**Name: - UPENDRA DAMA**

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

**Module: - 7**

**Prepare a prediction model for profit of 50\_startups data**

**Problem Statement: -** Prepare a prediction model for profit of 50\_startups data.

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

|  |
| --- |
| > library(readr)  > X50\_Startups <- read\_csv("Desktop/Digi 360/Module 7/50\_Startups.csv")  Parsed with column specification:  cols(  `R&D Spend` = col\_double(),  Administration = col\_double(),  `Marketing Spend` = col\_double(),  State = col\_character(),  Profit = col\_double()  )  ###Let’s do Exploratory Data Analysis  > summary(X50\_Startups)  R&D Spend Administration Marketing Spend State Profit  Min. : 0 Min. : 51283 Min. : 0 Length:50 Min. : 14681  1st Qu.: 39936 1st Qu.:103731 1st Qu.:129300 Class :character 1st Qu.: 90139  Median : 73051 Median :122700 Median :212716 Mode :character Median :107978  Mean : 73722 Mean :121345 Mean :211025 Mean :112013  3rd Qu.:101603 3rd Qu.:144842 3rd Qu.:299469 3rd Qu.:139766  Max. :165349 Max. :182646 Max. :471784 Max. :192262 |

Here, the mean is greater than the median for R&D Spend so the distribution is **right skewed**.

Similarly, the mean is greater than the median for profit so the distribution is **right skewed**.

Whereas mean is less than median for admin and market spend so the distribution is **left skewed**.

**Relationship: -**

Let’s draw scatter diagram to see the relationship among R&D spend, Admin, market spend, State and Profit.

As state is not numeric and no relation in predicting model for profit. So, let’s assign the dummy variables for State.

|  |
| --- |
| > colnames(X50\_Startups) <- c("rd","ad","mk","st","pf")  > attach(X50\_Startups)  The following objects are masked from X50\_Startups (pos = 3):  ad, mk, pf, rd, st  The following objects are masked from X50\_Startups (pos = 4):  ad, mk, pf, rd, st  ###State is not numeric so let’s assign dummy variables.  > X50\_Startups <- cbind(X50\_Startups,ifelse(X50\_Startups$st=="New York",1,0), ifelse(X50\_Startups$st=="California",1,0), ifelse(X50\_Startups$st=="Florida",1,0))  ###Rename the dummy variable columns  > colnames(X50\_Startups)[6] <- "ny"  > colnames(X50\_Startups)[7] <- "cf"  > colnames(X50\_Startups)[8] <- "fl"  ###Dropping the state column since dummy values are already assigned  > X50\_Startups$st <- NULL  ##Plotting the scatter plot for all the columns  >plot(X50\_Startups)    ##Finding the correlation coefficient for the entire dataset  > cor(X50\_Startups)  rd ad mk pf ny cf fl  rd 1.00000000 0.241955245 0.72424813 0.9729005 0.039068162 -0.14316522 0.10571106  ad 0.24195525 1.000000000 -0.03215388 0.2007166 0.005145226 -0.01547811 0.01049309  mk 0.72424813 -0.032153875 1.00000000 0.7477657 -0.033669800 -0.16887523 0.20568545  pf 0.97290047 0.200716568 0.74776572 1.0000000 0.031367600 -0.14583704 0.11624426  ny 0.03906816 0.005145226 -0.03366980 0.0313676 1.000000000 -0.51515152 -0.49236596  cf -0.14316522 -0.015478106 -0.16887523 -0.1458370 -0.515151515 1.00000000 -0.49236596  fl 0.10571106 0.010493089 0.20568545 0.1162443 -0.492365964 -0.49236596 1.00000000  ##All dummy variables for state not at all useful to or model because state is not numeric . So, we can drop the dummy variables and plot scatter diagram for remaining columns  > X50\_Startups[5:7] <- NULL  > plot(X50\_Startups)    ##Finding correlation coefficient for remaining columns  > cor(X50\_Startups)  rd ad mk pf  rd 1.0000000 0.24195525 0.72424813 0.9729005  ad 0.2419552 1.00000000 -0.03215388 0.2007166  mk 0.7242481 -0.03215388 1.00000000 0.7477657  pf 0.9729005 0.20071657 0.74776572 1.0000000 |

Here we observe the correlation coefficient is 0.97 which is close to 1 between profit and R&D spend which we can clearly see in the scatter plot above.

