NAME:- DUBEY KARAN SANJEEV
CLASS:- B.E - 4
ROLL NO:- 04
BATCH:- A

Experiment No-10

AIM : To implement predictive Analytics technique- Regression using R.

THEORY:

The simple linear regression is used to predict a quantitative outcome y on the basis of one single predictor variable x. The goal is to build a mathematical model (or formula) that defines y as a function of the x variable. Once, we built a statistically significant model, it's possible to use it for predicting future outcome on the basis of new x values.

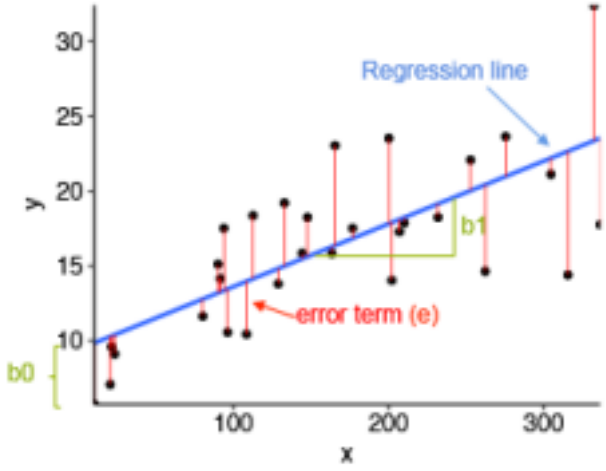
Consider that, we want to evaluate the impact of advertising budgets of three medias (youtube, facebook and newspaper) on future sales. This example of problem can be modeled with linear regression.

The mathematical formula of the linear regression can be written as y = b0 + b1*x + e, where:

- b0 and b1 are known as the regression *beta coefficients* or *parameters*:
  - b0 is the *intercept* of the regression line; that is the predicted value when x = 0.
  - b1 is the *slope* of the regression line.
- e is the *error term* (also known as the *residual errors*), the part of y that can be explained by the regression model

The figure below illustrates the linear regression model, where:

- the best-fit regression line is in blue
- the intercept (b0) and the slope (b1) are shown in green
- the error terms (e) are represented by vertical red lines



From the scatter plot above, it can be seen that not all the data points fall exactly on the fitted regression line. Some of the points are above the blue curve and some are below it; overall, the residual errors (e) have

approximately mean zero.

The sum of the squares of the residual errors are called the Residual Sum of Squares or RSS.
The average variation of points around the fitted regression line is called the Residual Standard Error (RSE). This is one the metrics used to evaluate the overall quality of the fitted regression model. The lower the RSE, the better it is.
Since the mean error term is zero, the outcome variable y can be approximately estimated as follow:

$y \sim b0 + b1*x$

Mathematically, the beta coefficients (b0 and b1) are determined so that the RSS is as minimal as possible. This method of determining the beta coefficients is technically called least squares regression or ordinary least squares (OLS) regression.
Once, the beta coefficients are calculated, a t-test is performed to check whether or not these coefficients are significantly different from zero. A non-zero beta coefficients means that there is a significant relationship between the predictors (x) and the outcome variable (y).

CONCLUSION:

After computing a regression model, a first step is to check whether, at least, one predictor is significantly associated with outcome variables. If one or more predictors are significant, the second step is to assess how well the model fits the data by inspecting the Residuals Standard Error (RSE), the R2 value and the F-statistics. These metrics give the overall quality of the model.

Program formation/ Execution/ ethical practices (06)	**Timely Submission and Documentation (02)**	**Viva Answer (02)**	**Experiment Marks (10)**	**Teacher Signature with date**

Step 1: Create the training and test data

This can be done using the function. Just make sure you set the seed

```
sample(
   )
```

using so the samples can be recreated for future use.

```
set.seed(
    )
```

```
# Create Training and Test data -
set.seed(100) # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(cars), 0.8*nrow(cars)) # row
indices for training data
trainingData <- cars[trainingRowIndex, ] # model training
data testData <- cars[-trainingRowIndex, ] # test data
```

```
dis
 t
```

Step 2: Fit the model on training data and predict on test data

```
# Build the model on training data
lmMod <- lm(dist ~ speed, data=trainingData) # build the
model distPred <- predict(lmMod, testData) # predict distance
```

Step 3: Review diagnostic measures.

```
summary (lmMod) # model summary
#>
#> Call:
#> lm(formula = dist ~ speed, data = trainingData)
#>
#> Residuals:
#> Min 1Q Median 3Q Max
#> -23.350 -10.771 -2.137 9.255 42.231
#>
#> Coefficients:
#> Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -22.657 7.999 -2.833 0.00735 **
#> speed 4.316 0.487 8.863 8.73e-11 ***
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
1 #>
#> Residual standard error: 15.84 on 38 degrees of freedom
#> Multiple R-squared: 0.674, Adjusted R-squared: 0.6654
#> F-statistic: 78.56 on 1 and 38 DF, p-value: 8.734e-11
AIC (lmMod) # Calculate akaike information criterion
#> [1] 338.4489
```

From the model summary, the model p value and predictor's p value are less than the significance level.
So you have a statistically significant model.
Also, the R-Sq and Adj R-Sq are comparative to the original model built on full data.

Step 4: Calculate prediction accuracy and error rates

A simple correlation between the actuals and predicted values can be used as a form of accuracy measure.
A higher correlation accuracy implies that the actuals and predicted values have similar directional movement, i.e. when the actuals values increase the predicted values also increase and vice-versa.

```
actuals_preds <- data.frame(cbind(actuals=testData$dist,
predicteds=distPred)) # make actuals_predicteds
dataframe. correlation_accuracy <- cor(actuals_preds) #
82.7%
head(actuals_preds)
#> actuals predicteds
#> 1 2 -5.392776
#> 4 22 7.555787
#> 8 26 20.504349
#> 20 26 37.769100
#> 26 54 42.085287
#> 31 50 50.717663
```

Now lets calculate the Min Max accuracy and MAPE:
$$MinMaxAccuracy = mean \left(\frac{min\left(actuals, predicteds\right)}{max\left(actuals, predicteds \right)} \right)$$
$$MeanAbsolutePercentageError \ (MAPE) = mean\left(\frac{abs\left(predicteds−actuals\right)}{actuals}\right)$$

```
# Min-Max Accuracy Calculation
min_max_accuracy <- mean(apply(actuals_preds, 1, min) /
apply(actuals_preds, 1, max))
# => 38.00%, min_max accuracy

# MAPE Calculation
mape <- mean(abs((actuals_preds$predicteds -
actuals_preds$actuals))/actuals_preds$actuals)
# => 69.95%, mean absolute percentage deviation
```

Alternately, you can compute all the error metrics in one go using the function in DMwR package. You will have

```
regr.eval(
    )
```

to for this if you are using it for the first time.

```
install.packages('DMwR')
```

```
DMwR::regr.eval(actuals_preds$actuals,
actuals_preds$predicteds) #=> mae mse rmse mape
```

```
#=> 12.0082829 205.9652710 14.3514902 0.6995032
```