.439 < 2e-16 \*\*\*  I(hd^4) -1.015e-08 1.111e-09 -9.131 < 2e-16 \*\*\*  I(hd^5) 1.933e-12 2.351e-13 8.220 2.46e-16 \*\*\*  ram -4.900e+02 3.659e+01 -13.393 < 2e-16 \*\*\*  I(ram^2) 9.027e+01 6.509e+00 13.870 < 2e-16 \*\*\*  I(ram^3) -6.427e+00 4.959e-01 -12.962 < 2e-16 \*\*\*  I(ram^4) 2.013e-01 1.668e-02 12.067 < 2e-16 \*\*\*  I(ram^5) -2.281e-03 2.035e-04 -11.208 < 2e-16 \*\*\*  screen -1.408e+03 1.123e+02 -12.537 < 2e-16 \*\*\*  I(screen^2) 4.930e+01 3.649e+00 13.509 < 2e-16 \*\*\*  I(screen^3) NA NA NA NA  I(screen^4) NA NA NA NA  I(screen^5) NA NA NA NA  cd 5.617e+01 8.852e+00 6.345 2.38e-10 \*\*\*  I(cd^2) NA NA NA NA  I(cd^3) NA NA NA NA  I(cd^4) NA NA NA NA  I(cd^5) NA NA NA NA  multi 9.704e+01 1.025e+01 9.465 < 2e-16 \*\*\*  I(multi^2) NA NA NA NA  I(multi^3) NA NA NA NA  I(multi^4) NA NA NA NA  I(multi^5) NA NA NA NA  premium -5.971e+02 1.183e+01 -50.480 < 2e-16 \*\*\*  I(premium^2) NA NA NA NA  I(premium^3) NA NA NA NA  I(premium^4) NA NA NA NA  I(premium^5) NA NA NA NA  ads -1.588e+01 7.942e+00 -1.999 0.045623 \*  I(ads^2) 2.008e-01 9.427e-02 2.130 0.033191 \*  I(ads^3) -1.109e-03 5.201e-04 -2.132 0.033062 \*  I(ads^4) 2.769e-06 1.349e-06 2.053 0.040095 \*  I(ads^5) -2.575e-09 1.328e-09 -1.939 0.052561 .  trend -8.465e+01 2.236e+01 -3.786 0.000154 \*\*\*  I(trend^2) 6.212e+00 3.436e+00 1.808 0.070707 .  I(trend^3) -3.473e-01 2.370e-01 -1.465 0.142836  I(trend^4) 6.548e-03 7.307e-03 0.896 0.370206  I(trend^5) -2.865e-05 8.268e-05 -0.346 0.728983  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 245.9 on 6228 degrees of freedom  Multiple R-squared: 0.8217, Adjusted R-squared: 0.8208  F-statistic: 956.5 on 30 and 6228 DF, p-value: < 2.2e-16  **##7th degree polynomial, since R^2 is still less than 0.85**  > model4 <- lm(price~speed+I(speed^2)+I(speed^3)+I(speed^4)+I(speed^5)+I(speed^6)+I(speed^7)+hd+I(hd^2)+I(hd^3)+I(hd^4)+I(hd^5)+I(hd^6)+I(hd^7)+ram+I(ram^2)+I(ram^3)+I(ram^4)+I(ram^5)+I(ram^6)+I(ram^7)+screen+I(screen^2)+I(screen^3)+I(screen^4)+I(screen^5)+I(screen^6)+I(screen^7)+cd+I(cd^2)+I(cd^3)+I(cd^4)+I(cd^5)+I(cd^6)+I(cd^7)+multi+I(multi^2)+I(multi^3)+I(multi^4)+I(multi^5)+I(multi^6)+I(multi^7)+premium+I(premium^2)+I(premium^3)+I(premium^4)+I(premium^5)+I(premium^6)+I(premium^7)+ads+I(ads^2)+I(ads^3)+I(ads^4)+I(ads^5)+I(ads^6)+I(ads^7)+trend+I(trend^2)+I(trend^3)+I(trend^4)+I(trend^5)+I(trend^6)+I(trend^7), Computer\_Data)  > summary(model4)  Call:  lm(formula = price ~ speed + I(speed^2) + I(speed^3) + I(speed^4) +  I(speed^5) + I(speed^6) + I(speed^7) + hd + I(hd^2) + I(hd^3) +  I(hd^4) + I(hd^5) + I(hd^6) + I(hd^7) + ram + I(ram^2) +  I(ram^3) + I(ram^4) + I(ram^5) + I(ram^6) + I(ram^7) + screen +  I(screen^2) + I(screen^3) + I(screen^4) + I(screen^5) + I(screen^6) +  I(screen^7) + cd + I(cd^2) + I(cd^3) + I(cd^4) + I(cd^5) +  I(cd^6) + I(cd^7) + multi + I(multi^2) + I(multi^3) + I(multi^4) +  I(multi^5) + I(multi^6) + I(multi^7) + premium + I(premium^2) +  I(premium^3) + I(premium^4) + I(premium^5) + I(premium^6) +  