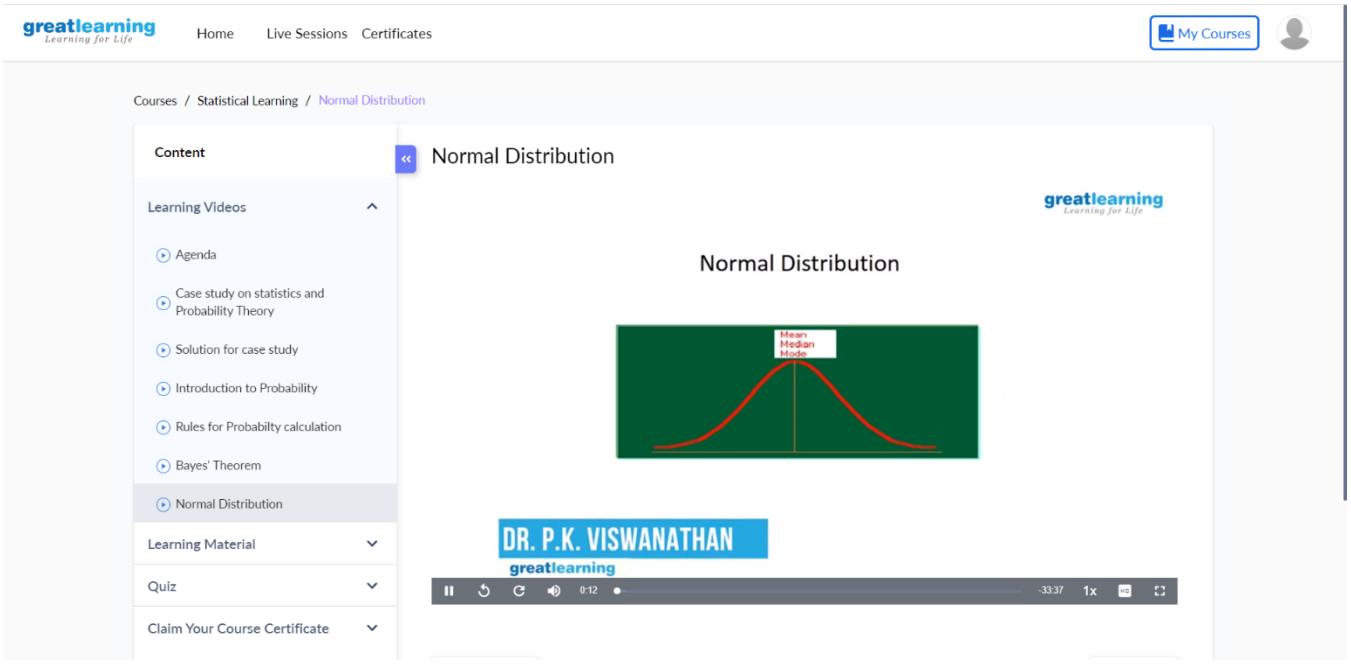
DAILY ASSESSMENT FORMAT

Date:	**17 JUNE 2020**	**Name:**	**PAVITHRAN S**
Course:	**STATISTICAL ANALYSIS**	**USN:**	**4AL17EC068**
Topic:	**STATISTICAL ANALYSIS**	**Semester & Section:**	**6TH B**
Github Repository:	**Pavithran**		

FORENOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

What is Statistical Learning?

Let, suppose that we observe a response Y and p different predictors X = (X_1, X_2,…., Xp). In general, we can say:

$$Y = f(X) + \varepsilon$$

Here **f** is an unknown function, and **ε** is the *random error term*.

In essence, statistical learning refers to a set of approaches for estimating f.

In cases where we have set of X readily available, but the output Y, not so much, the error averages to zero, and we can say:

$$¥ = f(X)$$

where ***f*** represents our estimate of **f** and **¥** represents the resulting prediction.

