Module-1: **Cross Validation**

CROSS VALIDATION

Why needed this in model-building..??

> To select no. of hyper-parameters *(or meta-parameters* of a model.

Motivation

- How to select the optimal number of meta or hyper-parameters of a model?
  - Number of principal components in principal components analysis — *Like, no of components to be chosen*
  - Number of clusters in K-means clusterin — *-- How many clusters are to formed..??*
  - Number of terms 'n' in polynomial or nonlinear regression — *What is the highest degree to be chosen?* | For nth degree there will be `n+1` co-efficients

    $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots \beta_\eta x^n$

    (equivalent to multilinear regression by treating x, x^2, …x^n as different variables)

May say that.. With MSE one can obtain the parameters, but...

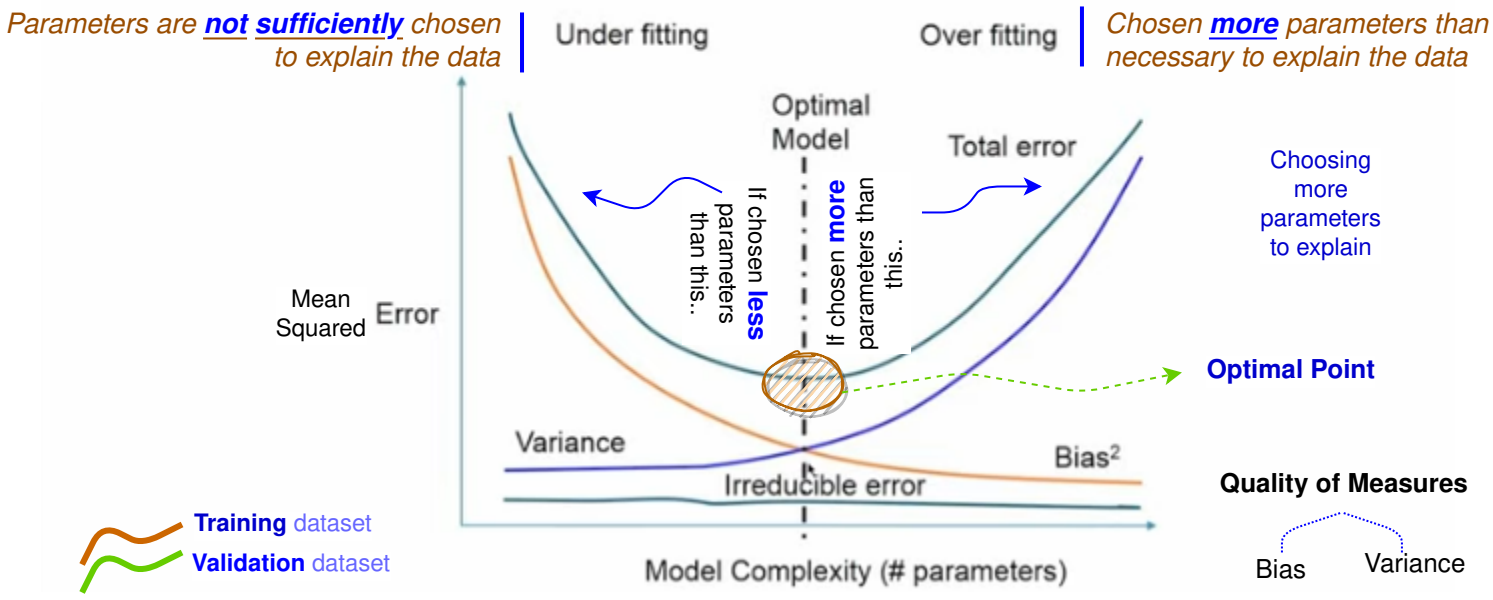
- MSE of training data set not useful as a measure
  - MSE will decrease with increasing number of parameters (can be reduced to zero)
- Use cross validation on a validation data set to determine optimal number of parameters

so, we don't use training dataset to do this, instead on a Validation dataset

Polynomial regression is used as a medium to explain further discussion on this topic..

Schematically..

Bias-Variance trade-off on test data set



So, the value to be chosen is said by: **Minimum value of Validation dataset curve**.

It is actually the trade-off between the Variance and Bias which gives the minimum value.

---------- We're going to find it out now

Training and Validation data sets

- For large data sets divide data set into training data set (~ 70% of the samples) and remaining validation/test data
  - Training set: $\{(x_1, y_1);(x_2, y_2);\ldots;(x_n, y_n)\}$
  - Test set: $(x_{0,i}, y_{0,i})$: i = 1...n_t observations
- Training error rate

  Measured value, **Predicted** value

  $$MSE_{Training} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\hat{\boldsymbol{\beta}})^2$$ ← Not of our interest for predictive ability of the model
- Test error rates

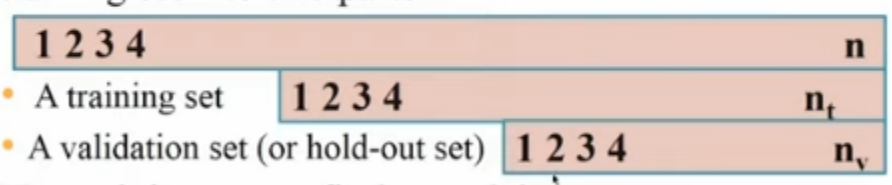
  $$MSE_{Test} = \frac{1}{n_t}\sum_{i=1}^{n}(y_{0,i} - \mathbf{x}_{0,i}^T\hat{\boldsymbol{\beta}})^2$$ ← Of our interest

Data scarcity: Test data are not available

Notice the ^(caret/cap) symbol above the \beta-- which indicates that: they are of predicted ones.

Validation Set Approach

- Enough data: (1) Training set, (2) Validation set, and (3) Test set
- Not enough data: Generate validation sets from a training set
- Validation set approach: Divides (often randomly) the training set into two parts



- Use training set, to fit the model
- Use validation set, to predict validation set errors

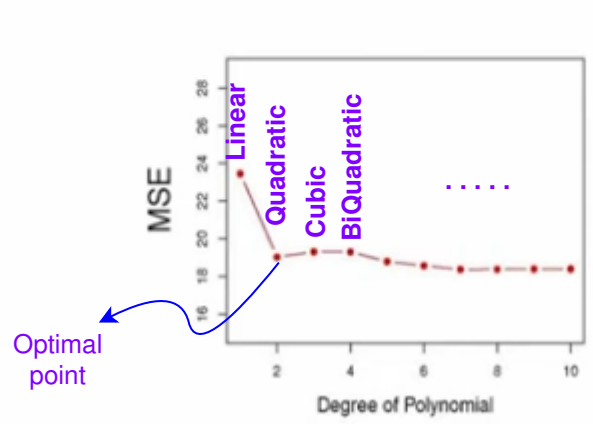
  Provides an estimate of test error rates

Training dataset: *to train the model*
Validation dataset: *to findout the optimal no. of parameters to be used.*
Test dataset: *to test out the accuracy*

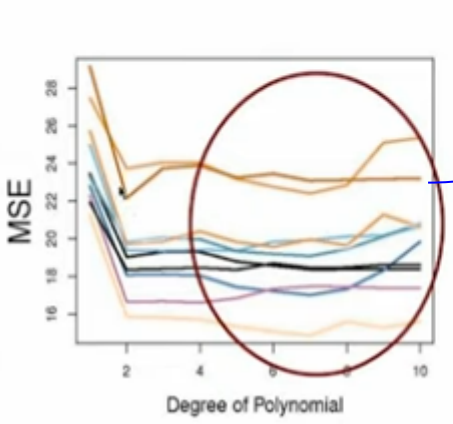
Fit the model in successive manner: i.e., first apply on Linear regression, the Multiple, then polynomial. .. each time get the MSE and use it for plot.
Let's see it in action with an example..

Validation Set Approach: Example

- Example: mileage~ horsepower[1] (> 300 data points on horsepower of automobiles and mileage)
- Polynomial Model: mileage~*f*(horsepower)



On a Validation dataset



Chosen for various random samples of sets like... 250, 200....

High variability means overfitting..

High variability in estimates of test error

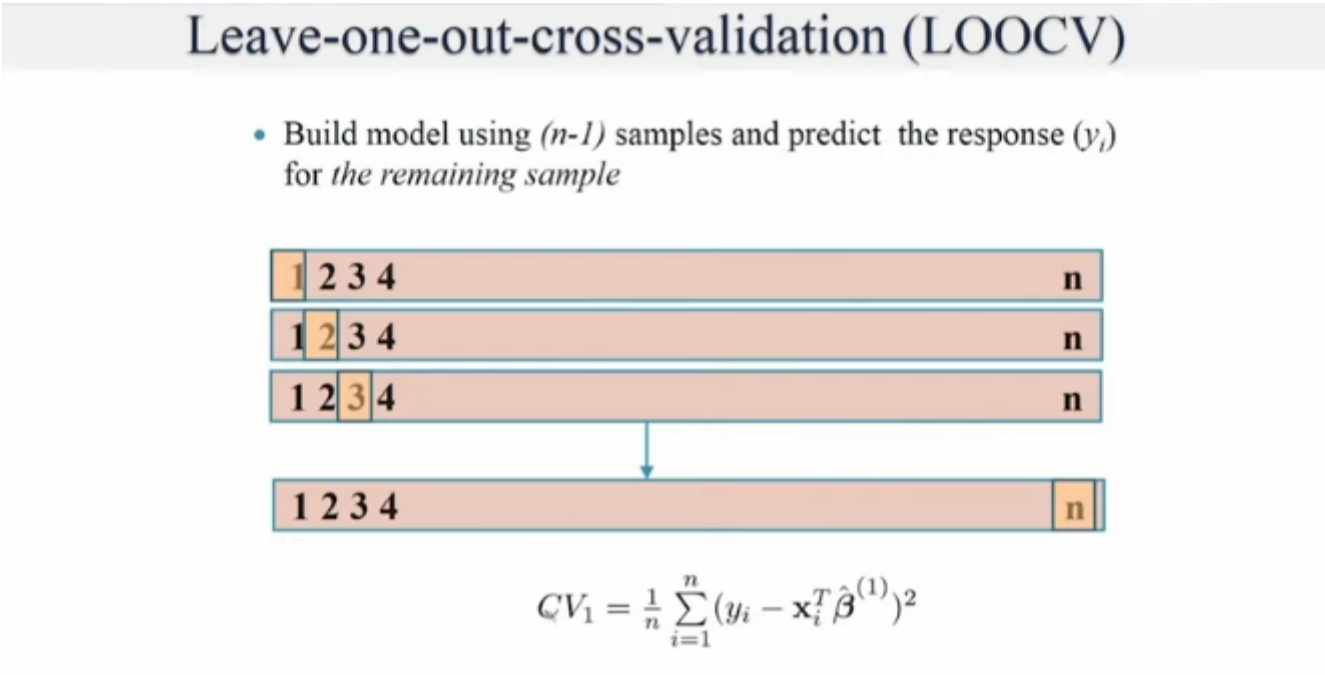
[1]Tibshirani et al (2013)

These are fine.. as long as we have sufficiently large datasets.
But what if had small datasets..??