I(premium^7) + ads + I(ads^2) + I(ads^3) + I(ads^4) + I(ads^5) +  I(ads^6) + I(ads^7) + trend + I(trend^2) + I(trend^3) + I(trend^4) +  I(trend^5) + I(trend^6) + I(trend^7), data = Computer\_Data)  Residuals:  Min 1Q Median 3Q Max  -987.02 -155.19 -20.29 127.04 1868.28  Coefficients: (27 not defined because of singularities)  Estimate Std. Error t value Pr(>|t|)  (Intercept) 1.041e+04 1.473e+03 7.063 1.80e-12 \*\*\*  speed 3.646e+02 8.789e+01 4.148 3.40e-05 \*\*\*  I(speed^2) -1.265e+01 3.650e+00 -3.466 0.000532 \*\*\*  I(speed^3) 2.215e-01 7.199e-02 3.076 0.002104 \*\*  I(speed^4) -1.898e-03 6.740e-04 -2.816 0.004874 \*\*  I(speed^5) 6.312e-06 2.393e-06 2.637 0.008375 \*\*  I(speed^6) NA NA NA NA  I(speed^7) NA NA NA NA  hd 2.215e+01 1.560e+00 14.203 < 2e-16 \*\*\*  I(hd^2) -9.921e-02 8.502e-03 -11.669 < 2e-16 \*\*\*  I(hd^3) 2.401e-04 2.264e-05 10.606 < 2e-16 \*\*\*  I(hd^4) -3.227e-07 3.209e-08 -10.056 < 2e-16 \*\*\*  I(hd^5) 2.394e-10 2.454e-11 9.753 < 2e-16 \*\*\*  I(hd^6) -9.101e-14 9.504e-15 -9.576 < 2e-16 \*\*\*  I(hd^7) 1.374e-17 1.451e-18 9.464 < 2e-16 \*\*\*  ram -6.800e+02 4.177e+01 -16.280 < 2e-16 \*\*\*  I(ram^2) 1.247e+02 7.476e+00 16.676 < 2e-16 \*\*\*  I(ram^3) -9.090e+00 5.735e-01 -15.850 < 2e-16 \*\*\*  I(ram^4) 2.921e-01 1.944e-02 15.026 < 2e-16 \*\*\*  I(ram^5) -3.393e-03 2.388e-04 -14.210 < 2e-16 \*\*\*  I(ram^6) NA NA NA NA  I(ram^7) NA NA NA NA  screen -1.463e+03 1.116e+02 -13.112 < 2e-16 \*\*\*  I(screen^2) 5.105e+01 3.624e+00 14.086 < 2e-16 \*\*\*  I(screen^3) NA NA NA NA  I(screen^4) NA NA NA NA  I(screen^5) NA NA NA NA  I(screen^6) NA NA NA NA  I(screen^7) NA NA NA NA  cd 6.136e+01 8.764e+00 7.001 2.81e-12 \*\*\*  I(cd^2) NA NA NA NA  I(cd^3) NA NA NA NA  I(cd^4) NA NA NA NA  I(cd^5) NA NA NA NA  I(cd^6) NA NA NA NA  I(cd^7) NA NA NA NA  multi 9.326e+01 1.014e+01 9.193 < 2e-16 \*\*\*  I(multi^2) NA NA NA NA  I(multi^3) NA NA NA NA  I(multi^4) NA NA NA NA  I(multi^5) NA NA NA NA  I(multi^6) NA NA NA NA  I(multi^7) NA NA NA NA  premium -6.126e+02 1.177e+01 -52.035 < 2e-16 \*\*\*  I(premium^2) NA NA NA NA  I(premium^3) NA NA NA NA  I(premium^4) NA NA NA NA  I(premium^5) NA NA NA NA  I(premium^6) NA NA NA NA  I(premium^7) NA NA NA NA  ads -9.647e+01 4.679e+01 -2.062 0.039262 \*  I(ads^2) 2.167e+00 9.610e-01 2.255 0.024189 \*  I(ads^3) -2.458e-02 1.025e-02 -2.398 0.016526 \*  I(ads^4) 1.533e-04 6.148e-05 2.494 0.012660 \*  I(ads^5) -5.349e-07 2.088e-07 -2.561 0.010451 \*  I(ads^6) 9.782e-10 3.747e-10 2.611 0.009056 \*\*  I(ads^7) -7.301e-13 2.758e-13 -2.648 0.008125 \*\*  trend -3.596e+02 6.010e+01 -5.984 2.30e-09 \*\*\*  I(trend^2) 1.010e+02 1.934e+01 5.220 1.85e-07 \*\*\*  I(trend^3) -1.310e+01 2.727e+00 -4.803 1.60e-06 \*\*\*  I(trend^4) 8.615e-01 1.988e-01 4.333 1.49e-05 \*\*\*  I(trend^5) -3.037e-02 7.822e-03 -3.883 0.000104 \*\*\*  I(trend^6) 5.441e-04 1.573e-04 3.459 0.000547 \*\*\*  I(trend^7) -3.875e-06 1.265e-06 -3.062 0.002208 \*\*  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 243 on 6222 degrees of freedom  Multiple R-squared: 0.8259, Adjusted R-squared: 0.8249  F-statistic: 820.2 on 36 and 6222 DF, p-value: < 2.2e-16 |