Hence for a set of predictors X, we can say:

$$E(Y - ¥)^2 = E[f(X) + \varepsilon - f(X)]^2 => E(Y - ¥)^2 = [f(X) - f(X)]^2 + Var(\varepsilon)$$

where,

- $E(Y - ¥)^2$ represents *the expected value* of the squared difference between actual and expected result.
- $[f(X) - f(X)]^2$ represents the ***reducible error***. It is reducible because we can potentially improve the accuracy of f by better modeling.
- $Var(\varepsilon)$ represents the **irreducible error**. It is irreducible because no matter how well we estimate f, we cannot reduce the error introduced by *variance* in ε.

Regression Vs Classification Problem

Variables, Y, can be broadly be characterized as *quantitative* or *qualitative*(also known as *categorical*). Quantitative variables take on numerical values, e.g., age, height, income, price, and much more. Estimating qualitative responses is often termed as *a **regression problem***. Qualitative variables take on categorical values, e.g., gender, brand, parts of speech, and much more. Estimating qualitative responses is often termed as *a **classification problem***.

There is no free lunch in statistics: no one method dominates all other over all possible data sets.

Variance And Bias

Variance refers to the amount by which f would change if we estimated with different training data sets. In general, when we over-fit a model on a given training data set(reducible error in training set is very low but on test set is very high), we get a model that has higher variance since any change in the data points would results in a significantly different model.

Bias refers to the error introduced by approximating a real-life problem, which may be extremely complicated by a much simpler model — for example, modeling non-linear problems with a linear model. In general, when we over-fit, a model on given data set it results in very less bias.

This results in the variance bias trade-off.

As we fit the model over a given data set, the bias tends to decrease faster than the variance increases initially. Consequently, the expected test error(reducible) declines. However, at some point, when over-fitting starts, there is a little impact on the bias, but variance starts to increase rapidly when this happens the test error increases.

Linear Regression

Linear regression is a statistical method belonging to supervised learning used for predicting quantitative responses.

Simple Linear Regression approach predicts a quantitative response ¥ based on a single variable X

assuming a linear relationship. We can say :

$$¥ \approx \beta_0 + \beta_1 X$$

Our job is now to *estimate* β_0 and β_1, the parameters/coefficients of our model based on the training data set, such that the hyperplane(in this case a line) is *close* to the training data set. Many criteria can estimate the closeness, the most common being *least square*.

The sum of the square of the difference between all observed response and the predicted response formulates to *Residual Sum Of Squares(RSS).*

Problems in Linear Regression

- *Non-linearity of the response-predictor relationships.*
- *Correlation of error terms.*
- *The non-constant variance of error terms*.
- *Outliers:* when the actual prediction is very *far* from the estimated one, can arise due to inaccurate recording of data.
- *High-leverage points:* Unusual values of the predictors impact the regression line known as high leverage points.
- *Collinearity:* where two or more predictor variables are closely related to each other, it may be challenging to weed out the individual effect of a single predictor variable**.**

KNN Regression

KNN Regression is a non-parametric approach towards estimating or predicting values, which do not assume the form of $f(X)$. It estimates/predicts $f(x_0)$ where x_0 is a prediction point by averaging out all N_0

responses closest to x_0. We can say:

$$f(x_o) = \frac{1}{K}\sum_{x_i \epsilon N_0} y_i$$

Note: If K is small, the fit would be flexible and any change in the data would result in a different fit, hence for small K the variance would be high and bias low; conversely, if K is large, it might mask some structure in the data hence the bias would be high.

The Classification Problem

The responses as we discussed till now, may not always be *quantitative*, it can be also *qualitative*, predicting these qualitative responses is called classification.

We will discuss various statistical approaches to classification including:

- SVM,*Logistic Regression,KNN Classifier*
- *GAM*
- *Trees*
- *Random Forest*
- *Boosting*

Support Vector Machine(SVM)

SVM or support vector machine is the classifier that maximizes the margin. The goal of a classifier in our example below is to find a line or (n-1) dimension hyper-plane that separates the two classes present in the n-dimensional space. I have written a detailed article explaining the derivation and formulation of SVM. In my opinion, it is one of the most powerful techniques in our tool box of statistical methods in AI.