**Model Building: -**

Now, let’s build the model.

|  |
| --- |
| > #Building the linear model of interest  > model1 <- lm(pf~rd+ad+mk)  > summary(model1)  Call:  lm(formula = pf ~ rd + ad + mk)  Residuals:  Min 1Q Median 3Q Max  -33534 -4795 63 6606 17275  Coefficients:  Estimate Std. Error t value Pr(>|t|)  (Intercept) 5.012e+04 6.572e+03 7.626 1.06e-09 \*\*\*  rd 8.057e-01 4.515e-02 17.846 < 2e-16 \*\*\*  ad -2.682e-02 5.103e-02 -0.526 0.602  mk 2.723e-02 1.645e-02 1.655 0.105  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 9232 on 46 degrees of freedom  Multiple R-squared: 0.9507, Adjusted R-squared: 0.9475  F-statistic: 296 on 3 and 46 DF, p-value: < 2.2e-16 |

So, the linear equation with the above coefficients won’t be considered a good equation because the p value is greater than 0.05 for admin and market spend.

**Model Evolution: -**

Here p value is greater than 0.05 for admin and market spend. So, let’s identify and drop the variable which is colinear with other input variables. After that let’s build the model again with remaining variables to see if we get correct equation with significance p value.

|  |
| --- |
| > #Variance Inflation Factor to identify which is most colinear variable with other input variables.  > library(car)  Loading required package: carData  > vif(model1)  rd ad mk  2.468903 1.175091 2.326773  > #Let's remove admin since it has high p value 0.6 that means no significance of this value for our model.  ##Building the model after dropping the admin variable  > model2 <- lm(pf~rd+mk)  > summary(model2)  Call:  lm(formula = pf ~ rd + mk)  Residuals:  Min 1Q Median 3Q Max  -33645 -4632 -414 6484 17097  Coefficients:  Estimate Std. Error t value Pr(>|t|)  (Intercept) 4.698e+04 2.690e+03 17.464 <2e-16 \*\*\*  rd 7.966e-01 4.135e-02 19.266 <2e-16 \*\*\*  mk 2.991e-02 1.552e-02 1.927 0.06 .  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 9161 on 47 degrees of freedom  Multiple R-squared: 0.9505, Adjusted R-squared: 0.9483  F-statistic: 450.8 on 2 and 47 DF, p-value: < 2.2e-16 |

Here p value is significant so we reject the null hypothesis. That means there is significant correlation between profit vs R&D spend and Market spend.

Here we also can see R-squared value is 0.9505 which is greater than 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.

|  |
| --- |
| > ###let's partition the dataset  > n=nrow(X50\_Startups)  > n1=n\*0.7  > n2=n-n1  > train=sample(1:n,n1)  > train  [1] 1 40 24 10 19 50 11 14 26 34 29 31 7 18 30 42 16 5 45 9 17 39 12 36 32 22 38 44  [29] 25 28 4 35 13 27 43  > test=X50\_Startups[-train,]  > test  # A tibble: 15 x 4  rd ad mk pf  <dbl> <dbl> <dbl> <dbl>  1 162598. 151378. 443899. 191792.  2 153442. 101146. 407935. 191050.  3 131877. 99815. 362861. 156991.  4 130298. 145530. 323877. 155753.  5 119943. 156547. 256513. 132603.  6 86420. 153514. 0 122777.  7 76254. 113867. 298664. 118474.  8 73995. 122783. 303319. 110352.  9 63409. 129220. 46085. 97428.  10 28664. 127056. 201127. 90708.  11 28754. 118546. 172796. 78240.  12 1000. 124153. 1904. 64926.  13 1315. 115816. 297114. 49491.  14 0 135427. 0 42560.  15 542. 51743. 0 35673.  ##Let’s predict the values with model2.  > pred=predict(model2,newdata = test)  > actual=test$pf  ##Finding the errors  > error=actual-pred  > error  1 2 3 4 5 6 7 8  2017.401 9645.012 -5888.191 -4703.139 -17589.842 6960.442 1823.138 -4638.135  9 10 11 12 13 14 15  -1436.822 14883.956 3190.854 17096.506 -7419.051 -4416.134 -11734.243  ###Finding the accuracy  > test.rmse=sqrt(mean(error\*\*2))  > test.rmse  [1] 8712.105  > train.rmse=sqrt(mean(model2$residuals\*\*2))  > train.rmse  [1] 8881.886 |