So, the linear equation with the above coefficients will be considered a good equation because the p value is less than 0.05 and R^2 is also significant.

**Model Evolution: -**

Here p value is significant so we reject the null hypothesis. That means there is significant correlation between price vs other independent variables.

Here we also can see R-squared value is 0.8259 which is significantly close to 0.85. **Hence our model is good** and we don’t need further transformations.

Since we are ready with equation, let’s **predict** the values and accuracy after **splitting** the dataset.

|  |
| --- |
| > n=nrow(Computer\_Data)  > n1=n\*0.7  > n2=n-n1  > n2  [1] 1877.7  > train=sample(1:n,n1)  > test=Computer\_Data[-train,]  > pred=predict(model4,newdata = test)  Warning message:  In predict.lm(model4, newdata = test) :  prediction from a rank-deficient fit may be misleading  > actual=test$price  > error=actual-pred  > test.rmse=sqrt(mean(error\*\*2))  > test.rmse  [1] 247.1674  > train.rmse=sqrt(mean(model4$residuals\*\*2))  > train.rmse  [1] 242.2887  > |

Here we can see test rmse value is very close to train rmse value. So, our model is significantly fit.

**Table of R^2 value for two models: -**

|  |  |
| --- | --- |
| Model 1 | 0.7756 |
| Model 2 | 0.8169 |
| Model 3 | 0.8217 |
| Model 4 | 0.8259 |

**Let’s do the same in Python.**

|  |
| --- |
| **Python Code** |
| ####################################################  ########Predict computer sales of a dataset#############  ####################################################  import pandas as pd  import numpy as np  comp = pd.read\_csv ("~/desktop/Digi 360/Module 7/Computer\_data.csv")  comp.head()   |  | Unnamed: 0 | price | speed | hd | ram | screen | cd | multi | premium | ads | trend | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 1499 | 25 | 80 | 4 | 14 | no | no | yes | 94 | 1 | | 1 | 2 | 1795 | 33 | 85 | 2 | 14 | no | no | yes | 94 | 1 | | 2 | 3 | 1595 | 25 | 170 | 4 | 15 | no | no | yes | 94 | 1 | | 3 | 4 | 1849 | 25 | 170 | 8 | 14 | no | no | no | 94 | 1 | | 4 | 5 | 3295 | 33 | 340 | 16 | 14 | no | no | yes | 94 | 1 |   ###Let's assign dummy values to categorical values  comp1 = comp.replace(to\_replace ="yes", value =1)  comp1 = comp1.replace(to\_replace ="no", value =0)  comp1.head()   |  | Unnamed: 0 | price | speed | hd | ram | screen | cd | multi | premium | ads | trend | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 1499 | 25 | 80 | 4 | 14 | 0 | 0 | 1 | 94 | 1 | | 1 | 2 | 1795 | 33 | 85 | 2 | 14 | 0 | 0 | 1 | 94 | 1 | | 2 | 3 | 1595 | 25 | 170 | 4 | 15 | 0 | 0 | 1 | 94 | 1 | | 3 | 4 | 1849 | 25 | 170 | 8 | 14 | 0 | 0 | 0 | 94 | 1 | | 4 | 5 | 3295 | 33 | 340 | 16 | 14 | 0 | 0 | 1 | 94 | 1 |   comp2 = comp1.drop(['Unnamed: 0'], axis=1)  comp2.head()   |  | price | speed | hd | ram | screen | cd | multi | premium | ads | trend | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1499 | 25 | 80 | 4 | 14 | 0 | 0 | 1 | 94 | 1 | | 1 | 1795 | 33 | 85 | 2 | 14 | 0 | 0 | 1 | 94 | 1 | | 2 | 1595 | 25 | 170 | 4 | 15 | 0 | 0 | 1 | 94 | 1 | | 3 | 1849 | 25 | 170 | 8 | 14 | 0 | 0 | 0 | 94 | 1 | | 4 | 3295 | 33 | 340 | 16 | 14 | 0 | 0 | 1 | 94 | 1 |   ##Let's verify the EDA  comp2.describe()   |  | **price** | **speed** | **hd** | **ram** | **screen** | **cd** | **multi** | **premium** | **ads** | **trend** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | count | 6259.000000 | 6259.000000 | 6259.000000 | 6259.000000 | 6259.000000 | 6259.000000 | 6259.000000 | 6259.000000 | 6259.000000 | 6259.000000 | | mean | 2219.576610 | 52.011024 | 416.601694 | 8.286947 | 14.608723 | 0.464611 | 0.139479 | 0.902221 | 221.301007 | 15.926985 | | std | 580.803956 | 21.157735 | 258.548445 | 5.631099 | 0.905115 | 0.498786 | 0.346474 | 0.297040 | 74.835284 | 7.873984 | | min | 949.000000 | 25.000000 | 80.000000 | 2.000000 | 14.000000 | 0.000000 | 0.000000 | 0.000000 | 39.000000 | 1.000000 | | 25% | 1794.000000 | 33.000000 | 214.000000 | 4.000000 | 14.000000 | 0.000000 | 0.000000 | 1.000000 | 162.500000 | 10.000000 | | 50% | 2144.000000 | 50.000000 | 340.000000 | 8.000000 | 14.000000 | 0.000000 | 0.000000 | 1.000000 | 246.000000 | 16.000000 | | 75% | 2595.000000 | 66.000000 | 528.000000 | 8.000000 | 15.000000 | 1.000000 | 0.000000 | 1.000000 | 275.000000 | 21.500000 | | max | 5399.000000 | 100.000000 | 2100.000000 | 32.000000 | 17.000000 | 1.000000 | 1.000000 | 1.000000 | 339.000000 | 35.000000 |   ###let's draw a pirplot among all the input variables vs output variable.  import seaborn as sns  sns.pairplot(comp2.iloc[:,:])  /var/folders/kv/w79zffc14fd2hj518gqdhnmc0000gn/T/com.microsoft.Word/Content.MSO/B8DF2A7F.tmp  ##finding the correlation coefficient  comp2.corr()   |  | **price** | **speed** | **hd** | **ram** | **screen** | **cd** | **multi** | **premium** | **ads** | **trend** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | price | 1.000000 | 0.300976 | 0.430258 | 0.622748 | 0.296041 | 0.197343 | -0.016651 | -0.080696 | 0.054540 | -0.199987 | | speed | 0.300976 | 1.000000 | 0.372304 | 0.234760 | 0.189074 | 0.258260 | 0.084172 | 0.114208 | -0.215232 | 0.405438 | | hd | 0.430258 | 0.372304 | 1.000000 | 0.777726 | 0.232802 | 0.503570 | 0.092805 | 0.196924 | -0.323222 | 0.577790 | | ram | 0.622748 | 0.234760 | 0.777726 | 1.000000 | 0.208954 | 0.438504 | 0.045497 | 0.197145 | -0.181670 | 0.276844 | | screen | 0.296041 | 0.189074 | 0.232802 | 0.208954 | 1.000000 | 0.129488 | -0.001740 | 0.018745 | -0.093919 | 0.188614 | | cd | 0.197343 | 0.258260 | 0.503570 | 0.438504 | 0.129488 | 1.000000 | 0.432179 | 0.216077 | -0.061091 | 0.445780 | | multi | -0.016651 | 0.084172 | 0.092805 | 0.045497 | -0.001740 | 0.432179 | 1.000000 | 0.124775 | -0.030394 | 0.210907 | | premium | -0.080696 | 0.114208 | 0.196924 | 0.197145 | 0.018745 | 0.216077 | 0.124775 | 