Logistic Regression

Logistic model models the probability of output response ¥ belonging to a particular category.

We can say:

$$\rho(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

Applying componendo dividendo we get:

$$\Rightarrow \frac{\rho(X)}{1 - \rho(X)} = e^{\beta_0 + \beta_1 X}$$

which is nothing but the odds.

$$\Rightarrow \underbrace{\ln\left(\frac{\rho(X)}{1 - \rho(X)}\right)}_{log - odds \,/\, logit} = \beta_0 + \beta_1 X$$

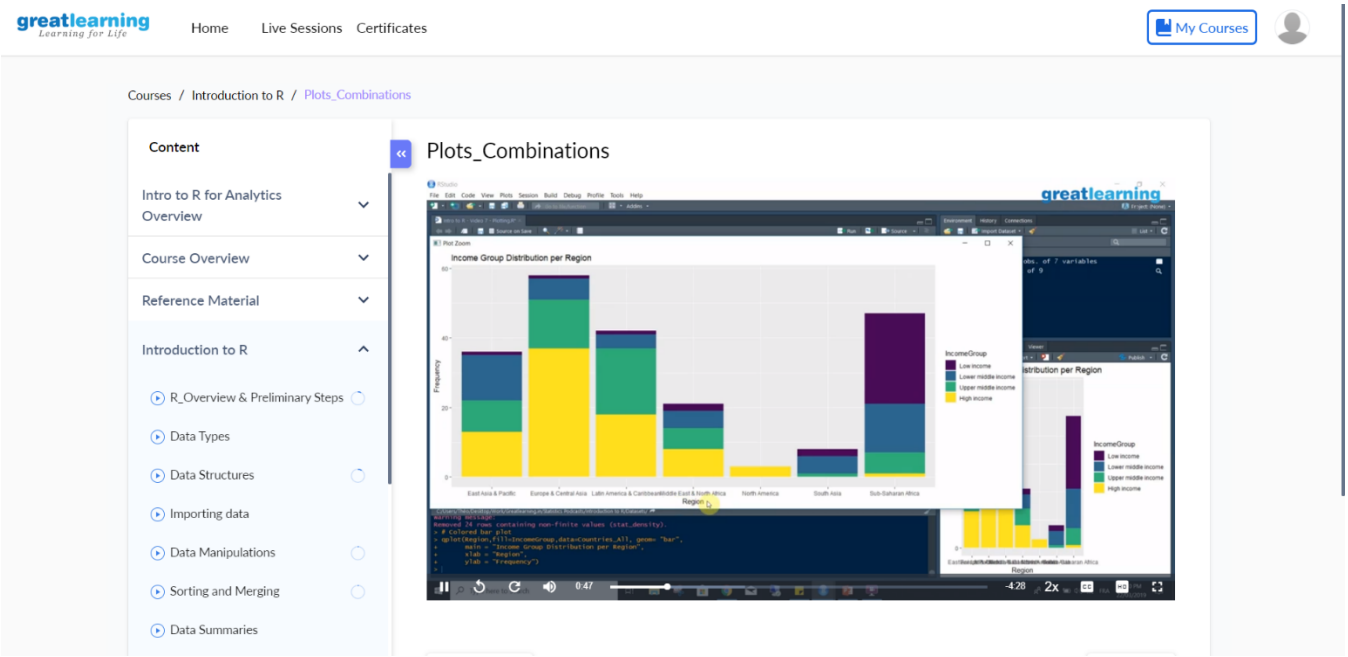
For estimating the beta coefficients, we can use maximum likelihood. The basic idea is to estimate the betas such that the estimated value and observed value of the results are as close as possible. In a binary classification, with observed classes as *1* and *0*, we can say the likelihood function would look like:

$$l(\beta_0, \beta_1) = \prod_{i\,:\,y_i = 1} \hat{p}(x_i) \prod_{\tilde{i}\,:\,y_{\tilde{i}} = 0} (1 - \hat{p}(x_{\tilde{i}}))$$

Date:	**17 JUNE 2020**	**Name:**	**PAVITHRAN S**
Course:	**R PROGRAMMING**	**USN:**	**4AL17EC068**
Topic:	**R PROGRAMMING**	**Semester & Section:**	**6TH B**
Github Repository:	**Pavithran**		

AFTERNOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

Data Reshaping in R is about changing the way data is organized into rows and columns. Most of the time data processing in R is done by taking the input data as a data frame. It is easy to extract data from the rows and columns of a data frame but there are situations when we need the data frame in a format that is different from format in which we received it. R has many functions to split, merge and change the rows to columns and vice-versa in a data frame.

Joining Columns and Rows in a Data Frame

We can join multiple vectors to create a data frame using the **cbind()**function. Also we can merge two data frames using **rbind()** function.

```
# Create vector objects.
city <- c("Tampa","Seattle","Hartford","Denver")
state <- c("FL","WA","CT","CO")
zipcode <- c(33602,98104,06161,80294)

# Combine above three vectors into one data frame.
addresses <- cbind(city,state,zipcode)

# Print a header.
cat("# # # # The First data frame\n")

# Print the data frame.
print(addresses)

# Create another data frame with similar columns
new.address <- data.frame(
   city = c("Lowry","Charlotte"),
   state = c("CO","FL"),
   zipcode = c("80230","33949"),
   stringsAsFactors = FALSE
)

# Print a header.
cat("# # # The Second data frame\n")

# Print the data frame.
print(new.address)

# Combine rows form both the data frames.
all.addresses <- rbind(addresses,new.address)

# Print a header.
cat("# # # The combined data frame\n")

# Print the result.
print(all.addresses)
```

When we execute the above code, it produces the following result −

```
# # # # The First data frame
     city       state zipcode
[1,] "Tampa"    "FL"  "33602"
[2,] "Seattle"  "WA"  "98104"
[3,] "Hartford" "CT"   "6161"
[4,] "Denver"   "CO"  "80294"

# # # The Second data frame
       city       state   zipcode
1      Lowry      CO      80230
2      Charlotte  FL      33949

# # # The combined data frame
       city      state zipcode
1      Tampa     FL    33602
2      Seattle   WA    98104
3      Hartford  CT     6161
4      Denver    CO    80294
5      Lowry     CO    80230
6     Charlotte  FL    33949
```

Merging Data Frames

We can merge two data frames by using the **merge()** function. The data frames must have same column names on which the merging happens.

In the example below, we consider the data sets about Diabetes in Pima Indian Women available in the library names "MASS". we merge the two data sets based on the values of blood pressure("bp") and body mass index("bmi"). On choosing these two columns for merging, the records where values of these two variables match in both data sets are combined together to form a single data frame.

```
library(MASS)
merged.Pima <- merge(x = Pima.te, y = Pima.tr,
   by.x = c("bp", "bmi"),
   by.y = c("bp", "bmi")
)
print(merged.Pima)
nrow(merged.Pima)
```

