To predict computer sales

BUSSINESS PROBLEM: To predict a model for computer sales

|  |  |  |
| --- | --- | --- |
| price:  HISTOGRAM | Data is right skewed | |
| BOX-PLOT | outliers are present in upper extreme | |
| QQPLOT | Shows data is not normally distributed  After transformation data are normally distributed | |
| > skewness(comp\_sales$price)  [1] 0.7115542  > kurtosis(comp\_sales$price)  [1] 3.728875 | | right skewed because value is greater than 0  leptokurtic, due to its peakedness and data is more distributed in  tails |

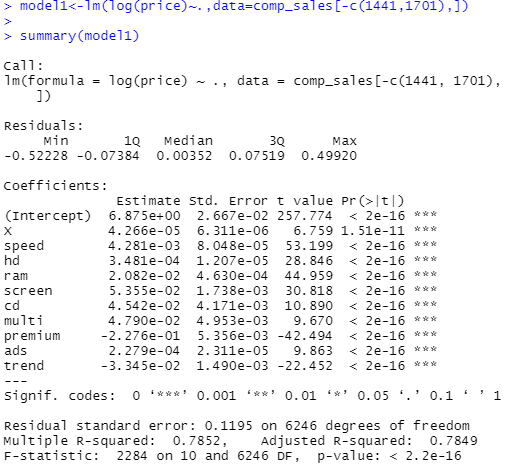
|  |  |  |
| --- | --- | --- |
| SPEED:  HISTOGRAM | Data shows multi model | |
| BOX-PLOT | No outliers ,and data is right skewed | |
| QQPLOT | Data is not normally distributed | |
| skewness(comp\_sales$speed)  [1] 0.6568505  > kurtosis(comp\_sales$speed)  [1] 2.723809 | | right skewed because value is greater than 0  leptokurtic, due to its peakedness and data is more distributed in  tails |

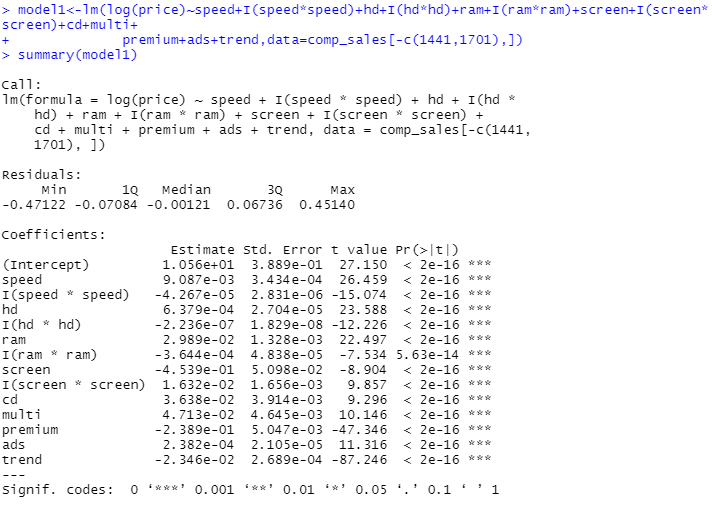
|  |  |  |
| --- | --- | --- |
| RAM  HISTOGRAM | Data is right skewed | |
| BOX-PLOT | outliers present ,data is more concentrated in upper quartile range | |
| QQPLOT | Data is not normally distributed | |
| skewness(comp\_sales$ram)  [1] 1.38587  > kurtosis(comp\_sales$ram)  [1] 4.460124 | | right skewed because value is greater than 0  leptokurtic, due to its peakedness and data is more distributed in  tails |

|  |  |  |
| --- | --- | --- |
| PROFITS:  HISTOGRAM | Data is right skewed | |
| BOX-PLOT | Outliers present in upper extreme ,and it has more variance in upperwhisker and data is right skewed | |
| QQPLOT | Data is not normally distributed | |
| skewness(comp\_sales$hd)  [1] 1.377689  > kurtosis(comp\_sales$hd)  [1] 5.449539 | | right skewed because value is greater than 0  leptokurtic and data is more distributed in tails |