1.000000 | -0.152023 | 0.042107 | | ads | 0.054540 | -0.215232 | -0.323222 | -0.181670 | -0.093919 | -0.061091 | -0.030394 | -0.152023 | 1.000000 | -0.318553 | | trend | -0.199987 | 0.405438 | 0.577790 | 0.276844 | 0.188614 | 0.445780 | 0.210907 | 0.042107 | -0.318553 | 1.000000 |   ###Preparing model with all variables  import statsmodels.formula.api as smf  model1 = smf.ols('price~speed+hd+ram+screen+cd+multi+premium+ads+trend',data=comp2).fit()  model1.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | **Dep. Variable:** | price | **R-squared:** | 0.776 | | **Model:** | OLS | **Adj. R-squared:** | 0.775 | | **Method:** | Least Squares | **F-statistic:** | 2399. | | **Date:** | Fri, 13 Mar 2020 | **Prob (F-statistic):** | 0.00 | | **Time:** | 02:59:50 | **Log-Likelihood:** | -44039. | | **No. Observations:** | 6259 | **AIC:** | 8.810e+04 | | **Df Residuals:** | 6249 | **BIC:** | 8.817e+04 | | **Df Model:** | 9 |  |  | | **Covariance Type:** | nonrobust |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** | | **Intercept** | 307.9880 | 60.353 | 5.103 | 0.000 | 189.675 | 426.301 | | **speed** | 9.3203 | 0.185 | 50.364 | 0.000 | 8.958 | 9.683 | | **hd** | 0.7818 | 0.028 | 28.311 | 0.000 | 0.728 | 0.836 | | **ram** | 48.2560 | 1.066 | 45.265 | 0.000 | 46.166 | 50.346 | | **screen** | 123.0890 | 3.999 | 30.776 | 0.000 | 115.249 | 130.929 | | **cd** | 60.9167 | 9.516 | 6.402 | 0.000 | 42.263 | 79.571 | | **multi** | 104.3238 | 11.413 | 9.141 | 0.000 | 81.951 | 126.697 | | **premium** | -509.2247 | 12.342 | -41.259 | 0.000 | -533.420 | -485.030 | | **ads** | 0.6573 | 0.051 | 12.809 | 0.000 | 0.557 | 0.758 | | **trend** | -51.8496 | 0.629 | -82.470 | 0.000 | -53.082 | -50.617 |  |  |  |  |  | | --- | --- | --- | --- | | **Omnibus:** | 1014.821 | **Durbin-Watson:** | 1.939 | | **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 3190.887 | | **Skew:** | 0.832 | **Prob(JB):** | 0.00 | | **Kurtosis:** | 6.077 | **Cond. No.** | 9.16e+03 |   Warnings: [1] Standard Errors assume that the covariance matrix of the errors is correctly specified. [2] The condition number is large, 9.16e+03. This might indicate that there are strong multicollinearity or other numerical problems.  ###calculating VIF values for speed  rsq\_sp = smf.ols('speed~hd+ram+screen+cd+multi+premium+ads+trend',data=comp2).fit().rsquared  vif\_sp= 1/(1-rsq\_sp)  print(vif\_sp)  1.2653637174997174  ###calculating VIF values for hd  rsq\_hd = smf.ols('hd~speed+ram+screen+cd+multi+premium+ads+trend',data=comp2).fit().rsquared  vif\_hd= 1/(1-rsq\_hd)  print(vif\_hd)  4.207394801604396  ###calculating VIF values for ram  rsq\_rm = smf.ols('ram~hd+speed+screen+cd+multi+premium+ads+trend',data=comp2).fit().rsquared  vif\_rm= 1/(1-rsq\_rm)  print(vif\_rm)  2.9746283364519175  ###calculating VIF values for Screen  rsq\_sc = smf.ols('screen~hd+ram+speed+cd+multi+premium+ads+trend',data=comp2).fit().rsquared  vif\_sc= 1/(1-rsq\_sc)  print(vif\_sc)  1.0816435677723715  ###calculating