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

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

|  |  |
| --- | --- |
| Model 1 | 0.9507 |
| Model 2 | 0.9505 |

**Let’s do the same in Python: -**

|  |
| --- |
| Python Code |
| import pandas as pd  import numpy as np  startups = pd.read\_csv ("~/desktop/Digi 360/Module 7/50\_Startups.csv")  startups.head()   |  | **R&D Spend** | **Administration** | **Marketing Spend** | **State** | **Profit** | | --- | --- | --- | --- | --- | --- | | 0 | 165349.20 | 136897.80 | 471784.10 | New York | 192261.83 | | 1 | 162597.70 | 151377.59 | 443898.53 | California | 191792.06 | | 2 | 153441.51 | 101145.55 | 407934.54 | Florida | 191050.39 | | 3 | 144372.41 | 118671.85 | 383199.62 | New York | 182901.99 | | 4 | 142107.34 | 91391.77 | 366168.42 | Florida | 166187.94 |   ##Let's verify the EDA  startups.describe()   | **R&D Spend** | **Administration** | **Marketing Spend** | **Profit** | | --- | --- | --- | --- | | count | 50.000000 | 50.000000 | 50.000000 | 50.000000 | | mean | 73721.615600 | 121344.639600 | 211025.097800 | 112012.639200 | | std | 45902.256482 | 28017.802755 | 122290.310726 | 40306.180338 | | min | 0.000000 | 51283.140000 | 0.000000 | 14681.400000 | | 25% | 39936.370000 | 103730.875000 | 129300.132500 | 90138.902500 | | 50% | 73051.080000 | 122699.795000 | 212716.240000 | 107978.190000 | | 75% | 101602.800000 | 144842.180000 | 299469.085000 | 139765.977500 | | max | 165349.200000 | 182645.560000 | 471784.100000 | 192261.830000 |   ##Creating dummy variables for state since state is categorical variable  startups1 = pd.get\_dummies(startups, columns=['State'])  startups1.head()   |  | **R&D Spend** | **Administration** | **Marketing Spend** | **Profit** | **State\_California** | **State\_Florida** | **State\_New York** | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 165349.20 | 136897.80 | 471784.10 | 192261.83 | 0 | 0 | 1 | | 1 | 162597.70 | 151377.59 | 443898.53 | 191792.06 | 1 | 0 | 0 | | 2 | 153441.51 | 101145.55 | 407934.54 | 191050.39 | 0 | 1 | 0 | | 3 | 144372.41 | 118671.85 | 383199.62 | 182901.99 | 0 | 0 | 1 | | 4 | 142107.34 | 91391.77 | 366168.42 | 166187.94 | 0 | 1 | 0 |   ##Renaming the columns  startups2 = startups1.rename(columns ={'R&D Spend' : 'rd', 'Administration' : 'ad', 'Marketing Spend' :'mk', 'Profit' : 'pf',  'State\_California' : 'cf', 'State\_Florida' : 'fl', 'State\_New York' : 'ny'})  startups2.head()   |  | **rd** | **ad** | **mk** | **pf** | **cf** | **fl** | **ny** | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 165349.20 | 136897.80 | 471784.10 | 192261.83 | 0 | 0 | 1 | | 1 | 162597.70 | 151377.59 | 443898.53 | 191792.06 | 1 | 0 | 0 | | 2 | 153441.51 | 101145.55 | 407934.54 | 191050.39 | 0 | 1 | 0 | | 3 | 144372.41 | 118671.85 | 383199.62 | 182901.99 | 0 | 0 | 1 | | 4 | 142107.34 | 91391.77 | 366168.42 | 166187.94 | 0 | 1 | 0 |   ###let's draw a pirplot among all the input variables vs output variable.  