When we execute the above code, it produces the following result −

```
   bp  bmi npreg.x glu.x skin.x ped.x age.x type.x npreg.y glu.y
skin.y ped.y
1  60 33.8       1   117     23 0.466    27     No       2   125
20 0.088
2  64 29.7       2    75     24 0.370    33     No       2   100
23 0.368
3  64 31.2       5   189     33 0.583    29    Yes       3   158
13 0.295
```

```
4  64 33.2       4   117     27 0.230    24     No       1    96
27 0.289
5  66 38.1       3   115     39 0.150    28     No       1   114
36 0.289
6  68 38.5       2   100     25 0.324    26     No       7   129
49 0.439
7  70 27.4       1   116     28 0.204    21     No       0   124
20 0.254
8  70 33.1       4    91     32 0.446    22     No       9   123
44 0.374
9  70 35.4       9   124     33 0.282    34     No       6   134
23 0.542
10 72 25.6       1   157     21 0.123    24     No       4    99
17 0.294
11 72 37.7       5    95     33 0.370    27     No       6   103
32 0.324
12 74 25.9       9   134     33 0.460    81     No       8   126
38 0.162
13 74 25.9       1    95     21 0.673    36     No       8   126
38 0.162
14 78 27.6       5    88     30 0.258    37     No       6   125
31 0.565
15 78 27.6      10   122     31 0.512    45     No       6   125
31 0.565
16 78 39.4       2   112     50 0.175    24     No       4   112
40 0.236
17 88 34.5       1   117     24 0.403    40    Yes       4   127
11 0.598
   age.y type.y
1     31     No
2     21     No
3     24     No
4     21     No
5     21     No
6     43    Yes
7     36    Yes
8     40     No
9     29    Yes
10    28     No
11    55     No
12    39     No
13    39     No
14    49    Yes
15    49    Yes
16    38     No
17    28     No
[1] 17
```

Melting and Casting

One of the most interesting aspects of R programming is about changing the shape of the

data in multiple steps to get a desired shape. The functions used to do this are called **melt()** and **cast()**.

We consider the dataset called ships present in the library called "MASS".

```
library(MASS)
print(ships)
```

When we execute the above code, it produces the following result −

```
     type year   period   service   incidents
1      A   60      60         127          0
2      A   60      75          63          0
3      A   65      60        1095          3
4      A   65      75        1095          4
5      A   70      60        1512          6
.............
.............
8      A   75      75        2244         11
9      B   60      60       44882         39
10     B   60      75       17176         29
11     B   65      60       28609         58
............
............
17     C   60      60        1179          1
18     C   60      75         552          1
19     C   65      60         781          0
............
............
```

Melt the Data

Now we melt the data to organize it, converting all columns other than type and year into multiple rows.

```
molten.ships <- melt(ships, id = c("type","year"))
print(molten.ships)
```

When we execute the above code, it produces the following result −

```
      type year  variable  value
1      A   60    period      60
2      A   60    period      75
3      A   65    period      60
4      A   65    period      75
............
............
9      B   60    period      60
10     B   60    period      75
11     B   65    period      60
12     B   65    period      75
13     B   70    period      60
...........
```

```
...........
41     A   60    service    127
42     A   60    service     63
43     A   65    service   1095
...........
...........
70     D   70    service   1208
71     D   75    service      0
72     D   75    service   2051
73     E   60    service     45
74     E   60    service      0
75     E   65    service    789
...........
...........
101    C   70    incidents    6
102    C   70    incidents    2
103    C   75    incidents    0
104    C   75    incidents    1
105    D   60    incidents    0
106    D   60    incidents    0
...........
...........
```

Cast the Molten Data

We can cast the molten data into a new form where the aggregate of each type of ship for each year is created. It is done using the **cast()** function.

```
recasted.ship <- cast(molten.ships, type+year~variable,sum)
print(recasted.ship)
```

When we execute the above code, it produces the following result −

```
     type year  period  service  incidents
1      A   60    135       190       0
2      A   65    135      2190       7
3      A   70    135      4865      24
4      A   75    135      2244      11
5      B   60    135     62058      68
6      B   65    135     48979     111
7      B   70    135     20163      56
8      B   75    135      7117      18
9      C   60    135      1731       2
10     C   65    135      1457       1
11     C   70    135      2731       8
12     C   75    135       274       1
13     D   60    135       356       0
14     D   65    135       480       0
15     D   70    135      1557      13
16     D   75    135      2051       4
17     E   60    135        45       0
18     E   65    135      1226      14
```