Sampling for small data sets

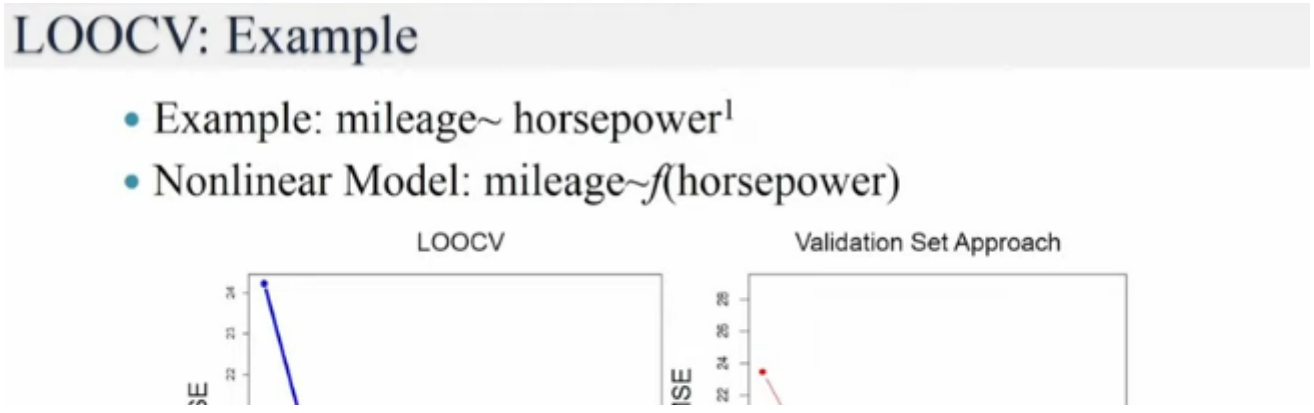
- Validation of models by repeatedly drawing random samples from a training set
  - Validation set (random sampling)
  - K-fold cross validation
  - Bootstrap
- Objective: Predict the performance of model(s) on the validation/test sets (drawn from training data)
- Resampling methods useful for data scarce situations

LOOCV

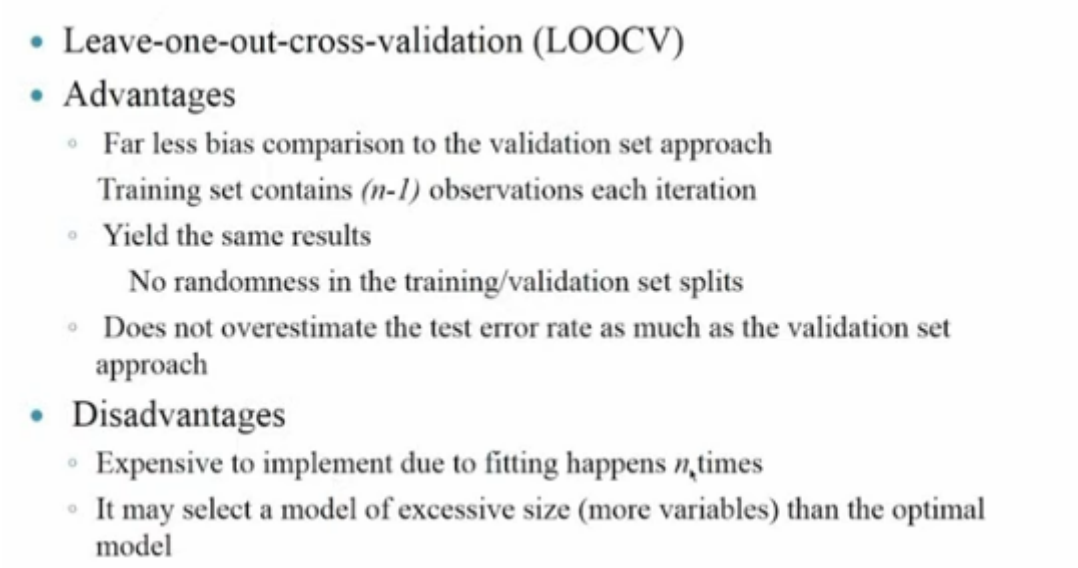


Here `n` nearly equal to `20`..

- Leave out one sample, and perform training on remaining dataset. Say if had 10 samples, keep 1st out, do training on remaining 9, then keep 2nd out and do on remaining 9..... so on.. Find the SSE for each one.
- At the end find the `MSE` or `Cross Validation(CV) error` for (say) Linear regression, then repeat the same procedure for Quadratic..

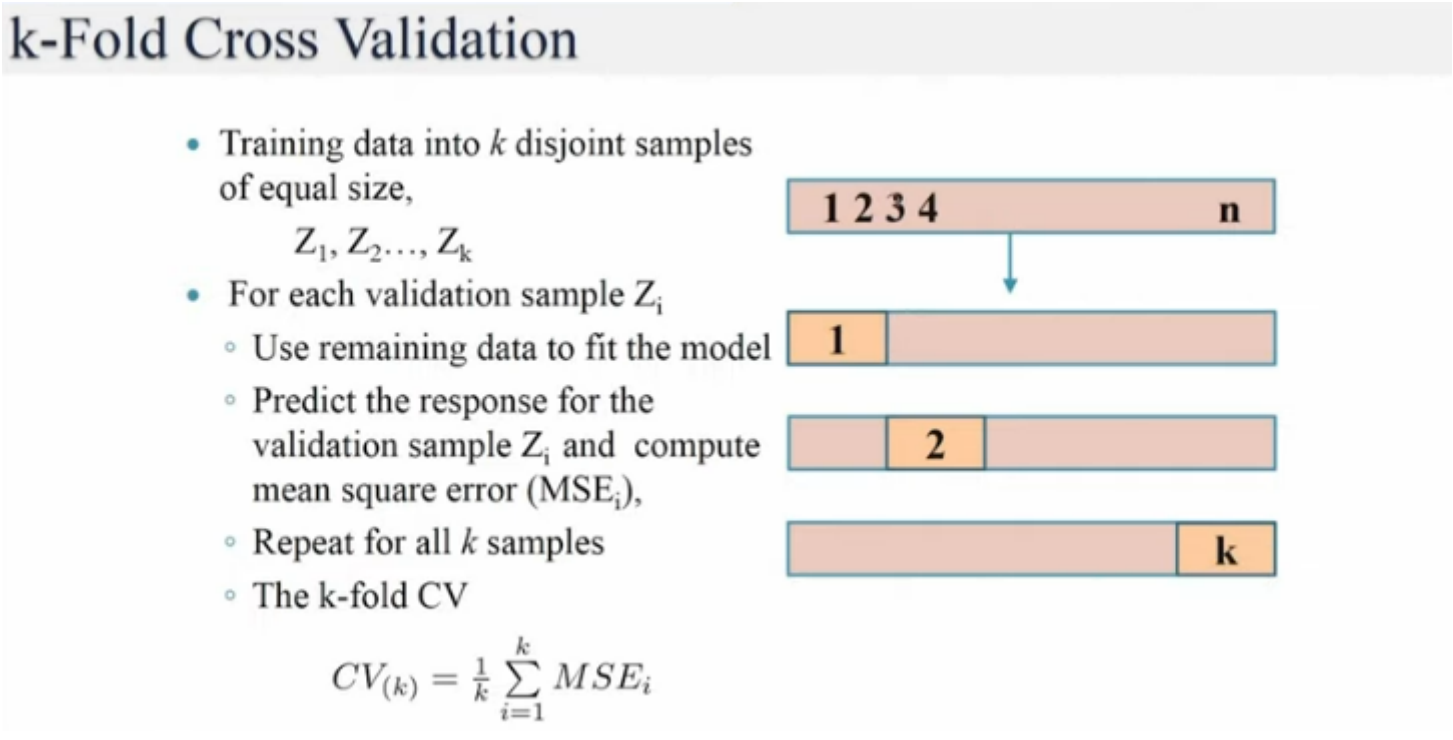


LOOCV-pros and cons



k-Fold Cross Validation

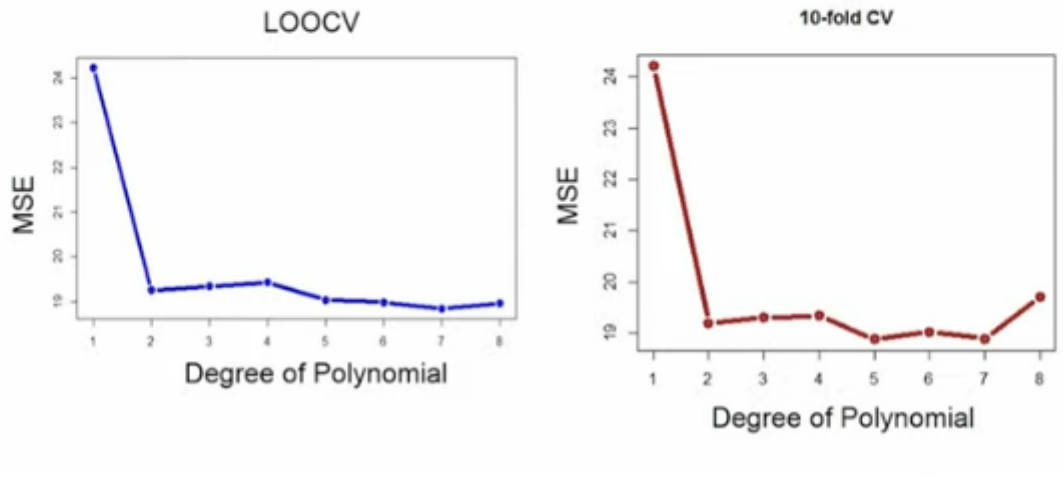
Here too much similar. Instead of leaving `1`, we choose `k` such (each as a group).



- For $k=n$, Leave-one-out-cross-validation (LOOCV)
- In practice, k=5 or 10 is taken,
- Less computation cost
- For computationally intensive learning methods
  - LOOCV fits the model n times
  - k-fold CV fits the model k times

k-fold CV: Example

- Example: mileage~ horsepower[1]
- Nonlinear Model: mileage~f(horsepower)



Module-2: **Multiple Linear Regression, Model Building and Selection**

Multiple Linear Regression Model Building and Selection

Summary of simple linear regression

- Steps in building simple linear regression models
- Model summary
- Residual analysis
- Checking need for refinement
- Refined model building

In this lecture

- Multiple linear regression
  - Build linear model with one dependent and multiple independent variables
  - Look at summary of the model to discard insignificant variable
  - Model selection ---- *Identifying subset of variables*

.png)

In [18]:
```
nyc <- read.csv("resources/datasets/nyc.csv")
#nyc    # View(nyc)
```

In [17]:
```
head(nyc)   # Gives first 6 rows from dataframe
```

Price	Food	Decor	Service	East
43	22	18	20	0
32	20	19	19	0

Price	Food	Decor	Service	East
34	21	13	18	0
41	20	20	17	0
54	24	19	21	0

In [16]:
```
tail(nyc)        # Gives last 6 rows from dataframe
```

	Price	Food	Decor	Service	East
163	31	19	16	18	0
164	31	17	15	16	0
165	26	20	16	17	0
166	31	18	16	17	0
167	38	22	17	21	0
168	34	24	10	16	0

Description of dataset

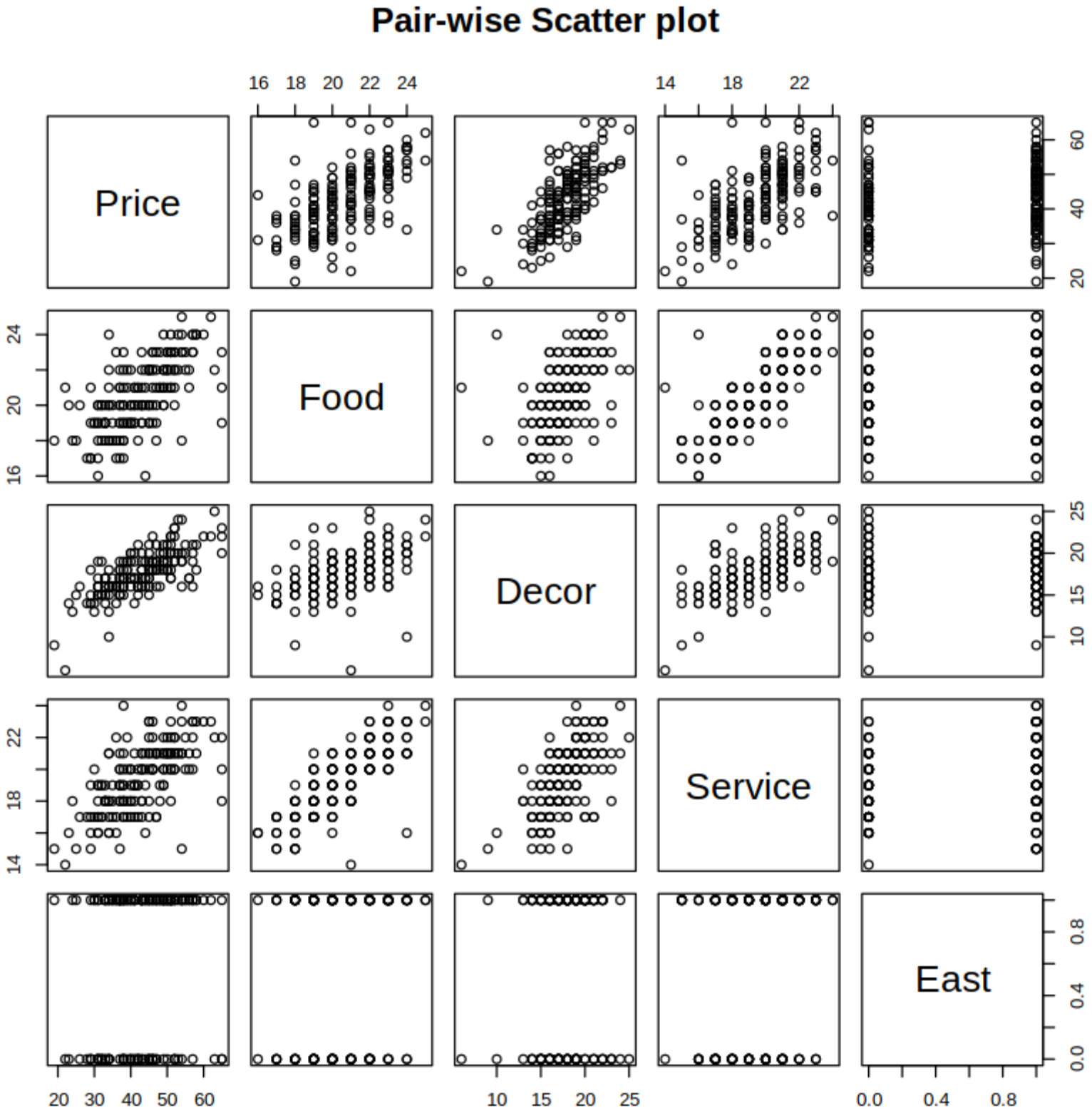
Menu pricing in restaurants of NYC

y : Price of dinner
x_1: Customer rating of the food (Food)
x_2: Customer rating of the décor (Décor)
x_3: Customer rating of the service (Service)
x_4: If the restaurant is east or west (East)