import seaborn as sns  sns.pairplot(startups2.iloc[:,:])  /var/folders/kv/w79zffc14fd2hj518gqdhnmc0000gn/T/com.microsoft.Word/Content.MSO/BFEBFCEC.tmp  ##finding the correlation coefficient  startups2.corr()   |  | **rd** | **ad** | **mk** | **pf** | **cf** | **fl** | **ny** | | --- | --- | --- | --- | --- | --- | --- | --- | | rd | 1.000000 | 0.241955 | 0.724248 | 0.972900 | -0.143165 | 0.105711 | 0.039068 | | ad | 0.241955 | 1.000000 | -0.032154 | 0.200717 | -0.015478 | 0.010493 | 0.005145 | | mk | 0.724248 | -0.032154 | 1.000000 | 0.747766 | -0.168875 | 0.205685 | -0.033670 | | pf | 0.972900 | 0.200717 | 0.747766 | 1.000000 | -0.145837 | 0.116244 | 0.031368 | | cf | -0.143165 | -0.015478 | -0.168875 | -0.145837 | 1.000000 | -0.492366 | -0.515152 | | fl | 0.105711 | 0.010493 | 0.205685 | 0.116244 | -0.492366 | 1.000000 | -0.492366 | | ny | 0.039068 | 0.005145 | -0.033670 | 0.031368 | -0.515152 | -0.492366 | 1.000000 |   ###Dropping dummy variables for state since they are not useful for preparing the model.  startups3 = startups2.drop(columns =['fl', 'cf','ny'])  startups3.head()   |  | **rd** | **ad** | **mk** | **pf** | | --- | --- | --- | --- | --- | | 0 | 165349.20 | 136897.80 | 471784.10 | 192261.83 | | 1 | 162597.70 | 151377.59 | 443898.53 | 191792.06 | | 2 | 153441.51 | 101145.55 | 407934.54 | 191050.39 | | 3 | 144372.41 | 118671.85 | 383199.62 | 182901.99 | | 4 | 142107.34 | 91391.77 | 366168.42 | 166187.94 |   ###let's draw a pirplot among remaining input variables vs output variable.  sns.pairplot(startups3.iloc[:,:])  /var/folders/kv/w79zffc14fd2hj518gqdhnmc0000gn/T/com.microsoft.Word/Content.MSO/9682DDA.tmp  ##Let's find the correlation coefficient for remaining varibales  startups3.corr()   |  | **rd** | **ad** | **mk** | **pf** | | --- | --- | --- | --- | --- | | rd | 1.000000 | 0.241955 | 0.724248 | 0.972900 | | ad | 0.241955 | 1.000000 | -0.032154 | 0.200717 | | mk | 0.724248 | -0.032154 | 1.000000 | 0.747766 | | pf | 0.972900 | 0.200717 | 0.747766 | 1.000000 |   ###Preparing model with all variables  import statsmodels.formula.api as smf  model1 = smf.ols('pf~rd+ad+mk',data=startups3).fit()  model1.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | **Dep. Variable:** | pf | **R-squared:** | 0.951 | | **Model:** | OLS | **Adj. R-squared:** | 0.948 | | **Method:** | Least Squares | **F-statistic:** | 296.0 | | **Date:** | Thu, 12 Mar 2020 | **Prob (F-statistic):** | 4.53e-30 | | **Time:** | 17:51:21 | **Log-Likelihood:** | -525.39 | | **No. Observations:** | 50 | **AIC:** | 1059. | | **Df Residuals:** | 46 | **BIC:** | 1066. | | **Df Model:** | 3 |  |  | | **Covariance Type:** | nonrobust |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** | | **Intercept** | 5.012e+04 | 6572.353 | 7.626 | 0.000 | 3.69e+04 | 6.34e+04 | | **rd** | 0.8057 | 0.045 | 17.846 | 0.000 | 0.715 | 0.897 | | **ad** | -0.0268 | 0.051 | -0.526 | 0.602 | -0.130 | 0.076 | | **mk** | 0.0272 | 0.016 | 1.655 | 0.105 | -0.006 | 0.060 |  |  |  |  |  | | --- | --- | --- | --- | | **Omnibus:** | 14.838 | **Durbin-Watson:** | 1.282 | | **Prob(Omnibus):** | 0.001 | **Jarque-Bera (JB):** | 21.442 | | **Skew:** | -0.949 | **Prob(JB):** | 2.21e-05 | | **Kurtosis:** | 5.586 | **Cond. No.** | 1.40e+06 |   Warnings: [1] Standard Errors assume that the covariance matrix of the errors is correctly specified. [2] The condition number is large, 1.4e+06. This might indicate that there are strong multicollinearity or other numerical problems.  ###calculating VIF values for R&D spend  rsq\_rd = smf.ols('rd~ad+mk',data=startups3).fit().rsquared  vif\_rd= 1/(1-rsq\_rd)  print(vif\_rd)  2.4689030699947017  ###calculating VIF values for admin spend  rsq\_ad = smf.ols('ad~rd+mk',data=startups3).fit().rsquared  vif\_ad= 1/(1-rsq\_ad)  print(vif\_ad)  1.1750910070550458  ###calculating VIF values for market spend  rsq\_mk = smf.ols('mk~rd+ad',data=startups3).fit().rsquared  vif\_mk = 1/(1-rsq\_mk)  print(vif\_mk)  2.3267732905308773  ##storing VIF values in a Dataframe  df1 = {'variables' :['rd','ad','mk'],'VIF' :[vif\_rd,vif\_ad,vif\_mk]}  vif\_df = pd.DataFrame(df1)  vif\_df   |  | **variables** | **VIF** | | --- | --- | --- | | 0 | rd | 2.468903 | | 1 | ad | 1.175091 | | 2 | mk | 2.326773 |   ###As ad is having p value high which is greater than 0.05, we are going to drop this and build model with remaining variables  model2 = smf.ols('pf~rd+mk',data=startups3).fit()  model2.summary()   |  |  |  |  | | --- | --- | --- | --- | | OLS Regression Results | | | | | **Dep. Variable:** | pf | **R-squared:** | 0.950 | | **Model:** | OLS | **Adj. R-squared:** | 0.948 | | **Method:** | Least Squares | **F-statistic:** | 450.8 | | **Date:** | Thu, 12 Mar 2020 | **Prob (F-statistic):** | 2.16e-31 | | **Time:** | 18:22:45 | **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]** | | **Intercept** | 4.698e+04 | 2689.933 | 17.464 | 0.000 | 4.16e+04 | 5.24e+04 | | **rd** | 0.7966 | 0.041 | 19.266 | 0.000 | 0.713 | 0.880 | | **mk** | 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 |   Warnings: [1] Standard Errors assume that the covariance matrix of the errors is correctly specified. [2] The condition number is large, 5.32e+05. This might indicate that there are strong multicollinearity or other numerical problems.  ##Splitting the data into train and test  from sklearn.model\_selection import train\_test\_split  st\_train,st\_test = train\_test\_split(startups3,test\_size=0.3) ##30% of test data  st\_train.head()   |  | **rd** | **ad** | **mk** | **pf** | | --- | --- | --- | --- | --- | | 45 | 1000.23 | 124153.04 | 1903.93 | 64926.08 | | 0 | 165349.20 | 136897.80 | 471784.10 | 192261.83 | | 44 | 22177.74 | 154806.14 | 28334.72 | 65200.33 | | 5 | 131876.90 | 99814.71 | 362861.36 | 156991.12 | | 40 | 28754.33 | 118546.05 | 172795.67 | 78239.91 |   ##Preparing the model on train data  model\_train = smf.ols('pf~rd+mk', data=st\_train).fit()  ###Train data prediction  train\_pred = model\_train.predict(st\_train)  ###Finding train Risedual values  train\_resid = train\_pred - st\_train.pf  ###rmse value for train data  train\_rmse = np.sqrt (np.mean(train\_resid \* train\_resid))  train\_rmse  8997.036384748422  ###Prediction on test data  test\_pred = model\_train.predict(st\_test)  ###Finding train Risedual values  test\_resid = test\_pred - st\_test.pf  ###rmse value for train data  test\_rmse = np.sqrt (np.mean(test\_resid \* test\_resid))  test\_rmse  8853.35144008655 |

**The rmse is very close between test and train data so that accuracy is high for our model.**