MULTI-VARIET ANALYSIS

|  |
| --- |
|  |
| cor(comp\_sales)  X price speed hd ram screen  X 1.00000000 -0.19992353 0.38856624 0.55549224 0.26537590 0.184699500  price -0.19992353 1.00000000 0.30097646 0.43025779 0.62274824 0.296041474  speed 0.38856624 0.30097646 1.00000000 0.37230410 0.23476050 0.189074122  hd 0.55549224 0.43025779 0.37230410 1.00000000 0.77772630 0.232801530  ram 0.26537590 0.62274824 0.23476050 0.77772630 1.00000000 0.208953740  screen 0.18469950 0.29604147 0.18907412 0.23280153 0.20895374 1.000000000  cd 0.45864221 0.19734334 0.25825980 0.50357041 0.43850441 0.129487662  multi 0.21806644 -0.01665139 0.08417193 0.09280483 0.04549689 -0.001740414  premium 0.03736416 -0.08069636 0.11420791 0.19692359 0.19714459 0.018745223  ads -0.27271422 0.05454047 -0.21523206 -0.32322200 -0.18166971 -0.093919429  trend 0.98981842 -0.19998694 0.40543833 0.57779013 0.27684384 0.188614445  cd multi premium ads trend  X 0.45864221 0.218066438 0.03736416 -0.27271422 0.98981842  price 0.19734334 -0.016651388 -0.08069636 0.05454047 -0.19998694  speed 0.25825980 0.084171934 0.11420791 -0.21523206 0.40543833  hd 0.50357041 0.092804830 0.19692359 -0.32322200 0.57779013  ram 0.43850441 0.045496894 0.19714459 -0.18166971 0.27684384  screen 0.12948766 -0.001740414 0.01874522 -0.09391943 0.18861444  cd 1.00000000 0.432179298 0.21607660 -0.06109108 0.44578018  multi 0.43217930 1.000000000 0.12477474 -0.03039426 0.21090743  premium 0.21607660 0.124774741 1.00000000 -0.15202274 0.04210738  ads -0.06109108 -0.030394260 -0.15202274 1.00000000 -0.31855251  trend 0.44578018 0.210907431 0.04210738 -0.31855251 1.00000000 |
| cor2pcor(cor(comp\_sales))  [,1] [,2] [,3] [,4] [,5] [,6]  [1,] 1.00000000 0.10001454 -0.10913556 -0.1428885 -0.01978897 -0.02293256  [2,] 0.10001454 1.00000000 0.54235426 0.3463231 0.49629943 0.36317631  [3,] -0.10913556 0.54235426 1.00000000 -0.1182390 -0.28479266 -0.11670216  [4,] -0.14288854 0.34632308 -0.11823898 1.0000000 0.42422816 -0.10267514  [5,] -0.01978897 0.49629943 -0.28479266 0.4242282 1.00000000 -0.12000744  [6,] -0.02293256 0.36317631 -0.11670216 -0.1026751 -0.12000744 1.00000000  [7,] 0.13224079 0.06638293 0.00624314 0.1044657 0.10660806 -0.02136852  [8,] -0.02350856 0.11661356 -0.07591482 -0.1262570 -0.10320746 -0.07476176  [9,] 0.05542536 -0.46521261 0.29666076 0.2118630 0.25217440 0.14460312  [10,] 0.25027844 0.12905000 -0.10778794 -0.1576715 -0.04165688 -0.05895304  [11,] 0.96718291 -0.27923776 0.23308673 0.2771775 0.05952205 0.09880704  [,7] [,8] [,9] [,10] [,11]  [1,] 0.13224079 -0.02350856 0.05542536 0.25027844 0.96718291  [2,] 0.06638293 0.11661356 -0.46521261 0.12905000 -0.27923776  [3,] 0.00624314 -0.07591482 0.29666076 -0.10778794 0.23308673  [4,] 0.10446568 -0.12625698 0.21186303 -0.15767153 0.27717749  [5,] 0.10660806 -0.10320746 0.25217440 -0.04165688 0.05952205  [6,] -0.02136852 -0.07476176 0.14460312 -0.05895304 0.09880704  [7,] 1.00000000 0.39952523 0.14670004 0.12468585 -0.07394534  [8,] 0.39952523 1.00000000 0.11022706 -0.04235615 0.05869848  [9,] 0.14670004 0.11022706 1.00000000 -0.05922021 -0.16098926  [10,] 0.12468585 -0.04235615 -0.05922021 1.00000000 -0.24252183  [11,] -0.07394534 0.05869848 -0.16098926 -0.24252183 1.00000000 |
| INFERENCE  Ram and price are moderately correlated, screen and multi are weak negatively correlated ,speed ,hd , screen are weak positively correlated  MODEL-BUILDING |