Objective: Build a linear model

Before heading to build a model, let's know the independency between the variables.. -- **pair-wise scatter plot** comes to handy and the **Correlation values**..

In [20]:
```
plot(nyc, main="Pair-wise Scatter plot")
```



Interpreting the plot.....

- Variables are mentioned acroos the daigonals.. When one moves from Left to Right..
- The variables of left will be the Y-axis and variables above or below, becomes the X-axis.

Finding the correlation (~~ Numerical meaning of the above scatter plots..)..

In [21]:
```
round(cor(nyc), 4)
```

	Price	Food	Decor	Service	East
Price	1.0000	0.6270	0.7244	0.6411	0.1866
Food	0.6270	1.0000	0.5039	0.7945	0.1804
Decor	0.7244	0.5039	1.0000	0.6453	0.0357
Service	0.6411	0.7945	0.6453	1.0000	0.2091
East	0.1866	0.1804	0.0357	0.2091	1.0000

Food vs Service seems a good match.. close to 8

Building multiple linear regression model

- Dependent variable (y) depends on p independent variables x_i, i=1,2..p

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p + \epsilon$$

- For i^{th} observation,

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1,i} + \hat{\beta}_2 x_{2,i} + \cdots + \hat{\beta}_p x_{p,i} + \epsilon_i$$

ϵ is the error, which is assumed as:

- It is purely in the measurement error of the dependent one(y), and not on independent one(x).
- It is an unknown quantity, which has **zero mean and some variance**.

Building multiple linear regression model

- Building multiple linear model using the function `lm()`
- Syntax: `lm(formula,data)`

```
lm(dependent var~indep.var1+ indep.var2)
```

In [29]:
```
nyc_model = lm(Price~Food+Decor+Service+East, data=nyc)   # Use this format, when need to regress on a subset of var
nyc_model
```

```
Call:
lm(formula = Price ~ Food + Decor + Service + East, data = nyc)

Coefficients:
(Intercept)         Food        Decor      Service         East
 -24.023800     1.538120     1.910087    -0.002727     2.068050
```

In [30]:
```
nyc_model_test = lm(Price~., data=nyc)                    # Use this format, when need to regress with all (excludin
nyc_model_test
```

```
Call:
lm(formula = Price ~ ., data = nyc)

Coefficients:
(Intercept)         Food        Decor      Service         East
 -24.023800     1.538120     1.910087    -0.002727     2.068050
```

Model Summary

In [26]:
```
summary(nyc_model)
```

```
Call:
lm(formula = Price ~ Food + Decor + Service + East, data = nyc)

Residuals:
     Min       1Q   Median       3Q      Max
-14.0465  -3.8837   0.0373   3.3942  17.7491

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.023800   4.708359  -5.102 9.24e-07 ***
Food          1.538120   0.368951   4.169 4.96e-05 ***
Decor         1.910087   0.217005   8.802 1.87e-15 ***
Service      -0.002727   0.396232  -0.007   0.9945
East          2.068050   0.946739   2.184   0.0304 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.738 on 163 degrees of freedom
Multiple R-squared:  0.6279,    Adjusted R-squared:  0.6187
F-statistic: 68.76 on 4 and 163 DF,  p-value: < 2.2e-16
```

Model summary

```
nycmod_1 <- lm(Price~Food+Decor+Service+East,data = nyc)

summary(nycmod_1)
```

```
Call:
lm(formula = Price ~ Food + Decor + Service + East, data = nyc)

Residuals:
     Min       1Q   Median       3Q      Max
-14.0465  -3.8837   0.0373   3.3942  17.7491

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.023800   4.708359  -5.102 9.24e-07 ***
Food          1.538120   0.368951   4.169 4.96e-05 ***
Decor         1.910087   0.217005   8.802 1.87e-15 ***
Service      -0.002727   0.396232  -0.007   0.9945
East          2.068050   0.946739   2.184   0.0304 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.738 on 163 degrees of freedom
Multiple R-squared:  0.6279,    Adjusted R-squared:  0.6187
F-statistic: 68.76 on 4 and 163 DF,  p-value: < 2.2e-16
```

$\hat{\beta}_i$
$i = 0, 1 \ldots 4$

Service is an insignificant term (see that it has no stars -- which indicate the level of significance).

So, let's build the model by removing it..

In [28]:
```
nyc_model_2 = lm(Price ~ Food+Decor+East, data=nyc)
nyc_model_2
```

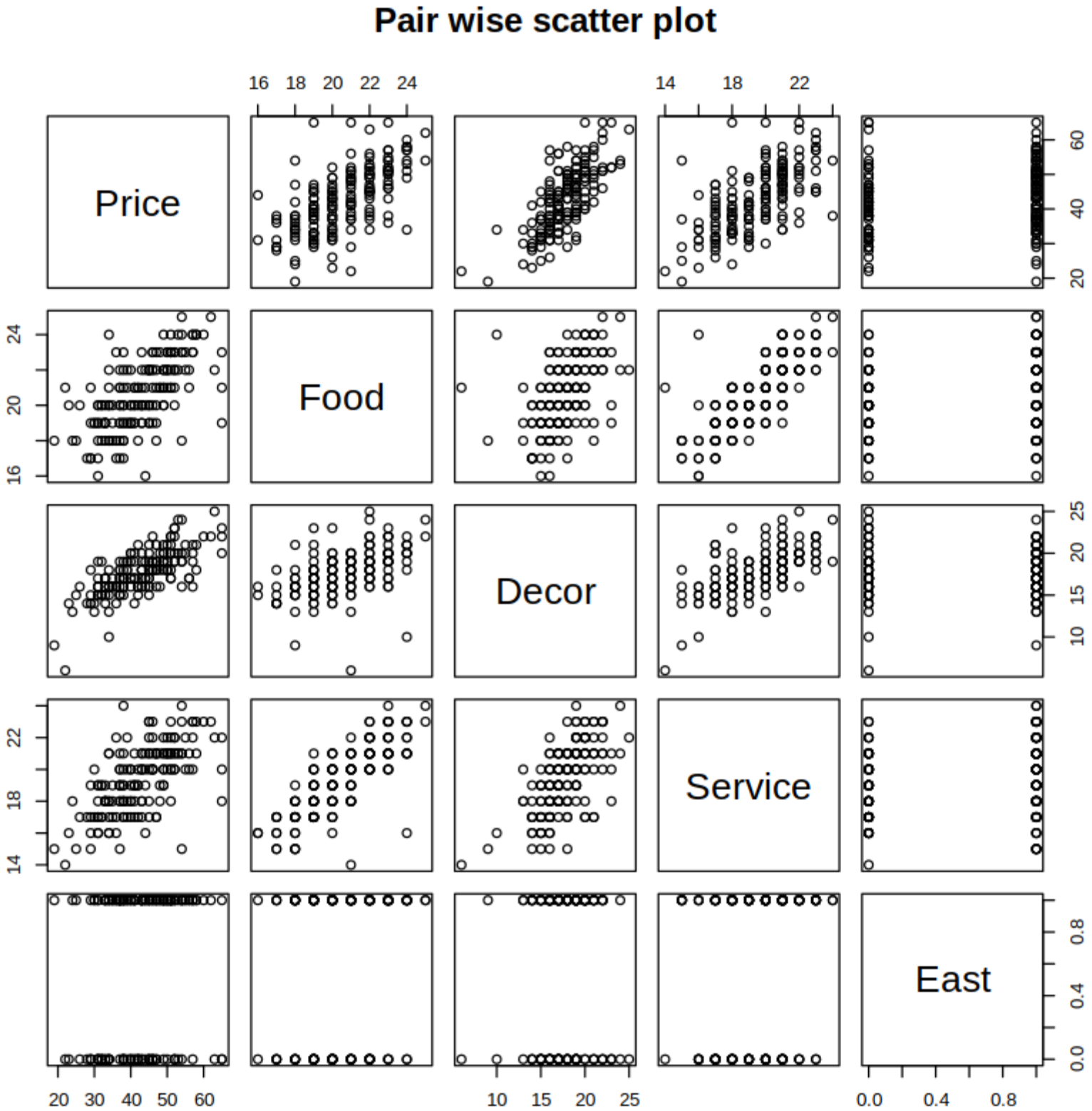
```
Call:
lm(formula = Price ~ Food + Decor + East, data = nyc)

Coefficients:
(Intercept)         Food        Decor         East
    -24.027        1.536        1.909        2.067
```

In [31]:
```
summary(nyc_model_2)
```

```
Call:
lm(formula = Price ~ Food + Decor + East, data = nyc)

Residuals:
     Min       1Q   Median       3Q      Max
-14.0451  -3.8809   0.0389   3.3918  17.7557

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.0269     4.6727  -5.142 7.67e-07 ***
Food          1.5363     0.2632   5.838 2.76e-08 ***
Decor         1.9094     0.1900  10.049  < 2e-16 ***
East          2.0670     0.9318   2.218   0.0279 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.72 on 164 degrees of freedom
Multiple R-squared:  0.6279,    Adjusted R-squared:  0.6211
F-statistic: 92.24 on 3 and 164 DF,  p-value: < 2.2e-16
```

- See that, estimates didn't changed drastically -- indicates that: Service has no that impact. And, Pr(>|t|) hasn't changed.
- Moreover, the **R-Squared value** didn'changed.. indication that, Serive is not that significant.
- But, the **Adjusted R-squared** changed slightly.
- **F-Statistic** really hight, tells that *Full model with Fod, decor and east* performs better comparing with reduced model having only intercept.

In [33]:
```
plot(nyc, main="Pair wise scatter plot")  # See the co-relation between Food and service..
```



As now, we built model w/o Service, let's try by removing the Food...