VIF values for CD  rsq\_cd = smf.ols('cd~hd+ram+screen+speed+multi+premium+ads+trend',data=comp2).fit().rsquared  vif\_cd= 1/(1-rsq\_cd)  print(vif\_cd)  1.859370474832228  ###calculating VIF values for multi  rsq\_ml = smf.ols('multi~hd+ram+screen+cd+speed+premium+ads+trend',data=comp2).fit().rsquared  vif\_ml= 1/(1-rsq\_ml)  print(vif\_ml)  1.2905684545693232  ###calculating VIF values for premium  rsq\_pm = smf.ols('premium~hd+ram+screen+cd+multi+speed+ads+trend',data=comp2).fit().rsquared  vif\_pm= 1/(1-rsq\_pm)  print(vif\_pm)  1.1093879133259468  ###calculating VIF values for ads  rsq\_ad = smf.ols('ads~hd+ram+screen+cd+multi+premium+speed+trend',data=comp2).fit().rsquared  vif\_ad= 1/(1-rsq\_ad)  print(vif\_ad)  1.2172178284996034  ###calculating VIF values for trend  rsq\_tr = smf.ols('trend~hd+ram+screen+cd+multi+premium+ads+speed',data=comp2).fit().rsquared  vif\_tr= 1/(1-rsq\_tr)  print(vif\_tr)  2.022789919481578  ###Since VIF for all variables is below 5, we can say that the input variables are not collinear.  ####So, we don't need to drop any variable. All are significant here.  ####Let's build the model using transformations to get better R^2  ####Let's go for 7th degree polynomial since we already checked in R.  model2 = smf.ols('price~speed+I(speed^2)+I(speed^3)+I(speed^4)+I(speed^5)+I(speed^6)+I(speed^7)+'  'hd+I(hd^2)+I(hd^3)+I(hd^4)+I(hd^5)+I(hd^6)+I(hd^7)+ram+I(ram^2)+I(ram^3)+I(ram^4)+'  'I(ram^5)+I(ram^6)+I(ram^7)+screen+I(screen^2)+I(screen^3)+I(screen^4)+I(screen^5)+'  'I(screen^6)+I(screen^7)+cd+I(cd^2)+I(cd^3)+I(cd^4)+I(cd^5)+I(cd^6)+I(cd^7)+multi+'  'I(multi^2)+I(multi^3)+I(multi^4)+I(multi^5)+I(multi^6)+I(multi^7)+premium+I(premium^2)+'  'I(premium^3)+I(premium^4)+I(premium^5)+I(premium^6)+I(premium^7)+ads+I(ads^2)+I(ads^3)+'  'I(ads^4)+I(ads^5)+I(ads^6)+I(ads^7)+trend+I(trend^2)+I(trend^3)+I(trend^4)+I(trend^5)+'  'I(trend^6)+I(trend^7)',data=comp2).fit()  model2.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | **Dep. Variable:** | price | **R-squared:** | 0.802 | | **Model:** | OLS | **Adj. R-squared:** | 0.801 | | **Method:** | Least Squares | **F-statistic:** | 1051. | | **Date:** | Fri, 13 Mar 2020 | **Prob (F-statistic):** | 0.00 | | **Time:** | 03:25:49 | **Log-Likelihood:** | -43649. | | **No. Observations:** | 6259 | **AIC:** | 8.735e+04 | | **Df Residuals:** | 6234 | **BIC:** | 8.752e+04 | | **Df Model:** | 24 |  |  | | **Covariance Type:** | nonrobust |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** | | **Intercept** | 1.7654 | 0.083 | 21.347 | 0.000 | 1.603 | 1.928 | | **speed** | -32.4259 | 6.910 | -4.693 | 0.000 | -45.971 | -18.880 | | **I(speed ^ 2)** | -17.1458 | 2.535 | -6.763 | 0.000 | -22.116 | -12.176 | | **I(speed ^ 3)** | 16.8044 | 4.177 | 4.023 | 0.000 | 8.616 | 24.993 | | **I(speed ^ 4)** | -13.7250 | 3.885 | -3.533 | 0.000 | -21.340 | -6.110 | | **I(speed ^ 5)** | 20.2251 | 2.824 | 7.162 | 0.000 | 14.689 | 25.761 | | **I(speed ^ 6)** | 1.5551 | 3.424 | 0.454 | 0.650 | -5.157 | 8.267 | | **I(speed ^ 7)** | 35.5053 | 6.641 | 5.347 | 0.000 | 22.487 | 48.523 | | **hd** | 15.2069 | 1.229 | 12.369 | 0.000 | 12.797 | 17.617 | | **I(hd ^ 2)** | -1.3339 | 1.525 | -0.875 | 0.382 | -4.323 | 1.655 | | **I(hd ^ 3)** | 1.0154 | 2.262 | 0.449 | 0.654 | -3.419 | 5.450 | | **I(hd ^ 4)** | 3.5803 | 1.593 | 2.248 | 0.025 | 0.458 | 6.703 | | **I(hd ^ 5)** | 5.9297 | 2.275 | 2.607 | 0.009 | 1.471 | 10.389 | | **I(hd ^ 6)** | -12.9605 | 2.330 | -5.562 | 0.000 | -17.528 | -8.393 | | **I(hd ^ 7)** | -10.6112 | 1.649 | -6.436 | 0.000 | -13.843 | -7.379 | | **ram** | 9.6464 | 2.437 | 3.959 | 0.000 | 4.870 | 14.423 | | **I(ram ^ 2)** | 8.8164 | 0.947 | 9.311 | 0.000 | 6.960 | 10.673 | | **I(ram ^ 3)** | 10.5818 | 0.941 | 11.245 | 0.000 | 8.737 | 12.427 | | **I(ram ^ 4)** | 3.9103 | 1.449 | 2.698 | 0.007 | 1.069 | 6.751 | | **I(ram ^ 5)** | 5.6757 | 1.453 | 3.907 | 0.000 | 2.828 | 8.524 | | **I(ram ^ 6)** | 3.0803 | 1.714 | 1.797 | 0.072 | -0.280 | 6.441 | | **I(ram ^ 7)** | 4.8457 | 1.714 | 2.828 | 0.005 | 1.486 | 8.205 | | **screen** | 23.2339 | 1.365 | 17.026 | 0.000 | 20.559 | 25.909 | | **I(screen ^ 2)** | 15.2340 | 1.110 | 13.722 | 0.000 | 13.058 | 17.410 | | **I(screen ^ 3)** | 15.4939 | 1.422 | 10.895 | 0.000 | 12.706 | 18.282 | | **I(screen ^ 4)** | 7.2341 | 1.040 | 6.953 | 0.000 | 5.194 | 9.274 | | **I(screen ^ 5)** | 7.4940 | 1.270 | 5.902 | 0.000 | 5.005 | 9.983 | | **I(screen ^ 6)** | -0.7658 | 1.188 | -0.644 | 0.519 | -3.095 | 1.564 | | **I(screen ^ 7)** | -0.5059 | 1.297 | -0.390 | 0.696 | -3.048 | 2.036 | | **cd** | 10.0230 | 1.328 | 7.548 | 0.000 | 7.420 | 12.626 | | **I(cd ^ 2)** | 13.5538 | 1.365 | 9.932 | 0.000 | 10.879 | 16.229 | | **I(cd ^ 3)** | -4.7268 | 1.310 | -3.607 | 0.000 | -7.296 | -2.158 | | **I(cd ^ 4)** | 17.0847 | 1.420 | 12.033 | 0.000 | 14.301 | 19.868 | | **I(cd ^ 5)** | -1.1960 | 1.325 | -0.903 | 0.367 | -3.793 | 1.401 | | **I(cd ^ 6)** | 20.6155 | 1.491 | 13.823 | 0.000 | 17.692 | 23.539 | | **I(cd ^ 7)** | 2.3348 | 1.359 | 1.718 | 0.086 | -0.330 | 4.999 | | **multi** | 14.1674 | 1.543 | 9.181 | 0.000 | 11.142 | 17.192 | | **I(multi ^ 2)** | 17.6982 | 1.562 | 11.327 | 0.000 | 14.635 | 20.761 | | **I(multi ^ 3)** | -8.8711 | 1.547 | -5.735 | 0.000 | -11.904 | -5.839 | | **I(multi ^ 4)** | 21.2290 | 1.599 | 13.277 | 0.000 | 18.095 | 24.363 | | **I(multi ^ 5)** | -5.3403 | 1.571 | -3.398 | 0.001 | -8.421 | -2.260 | | **I(multi ^ 6)** | 24.7598 | 1.651 | 14.996 | 0.000 | 21.523 | 27.997 | | **I(multi ^ 7)** | -1.8095 | 1.613 | -1.122 | 0.262 | -4.971 | 1.352 | | **premium** | -77.7624 | 1.735 | -44.827 | 0.000 | -81.163 | -74.362 | | **I(premium ^ 2)** | -74.2315 | 1.719 | -43.184 | 0.000 | -77.601 | -70.862 | | **I(premium ^ 3)** | 83.0586 | 1.787 | 46.477 | 0.000 | 79.555 | 86.562 | | **I(premium ^ 4)** | -70.7007 | 1.719 | -41.129 | 0.000 | -74.071 | -67.331 | | **I(premium ^ 5)** | 86.5894 | 1.840 | 47.064 | 0.000 | 82.983 | 90.196 | | **I(premium ^ 6)** | -67.1699 | 1.735 | -38.716 | 0.000 | -70.571 | -63.769 | | **I(premium ^ 7)** | 90.1202 | 1.906 | 47.294 | 0.000 | 86.385 | 93.856 | | **ads** | 1.1637 | 0.908 | 1.282 | 0.200 | -0.616 | 2.943 | | **I(ads ^ 2)** | -1.3790 | 1.398 | -0.986 | 0.324 | -4.120 | 1.362 | | **I(ads ^ 3)** | 3.9568 | 1.458 | 2.713 | 0.007 | 1.098 | 6.816 | | **I(ads ^ 4)** | -2.1508 | 0.970 | -2.216 | 0.027 | -4.053 | -0.248 | | **I(ads ^ 5)** | 3.1851 | 1.996 | 1.595 | 0.111 | -0.728 | 7.098 | | **I(ads ^ 6)** | -4.6935 | 1.881 | -2.495 | 0.013 | -8.381 | -1.006 | | **I(ads ^ 7)** | 0.6423 | 1.125 | 0.571 | 0.568 | -1.563 | 2.847 | | **trend** | -28.9290 | 1.032 | -28.043 | 0.000 | -30.951 | -26.907 | | **I(trend ^ 2)** | -8.5916 | 0.982 | -8.746 | 0.000 | -10.517 | -6.666 | | **I(trend ^ 3)** | -9.2475 | 1.400 | -6.605 | 0.000 | -11.992 | -6.503 | | **I(trend ^ 4)** | -10.5718 | 1.063 | -9.942 | 0.000 | -12.656 | -8.487 | | **I(trend ^ 5)** | -11.2277 | 1.431 | -7.844 | 0.000 | -14.034 | -8.422 | | **I(trend ^ 6)** | 9.7656 | 1.478 | 6.608 | 0.000 | 6.868 | 12.663 | | **I(trend ^ 7)** | 9.1097 | 1.087 | 8.384 | 0.000 | 6.980 | 11.240 |  |  |  |  |  | | --- | --- | --- | --- | | **Omnibus:** | 979.472 | **Durbin-Watson:** | 1.948 | | **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 2767.401 | | **Skew:** | 0.837 | **Prob(JB):** | 0.00 | | **Kurtosis:** | 5.795 | **Cond. No.** | 4.53e+17 |   Warnings: [1] Standard Errors assume that the covariance matrix of the errors is correctly specified. [2] The smallest eigenvalue is 5.95e-26. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.  ###Predict the values and split the data  ##Splitting the data into train and test  from sklearn.model\_selection import train\_test\_split  cm\_train,cm\_test = train\_test\_split(comp2,test\_size=0.3) ##30% of test data  cm\_train.head()   |  | **price** | **speed** | **hd** | **ram** | **screen** | **cd** | **multi** | **premium** | **ads** | **trend** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4498 | 1778 | 33 | 340 | 4 | 17 | 0 | 0 | 1 | 205 | 21 | | 2985 | 2490 | 33 | 528 | 16 | 14 | 1 | 0 | 1 | 267 | 15 | | 4749 | 2593 | 50 | 528 | 8 | 14 | 1 | 1 | 1 | 162 | 22 | | 4633 | 1795 | 66 | 340 | 4 | 14 | 1 | 0 | 1 | 205 | 21 | | 6129 | 2540 | 100 | 1000 | 16 | 17 | 1 | 0 | 1 | 51 | 32 |   ##Preparing the model on train data  model\_train = smf.ols('price~speed+hd+ram+screen+cd+multi+premium+ads+trend', data=cm\_train).fit()  ###Train data prediction  train\_pred = model\_train.predict(cm\_train)  ###Finding train Risedual values  train\_resid = train\_pred - cm\_train.price  **###rmse value for train data**  train\_rmse = np.sqrt (np.mean(train\_resid \* train\_resid))  train\_rmse  275.76381072686615  ###Prediction on test data  test\_pred = model\_train.predict(cm\_test)  ###Finding train Risedual values  test\_resid = test\_pred - cm\_test.price  **###rmse value for train data**  test\_rmse = np.sqrt (np.mean(test\_resid \* test\_resid))  test\_rmse  273.9595571755634 |