R AND D SHOWS MORE SIGFICIANCE

LINE ASSUMPTIONS of model2

|  |  |
| --- | --- |
|  | Residuals are dependent of each other,  (no auto-correlation )  And residuals are symmetrically distributed, hence homoscedasticity in nature |
|  | Residuals are normally distributed |
|  | VIF values are <10,hence the input variables are independent of each other  ( No multi-collinearity) |
|  | Residuals have equal variance |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MODEL | MULTIPL R2 | ADJUSTED R2 | TRAIN ERROR | TEST ERROR | INFERENCE |
| Model1 | 0.785 | 0.784 | 271.1 | 270.4 | Right-fit model |
| MODEL2 | 0.81 | 0.81 | 252.1 | 247.2 | Right-fit model |

PYTHON CODE

import numpy as np

import pandas as pd

import matplotlib.pylab as plt

from scipy import stats

import pylab

import seaborn as sns

import statsmodels.formula.api as smf

import statsmodels.api as sm

comp\_sales=pd.read\_csv("C:/Users/USER/Desktop/Computer\_Data.csv")

comp\_sales.columns

dum1=pd.get\_dummies(comp\_sales.cd)

dum1.columns="cd\_no","cd\_yes"

dum2=pd.get\_dummies(comp\_sales.multi)

dum2.columns="multi\_no","multi\_yes"

dum3=pd.get\_dummies(comp\_sales.premium)

dum3.columns="premium\_no","premium\_yes"

merged\_comp=pd.concat([comp\_sales,dum1,dum2,dum3],axis="columns")

merged\_comp.columns

final=merged\_comp.drop(['Unnamed: 0','cd','multi','premium'],axis="columns")

########univariet analysis#######

####price

plt.hist(final.price)

plt.boxplot(final.price)

measurements = np.random.normal(loc = 20, scale = 5, size=100)

stats.probplot(final.price, dist="norm", plot=pylab)

pylab.show()

####speed

plt.hist(final.speed)

plt.boxplot(final.speed)

measurements = np.random.normal(loc = 20, scale = 5, size=100)

stats.probplot(final.speed, dist="norm", plot=pylab)

pylab.show()

####ram

plt.hist(final.ram)

plt.boxplot(final.ram)

measurements = np.random.normal(loc = 20, scale = 5, size=100)

stats.probplot(final.ram, dist="norm", plot=pylab)

pylab.show()

####hd

plt.hist(final.hd)

plt.boxplot(final.hd)

measurements = np.random.normal(loc = 20, scale = 5, size=100)

stats.probplot(final.hd, dist="norm", plot=pylab)

pylab.show()

#########multi-variet analysis########

sns.pairplot(final)

np.corrcoef(final)

corr = final.dropna().corr()

corr

final.columns

#########

model=smf.ols('price~speed+hd+ram+screen+ads+trend+cd\_no+multi\_no+premium\_no',data=final).fit()

model.summary()

sm.graphics.influence\_plot(model)

plt.show()

#########

final1=final.drop(final.index[1440])

final2=final1.drop(final1.index[1700])

model1=smf.ols('price~speed+I(speed^2)+hd+I(hd^2)+ram+I(ram^2)+screen+I(screen^2)+ads+trend+cd\_no+multi\_no+premium\_no',data=final2).fit()

model1.summary()

######train test error of model

model=smf.ols('price~speed+hd+ram+screen+ads+trend+cd\_no+multi\_no+premium\_no',data=final).fit()

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

final1\_train,final1\_test=train\_test\_split(final,test\_size=0.2)

pred=model.predict(final1\_train)

err=pred-final1\_train.price

merr=np.mean(err\*err)

rmse=np.sqrt(merr)

rmse

pred=model.predict(final1\_test)

err=pred-final1\_test.price

merr=np.mean(err\*err)

rmse=np.sqrt(merr)

rmse

####train error=283 and test=271 and r2 value =77.4

######train test error of model1

model1=smf.ols('price~speed+I(speed^2)+hd+I(hd^2)+ram+I(ram^2)+screen+I(screen^2)+ads+trend+cd\_no+multi\_no+premium\_no',data=final).fit()

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

final1\_train,final1\_test=train\_test\_split(final,test\_size=0.2)

pred=model1.predict(final1\_train)

err=pred-final1\_train.price

merr=np.mean(err\*err)

rmse=np.sqrt(merr)

rmse

pred=model1.predict(final1\_test)

err=pred-final1\_test.price

merr=np.mean(err\*err)

rmse=np.sqrt(merr)

rmse

######train-error=266.77 and test error=274.5###r2=78.6