In [36]:
```
nyc_mod_3 = lm(Price~Service+Decor+East, data=nyc)
nyc_mod_3
```

```
Call:
lm(formula = Price ~ Service + Decor + East, data = nyc)

Coefficients:
(Intercept)      Service        Decor         East
    -14.531        1.151        1.896        2.154
```

In [37]:
```
summary(nyc_mod_3)
```

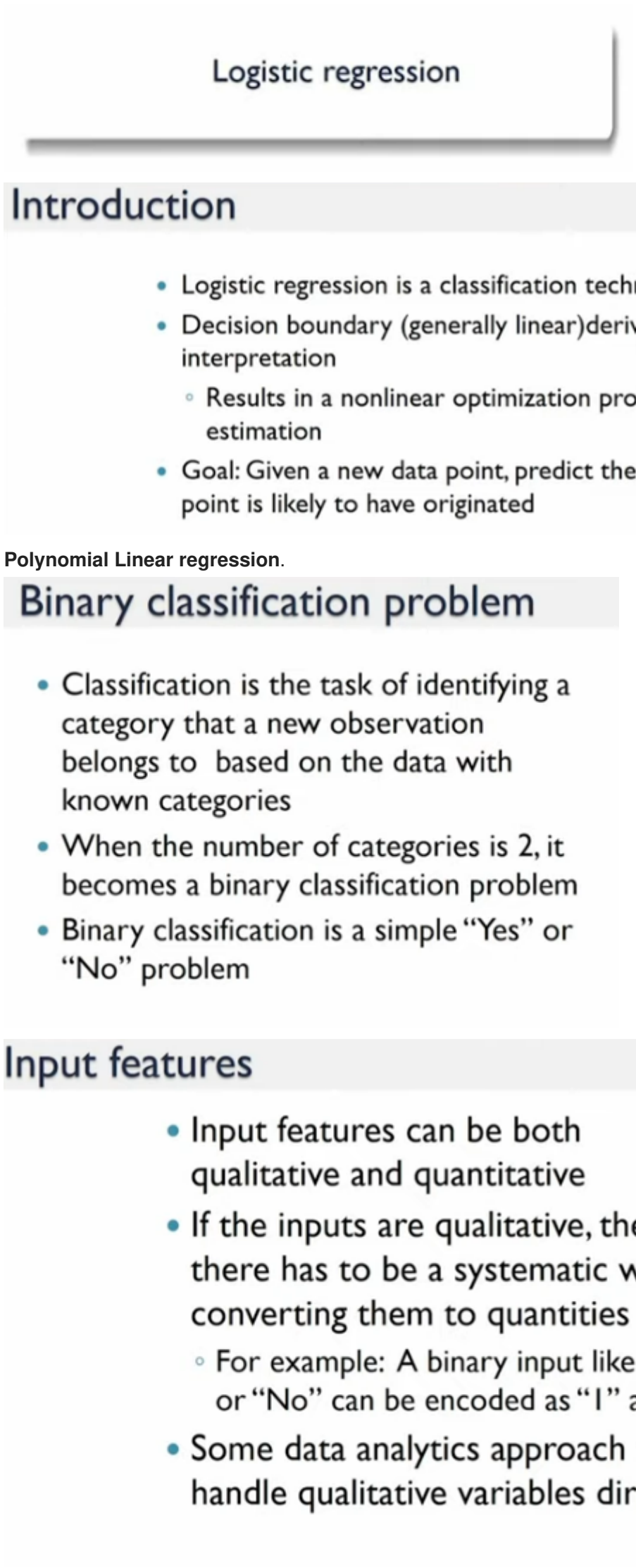
```
Call:
lm(formula = Price ~ Service + Decor + East, data = nyc)

Residuals:
    Min      1Q  Median      3Q     Max
-14.514  -3.668  -0.769   3.704  18.778

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -14.5308     4.3220  -3.362 0.000963 ***
Service       1.1513     0.2973   3.872 0.000156 ***
Decor         1.8956     0.2276   8.331 3.05e-14 ***
East          2.1535     0.9927   2.169 0.031489 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.018 on 164 degrees of freedom
Multiple R-squared:  0.5882,    Adjusted R-squared:  0.5807
F-statistic: 78.09 on 3 and 164 DF,  p-value: < 2.2e-16
```

Notice that, now the **R-squared** and **Adjusted R-squared** has changed drastically..

This tells that:

The scatter plot tells to use the Linear regresssion..
But we still need to verify the assumptions we make using residual analysis. -- its a task for you sir.. go ahead..

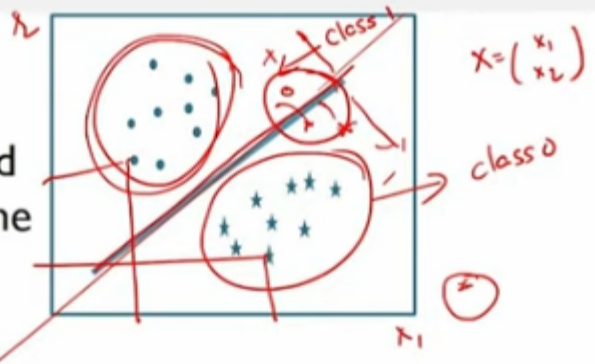
Module-3: **Logistic Regression**



For non-linear boundary we use the **Polynomial Linear regression**.

Linear classifier

- Decision function is linear
- Binary classification can be performed depending on the side of the half-plane that the data falls in
- We saw this before in the linear algebra module
- However, simply guessing "yes" or "no" is pretty crude
- Can we do something better using probabilities ?



Output

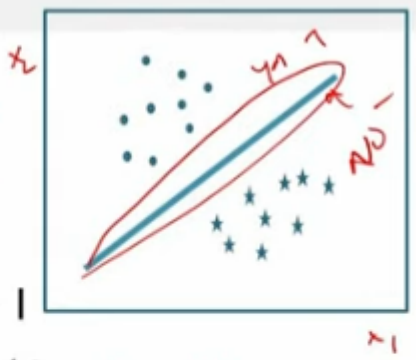
- Why model probabilities ?
  - The probability of a "Yes" or "No" gives a better understanding of the sample's membership to a particular category

Linear and log models

- Make p(x) a linear function of x

$$p(x) = \beta_0 + \beta_1 X$$

  - This makes p(x) unbounded below 0 and above 1
  - Might give nonsensical results making it difficult to interpret them as probabilities



- Make log(p(x)) a linear function of x

$$\log(p(x)) = \beta_0 + \beta_1 X$$

  - Bounded only on one side

Bounding only one side means....??

> - Linear model restricts to be in boundary of [0, 1], but that sometimes not sufficient.
> - Another chosen way is the: **Using logarithm**, this makes the -ve side trimmed off, but it makes +ve side Infinite -- but should have the notion of probability being bounded between 0 and 1.

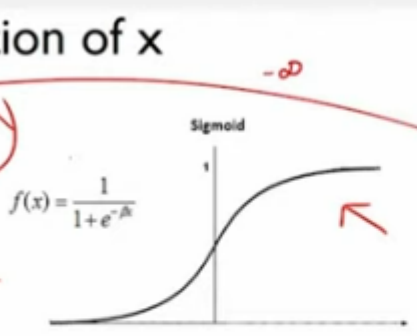
Next comes the sigmoid function..

Sigmoid function

- Make p(x) a sigmoid function of x

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

$$\text{or } \log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$



  - p(x) bounded above by 1 and below by 0
  - Good modeling choice for real life scenarios
  - The LHS can be interpreted as the log of odds-ratio in the second equation

-- this isn't properly clear, looks like some basic missing.. Come again later..

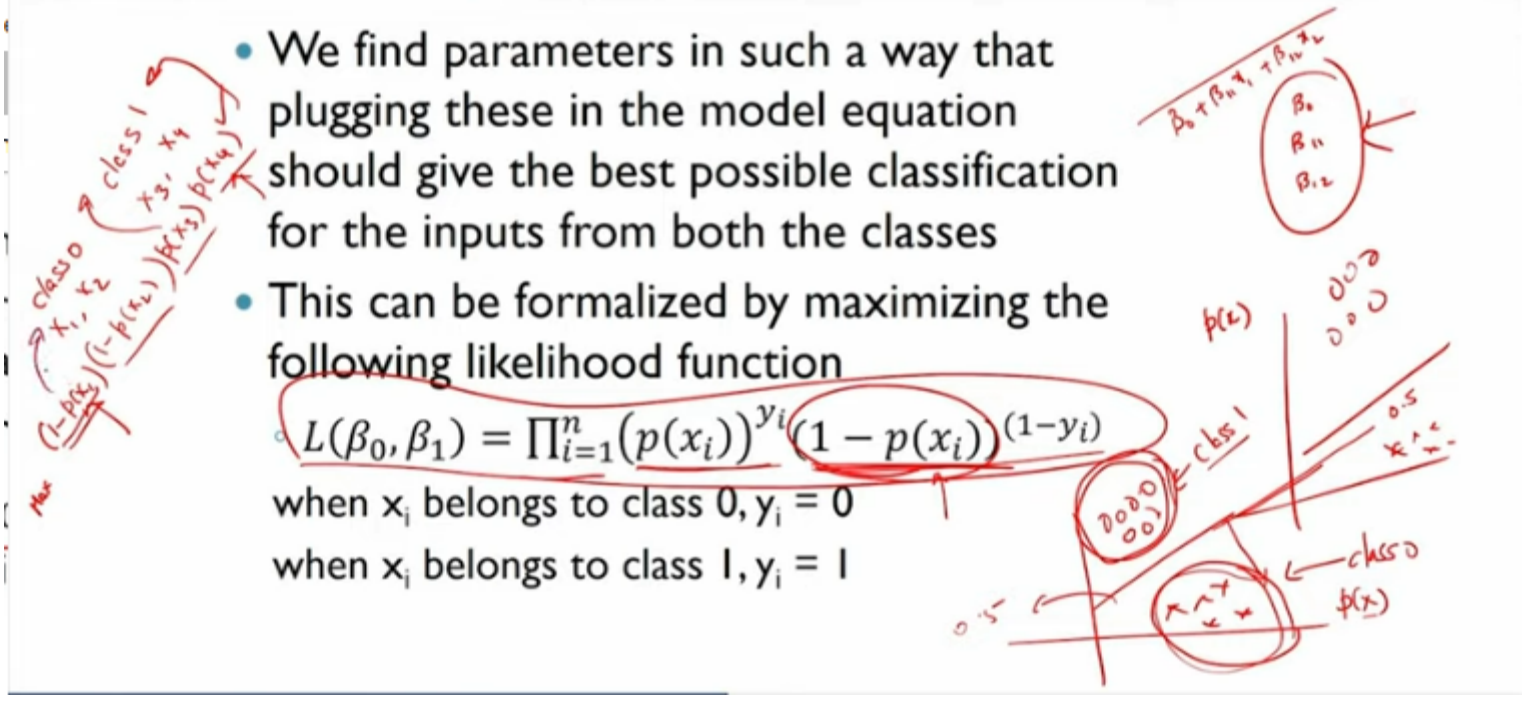
Estimation of the parameters

- We find parameters in such a way that plugging these in the model equation should give the best possible classification for the inputs from both the classes
- This can be formalized by maximizing the following likelihood function

$$L(\beta_0, \beta_1) = \prod_{i=1}^{n} (p(x_i))^{y_i} (1 - p(x_i))^{(1-y_i)}$$

when x_i belongs to class 0, $y_i = 0$

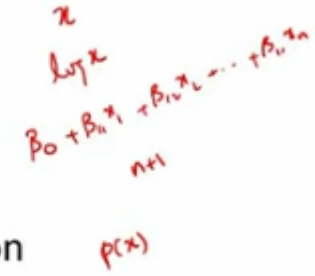
when x_i belongs to class 1, $y_i = 1$



-- little cloudy..

Log-likelihood function

- The log-likelihood function will become $l(\beta_0, \beta_1) = \sum_{i=1}^{n} y_i \log(p(x_i)) + (1 - y_i)\log(1 - p(x_i))$
- Simplifying this expression and using the definition for p(x) will result in an expression with the parameters of the linear decision boundary
- Now the parameters can be estimated by maximizing the above expression using any



Module-3.1: **Logistic Regression**-Continued

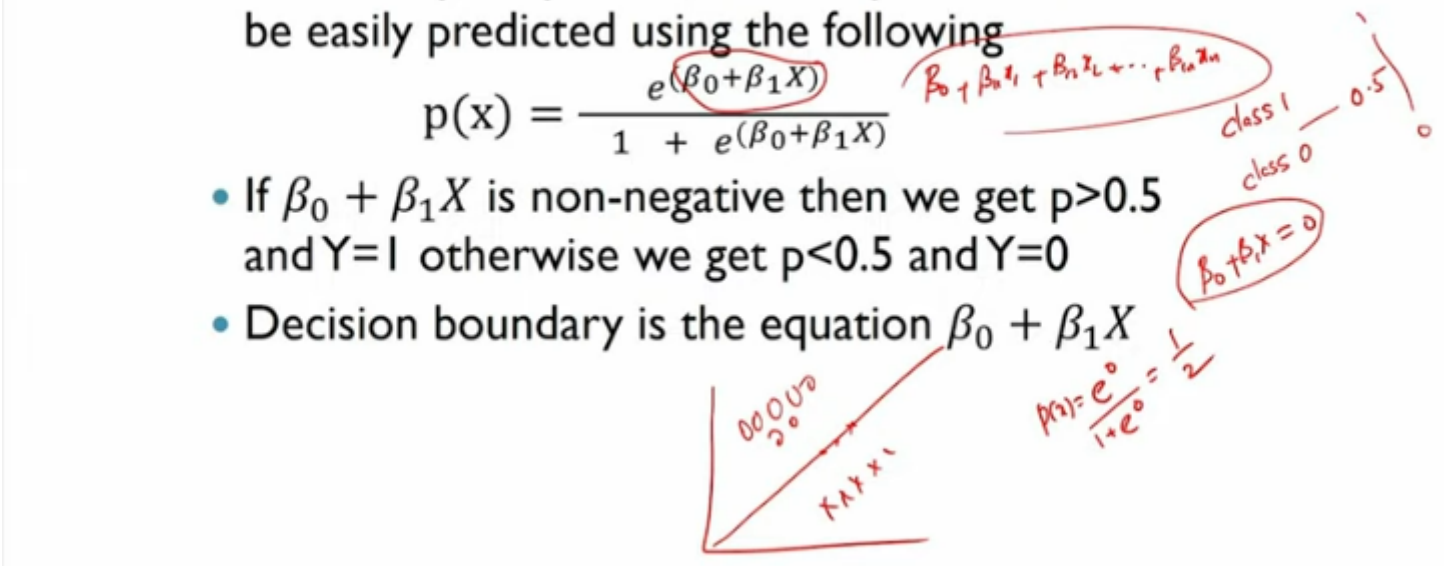
Logit Model

Logit model

- The binary output for new samples can now be easily predicted using the following

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

- If $\beta_0 + \beta_1 X$ is non-negative then we get p>0.5 and Y=1 otherwise we get p<0.5 and Y=0
- Decision boundary is the equation $\beta_0 + \beta_1 X$



Here, if the point falls on the Hyper-plane, it will have equal chance of belonging to either classes.

Example 1

X_1	X_2
1	1
2	1
3	1
4	1
5	1
1	2
2	2
3	2
4	2
5	2

Class 0

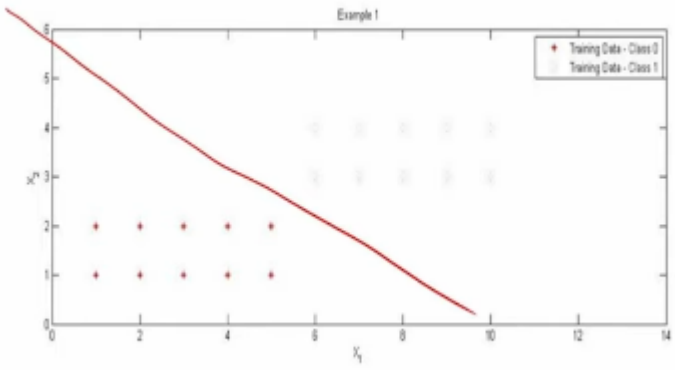
X_1	X_2
6	3
7	3
8	3
9	3
10	3
6	4
7	4
8	4
9	4
10	4

Class 1

X_1	X_2	Class ?
1	3	?
2	3	?
4	4	?
5	4	?
3	3	?
6	2	?
9	2	?
8	1	?
7	2	?
10	1	?

Test Data

2 dimensional



Results

- Input Features : X_1, X_2
- Classes : 0, 1
- Parameters:

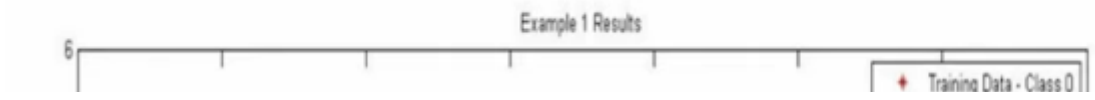
$$\beta_0 = -42.5487$$
$$\beta_{11} = 2.9509$$
$$\beta_{12} = 10.4012$$

Test Results

X_1	X_2	Prob	Class
1	3	0.0002	0
2	3	0.004	0
4	4	0.999	1
5	4	0.999	1
3	3	0.076	0
6	2	0.0172	0
9	2	0.991	1
8	1	0.0002	0
7	2	0.251	0
10	1	0.0667	0

if `Prob < 0.5` set to `class_0` and if `>`, set to `class_1`.

And said about, why one need to partition the dataset.



Regularization *-- method to control overfitting..*

Regularization

- General objective
  - $\min_{\theta} -L(\theta)$
  - where $L(\theta) = \left(\sum_{i=1}^{n} y_i \log\left(p(x_i)\right) + (1 - y_i) \log\left(1 - p(x_i)\right)\right)$
- When large number of independent variables are present, logistic regression tends to over-fit
- To prevent over-fitting, we need to penalize the coefficients
- This is known as regularization

, when had `n` variables, we need .....?????.. `n+1` ..

Every co-efficient has to contribute some reasonably max value, if it doesn't, removing it. -- that's the idea of regularization.
Why one needs to do that.??

> If doesn't it causes **Over-Fitting**.

In general, the use of regularization is:

> It helps in testing model as overfitting is cleared

Module-4: **Performance Measures**

Performance measures

Results : confusion matrix

confusion matrix.

```
> confusionMatrix(crashTest_1_TEST$LogisPred,crashTest_1_TEST$CarType)
Confusion Matrix and Statistics

          Reference
Prediction  Hatchback SUV
  Hatchback        10   1
  SUV               0   9

               Accuracy : 0.95
                 95% CI : (0.7513, 0.9987)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.003e-05

                  Kappa : 0.9
 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9000
         Pos Pred Value : 0.9091
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5500
      Balanced Accuracy : 0.9500

       'Positive' Class : Hatchback
```

Actual values

predicted values

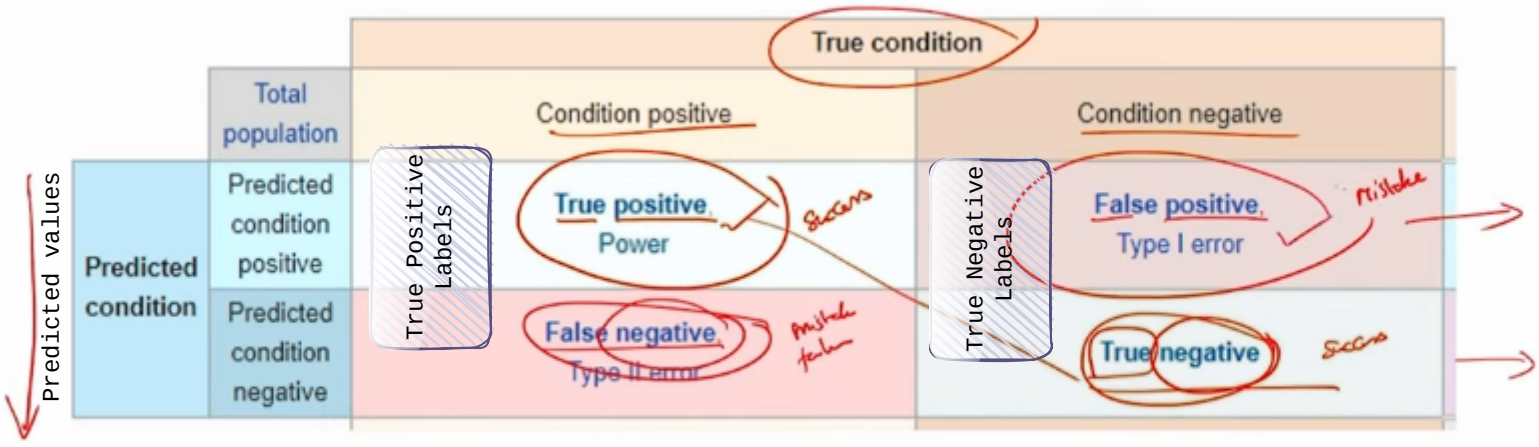
20 Totalling up all the values, gives the no. of observations

Performance metrics of classifiers

Confusion matrix

Each element tells: **`Row Column`**

`Actual Values`



Source: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Measures of performance

- Terminology
  - TP → true positives, TN → true negatives,
  - FP → false positives, FN → false negatives

$$N = TP + TN + FP + FN$$

  - TP – Correct identification of positive labels
  - TN – Correct identification of negative labels
  - FP – Incorrect identification of positive labels
  - FN – Incorrect identification of negative labels

Measures of performance

- Accuracy: Overall effectiveness of a classifier
  - $A = \frac{TP+TN}{N}$
  - Maximum value that accuracy can take is 1
  - This happens when the classifier exactly classifies two groups (i.e, $FP = 0$ and $FN = 0$)

TP, TN, FP, FN

- Remember
  - Total number of true positive labels = TP+FN

First column

- Similarly
  - Total number of true negative labels = TN+FP

Second column

Measures of performance

- Sensitivity: Effectiveness of a classifier to identify positive labels
  - $S_e = \frac{TP}{TP + FN}$ $\quad S_e = \frac{TruePositive}{TruePositive - labels}$
- Specificity: Effectiveness of a classifier to identify negative labels
  - $S_p = \frac{TN}{FP + TN}$ $\quad S_p = \frac{TrueNegative}{TrueNegative - labels}$
- Both S_e and S_p lie between 0 and 1, 1 is an ideal value for each of them
- Balanced accuracy
  - $BA = (sensitivity + specificity)/2$

TP, TN, FP, FN

Measures of performance

- Prevalence: How often does the yes condition actually occur in our sample

$$P = \frac{TP + FN}{N} \quad P = \frac{\text{True Positive\_Labels}}{\text{Number of observations}}$$

- Positive predictive value: Proportion of correct results in labels identified as positive
  - $PPV = \frac{(sensitivity * prevalence)}{((sensitivity*prevalence) + ((1-specificity)*(1-prevalence)))}$

$$P = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- Negative prediction value: Proportion of correct results in labels identified as negative
  - $NPV = \frac{specificity * (1-prevalence)}{(((1-sensitivity)*prevalence) + ((specificity)*(1-prevalence)))}$

$$N = \frac{\text{True Negative}}{\text{True Negative} + \text{False Negative}}$$

(PPV and PPN are said by sir

Measures of performance

- Detection rate:
  - $DR = \frac{TP}{N}$
- Detection prevalence: prevalence of predicted events
  - $DP = \frac{TP+FP}{N}$
- The Kappa statistic (or value) is a metric that compares an **observed accuracy** with an **expected accuracy** (random chance)
- $\text{Kappa} = \frac{observed\ accuracy - expected\ accuracy}{1 - expected\ accuracy}$

inthe final solving slide of conf_matrix) `Observed accuarcy` should be larger than the `Estimated accuracy`.

Measures of performance

- Observed accuracy
  - $OA = \frac{a+d}{N}$
- Expected accuracy
  - $EA = \frac{(a+c)(a+b)+(b+d)(c+d)}{N}$
- $\text{Kappa} = \frac{\frac{(a+d)}{N} - \left(\frac{(a+c)(a+b)+(b+d)(c+d)}{N}\right)}{\left(1 - \left(\frac{(a+c)(a+b)+(b+d)(c+d)}{N}\right)\right)}$
  - Where $a, b, c\ and\ d$ are $TP, FP, FN\ and\ TN$ respectively

Results : confusion matrix

confusion matrix.

```
> confusionMatrix(crashTest_1_TEST$LogisPred,crashTest_1_TEST$CarType)
Confusion Matrix and Statistics

          Reference
Prediction  Hatchback SUV
  Hatchback        10   1
  SUV               0   9

               Accuracy : 0.95
                 95% CI : (0.7513, 0.9987)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.003e-05

                  Kappa : 0.9
 Mcnemar's Test P-Value : 1
```

TP = 10 FP = 1
FN = 0 TN = 9
19/20

What range of values are good..??

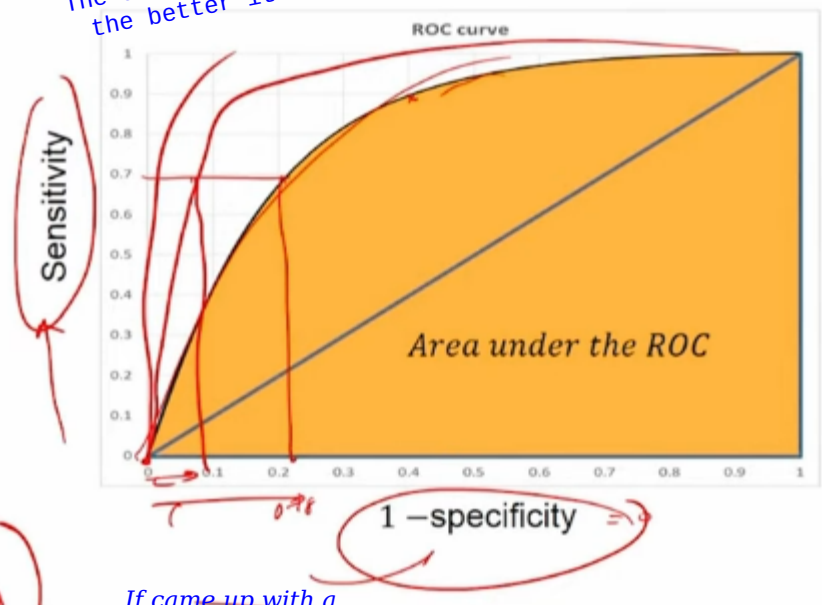
> Depends on subject

ROC curve

ROC

- ROC –An acronym for Receiver Operating Characteristics
- Originally developed and used in signal detection theory
- ROC graph:
  - Sensitivity as a function of specificity
  - sensitivity (Y-axis) and 1 –specificity (X-axis)

The steeper the curve.. the better it is..!!



$\frac{TP}{TP+FN}$

If came up with a strategy..

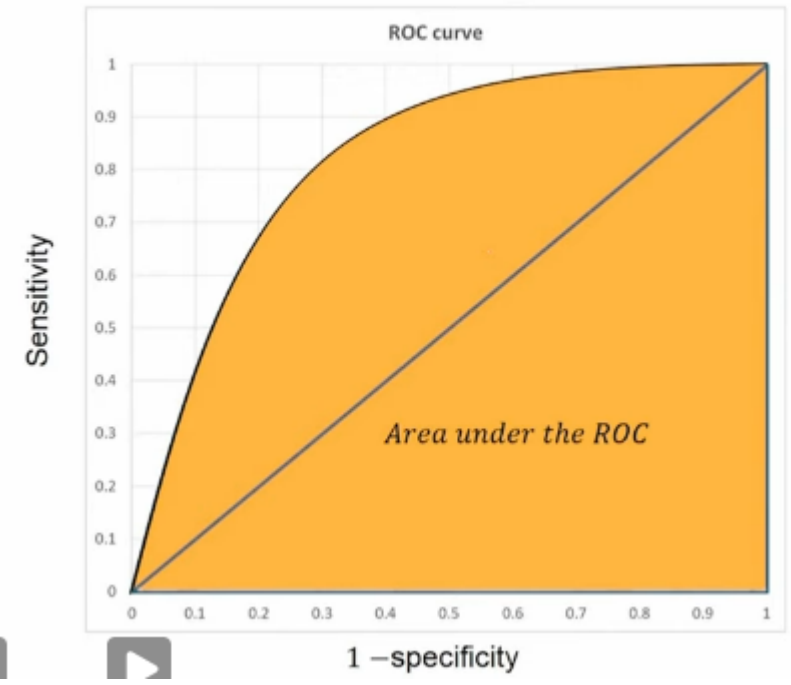
For **sensitivity** the good value is 1. So, I do no classification, and say every label as positive (then TP=1, so FN=0). then S_e yields1

For **Specificity** the good value is 1. So, without doing any work.. claim as every label is negative

This way, both are useless. There has to be some trade-off b/w these two ... that's what given by this curve.

ROC

- ROC can be used to
  - See the classifier performance at different threshold levels (from 0 to 1)
  - AUC- Area under the ROC
    - An area of 1 represents a perfect test; an area of 0.5 represents a worthless model.
    - .90 – 1 = excellent
    - .80 – .90 = good
    - .70 – .80 = fair
    - .60 – .70 = poor
  - AUC < 0.5, check whether your labels are marked in opposite

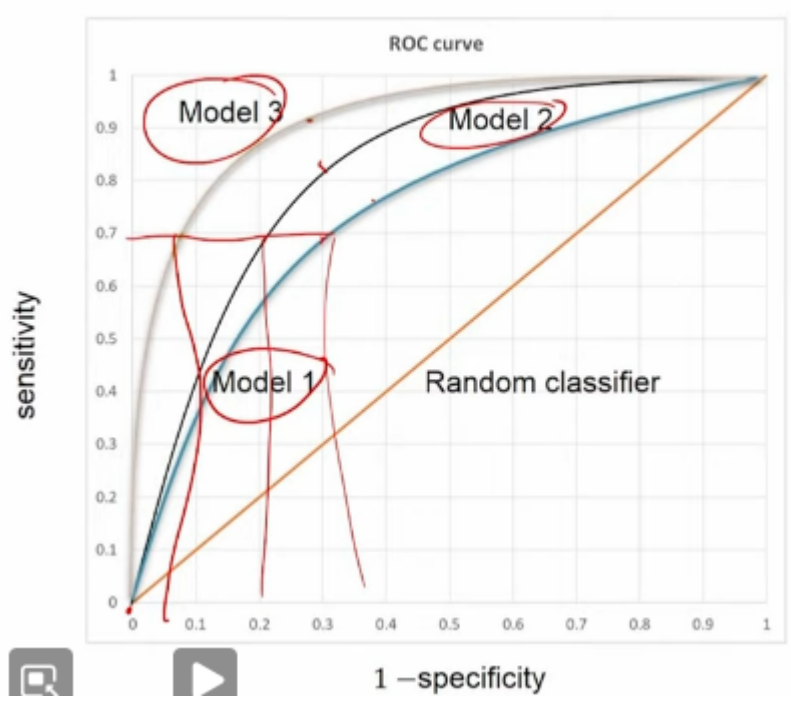


---------- this is for single classifier.

But if had several classifiers..

ROC

- ROC can be used to
  - Compare different classifiers at one threshold or overall threshold levels
  - Performance
  - Model 3 > Model 2 > Model 1



- How to benchmark the results
- How to interpret the results

Module 5: **Logistic Regression Implementation**

Logistic Regression implementation in R

In this lecture

- Case study
  - Problem statement
- Solve the case study using R
  - Read the data from a ".csv" file
  - Understand the data
  - glm() function
  - Interpret the results

Key points from previous lecture

- Logistic Regression is primarily used as a classification algorithm
- It is supervised learning algorithm
  - Data is labelled
- Parametric approach
- Decision boundary derived based on probability interpretation
- Decision boundary can be linear/ non-linear
- Probabilities are modelled as sigmoidal function

Automotive Crash Testing

Automotive Crash Testing- Problem Statement

- A crash test is a form of destructive testing that is performed in order to ensure high safety standards for various cars

Automotive Crash Testing

Hatchback SUV

Automotive Crash Testing

Automotive Crash Testing- Problem Statement

- Several cars have rolled into an independent audit unit for crash test
- They are being evaluated on a defined scale {poor (-10) to excellent(10)} on:

1) Manikin head impact
2) Manikin body impact
3) Interior impact
4) HVAC impact
5) Safety alarm system

HVAC: Heat Ventilation AC test

Automotive Crash Testing

- Each crash test is very expensive
- The crash test was performed for only 100 cars
- Type of car- Hatchback/SUV, was noted
- However with this data in future they should be able to predict the type of the car
- Part of data reserved for building a model and remaining kept for analysis

Automotive Crash Testing

- Data for 80 cars is given in crashTest_1.csv
- Data for remaining 20 cars is given in crashTest_1_TEST.csv
- Use logistic regression classification technique to classify the car types as

Solution to case study using R

Getting things ready

- Setting working directory, clearing variables in the workspace
- Installing or loading required packages

```
# Set the working directory as the directory which
#contains the data files
# setwd("Path of the directory with data files")
rm(list=ls()) # to clear the environment
# install.packages("caret",dependencies = TRUE)

library(caret) # for confusionMatrix
```

`glm()` is the in-built function, whereas for confusion matrix, need `caret`

In [1]:
```
library(caret)
```

```
Loading required package: lattice
Loading required package: ggplot2
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
```

Reading datasets...

In [2]:
```
crashTest = read.csv("resources/datasets/crashtest_1.csv")
crashTest_testData = read.csv("resources/datasets/crashTest_1_TEST.csv")
```

In [4]:
```
#crashTest  # View(crashTest)
```

Understanding the data

- crashTest_1 contains 80 observations of 6 variables
- crashTest_1_TEST contains 20 observations of 6 variables
- The variables are: Manikin head impact, Manikin body impact, Interior impact, HVAC impact, Safety alarm system
  - First five columns are the details about the car and last column is the label which says whether the cartype Hatchback/ SUV

In [5]:
```
str(crashTest)
```

```
'data.frame':   80 obs. of  7 variables:
 $ CarID  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ ManHI  : num  -5.27 -4.82 9.57 2.84 0 0.4 5.94 5.78 0.86 7.36 ...
 $ ManBI  : num  -1.3 -5.38 -7.5 -2.85 2.68 6.34 3.14 -1.75 -4.32 7.42 ...
 $ IntI   : num  2.86 9.72 -7.61 0.92 -4.15 0.83 -6.65 -6.85 8.1 0.27 ...
 $ HVACi  : num  -4.85 -0.97 1.33 5.51 0.85 5.03 6.62 0.73 -8.96 -8.62 ...
 $ Safety : num  4.04 -4.57 -5.1 -6.64 5.58 -8.1 -1.32 5.5 3.1 3.08 ...
 $ CarType: Factor w/ 2 levels "Hatchback","SUV": 2 1 1 1 2 2 1 1 1 2 ...
```

In [7]:
```
str(crashTest_testData)
```

```
'data.frame':   20 obs. of  7 variables:
 $ CarID  : int  81 82 83 84 85 86 87 88 89 90 ...
 $ ManHI  : num  1.94 -0.02 -0.49 5.76 2.51 -4.47 -9.89 -9.94 -8.37 8.48 ...
 $ ManBI  : num  2.21 -3.33 -4.48 1.35 -8.74 8.42 -2.25 -3.23 4.21 0.38 ...
 $ IntI   : num  3.38 0.79 5 7.92 4.53 -0.05 -5 2.81 -8.95 -3.02 ...
 $ HVACi  : num  1.78 -6.63 8.33 -0.43 -1.91 5.57 -9.23 -2.98 6.66 -1.92 ...
 $ Safety : num  -7.19 7.99 -2.77 4.29 3.95 9.62 9.38 -1.12 7.34 -7.43 ...
 $ CarType: Factor w/ 2 levels "Hatchback","SUV": 1 2 1 1 1 1 2 2 2 2 ...
```

Summary of the data

- Summary of data
  - The function invokes particular methods which depend on the class of the first argument.
- summary()

Summary gives a 5`point summary for numeric attributes in the data

SYNTAX

```
summary(object)
```

object	any R object about which you want to have some information.

In [8]:
```
summary(crashTest)
```

```
    CarID           ManHI             ManBI             IntI        
Min.   : 1.00   Min.   :-9.9300   Min.   :-9.9400   Min.   :-9.9900  
1st Qu.:20.75   1st Qu.:-5.1950   1st Qu.:-5.7050   1st Qu.:-5.5725  
Median :40.50   Median : 0.6350   Median :-1.8150   Median :-0.4150  
Mean   :40.50   Mean   :-0.0935   Mean   :-0.9277   Mean   :-0.1349  
3rd Qu.:60.25   3rd Qu.: 5.0500   3rd Qu.: 3.4175   3rd Qu.: 4.9775  
Max.   :80.00   Max.   : 9.5700   Max.   : 9.6100   Max.   : 9.7200  
    HVACi             Safety           CarType  
Min.   :-9.8200   Min.   :-9.8000   Hatchback:50  
1st Qu.:-5.6750   1st Qu.:-4.6775   SUV      :30  
Median : 0.8700   Median : 0.8300                 
Mean   : 0.1197   Mean   : 0.5437                 
3rd Qu.: 5.0625   3rd Qu.: 4.6225                 
Max.   : 9.8900   Max.   : 9.9900                 
```

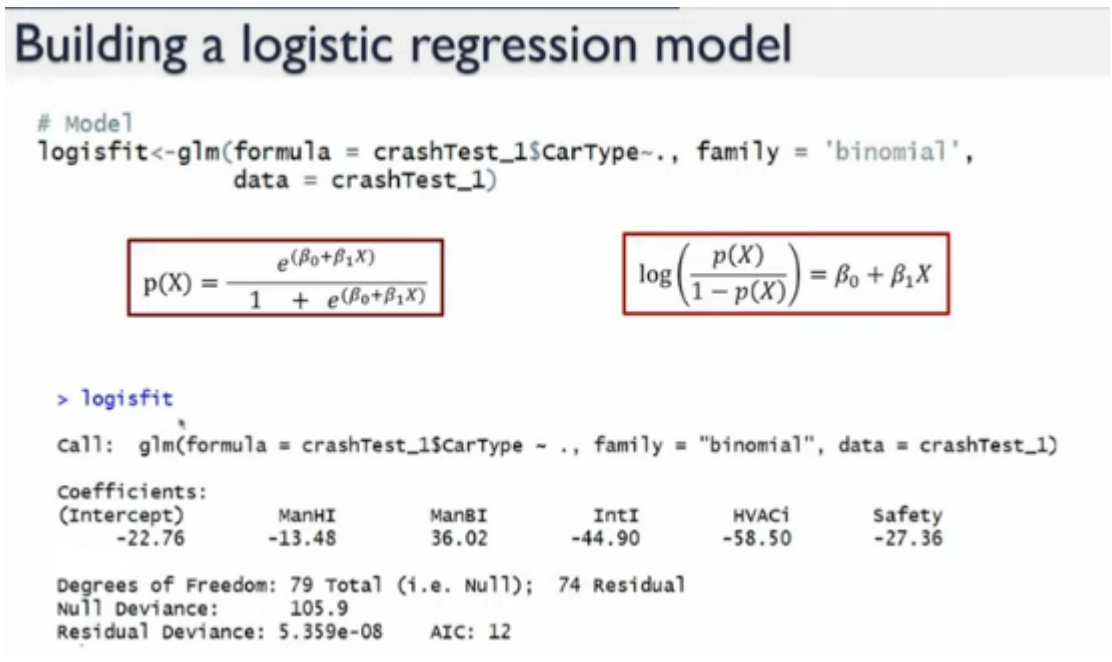
In [9]:
```
summary(crashTest_testData)
```

```
    CarID           ManHI             ManBI             IntI        
Min.   : 81.00   Min.   :-9.940   Min.   :-8.740   Min.   :-8.950  
1st Qu.: 85.75   1st Qu.:-5.535   1st Qu.:-2.502   1st Qu.:-3.272  
Median : 90.50   Median : 0.740   Median : 0.670   Median : 1.200  
```

```
Mean   : 90.50   Mean   : 0.047   Mean   : 0.328   Mean   : 0.524
3rd Qu.: 95.25   3rd Qu.: 5.110   3rd Qu.: 2.500   3rd Qu.: 3.908
Max.   :100.00   Max.   : 9.090   Max.   : 8.420   Max.   : 8.870
    HVACi             Safety           CarType
Min.   :-9.2300   Min.   :-8.660   Hatchback:10
1st Qu.:-2.4550   1st Qu.:-6.095   SUV      :10
Median : 0.6750   Median :-0.770
Mean   : 0.7235   Mean   : 0.191
3rd Qu.: 5.3375   3rd Qu.: 4.992
Max.   : 8.3300   Max.   : 9.620
```

For the numerical varible its: 5-point summary

For the categorical variable its: Frequency



.png)

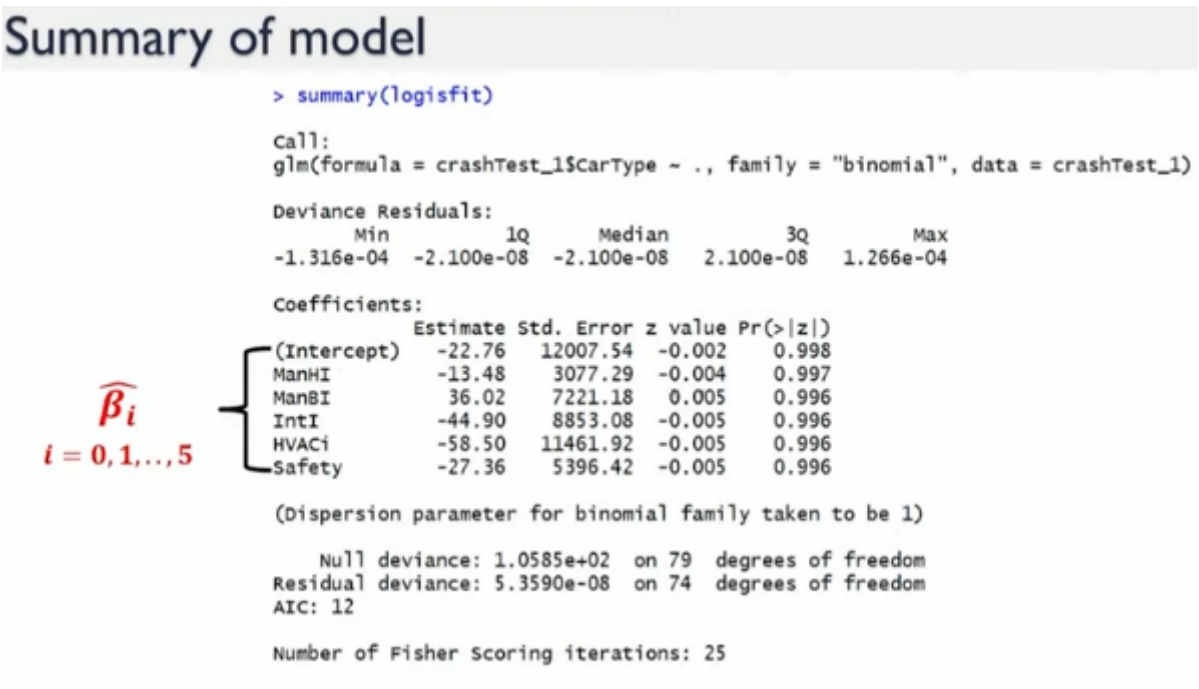
`P(X)` is the probability being: `HatchBack`, and `1-P(x)` is of `SUV`

In [10]:
```
logistic_model = glm(formula=crashTest$CarType~.,
                     family = 'binomial',
                     data=crashTest)
```

```
Warning message:
“glm.fit: algorithm did not converge”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
```

In [11]:
```
logistic_model
```

```
Call:  glm(formula = crashTest$CarType ~ ., family = "binomial", data = crashTest)

Coefficients:
(Intercept)        CarID        ManHI        ManBI         IntI        HVACi
   -22.6248       0.6066      -7.9116      23.9471     -29.0903     -37.2615
     Safety
   -16.3743

Degrees of Freedom: 79 Total (i.e. Null);  73 Residual
Null Deviance:      105.9
Residual Deviance: 2.592e-08    AIC: 14
```

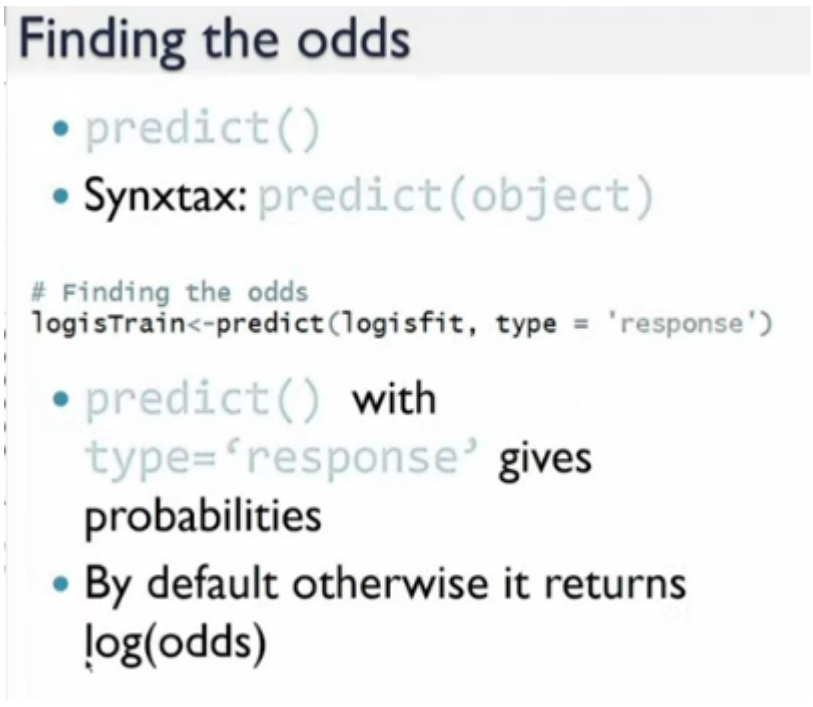


Understanding the summary

- Probabilities are very high, none of them are significant (as > 0.5)
- **Null deviance**: deviance of model when only intercept term is taken ----------**Reduced Model**: `80-1 = 79`, `1` tells the intercept
- **Residual deviance**: deviance of model when all terms are taken into account --**Full Model**: `80-6 = 74` <, `6` for the all varibles
- **Fisher Score**: Its used for maximum likelihood estimation and its a derivative of ***Newton Raphson Method***. Here it took 25 iterations

In [12]:
```
summary(logistic_model)
```

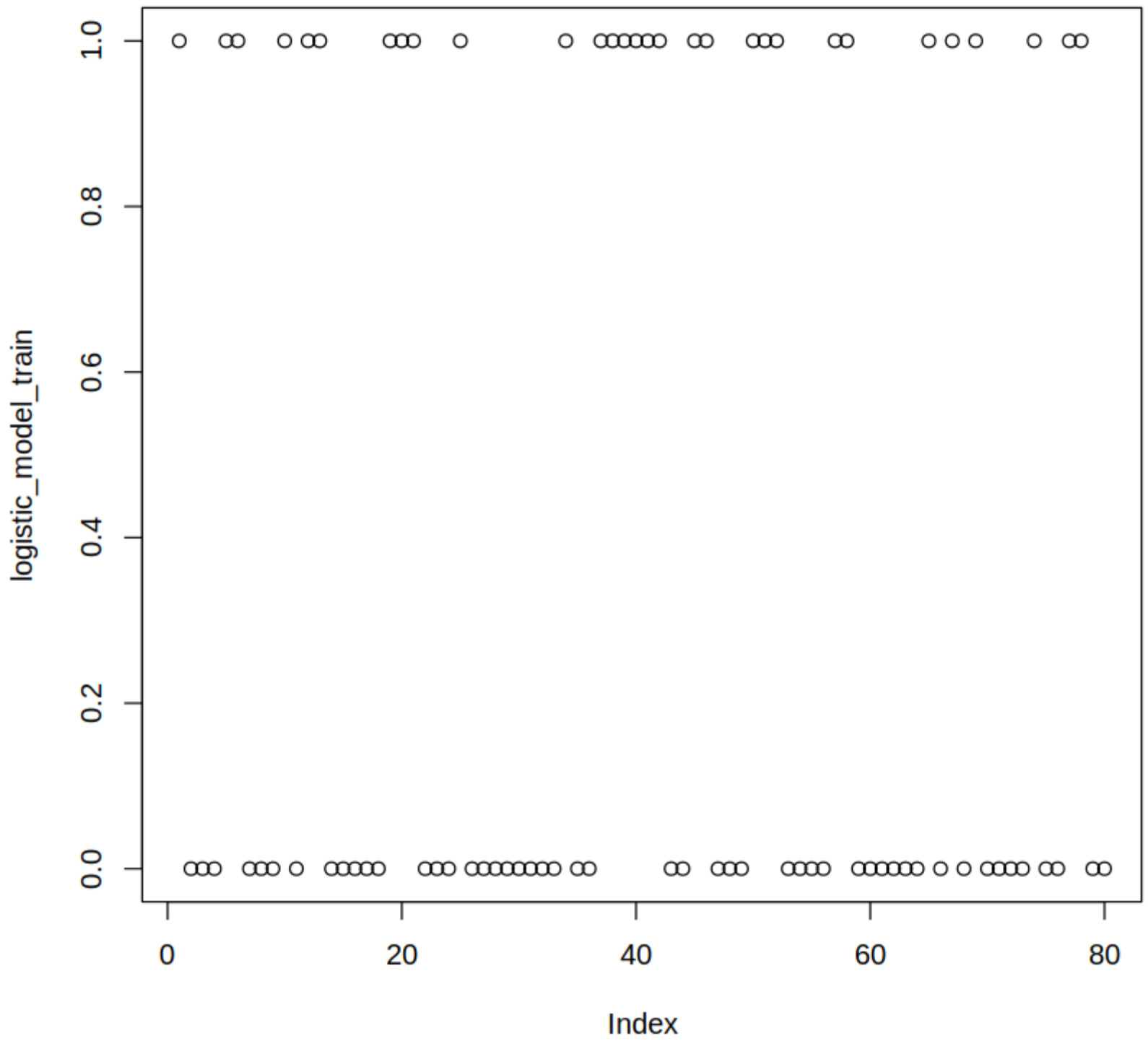
```
Call:
glm(formula = crashTest$CarType ~ ., family = "binomial", data = crashTest)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-7.708e-05  -2.100e-08  -2.100e-08   2.100e-08   9.186e-05

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -22.6248 14069.0777  -0.002    0.999
CarID           0.6066   509.6480   0.001    0.999
ManHI          -7.9116  4669.5451  -0.002    0.999
ManBI          23.9471  7002.6900   0.003    0.997
IntI          -29.0903  8411.3045  -0.003    0.997
HVACi         -37.2615 10222.2769  -0.004    0.997
Safety        -16.3743  4498.3291  -0.004    0.997

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.0585e+02  on 79  degrees of freedom
Residual deviance: 2.5924e-08  on 73  degrees of freedom
AIC: 14

Number of Fisher Scoring iterations: 25
```



In [15]:
```
logistic_model_train = predict(logistic_model, type='response')
#logistic_model_train
```

Plotting probabilities

In [16]:
```
plot(logistic_model_train)
```



The classes are well-separated, but we don't know which class they belong to..

Identifying probabilities associated with the CarType

- Mean of probabilities
- This helps us identify the probabilities associated with the two classes

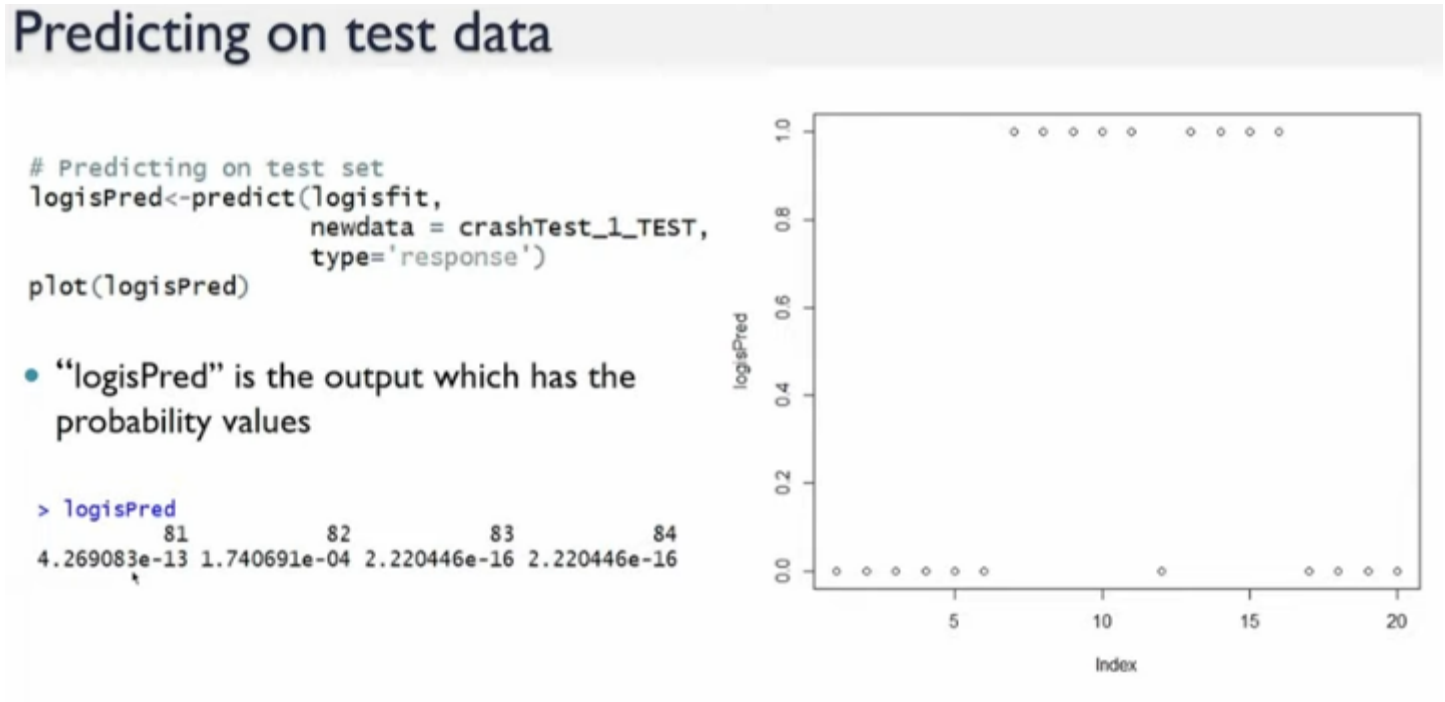
```
> tapply(logisTrain,crashTest_1$CarType,mean)
   Hatchback          SUV
2.851316e-10 1.000000e+00
```

Recollect the demonstartion of usage of this function from **Week_1**..

For `Hatchback`, its really low `2.851..`, and for `SUV` its `1`.

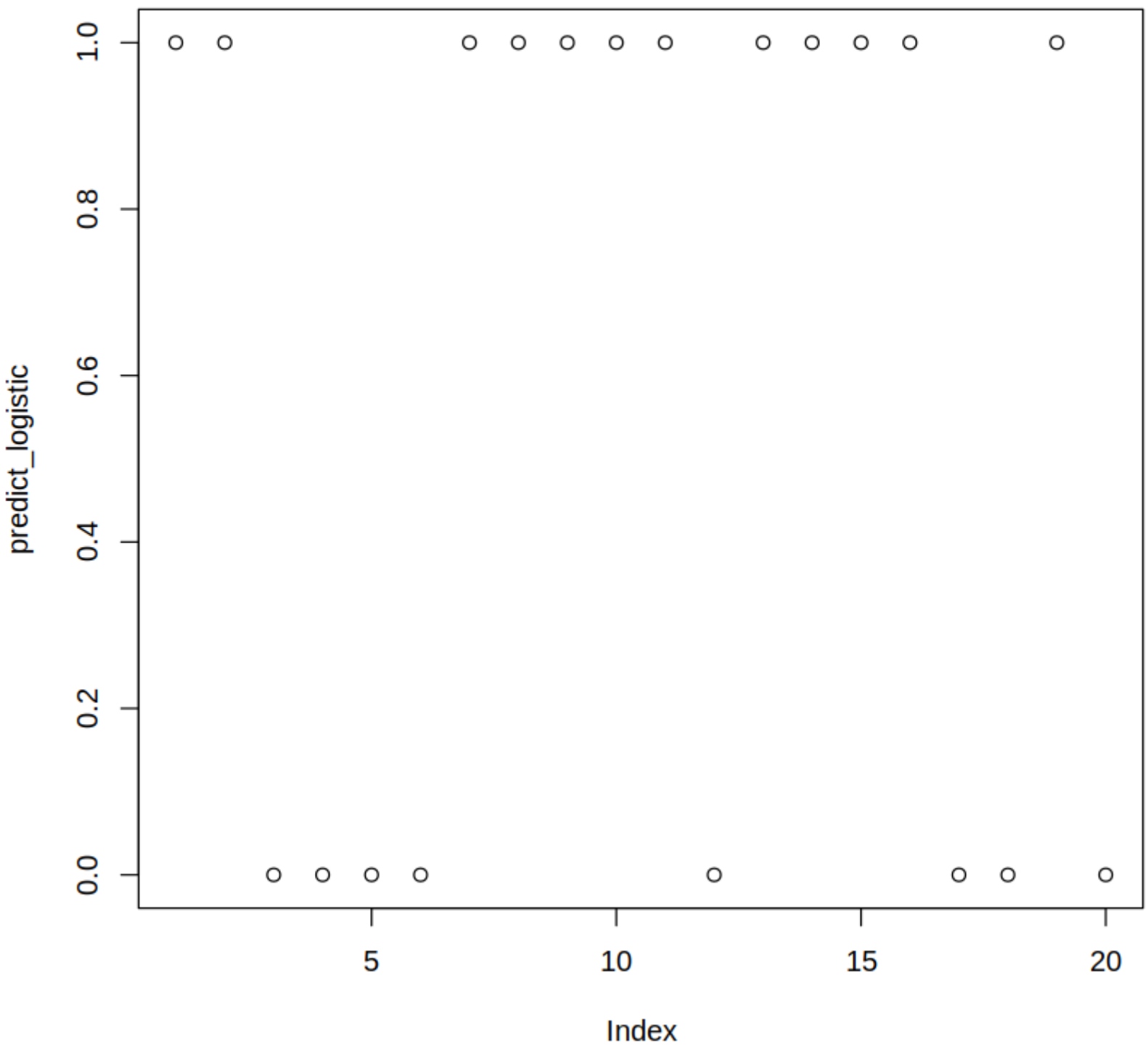
This tells that, lower valued probabilites are associated with `Hatchback`, and higher probabilities with `SUV`.

i.e., Bottom values are of `Hatchback`, and above ones are of `SUV`.



In [22]:
```
predict_logistic = predict(logistic_model,
                           newdata = crashTest_testData,
                           type='response')
#predict_logistic
```

In [23]:
```
plot(predict_logistic)
```



Bottom ones are of `Hatchback` and top ones are of `SUV`.

Results

- We classify whether the test point is Hatchback/ SUV by setting a threshold

```
crashTest_1_TEST[logisPred<=0.5,"LogisPred"]<-"Hatchback"
crashTest_1_TEST[logisPred>0.5,"LogisPred"]<-"SUV"
```

crashTest_1_TEST | Classification+Clustering R session cod... * | lda_logistic_finalLR

Filter

	ManHI	ManBI	Intl	HVACi	Safety	CarType	LogisPred
81	1.94	2.21	3.38	1.78	-7.19	Hatchback	Hatchback
82	-0.02	-3.33	0.79	-6.63	7.99	SUV	Hatchback

Hint: `dataset[condition, "new_col_name"]<-"What to assign if matched"`

Why needed this?

> To check, how accurately our classifier could able to predict.

Confusion matrix

```
confusionMatrix(table(crashTest_1_TEST[,7],
                      crashTest_1_TEST[,6]),positive = 'Hatchback')
```

```
Confusion Matrix and Statistics

           Reference
Prediction Hatchback SUV
Hatchback         10   1
SUV                0   9

               Accuracy : 0.95
                 95% CI : (0.7513, 0.9987)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.003e-05

                  Kappa : 0.9
 Mcnemar's Test P-Value : 1
```

```
            Sensitivity : 1.0000
            Specificity : 0.9000
         Pos Pred Value : 0.9091
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5500
      Balanced Accuracy : 0.9500

       'Positive' Class : Hatchback
```

If not given `positive='Hatchback'`, it selects the first encountered label as the postivel label.

$$\text{Accuracy} = \frac{\text{TruePostives} + \text{TrueNegatives}}{\text{Number of observations}}$$

- The positive label (`Hatchback`) is been identified correctly, hence `Senstivity:` .
- The negative label(`SUV`), has 1 mis-classification hence `` `Specificity: 0.9 ``
- Balanced accuracy = avg of Senstivity and Specificity.

In [24]:
```
help.stand(glm)
```

```
Error in help.stand(glm): could not find function "help.stand"
Traceback:
```

In [1]:
```
#?glm
```

In [29]:
```
3**4
```

81

In [ ]: