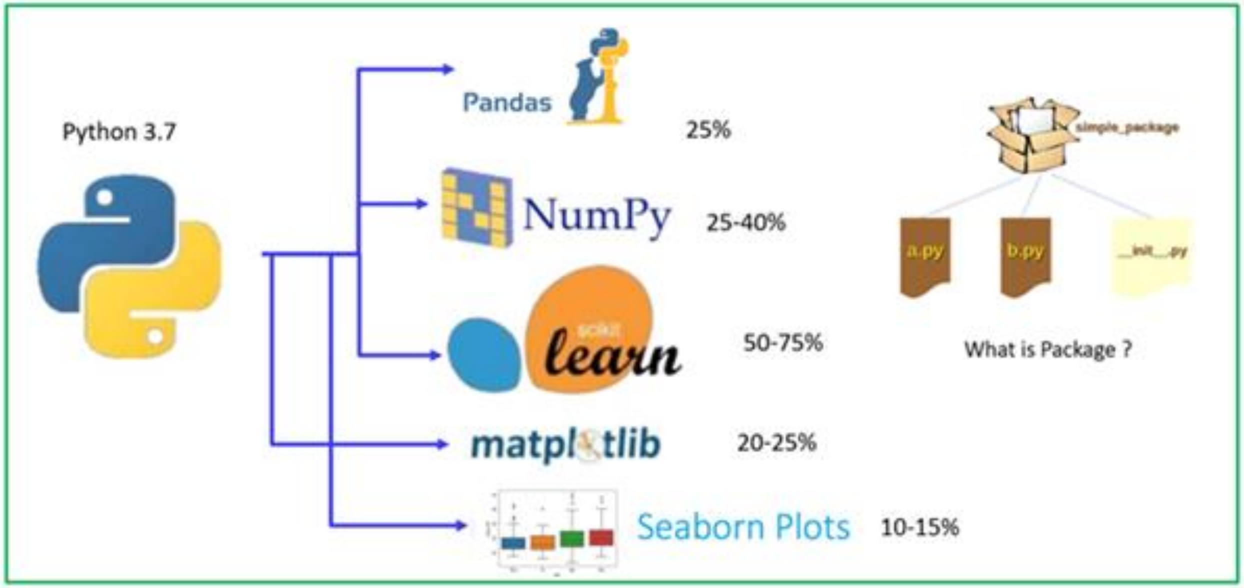
Data Analytics with Python

Indian Institute of Foreign Trade



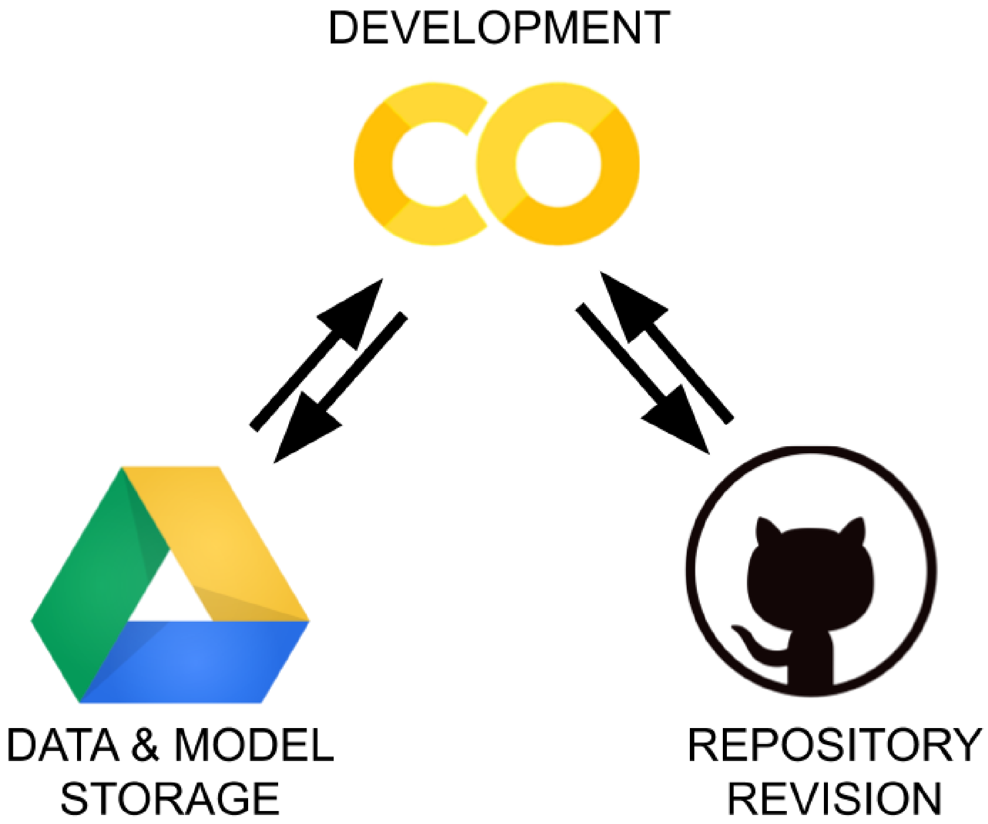
Dr. Tanujit Chakraborty

www.ctanujit.org

CONTENTS

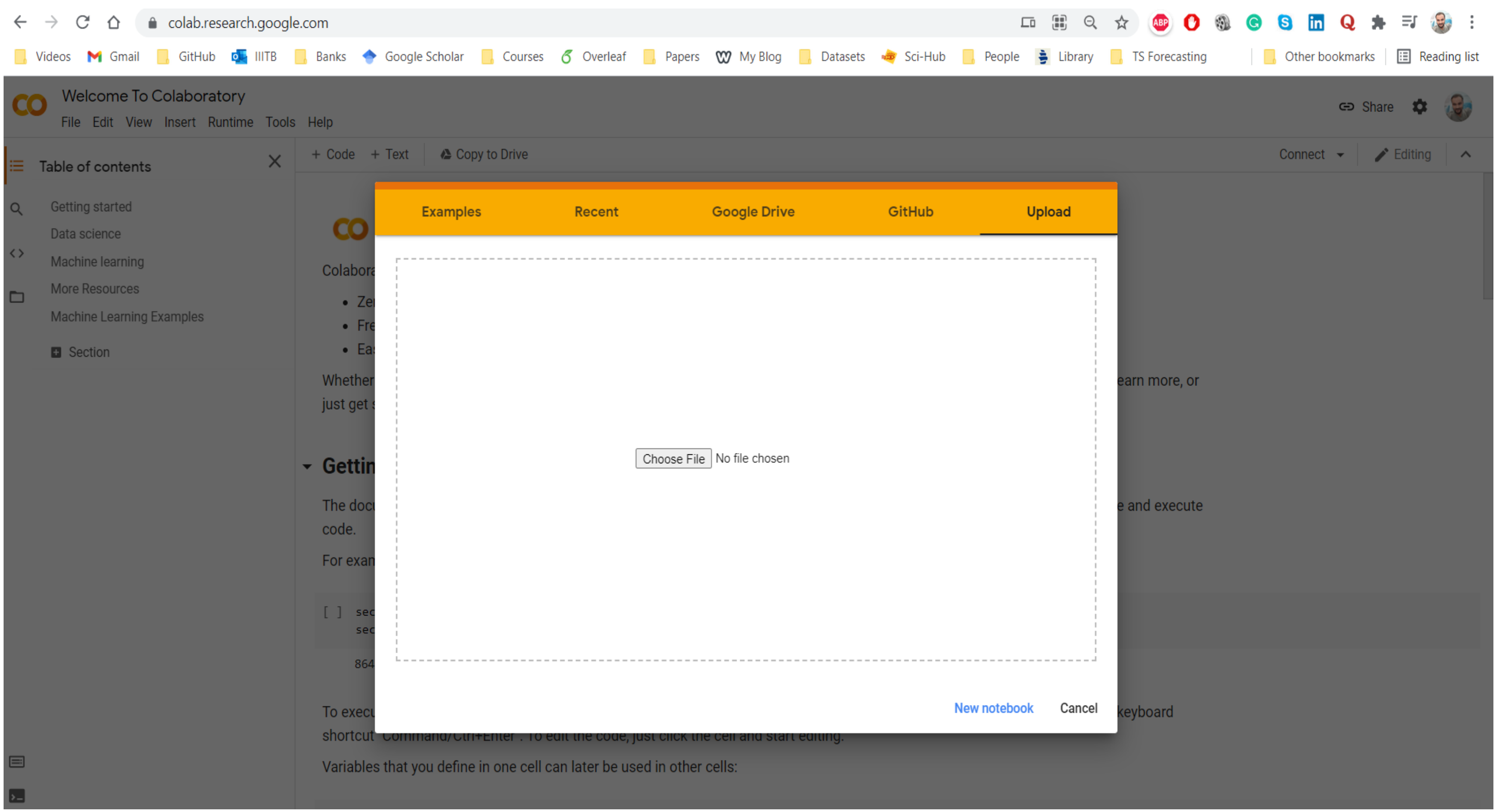
Getting started
with
Google Colab

GOOGLE COLAB



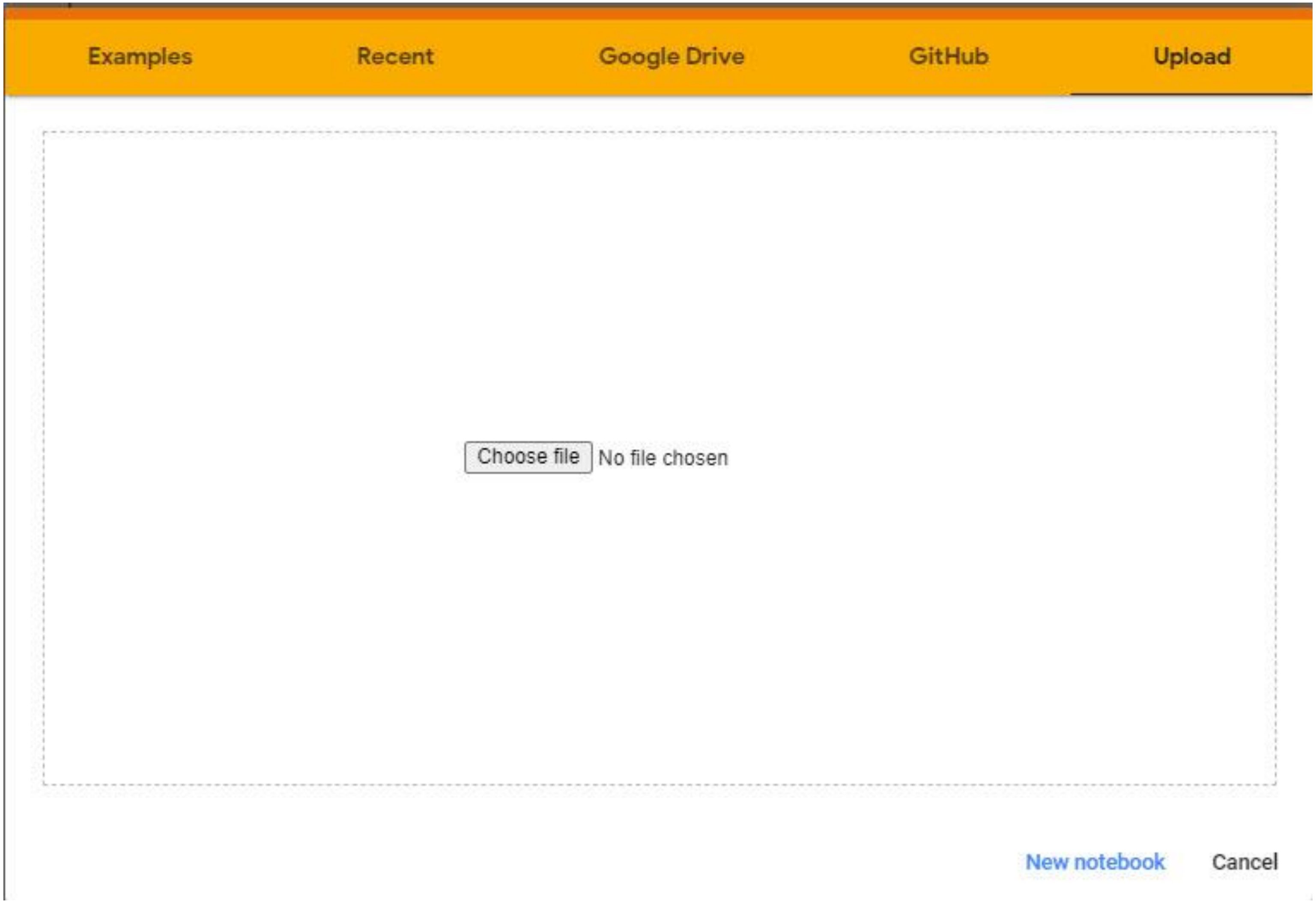
GOOGLE COLAB

Open Google Colab: https://colab.research.google.com/



GOOGLE COLAB

Upload the ipynb / py file or start with a new notebook



DESCRIPTIVE STATISTICS
using Python

DESCRIPTIVE STATISTICS

Exercise 1: The monthly credit card expenses of an individual in 1000 rupees is given in the file Credit_Card_Expenses.csv.

a. Read the dataset to Python

b. Compute mean, median minimum, maximum, range, variance, standard deviation, skewness, kurtosis and quantiles of Credit Card Expenses

c. Compute default summary of Credit Card Expenses

d. Draw Histogram of Credit Card Expenses

DESCRIPTIVE STATISTICS

Reading a csv file from local drive

```
from google.colab import files
uploaded = files.upload()
import io
import pandas as pd
data = pd.read_csv(io.BytesIO(uploaded['Credit_Card_Expenses.csv']))
data.head(5)  #shows the first 5 examples of the data
```

To read a particular column or variable of data set to a new variable

Example: Read CC_Expenses to CC

```
cc = mydata.CC_Expenses
cc
```

Operators - Arithmetic

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation
%	modulus (x mod y) 5%2 is 1

Operators - Logical

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
! =	not equal to

DESCRIPTIVE STATISTICS

Descriptive Statistics

Computation of descriptive statistics for variable CC

Function	Code	Value
Mean	cc.mean()	59.2
Median	cc.median()	59
Mode	cc.mode()	59
Standard deviation	cc.std()	3.105
Variance	cc.var()	9.642
Minimum	cc.min()	53
Maximum	cc.max()	65
Percentile	cc.quantile(0.9)	63
Skewness	cc.skew()	-0.09
Kurtosis	cc.kurt()	-0.436

DESCRIPTIVE STATISTICS

Descriptive Statistics

Arithmetic functions for variable CC

Function	Code	Value
Count	cc.count()	20
Sum	cc.sum()	1148
Product	cc.prod()	6.21447E+18

Function	Code	Value
Square root	Import math as mymath mymath.sqrt(49)	7
Sum of Squares	sum(cc**2)	70276

DESCRIPTIVE STATISTICS

Descriptive Statistics

Statistics	Code
Summary	cc.describe()

Statistics	Value
Count	20
Mean	59.2
Standard Deviation	3.1052
Minimum	53
Q1	57
Median	59
Q3	61
Maximum	65

DESCRIPTIVE STATISTICS

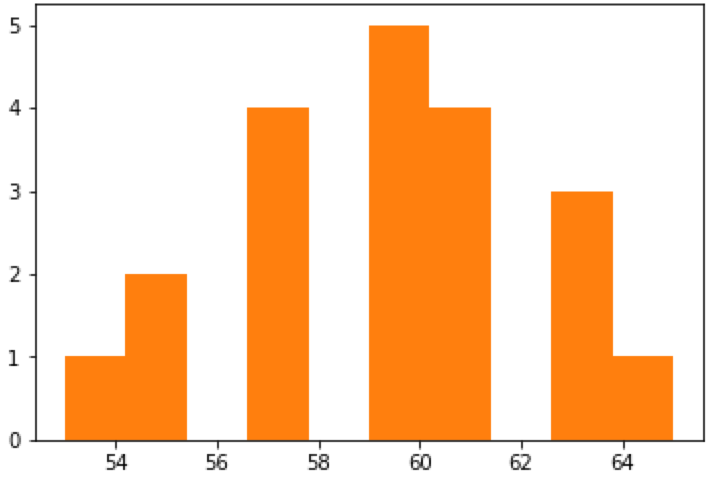
Graphs:

Graph	Code
Histogram	import matplotlib.pyplot as myplot myplot.hist(cc) myplot.show()
Box Plot	myplot.boxplot(cc) myplot.show()

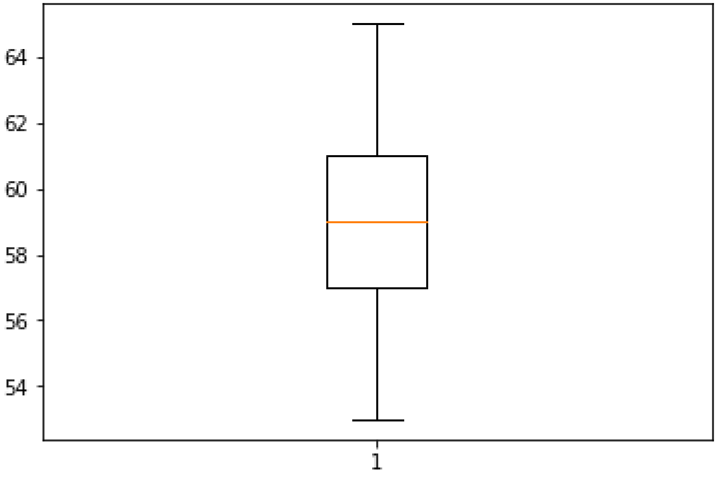
DESCRIPTIVE STATISTICS

Graphs:

Histogram



Box plot



TEST *of* HYPOTHESIS

TEST OF HYPOTHESIS

Introduction:

In many situations, it is required to accept or reject a statement or claim about some parameter

Example:

1. The average cycle time is less than 24 hours
2. The % of rejection is only 1%

The statement is called the hypothesis

The procedure for decision making about the hypothesis is called hypothesis testing

Advantages

1. Handles uncertainty in decision making
2. Minimizes subjectivity in decision making
3. Helps to validate assumptions or verify conclusions

TEST OF HYPOTHESIS

Some of the commonly used hypothesis tests:

- Checking mean equal to a specified value (mu = mu_0)
- Two means are equal or not (mu_1 = mu_2)
- Two variances are equal or not ($sigma_1^2$ = $sigma_2^2$)

- Proportion equal to a specified value ($P = P_0$)
- Two Proportions are equal or not ($P_1 = P_2$)

TEST OF HYPOTHESIS

Null Hypothesis:

A statement about the status quo

One of no difference or no effect

Denoted by H0

Alternative Hypothesis:

One in which some difference or effect is expected

Denoted by H1

TEST OF HYPOTHESIS

Types of errors in hypothesis testing

The decision procedure may lead to either of the two wrong conclusions

Type I Error

Rejecting the null hypothesis H0 when it is true

Type II Error

Failing to reject the null hypothesis H0 when it is false

Alpha (Significance level) = Probability of making type I error

Beta = Probability of making type II error

Power = 1 – Beta : Probability of correctly rejecting a false null hypothesis

TEST OF HYPOTHESIS

Hypothesis Testing: General Procedure

1. Formulate the null hypothesis H0 and the alternative hypothesis H1
2. Gather evidence (data collection)
3. Based on evidence take a decision to accept or reject H0

TEST OF HYPOTHESIS

Methodology demo: To Test Mean = Specified Value (mu = mu_0)

Suppose we want to test whether mean of a process characteristic is 5 based on the following sample data from the process

4	4	5	5	6
5	4.5	6.5	6	5.5

Calculate the mean of the sample, xbar = 5.15

Compare xbar with specified value 5

or xbar - specified value = xbar - 5 with 0

If xbar - 5 is close to 0

then conclude mean = 5

else mean $\neq$ 5

TEST OF HYPOTHESIS

Methodology demo : To Test Mean = Specified Value (mu = mu_0)

Consider another set of sample data. Check whether mean of the process characteristic is 500

400	400	500	500	600
500	450	650	600	550

Mean of the sample, xbar = 515

xbar - 500 = 515 - 500 = 15

Can we conclude mean $\neq$ 500?

Conclusion:

Difficult to say mean = specified value by looking at xbar - specified value alone

TEST OF HYPOTHESIS

Methodology demo: To Test Mean = Specified Value (mu = mu_0)

Test statistic is calculated by dividing (xbar - specified value) by a function of standard deviation

To test Mean = Specified value

Test Statistic t_0 = (xbar - Specified value) / (SD / $\sqrt{n}$)

If test statistic is close to 0, conclude that Mean = Specified value

To check whether test statistic is close to 0, find out p value from the sampling distribution of test statistic

TEST OF HYPOTHESIS

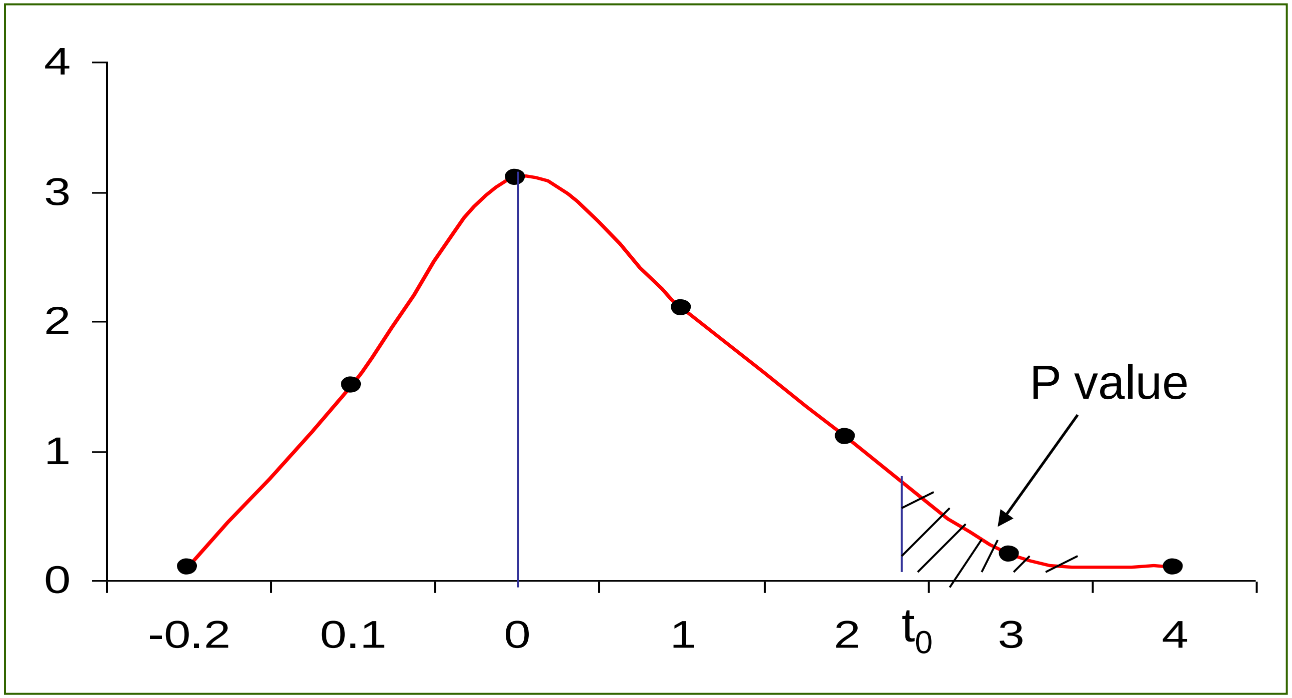
Methodology demo: To Test Mean = Specified Value

P value

The probability that such evidence or result will occur when H0 is true

Based on the reference distribution of test statistic

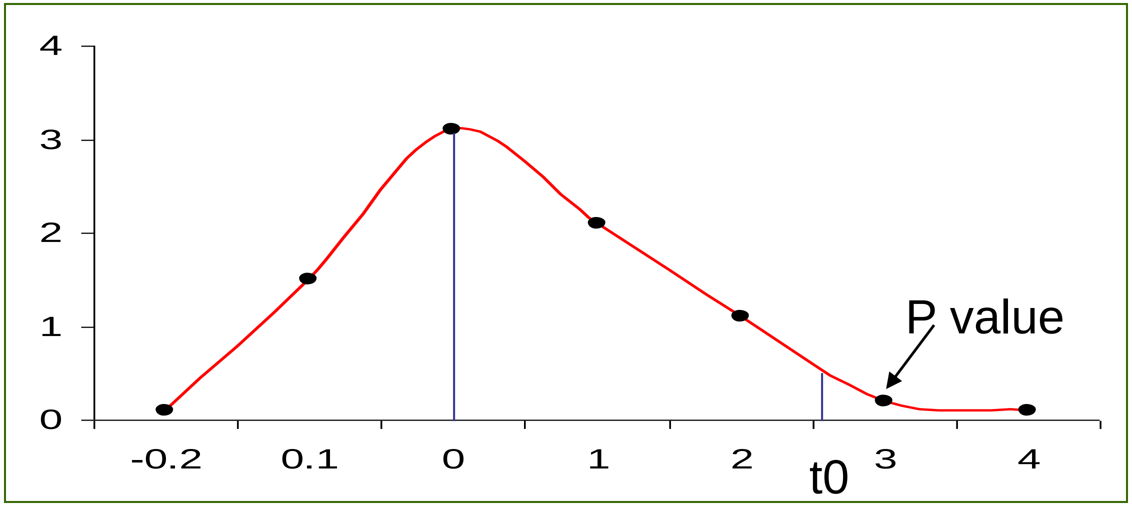
The tail area beyond the value of test statistic in reference distribution



TEST OF HYPOTHESIS

Methodology demo : To Test Mean = Specified Value

P value



If test statistic t_0 is close to 0 then p will be high

If test statistic t_0 is not close to 0 then p will be small

If p is small , $p < 0.05$ (with alpha = 0.05), conclude that $t \neq 0$, then

Mean $\neq$ Specified Value, H0 rejected

TEST OF HYPOTHESIS

To Test Mean = Specified Value (mu = mu_0)

Example: Suppose we want to test whether mean of the process characteristic is 5 based on the following sample data

4	4	5	5	6
5	4.5	6.5	6	5.5

H0: Mean = 5

H1: Mean $\neq$ 5

Calculate xbar = 5.15

SD = 0.8515

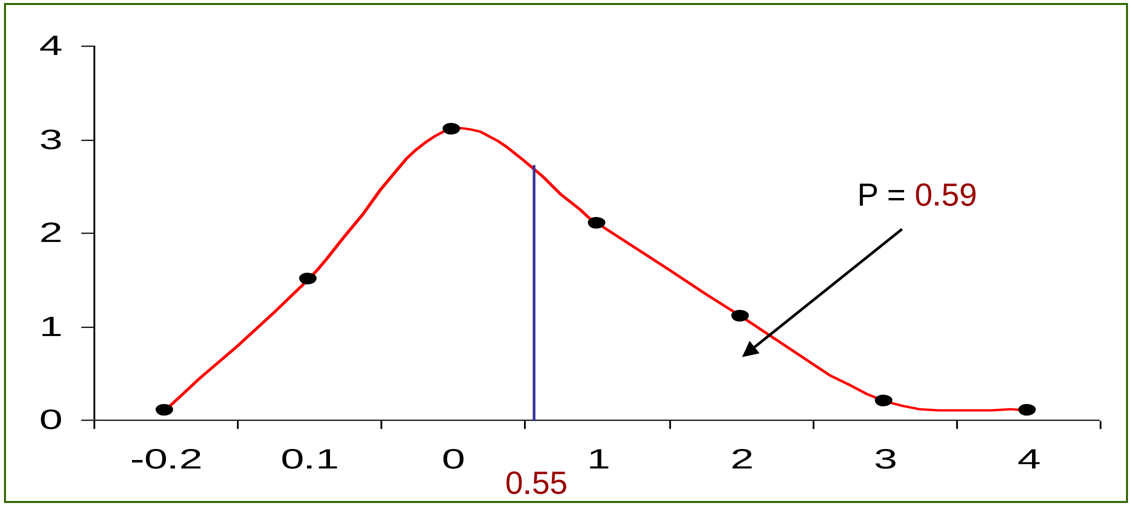
n = 10

Test statistic t_0 = (xbar - 5)/(SD / $\sqrt{}$n) = (5.15 - 5) / (0.8515 / $\sqrt{}$10) = 0.5571

TEST OF HYPOTHESIS

Example: To Test Mean = Specified Value (mu = mu_0)

$t_0 = 0.5571$



P ≥ 0.05, hence Mean = Specified value = 5.

H0: Mean = 5 is not rejected

TEST OF HYPOTHESIS

Hypothesis Testing: Steps

1. Formulate the null hypothesis H0 and the alternative hypothesis H1
2. Select an appropriate statistical test and the corresponding test statistic
3. Choose level of significance alpha (generally taken as 0.05)
4. Collect data and calculate the value of test statistic
5. Determine the probability associated with the test statistic under the null hypothesis using sampling distribution of the test statistic
6. Compare the probability associated with the test statistic with level of significance specified

TEST OF HYPOTHESIS

One sample t test

Exercise 1 : A company claims that on an average it takes only 40 hours to process any purchase order. Based on the data given below, can you validate the claim? The data is given in PO_Processing.csv

One sample t test

Exercise 1 : A company claims that on an average it takes only 40 hours to process any purchase order. Based on the data given below, can you validate the claim? The data is given in PO_Processing.csv

Reading data to data

```
from google.colab import files
uploaded = files.upload()
import io
import pandas as pd
data = pd.read_csv(io.BytesIO(uploaded['PO_Processing.csv']))
PT = data.Processing_Time
```

Performing one sample t test

```
stats.ttest_1samp(PT, 40)
```

TEST OF HYPOTHESIS

One sample t test

Exercise 1 : A company claims that on an average it takes only 40 hours to process any purchase order. Based on the data given below, can you validate the claim? The data is given in PO_Processing.csv

Statistics	Value
t	3.7031
P value	0.00035

TEST OF HYPOTHESIS

To Test Two Means are Equal:

Null hypothesis H0: $Mean_1 = Mean_2$ ($mu_1 = mu_2$)

Alternative hypothesis H1: $Mean_1 \neq Mean_2$ ($mu_1 \neq mu_2$)

or

H1: $Mean_1 > Mean_2$ ($mu_1 > mu_2$)

or

H1: $Mean_1 < Mean_2$ ($mu_1 < mu_2$)

TEST OF HYPOTHESIS

To Test Two Means are Equal: Methodology

Calculate both sample means xbar1 & xbar2

Calculate SD1 & SD2

Compare xbar1 with xbar2

Or xbar1 - xbar2 with 0

Calculate test statistic t_0 by dividing (xbar1 – xbar2) by a function of SD1 & SD2

$$t_0 = (\text{xbar1} - \text{xbar2}) / (Sp \sqrt{((1/n1)+(1/n2))})$$

Calculate p value from t distribution

If $p \geq 0.05$ then H0: $\text{Mean}_1 = \text{Mean}_2$ is not rejected

Two sample t test

Exercise 1: A super market chain has introduced a promotional activity in its selected outlets in the city to increase the sales volume. Based on the data given below, check whether the promotional activity resulted in increasing the sales. The outlets where promotional activity introduced are denoted by 1 and others by 2? The data is given in Sales_Promotion.csv

Outlet	Sales	Outlet	Sales
1	1217	2	1731
1	1416	2	1420
1	1381	2	1065
1	1413	2	1612
1	1800	2	1361
1	1724	2	1259
1	1310	2	1470
1	1616	2	622
1	1941	2	1711
1	1792	2	2315
1	1453	2	1180
1	1780	2	1515

TEST OF HYPOTHESIS

Two sample t test

Exercise 1: A super market chain has introduced a promotional activity in its selected outlets in the city to increase the sales volume. Based on the data given below, check whether the promotional activity resulted in increasing the sales. The outlets where promotional activity introduced are denoted by 1 and others by 2?

Reading data to mydata

```
from google.colab import files
uploaded = files.upload()
import io
import pandas as mypd
from scipy import stats
mydata = mypd.read_csv(io.BytesIO(uploaded['Sales_Promotion.csv']))
mydata
```

Reading the variables

```
sales_1 = mydata.Sales_Out1
sales_2 = mydata.Sales_Out2
```

Two sample t test

Exercise 1: A super market chain has introduced a promotional activity in its selected outlets in the city to increase the sales volume. Based on the data given below, check whether the promotional activity resulted in increasing the sales. The outlets where promotional activity introduced are denoted by 1 and others by 2?

2 sample t Test

```
stats.ttest_ind(sales_1, sales_2)
```

Statistics	Value
t	0.9625
p value	0.3463

TEST OF HYPOTHESIS

Paired t test:

A special case of two sample t test

When observations on two groups are collected in pairs

Each pair of observation is taken under homogeneous conditions

Procedure

Compute d: difference in paired observations

Let difference in means be $\mu_D = \mu_1 - \mu_2$

Null hypothesis H0: $\mu_D = 0$

Alternative hypothesis H1: $\mu_D \neq 0$ or $\mu_D > 0$ or $\mu_D < 0$

Test statistics t0 = $\dfrac{\bar{d}}{s_d / \sqrt{n}}$

Reject H0 if p – value < 0.05

Paired t test: Exercise 1

The manager of a fleet of automobiles is testing two brands of radial tires. He assigns one tire of each brand at random to the two rear wheels of eight cars and runs the cars until the tire wear out. Is both brands have equal mean life? The data in kilometers is given in tires.csv

Brand 1	Brand 2
36925	34318
45300	42280
36240	35500
32100	31950
37210	38015
48360	47800
38200	37810
33500	33215

TEST OF HYPOTHESIS

Paired t test: Exercise 1

The manager of a fleet of automobiles is testing two brands of radial tires. He assigns one tire of each brand at random to the two rear wheels of eight cars and runs the cars until the tire wear out. Is both brands have equal mean life? The data in kilometers is given in tires.csv

Reading the file and variables

```
from google.colab import files
uploaded = files.upload()
import io
import pandas as mypd
from scipy import stats
mydata = mypd.read_csv(io.BytesIO(uploaded['Tires.csv']))
b1 = mydata. Brand_1
b2 = mydata.Brand_2
```

Paired t test

```
stats.ttest_rel(b1,b2)
```

Statistics	Value
t	1.9039
P value	0.09863

NORMALITY TEST

NORMALITY TEST

Normality test

A methodology to check whether the characteristic under study is normally distributed or not

Two Methods

1. Quantile – Quantile (Q- Q) plot
2. Shapiro – Wilk test

NORMALITY TEST

Normality test - Quantile – Quantile (Q- Q) plot

- Plots the ranked samples from the given distribution against a similar number of ranked quantiles taken from a normal distribution
- If the sample is normally distributed then the line will be straight in the plot

NORMALITY TEST

Normality test – Shapiro – Wilk test

H0: Deviation from bell shape (normality) = 0

H1 : Deviation from bell shape $\neq$ 0

If p value $\geq$ 0.05 (5%), then H0 is not rejected, distribution is normal

NORMALITY TEST

Normality test

Exercise 1 : The processing times of purchase orders is given in PO_Processing.csv. Is processing time normally distributed?

Reading the data and variable

```
from google.colab import files
uploaded = files.upload()
import io
import pandas as pd
from scipy import stats
import matplotlib.pyplot as myplot
data = pd.read_csv(io.BytesIO(uploaded['PO_Processing.csv']))
PT = data.Processing_Time
```

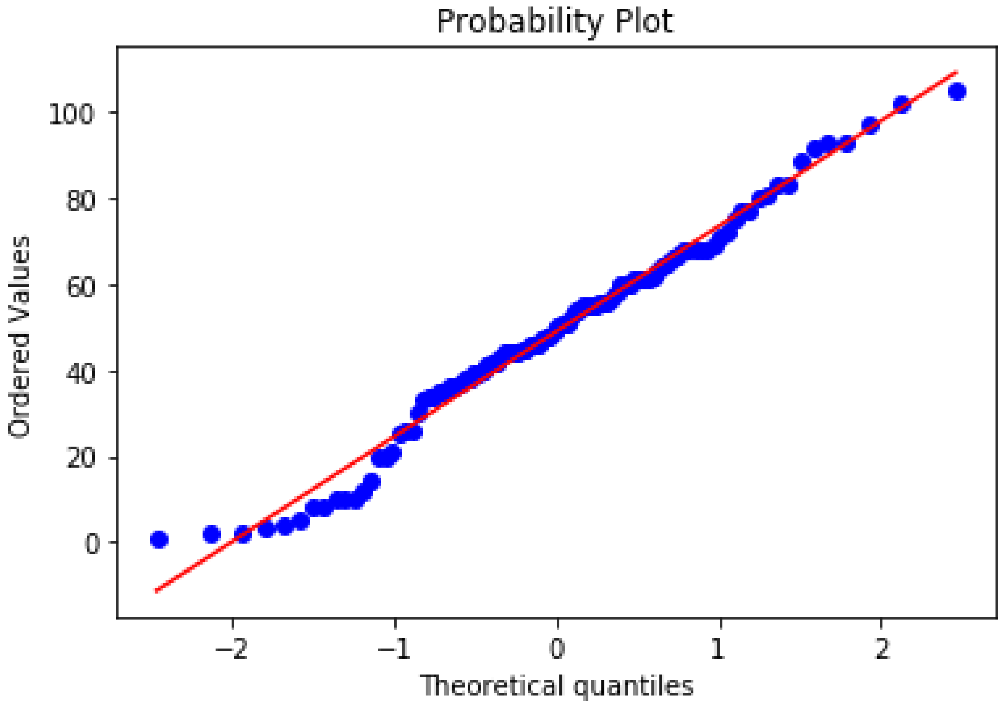
NORMALITY TEST

Normality test

Exercise 1 : The processing times of purchase orders is given in PO_Processing.csv. Is processing time normally distributed?

Normality Check using Normal Q – Q plot

```
stats.probplot(PT, plot = myplot)
myplot.show()
```



NORMALITY TEST

Normality test

Exercise 1 : The processing times of purchase orders is given in PO_Processing.csv. Is processing time normally distributed?

Normality test

```
stats.mstats.normaltest(PT)
```

Statistics	Value
W	0.33965
p value	0.84381

ANALYSIS
of
VARIANCE

ANALYSIS OF VARIANCE

ANOVA

Analysis of Variance is a test of means for two or more populations

Partitions the total variability in the variable under study to different components

H0 = $Mean_1$ = $Mean_2$ = - - - = $Mean_k$

Reject H0 if p – value < 0.05

Example:

To study location of shelf on sales revenue

One Way Anova : Example

An electronics and home appliance chain suspect the location of shelves where television sets are kept will influence the sales revenue. The data on sales revenue in lakhs from the television sets when they are kept at different locations inside the store are given in sales revenue data file. The location is denoted as 1:front, 2: middle & 3: rear. Verify the doubt? The data is given in Sales_Revenue_Anova.csv.

One Way Anova : Example

Factor: Location(A)

Levels : front, middle, rear

Response: Sales revenue

One Way Anova : Example

Step 1: Calculate the sum, average and number of response values for each level of the factor (location).

Level 1 Sum(A_1):

Sum of all response values when location is at level 1 (front)

$$= 1.55 + 2.36 + 1.84 + 1.72$$

$$= 7.47$$

nA_1: Number of response values with location is at level 1 (front)

$$= 4$$

One Way Anova : Example

Step 1: Calculate the sum, average and number of response values for each level of the factor (location).

Level 1 Average:

Sum of all response values when location is at level 1 / number of response values with location is at level 1

$= A_1 / nA_1 = 7.47 / 4 = 1.87$

ANALYSIS OF VARIANCE

One Way Anova : Example

Step 1: Calculate the sum, average and number of response values for each level of the factor (location).

	Level 1 (front)	Level 2 (middle)	Level 3 (rear)
Sum	A_1: 7.47	A_2: 30.31	A_3: 15.55
Number	nA_1: 4	nA_2: 8	nA_3: 6
Average	1.87	3.79	2.59

ANALYSIS OF VARIANCE

One Way Anova : Example

Step 2: Calculate the grand total (T)

T = Sum of all the response values

= 1.55 + 2.36 + - - - + 2.72 + 2.07 = 53.33

Step 3: Calculate the total number of response values (N)

N = 18

Step 4: Calculate the Correction Factor (CF)

CF = $(\text{Grand Total})^2$ / Number of Response values

$= T^2 / N = (537.33)^2 / 18 = 158.0049$

One Way Anova : Example

Step 5: Calculate the Total Sum of Squares (TSS)

TSS = Sum of square of all the response values - CF

$$= 1.55^2 + 2.36^2 + - - - + 2.72^2 + 2.07^2 – 158.0049$$

$$= 15.2182$$

One Way Anova : Example

Step 6: Calculate the between (factor) sum of square

$$SS_A = A_1^2 / nA_1 + A_2^2 / nA_2 + A_3^2 / nA_3 - CF$$

$$= 7.47^2 / 4 + 30.31^2 / 8 + 15.55^2 / 4 – 158.0049$$

$$= 11.0827$$

Step 7: Calculate the within (error) sum of square

$$SS_e = \text{Total sum of square – between sum of square}$$

$$= TSS - SS_A = 15.2182 – 11.0827 = 4.1354$$

ANALYSIS OF VARIANCE

One Way Anova : Example

Step 8: Calculate degrees of freedom (df)

Total df = Total Number of response values - 1

= 18 - 1 = 17

Between df

= Number of levels of the factor - 1

= 3 - 1 = 2

Within df = Total df – Between df

= 17 - 2 = 15

ANALYSIS OF VARIANCE

One Way Anova : Example

Anova Table:

Source	df	SS	MS	F	F Crit	P value
Between	2	11.08272	5.541358	20.09949	3.68	0.0000
Within	15	4.135446	0.275696			
Total	17	15.21816				

$MS = SS / df$

$F = MS_{Between} / MS_{Within}$

F Crit =finv (probability, between df, within df) , probability = 0.05

P value = fdist (F, between df, within df)

ANALYSIS OF VARIANCE

One Way Anova : Python Code

Import the packages

```
from google.colab import files
import io
import pandas as mypd
from scipy import stats
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
```

Import data

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Sales_Revenue_Anova.csv']))
sales = mydata.Sales_Revenue
location = mydata.Location
```

ANALYSIS OF VARIANCE

One Way Anova :

Computing ANOVA table

```
mymodel = ols('sales ~ C(location)', mydata).fit()
anova_table = anova_lm(mymodel)
anova_table
```

	df	SS	MS	F	p-value
Location	2	11.08272	5.541358	20.09949	5.7E-05
Residual	15	4.135446	0.275696		

ANALYSIS OF VARIANCE

One Way Anova : Decision Rule

If p value < 0.05, then

The factor has significant effect on the process output or response.

Meaning:

When the factor is changed from 1 level to another level, there will be significant change in the response.

One Way Anova : Example Result

For factor Location, $p = 0.000 < 0.05$

Conclusion:

Location has significant effect on sales revenue

Meaning:

The sales revenue is not same for different locations like front, middle & rear.

ANALYSIS OF VARIANCE

One Way Anova : Example Result

The expected sales revenue for different location under study is equal to level averages.

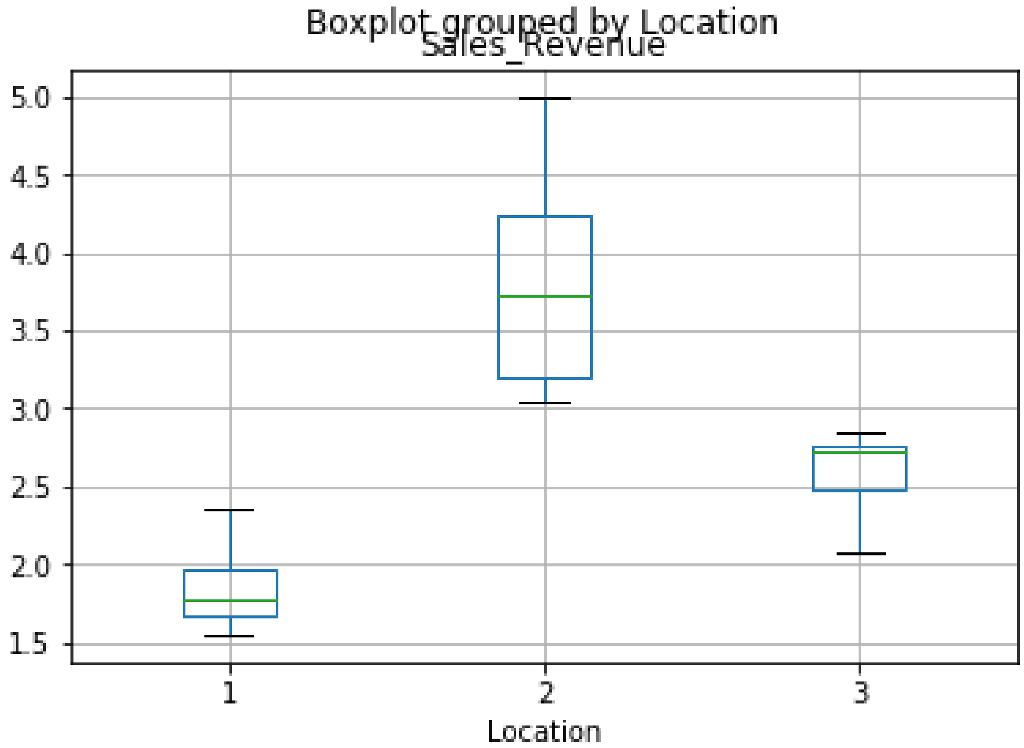
Location	Expected Sales Revenue
Front	1.8675
Middle	3.78875
Rear	2.591667

```
sales.groupby(location).mean()
```

ANALYSIS OF VARIANCE

One Way Anova : Example Result

```
import matplotlib.pyplot as myplot
mydata.boxplot(column ='Sales_Revenue', by = 'Location')
myplot.show()
```



MULTIPLE REGRESSION ANALYSIS

CORRELATION & REGRESSION

Regression

Correlation helps

To check whether two variables are related

If related

Identify the type & degree of relationship

CORRELATION & REGRESSION

Regression

Regression helps

- To identify the exact form of the relationship
- To model output in terms of input or process variables

Examples:

Expected (Yield) = 5 + 3 x Time - 2 x Temperature

Simple Linear Regression Illustration

Output variable is modeled in terms of only one variable

x	y
2	7
1	4
5	16
4	13
3	10
6	19

Regression Model

$y = 1 + 3x$

Simple Linear Regression

General Form:

$$y = a + bx + \varepsilon$$

where

a: intercept (the value of y when x is equal to 0)

b: slope (indicates the amount of change in y with every unit change in x)

CORRELATION & REGRESSION

Simple Linear Regression: Parameter Estimation

Model: $y = a + bx + \varepsilon$

$$\hat{a} = \bar{y} - \hat{b}\bar{x}$$

$$\hat{b} = S_{xy} / S_{xx}$$

Test for Significance (Testing b = 0 or not) of relation between x & y

H0: b = 0

H1: $b \neq 0$

Test Statistic $t_0 = (\hat{b} - 0)/se(\hat{b})$

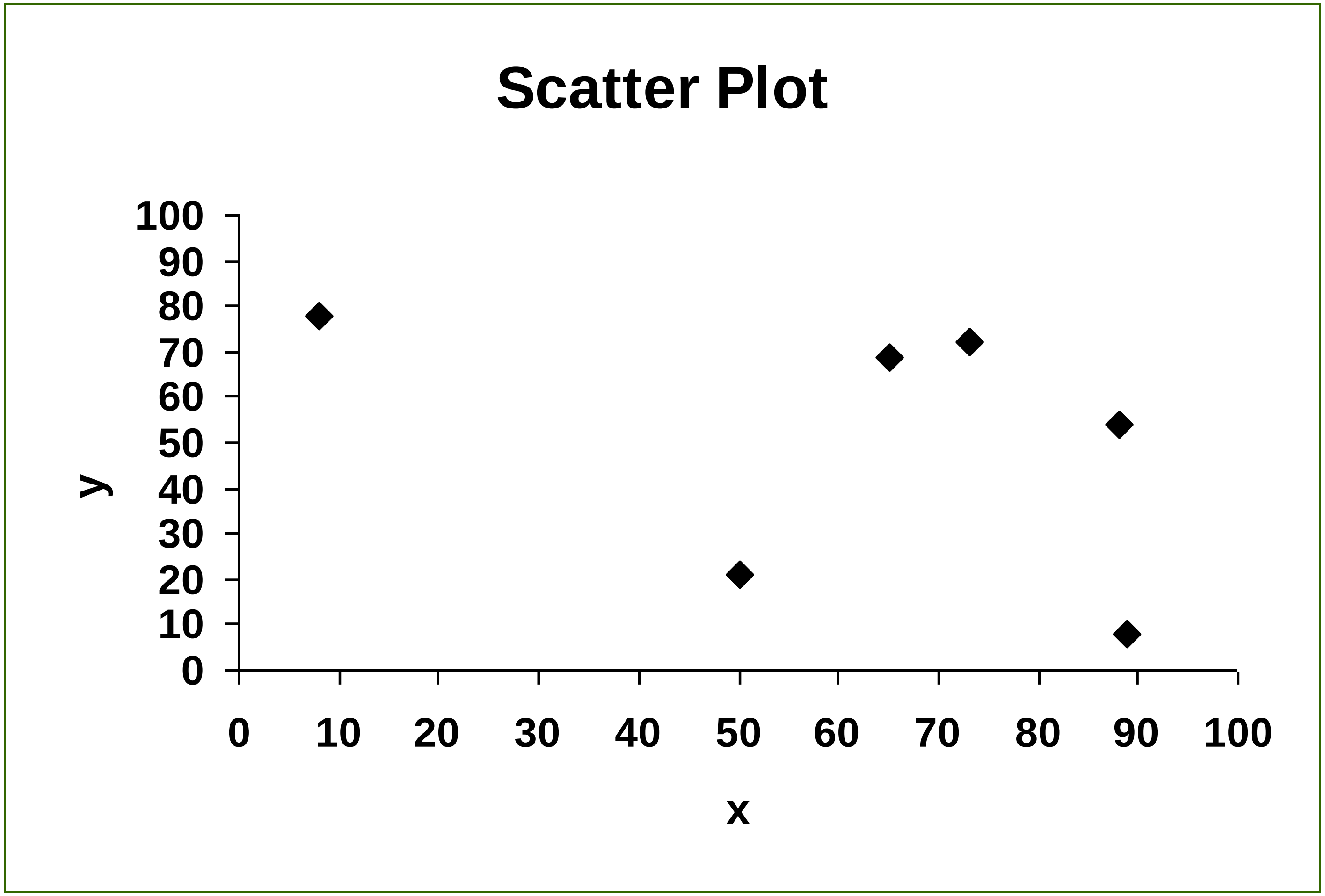
If p value < 0.05, then H0 is rejected & y can be modeled with x

Regression illustration: Issues

x	y
65	69
8	78
89	8
88	21
50	24
73	72

CORRELATION & REGRESSION

Regression Model $y = 76.32 - 0.42x + \varepsilon$



Regression: Issues

For any set of data,

a & b can be calculated

Regression model $y = a + bx + \varepsilon$ can be build

But all the models may not be useful

CORRELATION & REGRESSION

Coefficient of Regression: Measure of degree of Relationship

Symbol : R^2

$$R^2 = SS_R / Syy = b.Sxy / Syy$$

$$SS_R = \Sigma(y_{predicted} - \text{Mean } y)^2$$

$$Syy = \Sigma(y_{actual} - \text{Mean } y)^2$$

R^2 : amount variation in y explained by x

Range of R^2 : 0 to 1

If $R^2 \geq 0.6$, the model is reasonably good

Coefficient of Regression: Testing the significance of Regression

Regression ANOVA

Model	SS	df	MS	F	p value
Regression	SS_R				
Residual	$Syy – SS_R$				
Total	Syy				

If p value < 0.05, then the regression model is significant

REGRESSION ANALYSIS

Multiple Linear Regression

To model output variable y in terms of two or more variables.

General Form:

$$y = a + b_1x_1 + b_2x_2 + - - - + b_kx_k + \varepsilon$$

Two variable case:

$$y = a + b_1x_1 + b_2x_2 + \varepsilon$$

Where

a: intercept (the predicted value of y when all x’s are zero)

b_j: slope (the amount change in y for unit change in x_j keeping all other x’s constant, j = 1,2,---,k)

REGRESSION ANALYSIS

Exercise : The effect of temperature and reaction time affects the % yield. The data collected in given in the Mult-Reg_Yield file. Develop a model for % yield in terms of temperature and time?

Step 1: Import packages

```
from google.colab import files
import io
import pandas as mypd
from scipy import stats
import matplotlib.pyplot as myplot
import math as mymath
from pandas.plotting import scatter_matrix
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
```

REGRESSION ANALYSIS

Exercise : The effect of temperature and reaction time affects the % yield. The data collected in given in the Mult-Reg_Yield file. Develop a model for % yield in terms of temperature and time?

Step 2: Read Data

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Mult_Reg_Yield.csv']))
mydata.head()
time = mydata.Time
temp = mydata.Temperature
output = mydata["Yield"]
```

REGRESSION ANALYSIS

Exercise : The effect of temperature and reaction time affects the % yield. The data collected in given in the Mult-Reg_Yield file. Develop a model for % yield in terms of temperature and time?

Step 1: Correlation Analysis

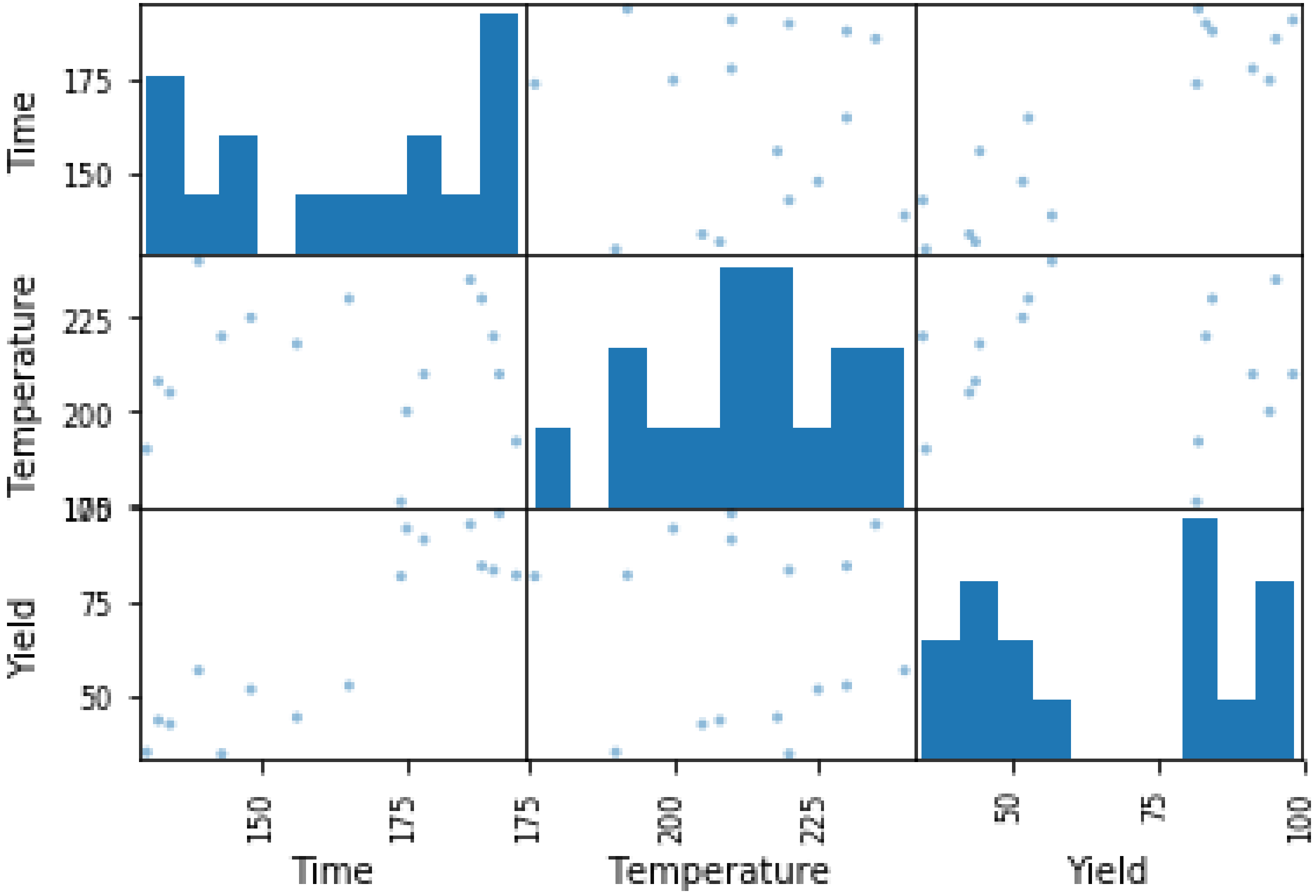
```
scatter_matrix(mydata)
myplot.show()
```

Correlation between xs & y should be high

Correlation between xs should be low

REGRESSION ANALYSIS

Exercise : The effect of temperature and reaction time affects the % yield. The data collected in given in the Mult-Reg_Yield file. Develop a model for % yield in terms of temperature and time?



REGRESSION ANALYSIS

Step 2: Regression Output

```
mymodel = ols("output ~ time + temp", mydata).fit()

mymodel.summary()
```

Statistics	Value	Criteria
R-squared:	0.806	≥ 0.6
Adj. R-squared:	0.777	≥ 0.6
F-statistic:	27.07	
Prob (F-statistic):	2.32e-05	< 0.05
Log-Likelihood:	-59.703	
AIC:	125.4	
BIC:	127.7	

REGRESSION ANALYSIS

Step 2: Regression Output

```
anova_table = anova_lm(mymodel)
anova_table
```

	df	SS	MS	F	p-value
Time	1	6777.81	6777.81	53.98722	0.000006
Temp	1	19.25253	19.25253	0.153352	0.701696
Residual	13	1632.081	125.5447		

Criteria: p value < 0.05

REGRESSION ANALYSIS

Step 2: Regression Output

Regression ANOVA

Model	SS	df	MS	F	p value
Regression	6797.063	2	3398.531	27.07	0.0000
Residual	1632.08138	13	125.5447		
Total	8429.14438	15			

Criteria: P value < 0.05

REGRESSION ANALYSIS

Step 2: Regression Output – Identify the model

	Coefficients	Std error	t	p-value	[0.025	0.975]
Intercept	-67.8844	40.587	-1.673	0.118	-155.57	19.797
Time	0.9061	0.123	7.344	0.000	0.64	1.173
Temp	-0.0642	0.164	-0.392	0.702	-0.418	0.29

Interpretation: Only time is related to yield or output as p value < 0.05

REGRESSION ANALYSIS

Step 2: Regression Output – Identify the model

	Coefficients	Std error	t	p-value	[0.025	0.975]
Intercept	- 81.6205	19.791	-4.124	0.001	-124.067	-39.174
Time	0.9065	0.120	7.580	0.000	0.650	1.163

Model Yield= 0.9065 x Time - 81.621

Statistics	Value	Criteria
R-squared:	0.804	≥ 0.6
Adj. R-squared:	0.79	≥ 0.6
F-statistic:	57.46	
Prob (F-statistic):	2.55e-06	< 0.05

REGRESSION ANALYSIS

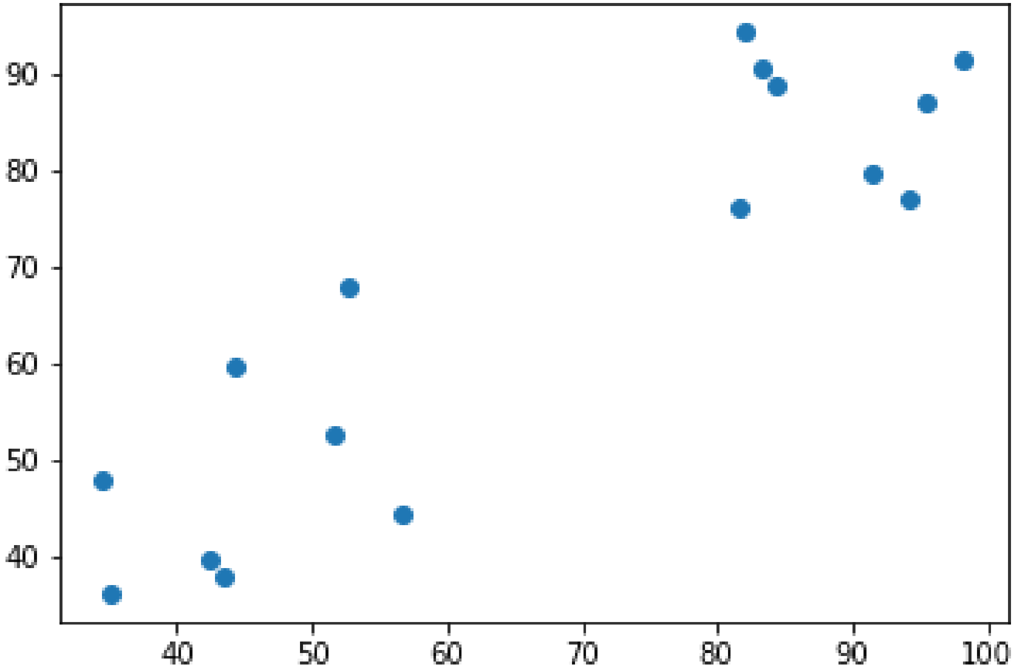
Step 3: Residual Analysis

```
pred = mymodel.predict()
res = output - pred
```

SL No	Actual	Predicted	Residuals
1	35	36.22	-1.22
2	81.7	76.10	5.60
3	42.5	39.84	2.66
4	98.3	91.51	6.79
5	52.7	67.94	-15.24
6	82	94.23	-12.23
7	34.5	48.00	-13.50
8	95.4	86.98	8.42
9	56.7	44.38	12.32
10	84.4	88.79	-4.39
11	94.3	77.01	17.29
12	44.3	59.79	-15.49
13	83.3	90.61	-7.31
14	91.4	79.73	11.67
15	43.5	38.03	5.47
16	51.7	52.53	-0.83

Step 3: Residual Analysis – Actual Vs Fitted

```
myplot.scatter(output, pred)
myplot.show()
```



Note: There need to be strong positive correlation between actual and fitted response

Step 3: Residual Analysis: Normality test

```
stats.mstats.normaltest(res)
```

Normality Test: Yield data	
W	p value
1.9835	0.3709

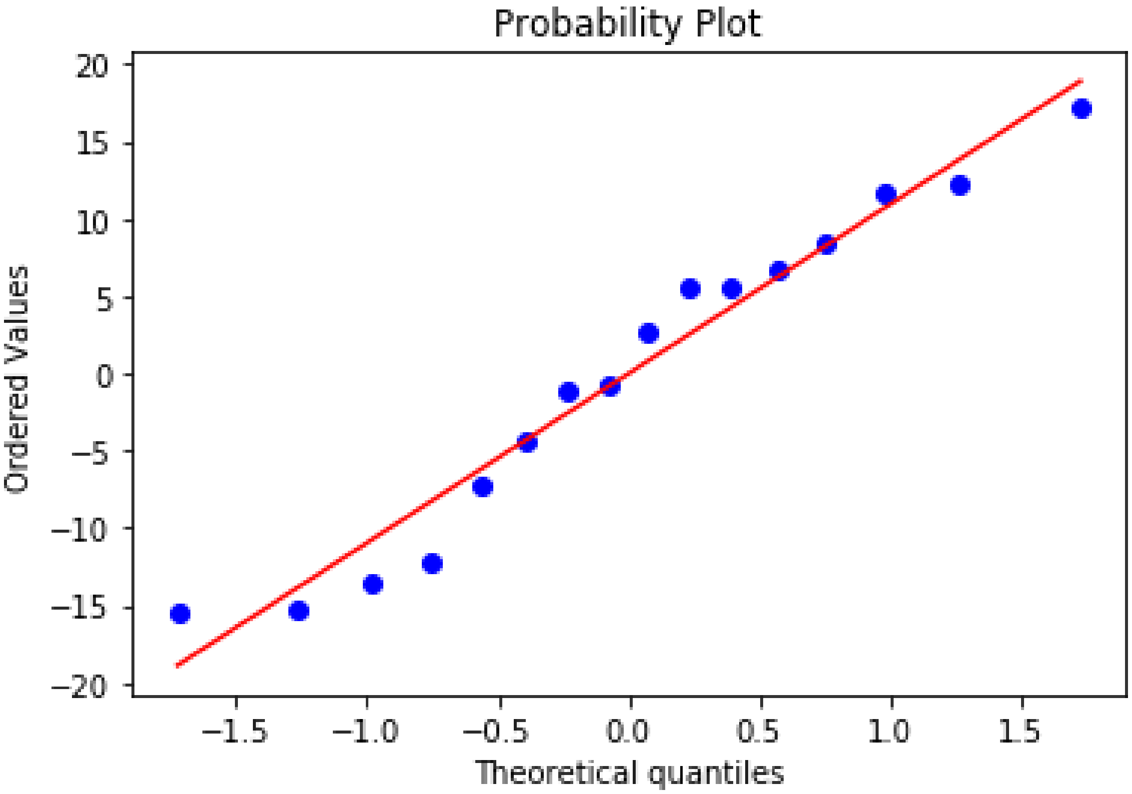
```
res_sq = res**2
mse = res_sq.mean()
print(mse)
import math as mymath
rmse = mymath.sqrt(mse)
print(rmse)
```

Statistic	Value
MSE	102.005
RMSE	10.099

REGRESSION ANALYSIS

7: Residual Analysis: Normality Plot

```
stats.probplot(res, plot = myplot)
myplot.show
```

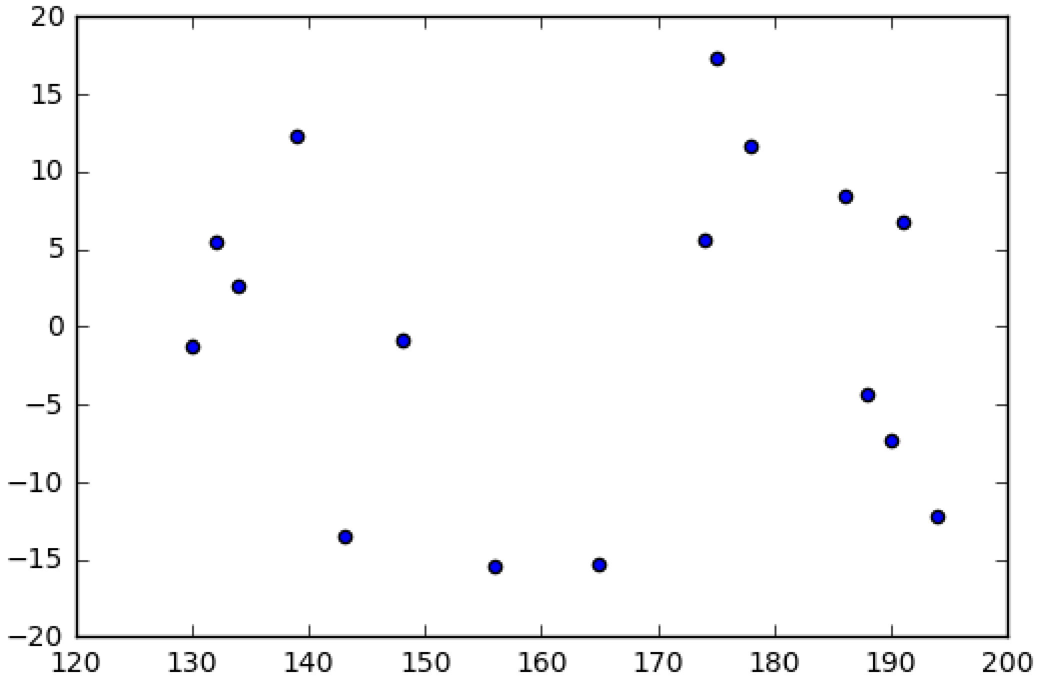


REGRESSION ANALYSIS

7: Model adequacy check

Residuals Vs Independent variables

```
myplot.scatter(time, res)
myplot.show()
```

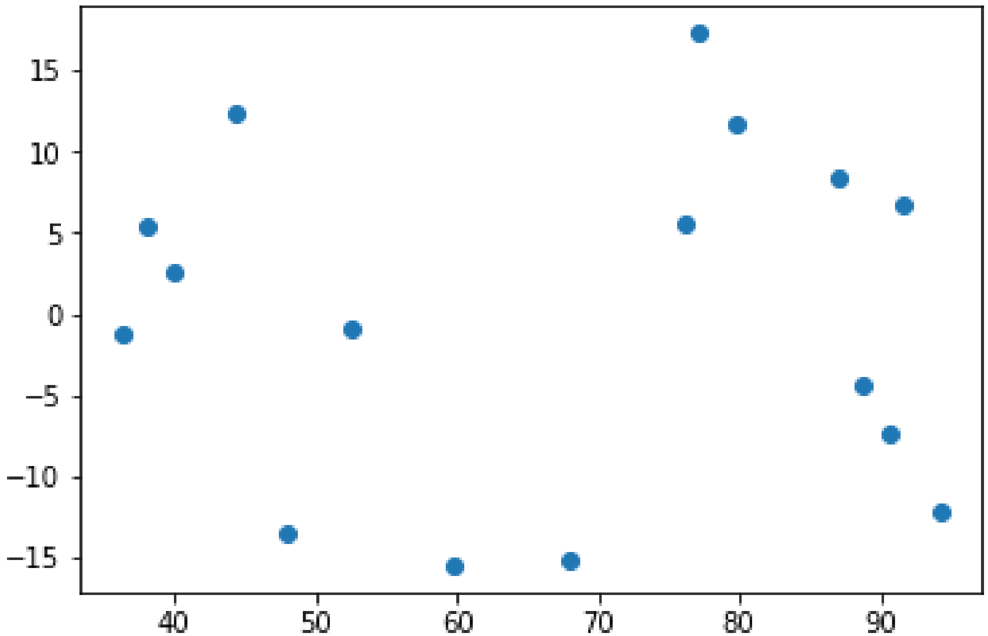


Note: There should not be any pattern or trend, the points should be distributed randomly

7: Model adequacy check

Residuals Vs Fitted

```
myplot.scatter(pred, res)
myplot.show()
```



Note: There should not be any pattern or trend, the points should be distributed randomly

Regression with dummy variables

When x's are not numeric but nominal each nominal or categorical variable is converted into dummy variables

Dummy variables takes values 0 or 1

Number of dummy variable for one x variable is equal to number of distinct values of that variable - 1

Example: A study was conducted to measure the effect of gender and income on attitude towards vocation. Data was collected from 30 respondents and is given in Travel_dummy_reg file. Attitude towards vocation is measured on a 9 point scale. Gender is coded as male = 1 and female = 2. Income is coded as low=1, medium = 2 and high = 3. Develop a model for attitude towards vocation in terms of gender and Income?

Regression with dummy variables

Variable		Dummy
Gender	Code	gender_Code
Male	1	0
Female	2	1

Variable		Dummy	
Income	Code	Income1	Income 2
Low	1	0	0
Medium	2	1	0
High	3	0	1

Regression with dummy variables

Import packages

```
from google.colab import files
import io
import pandas as mypd
from scipy import stats
import matplotlib.pyplot as myplot
import math as mymath
from pandas.plotting import scatter_matrix
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
mydata = mypd.read_csv("E:/ISI/Data/Travel_dummy_Reg.csv")
gender = mydata.Gender
income = mydata.Income
attitude = mydata.Attitude
```

Regression with dummy variables

Read the fie and variables

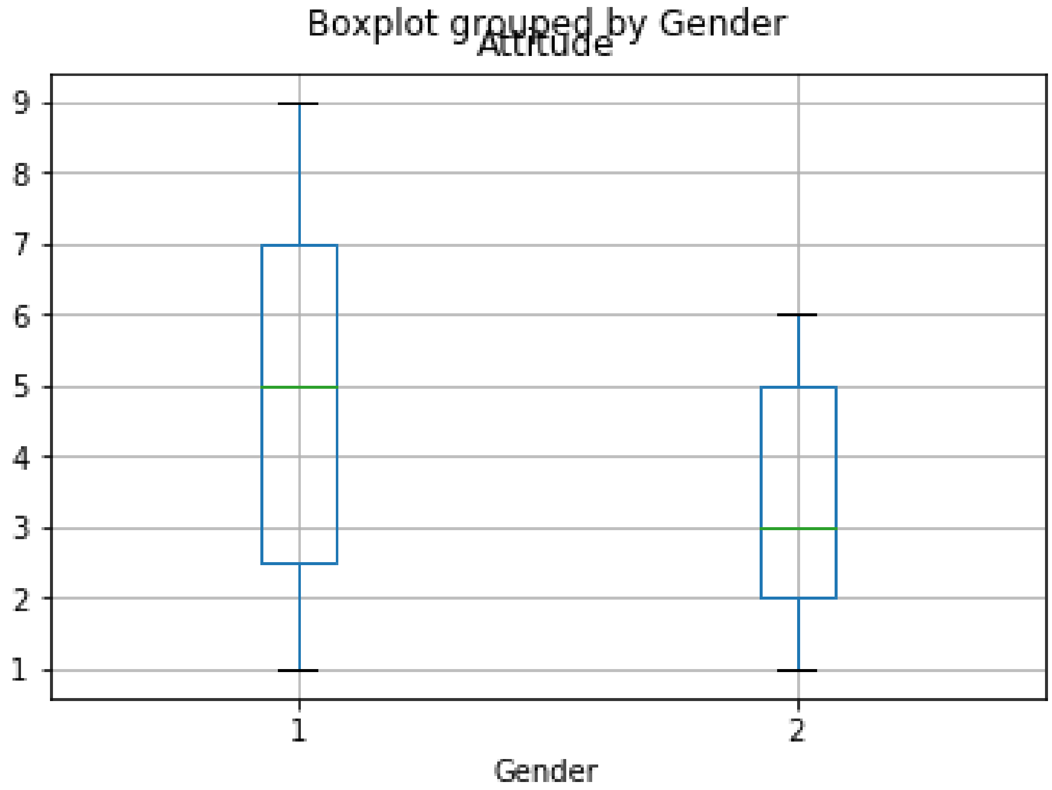
```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Travel_dummy_Reg.csv']))
gender = mydata.Gender
income = mydata.Income
attitude = mydata.Attitude
```

Regression with dummy variables

Checking relation between x and y

Attitude Vs gender

```
mydata.boxplot(column = 'Attitude', by = 'Gender')
myplot.show()
```

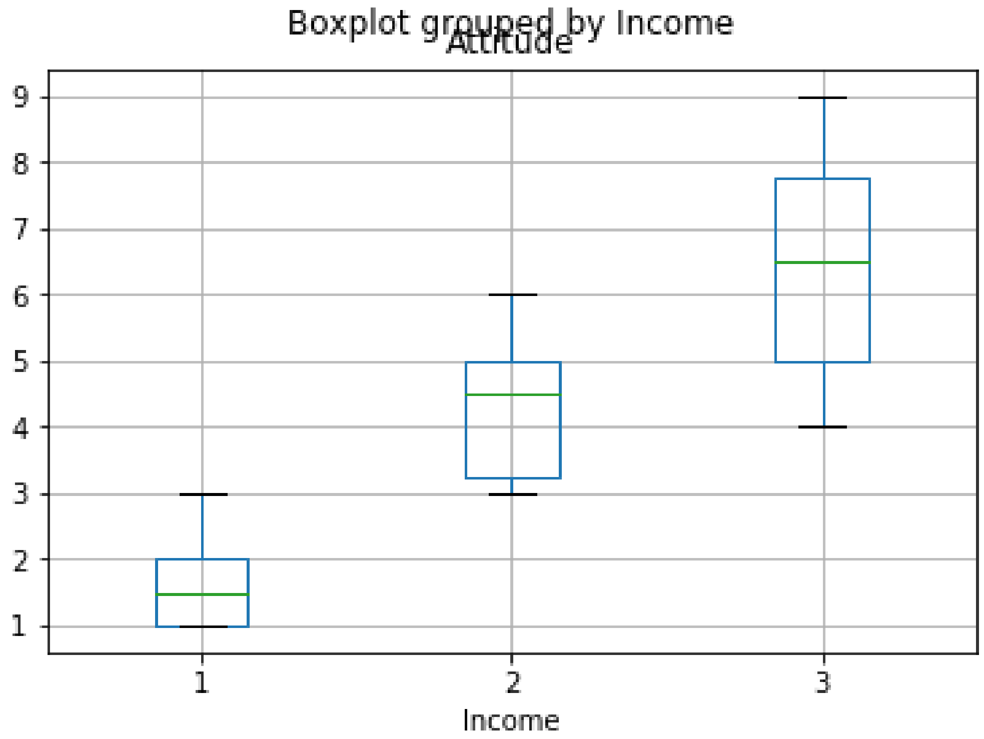


Regression with dummy variables

Checking relation between x and y

Attitude Vs income

```
mydata.boxplot(column = 'Attitude', by = 'Income')
myplot.show()
```



Regression with dummy variables – Output

```
mymodel = ols('attitude ~ C(gender) + C(income)', mydata).fit()

mymodel.summary()
```

R^2	0.86
Adjusted R^2	0.844
F Statistics	53.37
p value	0.0000

	Coef	Std err	t	p-value	[0.025	0.975]
Intercept	2.4	0.336	7.145	0.00	1.71	3.09
C(gender)[T.2]	-1.6	0.336	-4.763	0.00	-2.29	-0.91
C(income)[T.2]	2.8	0.411	6.806	0.00	1.954	3.646
C(income)[T.3]	4.8	0.411	11.668	0.00	3.954	5.646

REGRESSION ANALYSIS

Regression with dummy variables – Anova Table

```
anova_table = anova_lm(mymodel)
anova_table
```

	df	SS	MS	F	P-value
C(gender)	1	19.2	19.2	22.69091	0
C(income)	2	116.26667	58.13333	68.70303	0
Residual	26	22	0.846154		

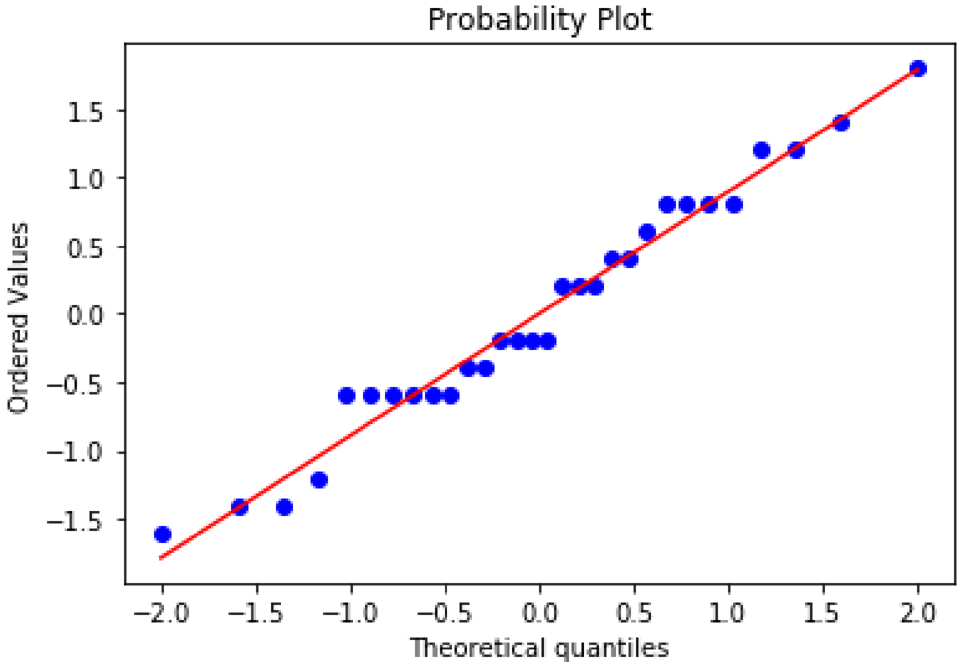
REGRESSION ANALYSIS

Regression with dummy variables

```
pred = mymodel.predict()
res = attitude – pred
```

Normality test of Residuals

```
stats.probplot(res, plot = myplot)
myplot.show()
```



Regression with dummy variables – Normality test of Residuals
stats.mstats.normaltest(res)

Statistics	Value
w	0.5211
p-value	0.7706

BINARY LOGISTIC REGRESSION

BINARY LOGISTIC REGRESSION

Used to develop models when the output or response variable y is binary

The output variable will be binary, coded as either success or failure

Models probability of success p which lies between 0 and 1

Linear model is not appropriate

$$p = \frac{e^{a+b_1x_1+b_2x_2+---+b_kx_k}}{1+e^{a+b_1x_1+b_2x_2+---+b_kx_k}}$$

p: probability of success
x_i's : independent variables
a, b_1, b_2, ---: coefficients to be estimated

If estimate of $p \geq 0.5$, then classified as success, otherwise as failure

Example: Develop a model to predict the non payment of overdrafts by customers of a multinational banking institution. The data collected is given in Logistic_Reg.csv file. The factors and response considered are given below.

SL No	Factor
1	Individual expected level of activity score
2	Transaction speed score
3	Peer comparison score in terms of transaction volume

Response	Values
Outcome	0: Not Paid and 1: Paid

Example: Develop a model to predict the non payment of overdrafts by customers of a multinational banking institution. The data collected is given in Logistic_Reg.csv file.

Step 1: Import Packages

```
import pandas as mypd
from sklearn.linear_model import LogisticRegression
import statsmodels.api as mysm
from google.colab import files
import io
from scipy import stats
import matplotlib.pyplot as myplot
```

Step 2: Read the data

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Logistic_Reg.csv']))
x = mydata[["Ind_Exp_Act_Score", "Tran_Speed_Score", "Peer_Comb_Score"]]
y = mydata.Outcome
x["Intercept"]=1
```

Example: Develop a model to predict the non payment of overdrafts by customers of a multinational banking institution. The data collected is given in Logistic_Reg.csv file.

Step 3: Developing the Model

```
mymodel = mysm.Logit(y,x)
myresult = mymodel.fit()
myresult.summary()
```

Step 4: Printing the Predicted values

```
pred = myresult.predict(x)
myoutput = mypd.DataFrame(pred)
```

Example: Develop a model to predict the non payment of overdrafts by customers of a multinational banking institution. The data collected is given in Logistic_Reg.csv file.

Logistic Regression Results

	Code	Coef	Std err	z	p-value	95 % CI
Ind_Exp_Act_Score	x_1	2.7957	0.355	7.867	0.00	2.099 3.492
Tran_Speed_Score	x_2	2.7532	0.343	8.032	0.00	2.081 3.425
Peer_Comb_Score	x_3	3.5153	0.434	8.095	0.00	2.664 4.366
Intercept		-35.5062	4.406	-8.058	0.00	-71.012

Criteria: $p\text{-value} < 0.05$

Example: Develop a model to predict the non payment of overdrafts by customers of a multinational banking institution. The data collected is given in Logistic_Reg.csv file.

Logistic Regression Results

	Code	Coef	Std err	z	p-value	95 % CI
Ind_Exp_Act_Score	x_1	2.7957	0.355	7.867	0.00	2.099 3.492
Tran_Speed_Score	x_2	2.7532	0.343	8.032	0.00	2.081 3.425
Peer_Comb_Score	x_3	3.5153	0.434	8.095	0.00	2.664 4.366
Intercept		-35.5062	4.406	-8.058	0.00	-71.012

The Model

$$y = \frac{e^{-35.5062+2.7957x_1+2.7532x_2+3.5153x_3}}{1+e^{-35.5062+2.7957x_1+2.7532x_2+3.5153x_3}}$$

NAÏVE BAYES CLASSIFIER

NAÏVE BAYES CLASSIFIER

- Used to predict the probability that the value of the output variable will fall in an interval for a given set of values of input or predictor variables
- Assigns each observation to the most likely class, given its predictor values
- Uses the conditional probability of *P(y / x)* for making prediction

Methodology

Assign a test observation with predictor vector x_0 to the class j for which

$$P(y = j / x = x_0)$$

is the largest

NAÏVE BAYES CLASSIFIER

Example : The data on code review duration and defect density obtained for 10 code reviews are given below. Predict the defect density for review duration = Low using Naïve Bayes classifier?

SL No	Review Duration	Defect Density
1	Low	High
2	Low	Medium
3	Low	Low
4	Low	Medium
5	Low	Medium
6	Low	High
7	High	Low
8	High	High
9	High	Low
10	High	High
11	High	Low
12	High	Low

Predict y given x = Low

NAÏVE BAYES CLASSIFIER

Example : The data on code review duration and defect density obtained for 10 code reviews are given below. Predict the defect density for review duration = Low using Naïve Bayes classifier?

SL No	Review Duration	Defect Density
1	Low	High
2	Low	Medium
3	Low	Low
4	Low	Medium
5	Low	Medium
6	Low	High
7	High	Low
8	High	High
9	High	Low
10	High	High
11	High	Low
12	High	Low

P(y = Low / x = Low)

= Number of cases when both x & y are Low / Number of cases x is Low = 1/6 = 0.17

P(y = Medium / x = Low)

= Number of cases both x is Low and y is Medium / Number of cases x is Low = 3/6 = 0.50

P(y = High / x = Low)

= Number of cases both x is Low and y is Medium / Number of cases x is Low = 2/6 = 0.33

NAÏVE BAYES CLASSIFIER

Example : The data on code review duration and defect density obtained for 10 code reviews are given below. Predict the defect density for review duration = Low using Naïve Bayes classifier?

SL No	Review Duration	Defect Density
1	Low	High
2	Low	Medium
3	Low	Low
4	Low	Medium
5	Low	Medium
6	Low	High
7	High	Low
8	High	High
9	High	Low
10	High	High
11	High	Low
12	High	Low

Maximum of

P(y = Low / x = Low), P(y = Medium / x = Low) and P(y = High / x = Low)

max (0.17, 0.50, 0. 33) = 0.50

= P(y = Medium / x = Low)

Predicted value of y for x = Low is y = Medium

NAÏVE BAYES CLASSIFIER

Used to develop models when the output or response variable y is categorical

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

NAÏVE BAYES CLASSIFIER

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Call libraries

```
import pandas as mypd
from google.colab import files
import io
from sklearn.naive_bayes import GaussianNB
```

Read Data

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Iris_data.csv']))
x = mydata.values[:, 0:4]
y = mydata.values[:, 4]
```

NAÏVE BAYES CLASSIFIER

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Develop Model

```
mymodel = GaussianNB()
mymodel.fit(x, y)
pred = mymodel.predict(x)
mytable = mypd.crosstab(y, pred)
mytable
```

NAÏVE BAYES CLASSIFIER

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Actual	predicted		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	37	0	0
Iris-versicolor	0	32	3
Iris-virginica	0	2	40

Statistics	Value
Accuracy	95.61
Misclassification Error	4.39

NAÏVE BAYES CLASSIFIER

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Validating the model on test data

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Iris_test.csv']))
test_x =mytestdata.values[:,0:4]
test_y = mytestdata.values[:,4]

pred_test = mymodel.predict(test_x)
mytesttable = mypd.crosstab(test_y, pred_test)
mytesttable
```

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Validation Results

Actual	Predicted		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	13	0	0
Iris-versicolor	0	14	1
Iris-virginica	0	1	7

Statistics	Value
Accuracy	94.44
Misclassification Error	5.56

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Results

Statistics	Training	Test
Accuracy	95.61	94.44
Misclassification Error	4.39	5.56

k-Nearest Neighbors

K - NEAREST NEIGHBORS

A non parametric approach for developing models

No assumptions are made about the shape of the decision boundary or underlying distribution

Performs better than regression when the relationship between x's and y is non linear

Will not provide information about which x's are important

K - NEAREST NEIGHBORS

Methodology

Let (x_i, y_i) be the training dataset consists of large number of n records

Let x_0 be the test observation set for which the value of y_0 need to be predicted

Step 1: Identify k records from (x_i, y_i) with x_i values are close to x_0

Step 2: Compute the predicted y_0 from the k y_i values

if y is continuous then y_0 = average of k y_i's

else y_0 = maximum occurring value of y_i in k y_i's

K - NEAREST NEIGHBORS

Example 1

A develop a methodology to predict the value of y in terms of x1 and x2 based on the data given below. Use k – nearest neighbors approach with k = 3. using the methodology predict the value of y for x1 = 15.2 and x2 = 33.1

Training Data set			
Record No.	**x1**	**x2**	**Y**
1	11.35	23	Blue
2	11.59	22.3	Blue
3	12.19	24.5	Blue
4	13.23	26.4	Blue
5	13.51	30.2	Blue
6	13.68	32	Blue
7	14.78	33.1	Blue
8	15.11	33	Blue
9	15.55	25.2	Blue
10	11.85	39.9	Red
11	12.09	39.5	Red
12	12.69	37.8	Red
13	13.73	38.2	Red
14	14.01	37.8	Red
15	14.18	36.5	Red
16	15.28	36	Red
17	15.61	37.1	Red
18	16.05	33.1	Red

Test data		
x1	**x2**	**y**
15.20	33.1	?

Example 1

A develop a methodology to predict the value of y in terms of x1 and x2 based on the data given below. Use k – nearest neighbors approach with k = 3. using the methodology predict the value of y for x1 = 15.2 and x2 = 33.1

Step 1: Compute the distance (Euclidean) of each record in training data from test data

Record No.	x1	x2	Y	Ecludean distance
1	11.35	23	Blue	10.81
2	11.59	22.3	Blue	11.39
3	12.19	24.5	Blue	9.11
4	13.23	26.4	Blue	6.98
5	13.51	30.2	Blue	3.36
6	13.68	32	Blue	1.88
7	14.78	33.1	Blue	0.42
8	15.11	33	Blue	0.13
9	15.55	25.2	Blue	7.91
10	11.85	39.9	Red	7.58
11	12.09	39.5	Red	7.12
12	12.69	37.8	Red	5.33
13	13.73	38.2	Red	5.31
14	14.01	37.8	Red	4.85
15	14.18	36.5	Red	3.55
16	15.28	36	Red	2.90
17	15.61	37.1	Red	4.02
18	16.05	33.1	Red	0.85

Example 1

A develop a methodology to predict the value of y in terms of x1 and x2 based on the data given below. Use k – nearest neighbors approach with k = 3. using the methodology predict the value of y for x1 = 15.2 and x2 = 33.1

Step 2: identify k = 3 records closest (with minimum distance) to test data

Record No.	x1	x2	Y	Ecludean distance
1	11.35	23	Blue	10.81
2	11.59	22.3	Blue	11.39
3	12.19	24.5	Blue	9.11
4	13.23	26.4	Blue	6.98
5	13.51	30.2	Blue	3.36
6	13.68	32	Blue	1.88
7	14.78	33.1	Blue	0.42
8	15.11	33	Blue	0.13
9	15.55	25.2	Blue	7.91
10	11.85	39.9	Red	7.58
11	12.09	39.5	Red	7.12
12	12.69	37.8	Red	5.33
13	13.73	38.2	Red	5.31
14	14.01	37.8	Red	4.85
15	14.18	36.5	Red	3.55
16	15.28	36	Red	2.90
17	15.61	37.1	Red	4.02
18	16.05	33.1	Red	0.85

Example 1

A develop a methodology to predict the value of y in terms of x1 and x2 based on the data given below. Use k – nearest neighbors approach with k = 3. using the methodology predict the value of y for x1 = 15.2 and x2 = 33.1

Step 3: Count different y values in k = 3 records. The predicted value is the mode

Record No.	x1	x2	Y	Euclidean distance
7	14.78	33.1	Blue	0.42
8	15.11	33	Blue	0.13
18	16.05	33.1	Red	0.85

y	Number of Occurrences
Blue	2
Red	1
Mode	Blue

x1	x2	Predicted y
15.20	33.1	Blue

K - NEAREST NEIGHBORS

Example 2 : The effect of temperature and reaction time affects the % yield. The data collected in given in the Mult-Reg_Yield file. Develop a model for % yield in terms of temperature and time? Use k–nearest neighbors approach with k = 2. Predict the yield for the following temperature & time?

Variable	Value
Time	185
Temperature	225
Yield	?

K - NEAREST NEIGHBORS

Example 2 : The effect of temperature and reaction time affects the % yield. The data collected in given in the Mult-Reg_Yield file. Develop a model for % yield in terms of temperature and time? Use k – nearest neighbors approach with k = 2. Predict the yield for the following temperature & time?

Record Number	Time	Temperature	%Yield	Euclidean Distance
1	130	190	35	65.192
2	174	176	81.7	50.220
3	134	205	42.5	54.781
4	191	210	98.3	16.155
5	165	230	52.7	20.616
6	194	192	82	34.205
7	143	220	34.5	42.297
8	186	235	95.4	10.050
9	139	240	56.7	48.384
10	188	230	84.4	5.831
11	175	200	94.3	26.926
12	156	218	44.3	29.833
13	190	220	83.3	7.071
14	178	210	91.4	16.553
15	132	208	43.5	55.660
16	148	225	51.7	37.000

Variable	Value
Time	185
Temperature	225
Yield	= (84.4 + 83.3)/2 = 83.85

Example 3: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data? Use k = 5

Call libraries

```
import pandas as mypd
from google.colab import files
import io
from sklearn.neighbors import KNeighborsClassifier
```

Import data

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Iris_data.csv']))
x = mydata.values[:, 0:4]
y = mydata.values[:, 4]
```

Example 3: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data? Use k = 5

Develop Model

```
mymodel = KNeighborsClassifier(n_neighbors = 5)
mymodel.fit(x, y)
mymodel.score(x, y)
pred = mymodel.predict(x)
mytable = mypd.crosstab(y, pred)
mytable
```

Example 3: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data? Use k = 5

Actual Vs. Predicted

Actual	Predicted		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	37	0	0
Iris-versicolor	0	34	1
Iris-virginica	0	1	41

Statistics	Value
Accuracy	98.24
Misclassification Error	1.76

Example 3: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data? Use k = 5

Validating the model on test data

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Iris_test.csv']))
test_x = mytestdata.values[:, 0:4]
test_y = mytestdata.values[:, 4]
pred_test = mymodel.predict(test_x)
mytesttable = mypd.crosstab(test_y, pred_test)
mytesttable
```

Example 3: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data? Use k = 5

Validating the model on test data:

Actual	Predicted		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	13	0	0
Iris-versicolor	0	13	2
Iris-virginica	0	0	8

Statistics	Value
Accuracy	94.44
Misclassification Error	5.56

Example 3: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Naïve Bayes classifier. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data? Use k = 5

Result:

Statistics	Training	Test
Accuracy	98.24	94.44
Misclassification Error	1.76	5.56

CLASSIFICATION and REGRESSION TREE

CLASSIFICATION AND REGRESSION TREE

Objective

To develop a predictive model to classify dependant or response metric (y) in terms of independent or exploratory variablesxs).

When to Use

Xs : Continuous or discrete

Y : Discrete or continuous

CLASSIFICATION AND REGRESSION TREE

Classification Tree

When response y is discrete

Method = “DecisionTreeClassifier”

Regression Tree

When response y is numeric

Method = “DecisionTreeRegressor”

CLASSIFICATION AND REGRESSION TREE

Example:

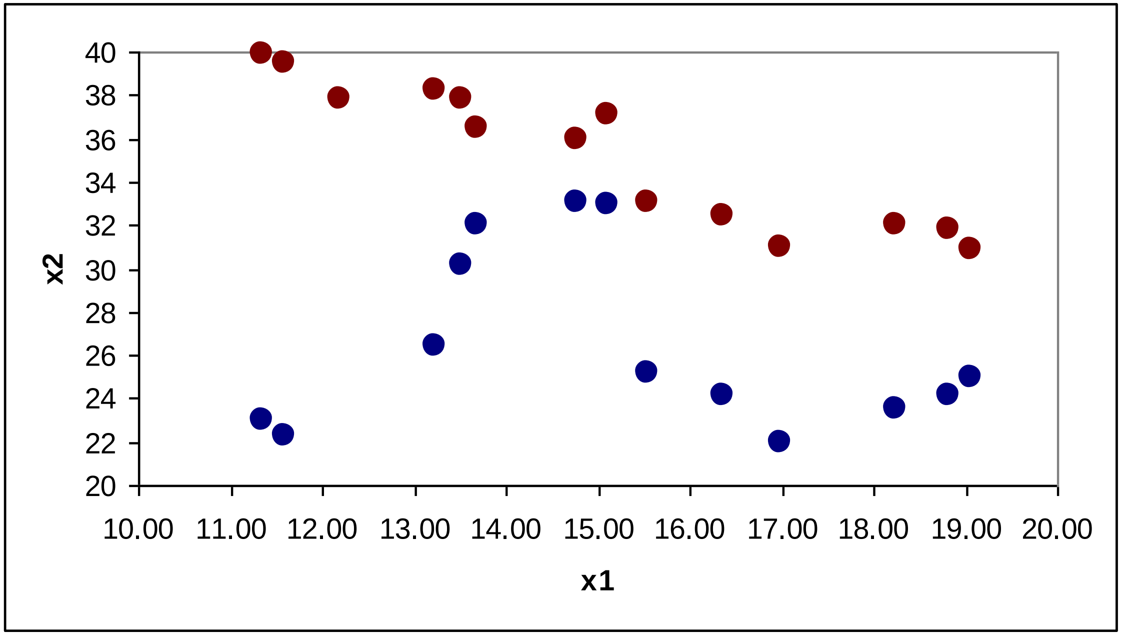
Attribute 1	x1
Attribute 2	x2
Label : y	y1 (Red) , y2 (Blue)

x1	x2	Y	x1	x2	Y
11.35	23	Blue	11.85	39.9	Red
11.59	22.3	Blue	12.09	39.5	Red
12.19	24.5	Blue	12.69	37.8	Red
13.23	26.4	Blue	13.73	38.2	Red
13.51	30.2	Blue	14.01	37.8	Red
13.68	32	Blue	14.18	36.5	Red
14.78	33.1	Blue	15.28	36	Red
15.11	33	Blue	15.61	37.1	Red
15.55	25.2	Blue	16.05	33.1	Red
16.37	24.1	Blue	16.87	32.4	Red
16.99	22	Blue	17.49	31	Red
18.23	23.5	Blue	18.73	32	Red
18.83	24.1	Blue	19.33	31.8	Red
19.06	25	Blue	19.56	30.9	Red

CLASSIFICATION AND REGRESSION TREE

Example:

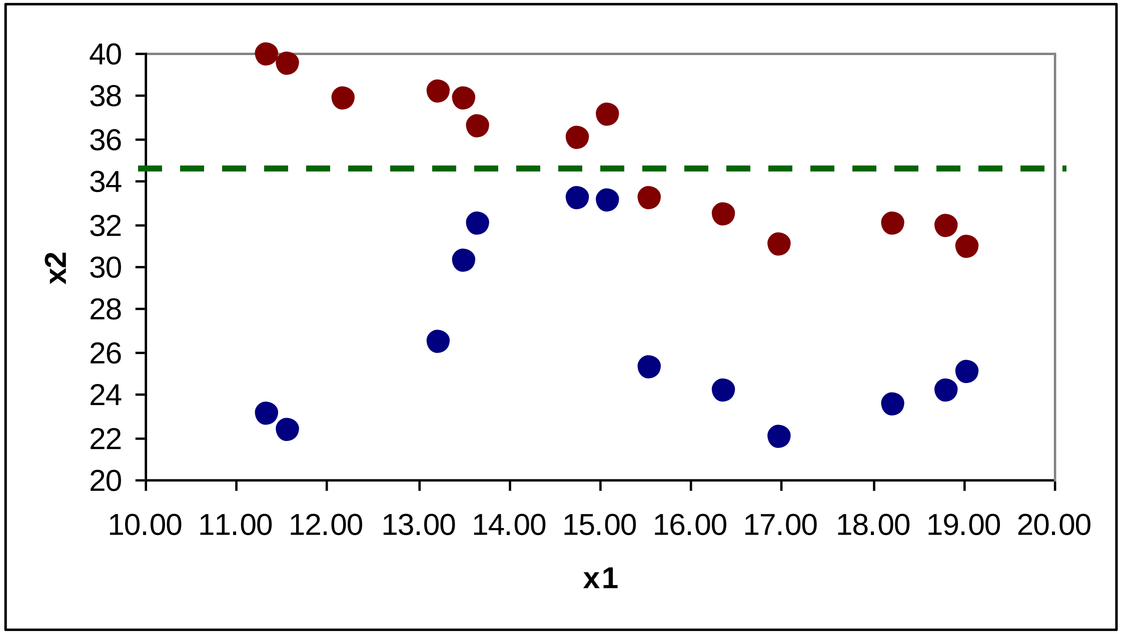
Attribute 1	x1
Attribute 2	x2
Label : y	y1 (Red) , y2 (Blue)

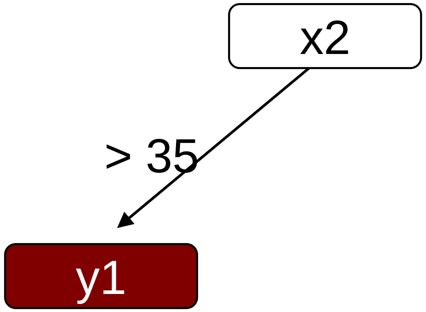


CLASSIFICATION AND REGRESSION TREE

Example:

Attribute 1	x1
Attribute 2	x2
Label : y	y1 (Red) , y2 (Blue)

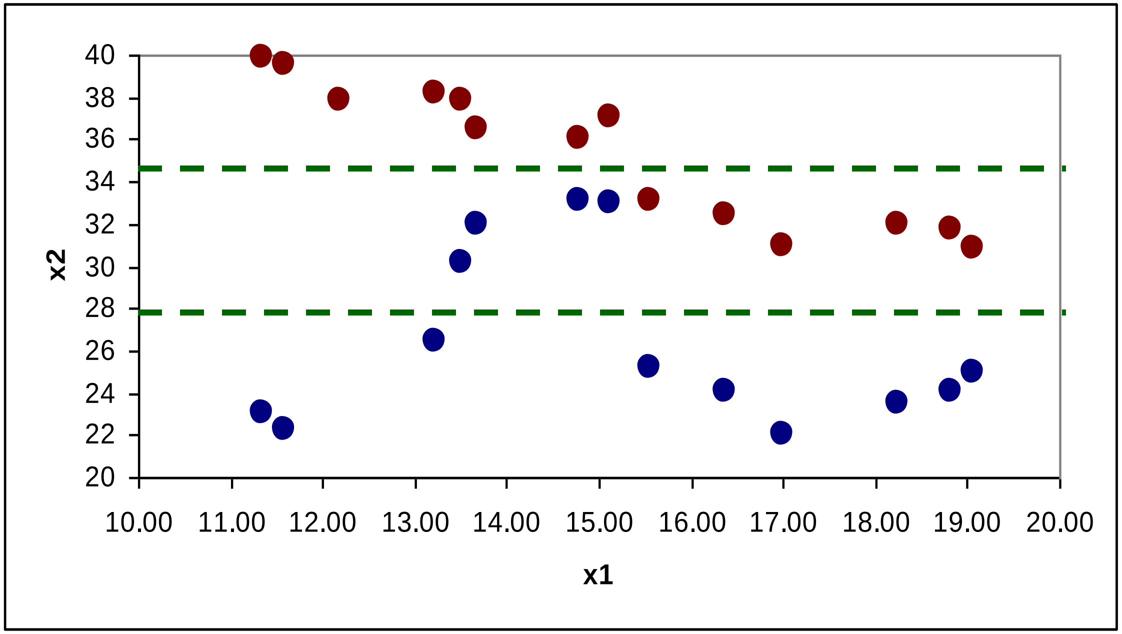


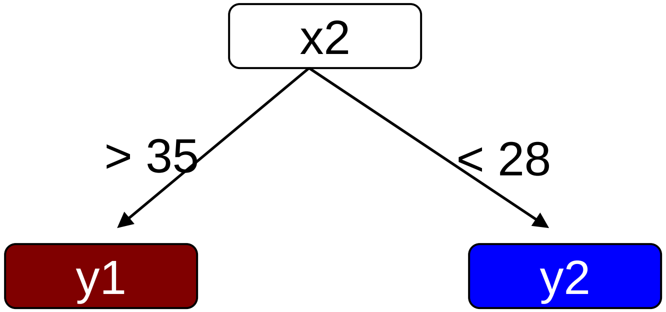


CLASSIFICATION AND REGRESSION TREE

Example:

Attribute 1	x1
Attribute 2	x2
Label : y	y1 (Red) , y2 (Blue)

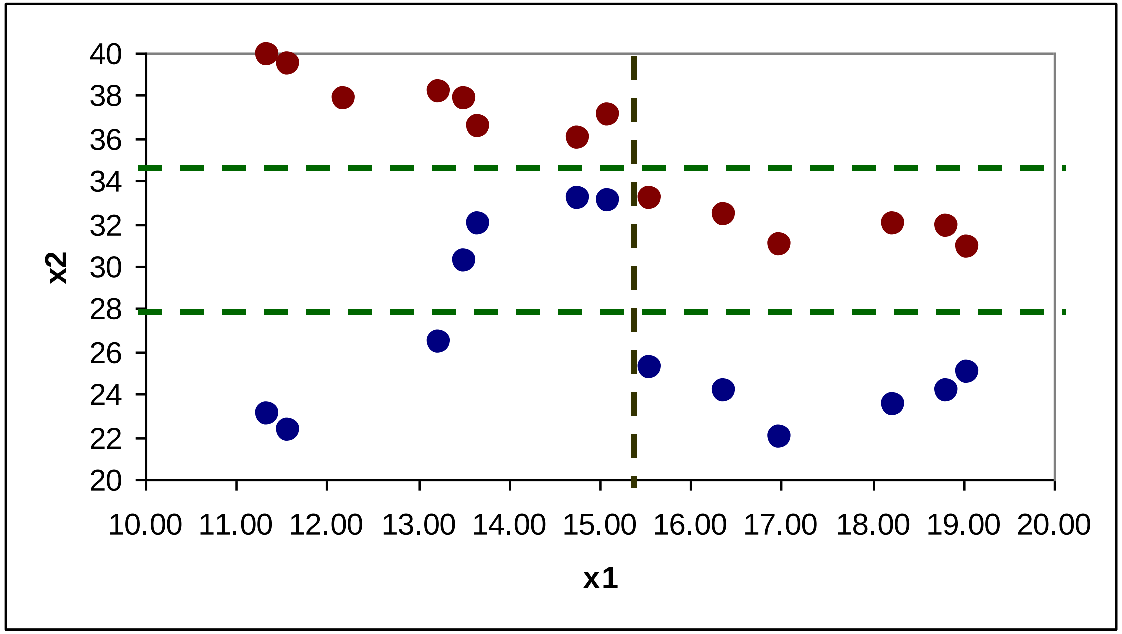


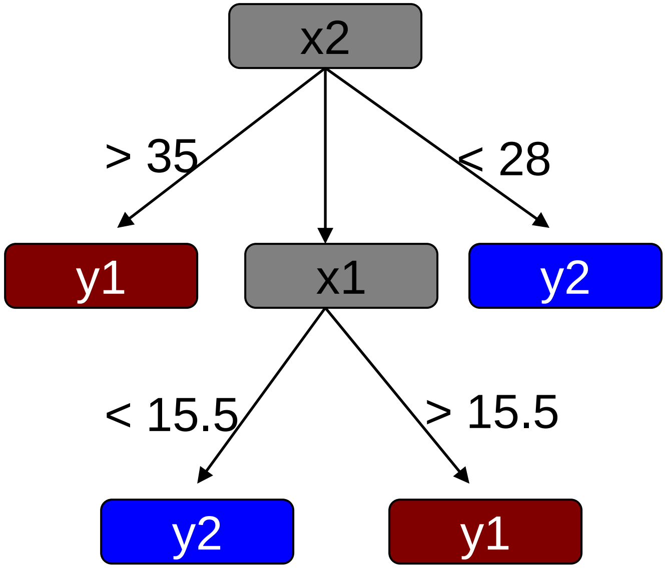


CLASSIFICATION AND REGRESSION TREE

Example:

Attribute 1	x1
Attribute 2	x2
Label : y	y1 (Red) , y2 (Blue)





CLASSIFICATION AND REGRESSION TREE

Example: Rules

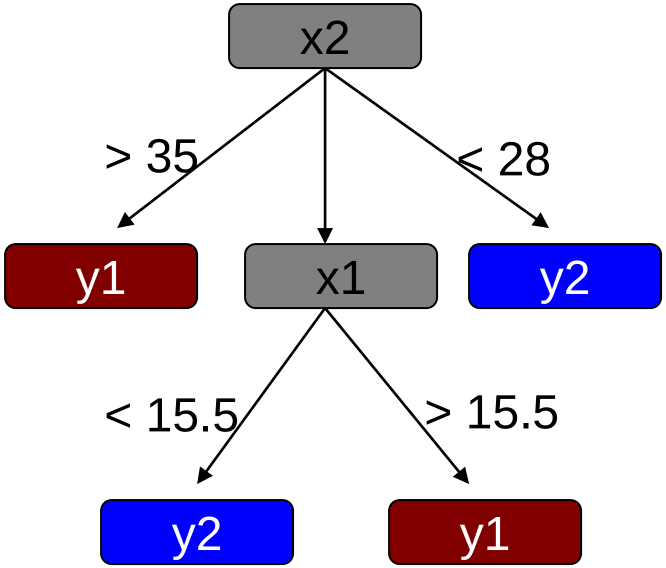
Attribute 1	x1
Attribute 2	x2
Label : y	y1 (Red) , y2 (Blue)

If x2 > 35 then y = y1

If x2 < 28, then y = y2

If 28 > x2 > 35 & x1 > 15.5, then y = y1

If 28 > x2 > 35 & x1 < 15.5, then y = y2



CLASSIFICATION AND REGRESSION TREE

Challenges

How to represent the entire information in the dataset using minimum number of rules?

How to develop the smallest tree?

Solution

Select the variable with maximum information (highest relation with y) for first split

Example: A marketing company wants to optimize their mailing campaign by sending the brochure mail only to those customers who responded to previous mail campaigns. The profile of customers are given below. Can you develop a rule to identify the profile of customers who are likely to respond (Mail_Respond.csv)?

Profile Variable	Values
District	0:Urban, 1: Suburban & 2: Rural
House Type	0:Detached, 1: Semi Detached & 2: Terrace
Income	0:Low & 1: High
Previous Customer	0:No & 1:Yes

Output Variable	Value
Outcome	0:No & 1:Yes

Example: A marketing company wants to optimize their mailing campaign by sending the brochure mail only to those customers who responded to previous mail campaigns. The profile of customers are given in mail_respond.csv? Can you develop a rule to identify the profile of customers who are likely to respond?

Number of variables = 4

SL No	Variable Name	Number of values
1	District	3
2	House Type	3
3	Income	2
4	Previous Customer	2

Total Combination of Customer Profiles = 3 x 3 x 2 x 2 = 36

CLASSIFICATION AND REGRESSION TREE

Import Packages

```
import pandas as mypd
from sklearn import tree
from google.colab import files
import io
```

Read file and variables

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Mail_Respond.csv']))
x = mydata[["District", "House_Type", "Income", "Previous_Customer"]]
y = mydata.Outcome
```

CLASSIFICATION AND REGRESSION TREE

Develop the model

```
mymodel = tree.DecisionTreeClassifier(min_samples_split = 10)
mymodel.fit(x,y)
mymodel.score(x,y)
```

Statistics	Value (%)
Accuracy	1.0
Misclassification Error	0.00

Model Accuracy measures

```
pred = mymodel.predict(x)

mytable = mypd.crosstab(y, pred)

mytable
```

Actual Vs predicted: %

Actual	Predicted	
	No	Yes
No	34	0
Yes	0	66

Accuracy = 34 + 66 = 100%

CLASSIFICATION AND REGRESSION TREE

Exercise 1: Develop a tree based model for predicting whether the customer will take pep (0: No & 1: Yes) using the customer profile data given in bank-data.csv? Use 80% of data to develop the model and validate the model using the remaining 20% of data?

Variables	Values
Age	Numeric
Sex	0:Male & 1: Female
Region	0: Inner City, 1: Rural, 2: Suburban & 3: Town
Income	Numeric
Married	0: No, 1: Yes
Children	Numeric
Car	0: No, 1: Yes
Saving Account	0: No, 1: Yes
Current Account	0: No, 1: Yes
Mortgage	0: No, 1: Yes

Exercise 1: Develop a tree based model for predicting whether the customer will take pep using the customer profile data given in bank-data.csv? Use 80% of data to develop the model and validate the model using the remaining 20% of data?

Reading data

```
import pandas as mypd
from sklearn import tree
from sklearn.cross_validation import train_test_split
from google.colab import files
import io

uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['bank-data.csv']))
x = mydata.values[:, 0:9]
y = mydata.values[:, 10]
```

CLASSIFICATION AND REGRESSION TREE

Exercise 1: Develop a tree based model for predicting whether the customer will take pep using the customer profile data given in bank-data.csv? Use 80% of data to develop the model and validate the model using the remaining 20% of data?

Split data into training and test data

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,
random_state = 100)
```

Develop model using training data

```
mymodel = tree.DecisionTreeClassifier(min_samples_split=50)
mymodel.fit(x_train, y_train)
mymodel.score(x_train, y_train)
```

Statistics	Value (%)
Accuracy	83.3
Misclassification Error	16.7

CLASSIFICATION AND REGRESSION TREE

Exercise 1: Develop a tree based model for predicting whether the customer will take pep using the customer profile data given in bank-data.csv? Use 80% of data to develop the model and validate the model using the remaining 20% of data?

```
pred = mymodel.predict(x_train)
mytable = mypd.crosstab(y_train, pred)
mytable
```

Actual vs Predicted

Actual	Predicted	
	No	Yes
No	232	30
Yes	50	168

CLASSIFICATION AND REGRESSION TREE

Exercise 1: Develop a tree based model for predicting whether the customer will take pep using the customer profile data given in bank-data.csv? Use 80% of data to develop the model and validate the model using the remaining 20% of data?

Validating the Model using test data

```
pred_test = mymodel.predict(x_test)
mytesttable = mypd.crosstab(y_test, pred_test)
mytesttable
```

Actual Vs predicted: %

Actual	Predicted	
	No	Yes
No	58	6
Yes	15	41

Accuracy = (58 + 41)/(58 + 6 + 15 +41) = 82.5 %

CLASSIFICATION AND REGRESSION TREE

Exercise 1: Develop a tree based model for predicting whether the customer will take pep using the customer profile data given in bank-data.csv? Use 80% of data to develop the model and validate the model using the remaining 20% of data?

Data	Accuracy	Misclassification Error
Training	83.33	16.67
Test	82.5	17.5

RANDOM FOREST

and

BAGGING

RANDOM FOREST

Improves predictive accuracy

Generates large number of bootstrapped trees

Classifies a new case using each tree in the new forest of trees

Final predicted outcome by combining the results across all of the trees

Regression tree – average

Classification tree – majority vote

RANDOM FOREST

- Uses trees as building blocks to construct more powerful prediction models
- Decision trees suffer from high variance

  If we split the data into two parts and construct two different trees for each half of the data, the trees can be quite different

- In contrast, a proceedure with low varaince will yield similar results if applied repeatedly to distinct datasets
- Bagging is a general purpose procedure for reducing the variance of a statistiocal learning method

RANDOM FOREST

Procedure

- Take many training sets from the population
- Build seperate prediction models using each training set
- Average the resulting predictions
- Averaging of a set of observatins reduce variance
- Different training datasets are taken using bootstrap sampling
- Generally bootstraped sample consists of two third of the observations and the model is tested on the remainng one third ofthe out of the bag observations

For discrete response – will take the majority vote instead of average

Major difference between bagging and Random Forest

Bagging generally uses all the p predictors while random forest uses $\sqrt{p}$ predictors

RANDOM FOREST

Example

Develop a model to predict the medain value of owner occupied homes using Bosten housing data ? Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

Call libraries

```
import pandas as mypd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import math as mymath
from google.colab import files
import io
```

RANDOM FOREST

Example

Develop a model to predict the medain value of owner occupied homes using Bosten housing data ? Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

Import data

```
uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Boston_Housing_Data.csv']))
x = mydata.values[:, 0:12]
y = mydata.values[:,13]
```

RANDOM FOREST

Example

Develop a model to predict the medain value of owner occupied homes using Bosten housing data ? Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

Python Code

Split data into training and test

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state
= 100)
```

Develop the model using training data - Bagging

```
mymodel = RandomForestRegressor(n_estimators = 500, min_sample_split = 40,
                  max_features = None)
mymodel.fit(x_train,y_train)
```

n_estimators : Number of trees
max_features = None, include all (p) explanatory variable (x's)
max_features = 'auto', include subset ($\sqrt{p}$) explanatory variable (x's)

RANDOM FOREST

Example

Develop a model to predict the medain value of owner occupied homes using Bosten housing data ? Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

Python Code

```
mymodel.score(x_train, y_train)
pred = mymodel.predict(x_train)
res = y_train – pred
res_sq = res**2
res_ss = res_sq.sum()
total_ss = y_train.var()*404
r_sq = 1 - res_ss/total_ss
mse = res_sq.mean()
rmse = mymath.sqrt(mse)
```

RANDOM FOREST

Example

Develop a model to predict the medain value of owner occupied homes using Bosten housing data ? Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

Statistics	Value
MSE	2.874
RMSE	1.695
R^2	96.46

RANDOM FOREST

Example

Develop a model to predict the medain value of owner occupied homes using Bosten housing data ? Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

Python Code

Validate the model using test data

```
pred_test = mymodel.predict(x_test)
res_test = y_test- pred_test
res_test_sq = res_test**2
res_test_ss = res_test_sq.sum()
total_test_ss = y_test.var()*101
r_test_sq = 1 - res_test_ss/total_test_ss
mse = res_test_sq.mean()
rmse = mymath.sqrt(mse)
```

RANDOM FOREST

Example

Develop a model to predict the medain value of owner occupied homes using Bosten housing data ? Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

Statistics	Training	Test
MSE	2.886	16.056
RMSE	1.699	4.007
R^2	96.44	83.21

RANDOM FOREST

Example

Develop a model to predict the medain value of owner occupied homes using Bosten housing data ? Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

Developing model with random forest

mymodel = RandomForestRegressor(n_estimators = 500, min_samples_split = 40, max_features= 'auto’]

Developing model with CART

mymodel = tree.DecisiontreeRegressor(min_samples_split=40)

Statistics	Bagging		Random Forest		Regression Tree	
	Training	Test	Training	Test	Training	Test
MSE	3.733	18.007	4.449	20.169	13.287	28.879
RMSE	1.932	4.243	2.109	4.491	3.645	5.373
R^2	95.41	81.17	94.52	78.91	83.65	69.81

SUPPORT VECTOR MACHINE

Hyperplane

In two dimensions, a hyperplane is a one dimension subspace namely a line

In three dimensions, a hyperplane is a flat two dimension subspace namely a plane

In a *p* dimensional space, a hyperplane is a flat affine subspace of *p-1* dimension

Mathematical Equation

In *2* dimension $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$

Any point x = (x_1, x_2) satisfying the above equation will be in the hyperplane

In *p* dimension

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + --- + \beta_p x_p = 0$$

Any point x = (x_1, x_2, - - - , x_p) satisfying the above equation will be in the hyperplane

Hyperplane

Suppose for a x = (x_1, x_2, - - - , x_p)

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + --- + \beta_p x_p > 0$$

Then the x = (x_1, x_2, - - - , x_p) lies in one side of the hyperplane

Suppose for a x = (x_1, x_2, - - - , x_p)

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + --- + \beta_p x_p < 0$$

Then the x = (x_1, x_2, - - - , x_p) lies on the other side of the hyperplane

Hence

Hyperplane is dividing *p* dimensional space into *2* halves

We can easily determine which side of the hyperplane a point lies by evaluating the hyperplane

SUPPORT VECTOR MACHINE

Procedure

Suppose it is possible to construct a hyperplane that separate the training observations perfectly into two classes according to their class labels (say y = 1 0r y = -1).

Then a separating hyperplane has the property that

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + - - - + \beta_p x_p > 0$$ If y = 1 and

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + - - - + \beta_p x_p < 0$$ If y = -1 and

If a separating hyperplane exists, it can be used to construct a natural classifier
A test observation is assigned to a class depending on which side of the hyperplane it is located

Maximal Marginal Classifier

If the data can be perfectly separated using a hyperplane, then there exists many such hyperplanes

Then the best separating hyperplane (maximal marginal hyperplane) is the one which is furthest from the training observations

Margin: The minimal distance from hyperplane to an observation

Maximal marginal classifier is the separating hyperplane with maximum margin.

SUPPORT VECTOR MACHINE

Support Vector Machine

In many cases no separating hyperplane exists
So no maximum marginal classifier exists

The generalisation of maximum marginal hyperplane to no separable cases is Support Vector Machine

In SVM, a hyperplane is chosen to separate most of the observations into the two classes but may misclassify a few observations

C: The number of misclassified observations. Optimum C can be obtained through cross validation.

SUPPORT VECTOR MACHINE

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Support Vector Machine. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

SUPPORT VECTOR MACHINE

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Support Vector Machine. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Call libraries and import data

```
import pandas as mypd
from sklearn import svm
from google.colab import files
import io

uploaded = files.upload()
mydata = mypd.read_csv(io.BytesIO(uploaded['Iris_data.csv']))
x = mydata.values[:, 0:4]
y = mydata.values[:, 4]
mydata.head()
```

SUPPORT VECTOR MACHINE

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Support Vector Machine. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Develop Model

```
mymodel = svm.SVC()
mymodel.fit(x, y)
mymodel.score(x, y)
pred = mymodel.predict(x)
mytable = mypd.crosstab(y, pred)
mytable
```

SUPPORT VECTOR MACHINE

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Support Vector Machine. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Actual Vs. Predicted

Actual	Predicted		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	37	0	0
Iris-versicolor	0	33	2
Iris-virginica	0	1	41

Statistics	Value
Accuracy	98.24
Misclassification Error	1.76

SUPPORT VECTOR MACHINE

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Support Vector Machine. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Validating the model on test data

```
uploaded = files.upload()
mytestdata = mypd.read_csv(io.BytesIO(uploaded['Iris_test.csv']))
test_x = mytestdata.values[:, 0:4]
test_y = mytestdata.values[:, 4]
pred_test = mymodel.predict(test_x)
mytesttable = mypd.crosstab(test_y, pred_test)
mytesttable
```

SUPPORT VECTOR MACHINE

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Support Vector Machine. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Validating the model on test data

Actual	Predicted		
	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	13	0	0
Iris-versicolor	0	14	1
Iris-virginica	0	0	8

Statistics	Value
Accuracy	97.22
Misclassification Error	2.78

SUPPORT VECTOR MACHINE

Example: Develop a model to predict the iris plant class (1: Iris-setosa, 2: Iris-versicolor & 3: Iris-virginica) based on sepal length, sepal width, petal length and petal width using Support Vector Machine. The data is given in Iris_data.csv file. Validate the model with Iris_test.csv data?

Result

Statistics	Training	Test
Accuracy	98.24	97.22
Misclassification Error	1.76	2.78

ARTIFICIAL NEURAL NETWORKS

USING SCIKIT LEARN

Introduction

One of the most fascinating machine learning modeling technique

Generally uses back propagation algorithm

Relatively complex (due to deep learning with many hidden layers)

Structure is inspired by brain functioning

Generally computationally expensive

Instructions

1. Normalize the data – Use Min – Max transformation (optional)

   Normalized data = Data – Minimum / (Maximum – Minimum)

2. Number of hidden layers required = 1 for vast number of application

3. Number of neurons required = 2/3 of the number of predictor variables or input layers

Remark: The optimum number of layers and neurons are the ones which would minimize mean square error or misclassification error which can be obtained by testing again and again

Example: Develop a model to predict the non payment of overdrafts by customers of a multinational banking institution. The data collected is given in Logistic_Reg.csv file. The factors and response considered are given below. Use 80% of the data to develop the model and validate the model using remaining 20% of the data?

SL No	Factor
1	Individual expected level of activity score
2	Transaction speed score
3	Peer comparison score in terms of transaction volume

Response	Values
Outcome	0: Not Paid and 1: Paid

ARTIFICAL NEURAL NETWORKS

Importing packages

```
import pandas as pd
from sklearn.cross_validation import train_test_split
from sklearn.neural_network import MLPClassifier
from google.colab import files
uploaded = files.upload()
import io
```

Reading the data

```
mydata = pd.read_csv(io.BytesIO(uploaded['Logistic_Reg.csv']))
x = mydata.drop("Outcome", axis=1)
y = mydata.Outcome
```

Splitting the data into training and test

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 100)
```

Example

Develop the model

```
mymodel =MLPClassifier(solver = 'lbfgs', alpha = 1e-5, hidden_layer_sizes = (2), random_state = 100)

mymodel.fit(x_train, y_train)
```

Note:

Classification problem: Use MLPCLassifier

Value estimation: Use MLPRegressor

Solver:
‘lbfgs’ : Uses quasi-Newton method optimization algorithm.
‘sgd’ :Uses stochastic gradient descent optimization algorithm.
‘adam’ :Uses stochastic gradient-based optimizer

Example: Interpretation

hidden_layer_sizes : a vector representing hidden layers and hidden neurons in each layer

hidden_layer_sizes = (l) : one hidden layers with *l* hidden neurons

Output

mymodel.score(x_train, y_train)

Statistics	Value
% Accuracy	96.81
% Error	3.19

mymodel.predict_proba(x_train)

Output: Validation

```
predtest = mymodel.predict(x_test)
mytable = mypd.crosstab(y_test, predtest)
mytable
```

Actual Vs Predicted

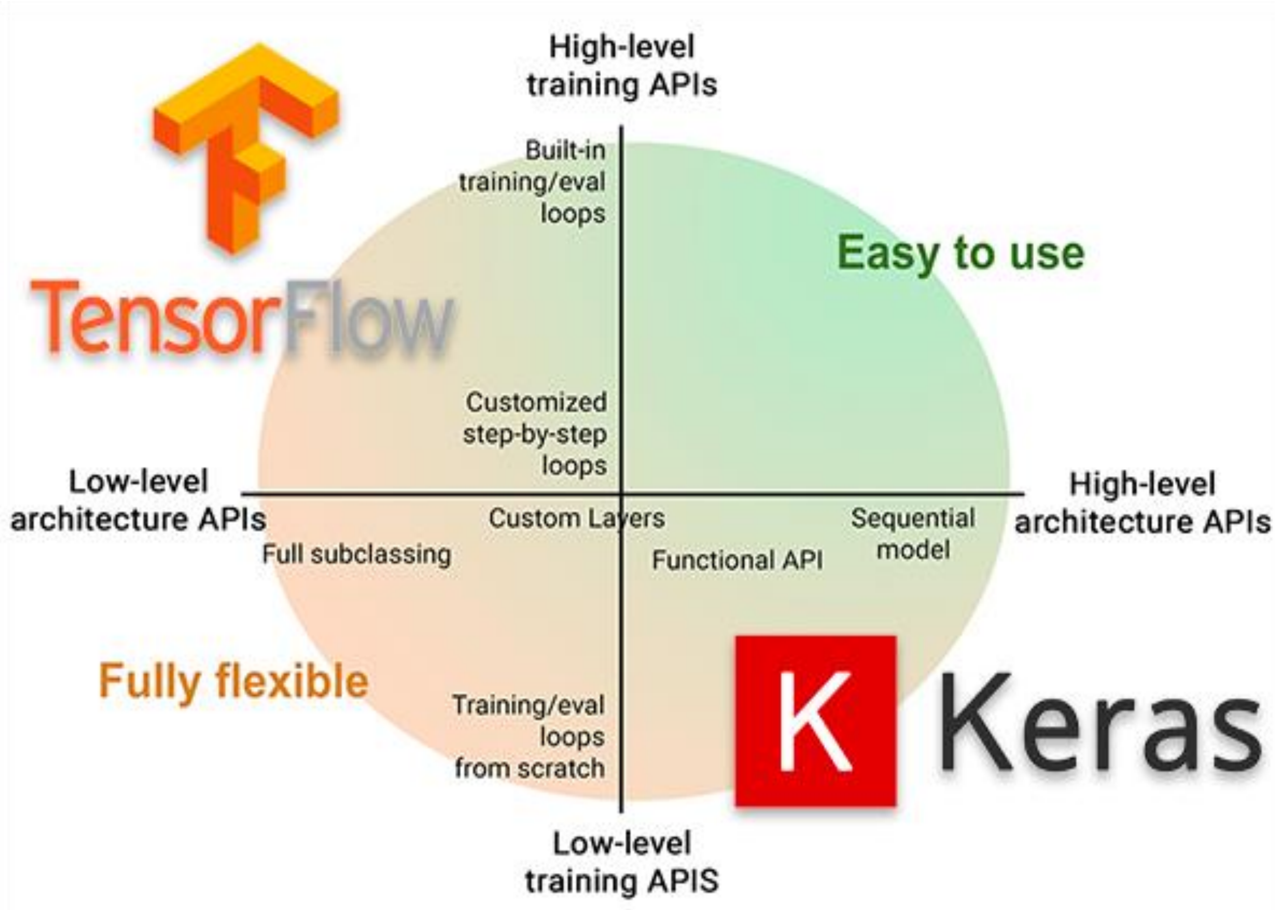
<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">Predicted</td></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td rowspan="2">Actual</td><td>0</td><td>54</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>138</td></tr>
</table>

DEEP LEARNING USING

TENSORFLOW & KERAS

IMPLEMENTATION OF DEEP LEARNING FRAMEWORKS

RELATIONSHIP BETWEEN KERAS & TENSORFLOW



WHAT IS KERAS?

- Francois Chollet, the author of Keras, says:

  "The framework was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research."

- Keras is open source framework written in Python
- ONEIROS (Open Ended Neuro-Electronic Intelligent Robot OS)
- Contains neural-network building blocks like layers, optimizer, activation functions
- Support CNN and RNN

DEVELOPMENT OF KERAS

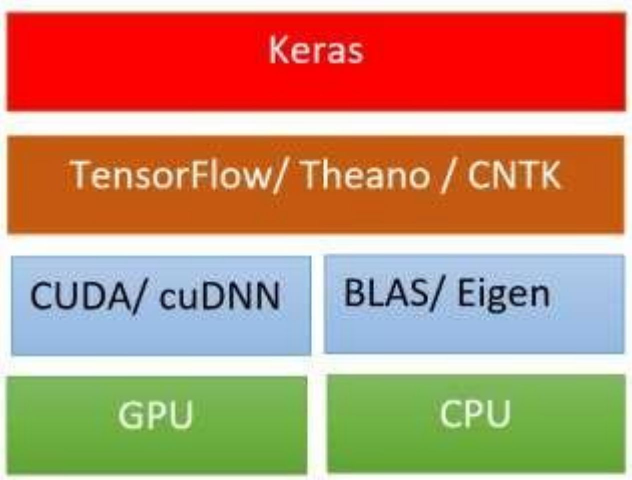
- Francois Chollet, Google AI Developer/Researcher developed Keras on 27 March 2015 to facilitate his own research and experiments.
- With the explosion of deep learning popularity, many developers, programmers, and machine learning practitioners flocked to Keras due to its easy-to-use API.
- At that time the popular deep learning libraries (Torch, Theano, and Caffe) **tedious, time-consuming, and inefficient.**
- **Keras, on the other hand, was extremely easy to use**

BACKEND OF KERAS

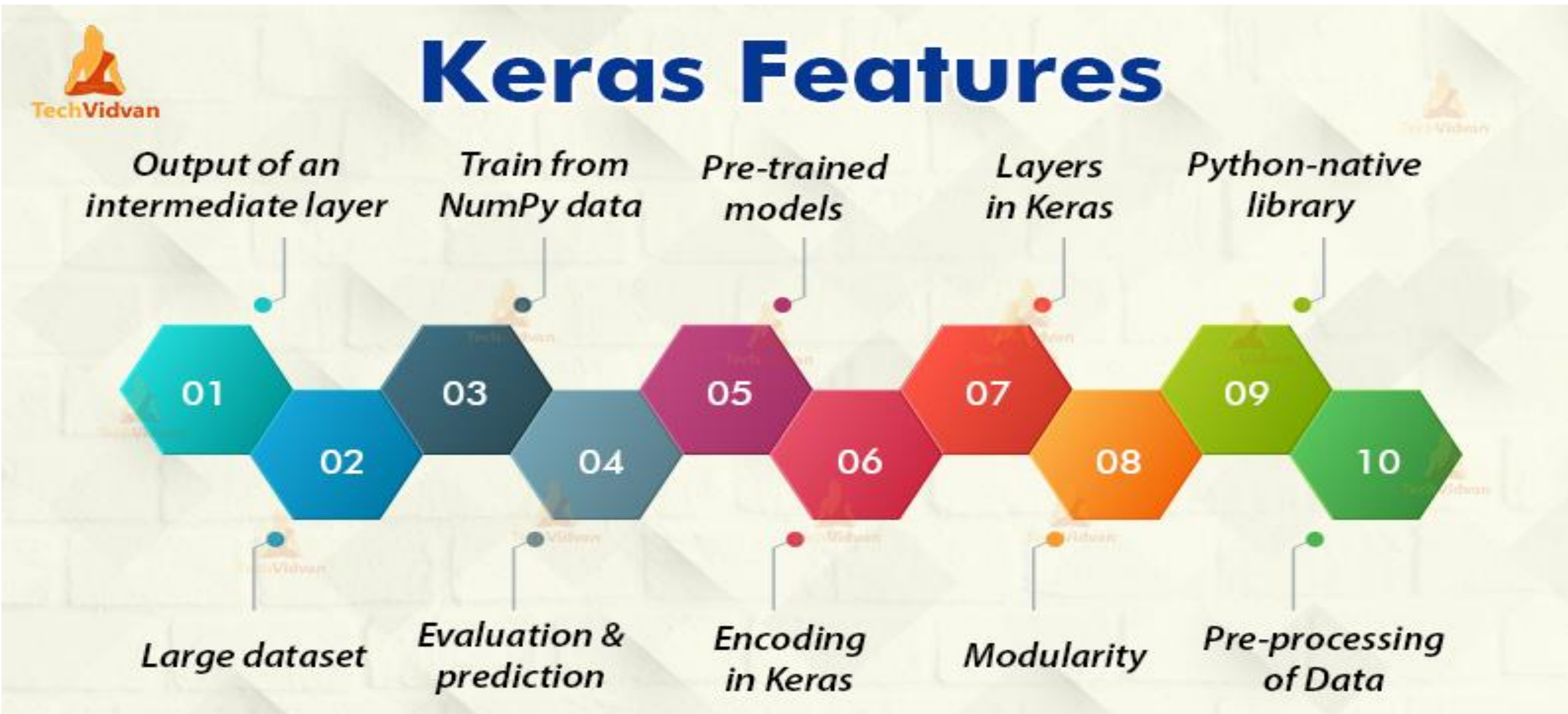
- ***A backend*** **is a computational engine** — it builds the network graph/topology, runs the optimizers, and performs the actual number crunching.
- Keras might have several backend one at a time and can be thought as a set of abstractions that makes it easier to perform deep learning.
- Keras' default backend was ***Theano*** until v1.1.0.
- With the release of **TensorFlow** by Google, Keras started supporting TensorFlow as a backend, **resulting in TensorFlow being the *default backend* starting from the release of Keras v1.1.0.**

KERAS FEATURES

- Contains datasets and some pre-trained deep learning applications.
- Model check-pointing, early stopping
- Uses libraries TensorFlow, Theano, CNTK as backend, only one at a time
- Backend does all computations
- Keras call backend functions
- Works for both CPU and GPU



KERAS FEATURES



KERAS FEATURES

- Rapid prototyping-
  1. Build neural network with minimal lines of code
  2. Build simple or complex neural networks within a few minutes
- Flexibility-

  Sometime it is desired to define own metrics, layers, a cost function, Keras provide freedom for the same.
- Two types of built in models
  1. Sequential
  2. Functional
- All models have following common properties

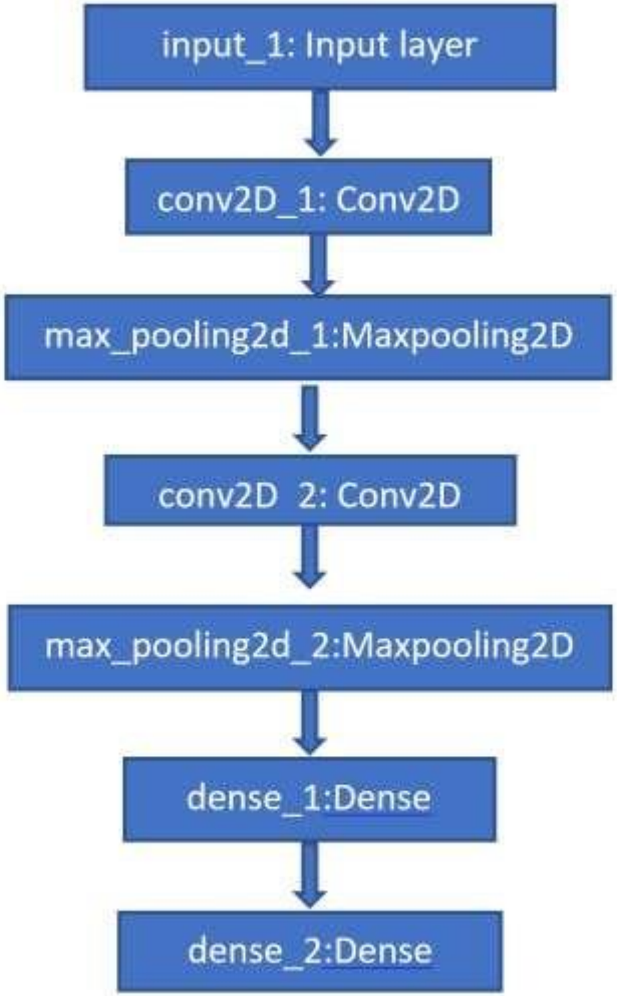
  Inputs that contain a list of input tensors

  Layers, which comprise the model graph

  Outputs, a list of output tensors.

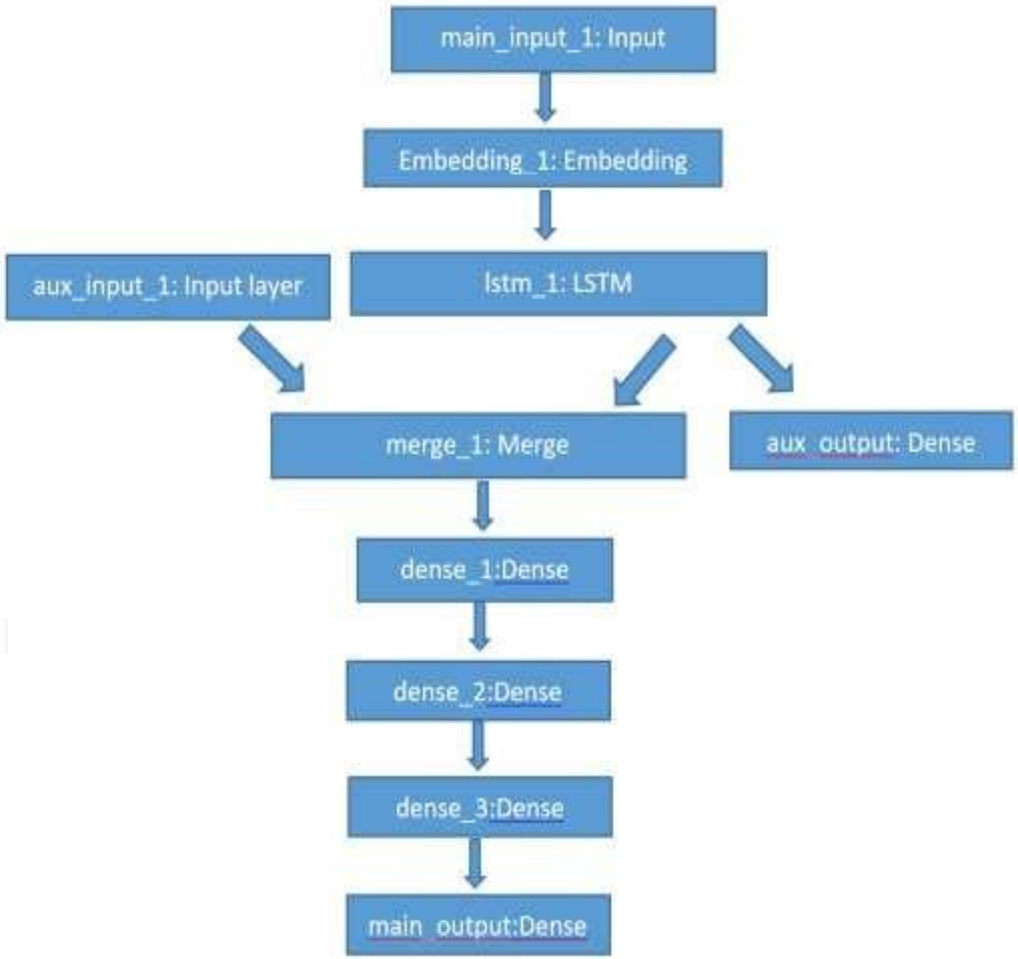
SEQUENTIAL MODEL

- It is linear stack of layers
- Output of previous layer is input to the next layer.
- Create models by stacking one layer on top of other
- Useful in situation where task is not complex
- Provides higher level of abstraction

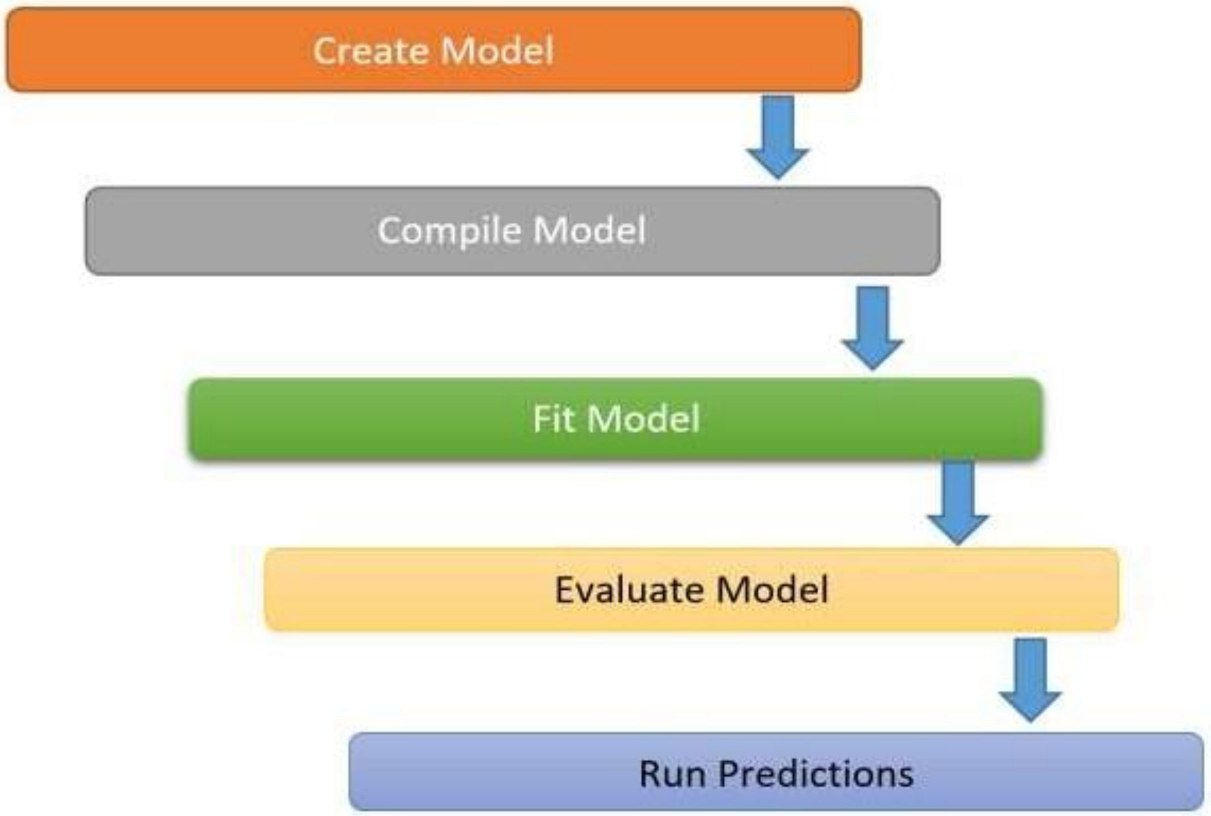


FUNCTIONAL MODEL

- It define more complex models
- Such as directed acylic graphs
- Multi input output models
- Model with shared layers
- Possible to connect a layer with any other layer



STEPS IN BUILDING A MODEL



MODEL METHODS

1. Compile: It is used to configure model. It accept following parameters

- **Optimizer**:
  - This specifies type of optimiser to use in back-propagation algorithm
  - SGD, Adadelta, Adagrad , Adam , Nadam optimizer and many others.

- **Loss:**
  - It is the objective function
  - It track losses or the drift from the function during training the model.
  - For regression: mean squared error, mean absolute error etc.
  - For classification: Categorical cross-entropy, binary cross entropy
  - Different loss functions for different outputs

MODEL METHODS

- **Metrics:**
  - It is similar to objective function.
  - Results from metric aren't used when training model.
  - It specifies the list of metrics that model evaluate during training and testing.
  - The commonly used metric is the accuracy metric.
  - Possible to specify different metrics for different output

MODEL METHODS

1. Compile: It is used to configure model. It accept following parameters

- **Optimizer**:
  - This specifies type of optimiser to use in back-propagation algorithm
  - SGD, Adadelta, Adagrad , Adam , Nadam optimizer and many others.
- **Loss:**
  - It is the objective function
  - It track losses or the drift from the function during training the model.
  - For regression: mean squared error, mean absolute error etc.
  - For classification: Categorical cross-entropy, binary cross entropy
  - Different loss functions for different outputs

MODEL METHODS

- **epochs**:
  - An epoch is an iteration
  - It specifies number of times training data is feed to the model.
- **validation_split:**
- Validation data is selected from the end samples
  - At the end of each epoch, loss and metrics are calculated for this data.
- **validation_data:**
  - Fraction of the data that is used as validation.

KERAS DATASETS

Keras contains various datasets that are used to build neural networks. The datasets are described below

1. **Boston House Pricing dataset:**
   It contains 13 attributes of houses of Boston suburbs in the late 1970s
   Used in regression problems

2. **CIFAR10:**
   - It is used for classification problems.
   - This dataset contains 50,000 32×32 colour images
   - Images are labelled over 10 categories
   - 10,000 test images.

3. **CIFAR100:**
   - Same as CIFAR10 but it has 100 categories

KERAS DATASETS

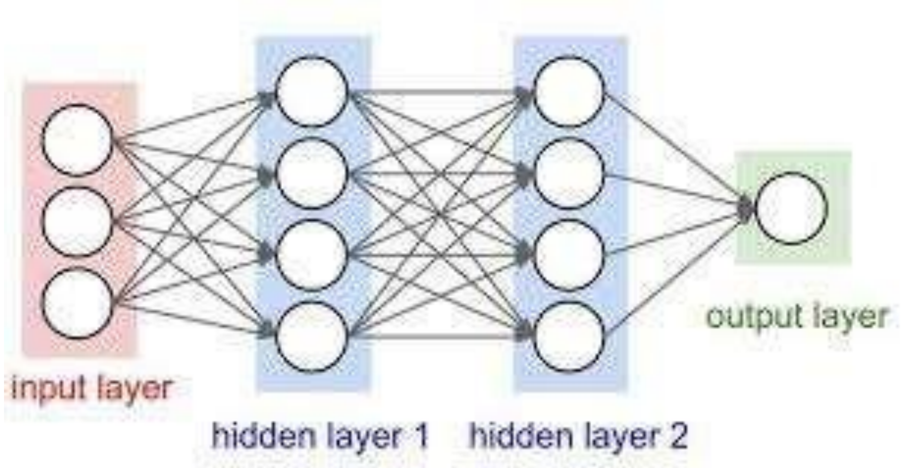
4. **MNIST:**
   - This dataset contains 60,000 28×28 grayscale images of 10 digits
   - Also include 10,000 test images.
5. **Fashion-MNIST**:
   - This dataset is used for classification problems.
   - This dataset contains 60,000 28×28 grayscale images of 10 categories, along
6. **IMDB movie reviews data:**
   - Dataset of 25,000 movies reviews from IMDB
   - labeled by sentiment (positive/negative).
7. **Reuters newswire topics classification:**
   - Dataset of 11,228 newswires from Reuters, labeled over 46 topics

KERAS LAYERS

It consist of different types of layers used in deep learning such as

:

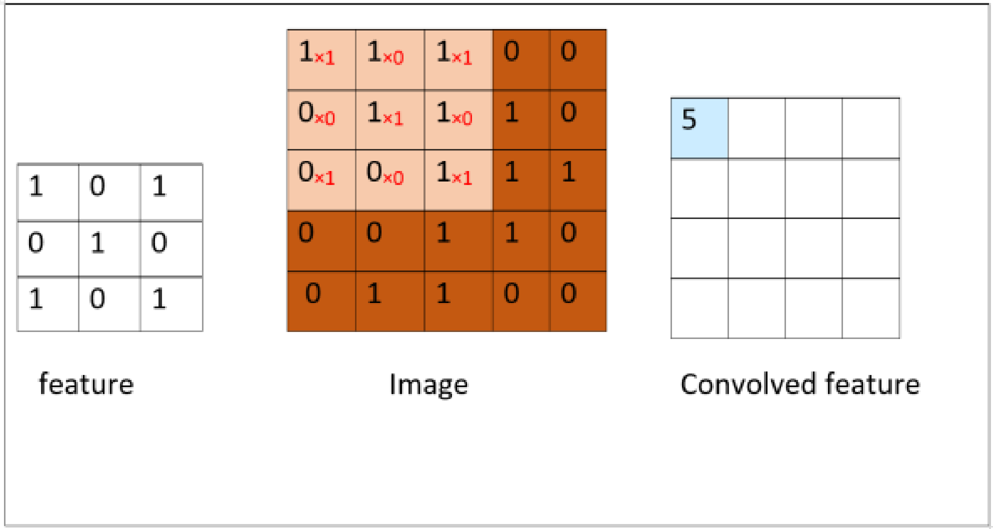
1. **Dense layer:**
   - A Dense layer is fully connected neural network layer



(Ramchoun et al, 2016)

2. Convolutional layer:

- Mostly used in computer vision.
- It extract features from the input image.
- It preserves the spatial relationships between pixels
- It learn image features using small squares of input data
- Finally, the obtained matrix is known as the feature map



3. Pooling layer

- Also called as subsampling or down-sampling layer
- Pooling reduces the dimensionality of feature map
- Retains the most important information
- There are many types of pooling layers such as
- MaxPooling and AveragePooling
- In case of max pooling, take the largest element from the rectified feature map within that window.
- In average pooling, average of all elements in that window is taken.

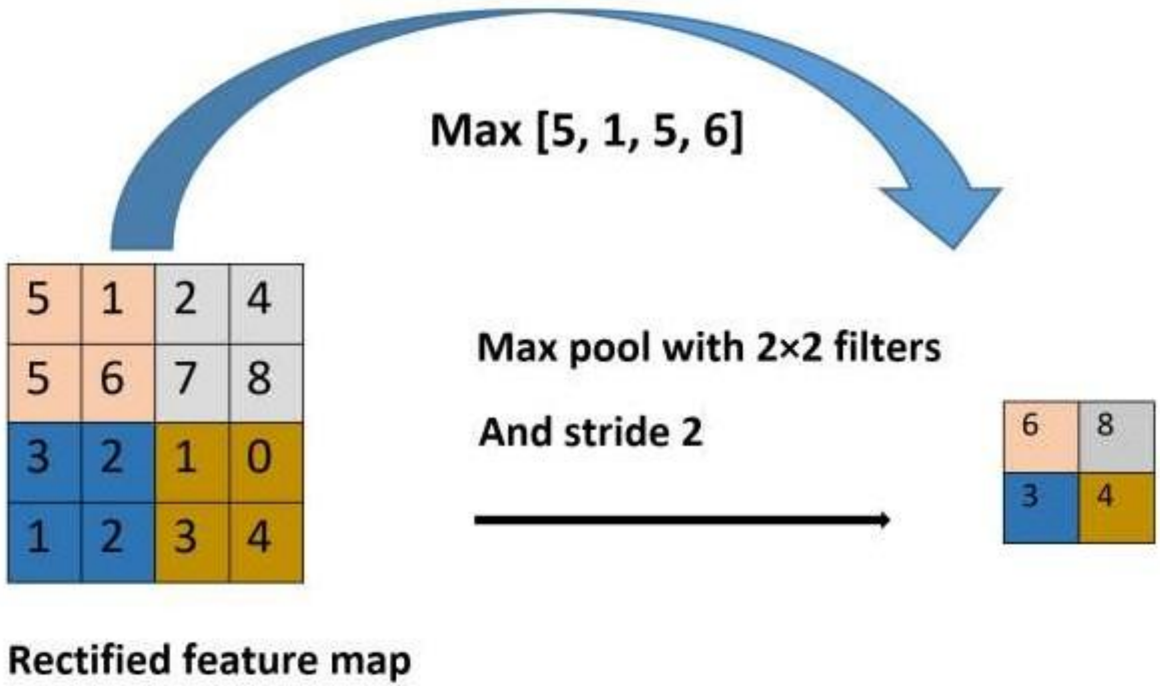


Figure: Max Pooling Concept

KERAS LAYERS

4. **Recurrent layer**
   - Basic building block of RNN
   - This is mostly used in sequential and time series modelling.
5. **Embedding layers**
   - Required when input is text
   - These are mostly used in Natural Language Processing.
6. **Batch Normalisation layer**
   - Normalize the activations of the previous layer at each batch
   - Applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

IMPLEMENTATION OF ACTIVATION FUNCTION

PACKAGE INSTALLATION

```
! pip install tensorflow
import tensorflow as tf
import numpy as np
import math
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

DEFINING PLOT FUNCTION

```
def do_plot(x, y, title):
    plt.figure(figsize=(10,5))
    plt.plot(x,y)
    plt.title(title)
    plt.ylabel('Y axis')
    plt.xlabel('X axis')
    plt.show()
```

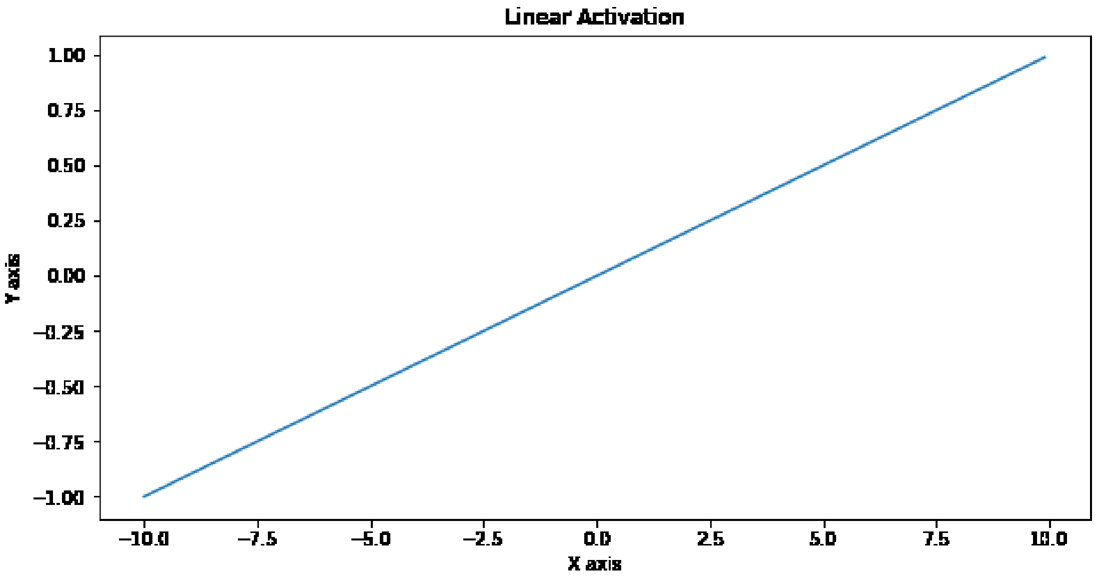
IMPLEMENTATION OF ACTIVATION FUNCTION

DATA

```
x = tf.Variable(tf.range(-10, 10, 0.1), dtype=tf.float32)
y_predicted = np.array([1,1,0,0,1])
y_true = np.array([0.30,0.7,1,0,0.5])
```

LINEAR ACTIVATION

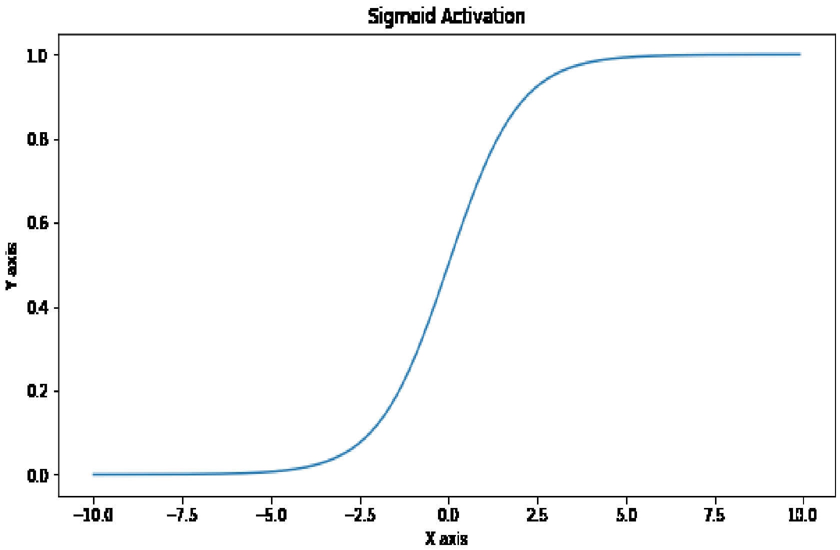
```
def linear_activation(x):
        c = 0.1
        return c*x.numpy()
        do_plot(x.numpy(), linear_activation(x), 'Linear Activation')
```

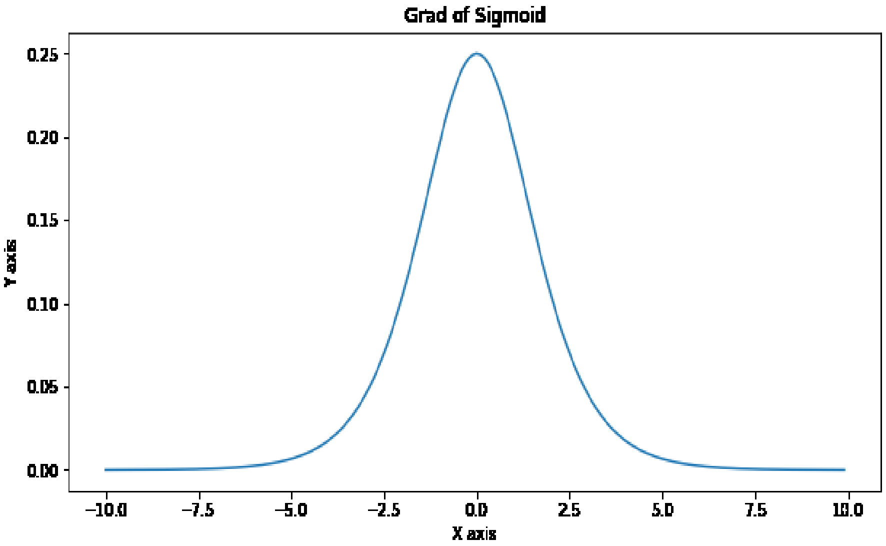


IMPLEMENTATION OF ACTIVATION FUNCTION

SIGMOID ACTIVATION

```
y = tf.nn.sigmoid(x)
do_plot(x.numpy(), y.numpy(), 'Sigmoid Activation')
with tf.GradientTape() as t:
        y = tf.nn.sigmoid(x)
do_plot(x.numpy(), t.gradient(y, x).numpy(), 'Grad of Sigmoid')
```

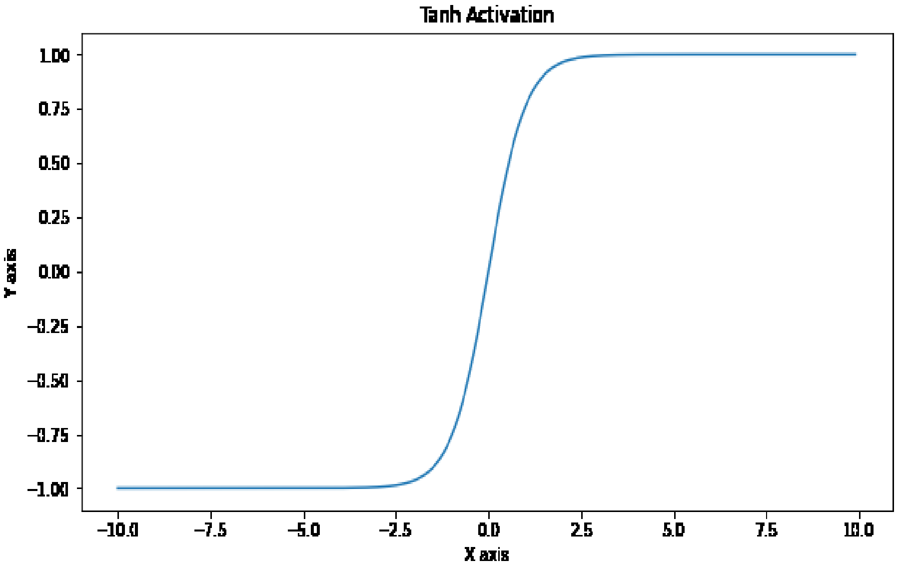


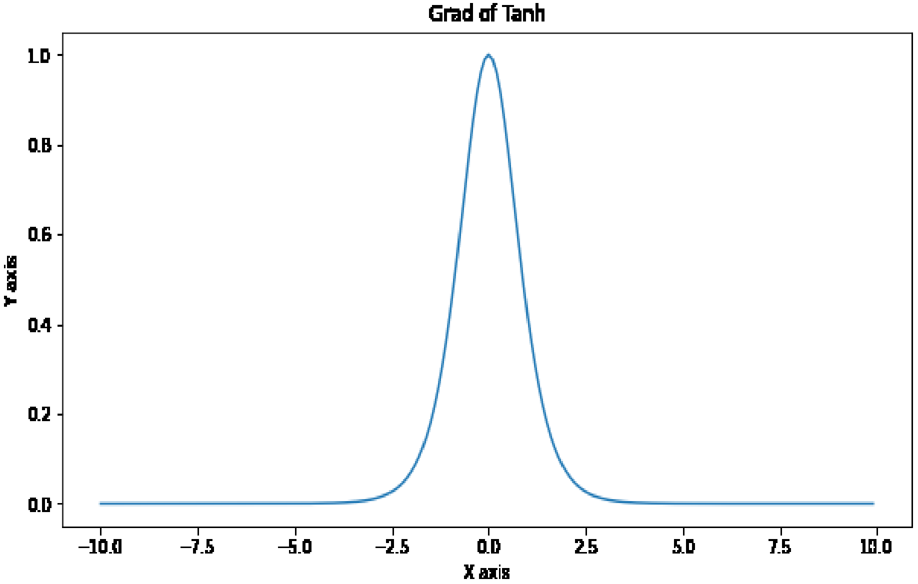


IMPLEMENTATION OF ACTIVATION FUNCTION

TANH ACTIVATION

```
def tanh(x):
return (math.exp(x) - math.exp(-x)) / (math.exp(x) + math.exp(-x))
y = tf.nn.tanh(x)
do_plot(x.numpy(), y.numpy(), 'Tanh Activation')
with tf.GradientTape() as t:
        y = tf.nn.tanh(x)
do_plot(x.numpy(), t.gradient(y, x).numpy(), 'Grad of Tanh')
```

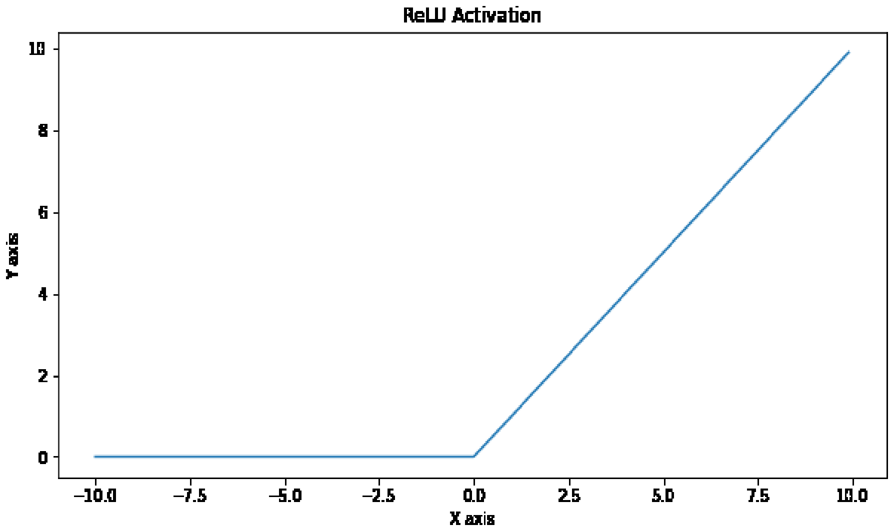


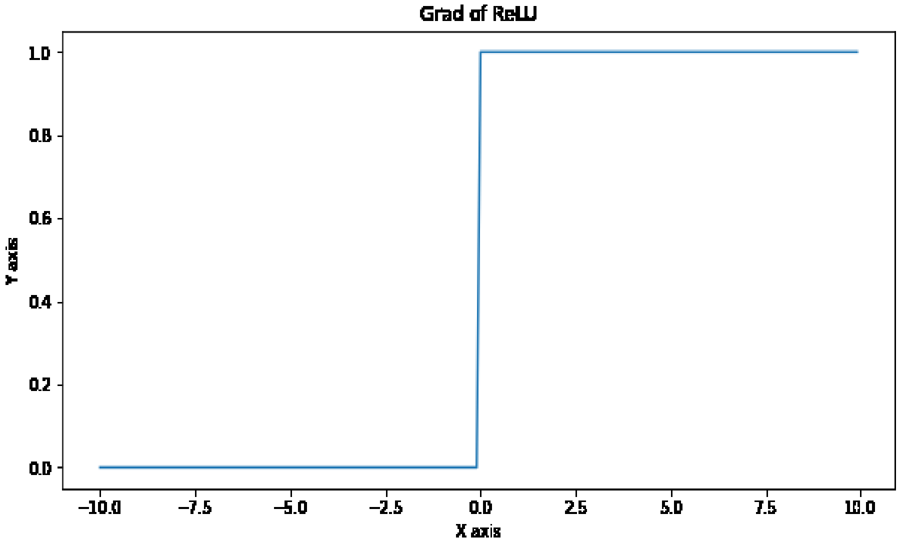


IMPLEMENTATION OF ACTIVATION FUNCTION

RELU ACTIVATION

```
def relu(x):
return max(0,x)
y = tf.nn.relu(x)
do_plot(x.numpy(), y.numpy(), 'ReLU Activation')
with tf.GradientTape() as t:
    y = tf.nn.relu(x)
do_plot(x.numpy(), t.gradient(y, x).numpy(), 'Grad of ReLU')
```

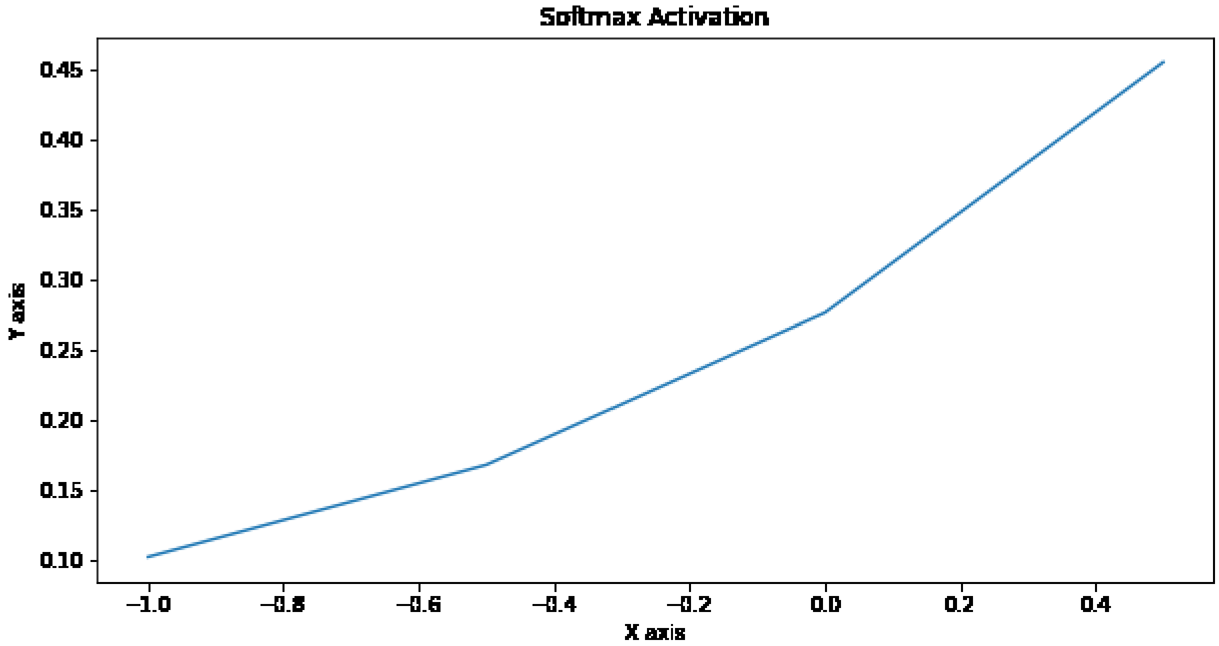




IMPLEMENTATION OF ACTIVATION FUNCTION

SOFTMAX ACTIVATION

```
x1 = tf.Variable(tf.range(-1, 1, .5), dtype=tf.float32)
y = tf.nn.softmax(x1)
do_plot(x1.numpy(), y.numpy(), 'Softmax Activation')
```



IMPLEMENTATION OF LOSS FUNCTION

MEAN ABSOLUTE ERROR

USING NUMPY FUNCTION

```
def mae_np(y_predicted, y_true):
    return np.mean(np.abs(y_predicted-y_true))
mae_np(y_predicted, y_true)
```

USER DEFINED FUNCTION

```
def mae(y_predicted, y_true):
        total_error = 0
        for yp, yt in zip(y_predicted, y_true):
                total_error += abs(yp - yt)
        print("Total error is:",total_error)
        mae = total_error/len(y_predicted)
        print("Mean absolute error is:",mae)
        return mae
mae(y_predicted, y_true)
```

OUTPUT

0.5

IMPLEMENTATION OF LOSS FUNCTION

MEAN SQUARE ERROR

USING NUMPY FUNCTION

```
np.mean(np.square(y_true-y_predicted))
```

USER DEFINED FUNCTION

```
def mse(y_true, y_predicted):
    total_error = 0
    for yt, yp in zip(y_true, y_predicted):
        total_error += (yt-yp)**2
    print("Total Squared Error:",total_error)
    mse = total_error/len(y_true)
    print("Mean Squared Error:",mse)
    return mse
mse(y_true, y_predicted)
```

OUTPUT
0.366

IMPLEMENTATION OF LOSS FUNCTION

LOG LOSS

USER DEFINED FUNCTION

```
epsilon = 1e-15
y_predicted_new = [max(i,epsilon) for i in y_predicted]
y_predicted_new = [min(i,1-epsilon) for i in y_predicted_new]
y_predicted_new = np.array(y_predicted_new)
-np.mean(y_true*np.log(y_predicted_new)+(1-y_true)*np.log(1-y_predicted_new))
```

OUTPUT
17.26

HANDWRITTEN DIGIT RECOGNITION PROBLEM

- The dataset was constructed from a number of scanned document dataset available from the National Institute of Standards & Technology (NIST).
- Images of digits were taken from a variety of scanned documents, normalized in size and centred.
- Each image is a 28 by 28 pixel square (784 pixels total). A standard split of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model and a separate set of 10,000 images are used to test it.
- It is a digit recognition task. As such there are 10 digits (0 to 9) or 10 classes to predict. Results are reported using prediction error, which is nothing more than the inverted classification accuracy.

HANDWRITTEN DIGIT RECOGNITION PROBLEM

Import the packages

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

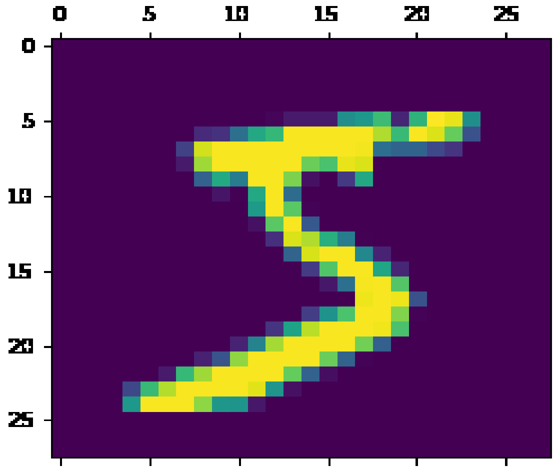
Load the data

```
(X_train, y_train) , (X_test, y_test) = keras.datasets.mnist.load_data()
```

Visualizing our 1st input data

```
plt.matshow(X_train[0])

y_train[0]
```

5



HANDWRITTEN DIGIT RECOGNITION PROBLEM

Data Scaling

```
X_train = X_train / 255
X_test = X_test / 255
```

Data Shape

```
X_train[0].shape
```

(28, 28)

Flatten the 28*28 grid data into a 1-dimensional data

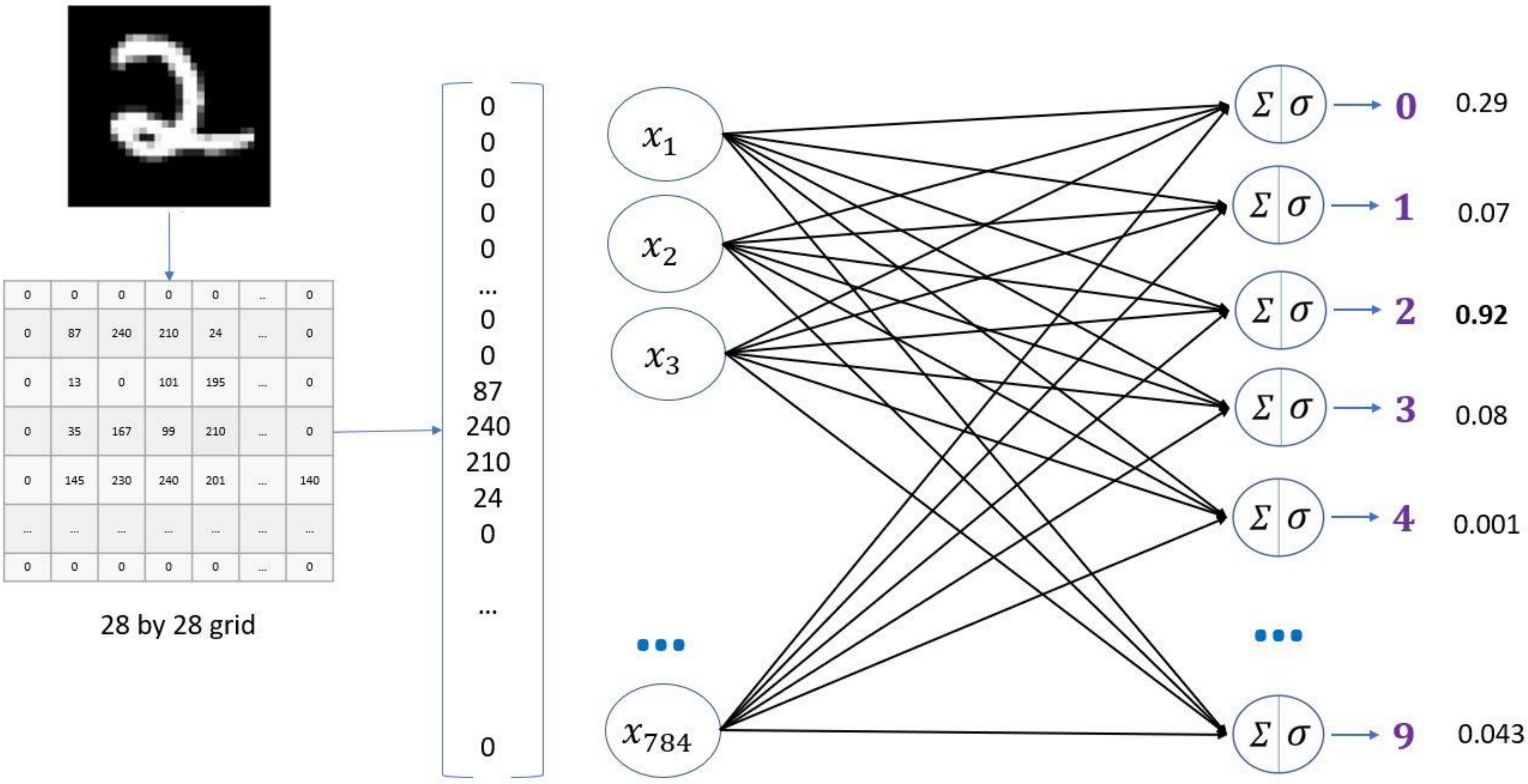
```
X_train_flattened = X_train.reshape(len(X_train), 28*28)
X_test_flattened = X_test.reshape(len(X_test), 28*28)
```

Flattened Data Shape

```
X_train_flattened.shape
```

(60000, 784)

HANDWRITTEN DIGIT RECOGNITION PROBLEM



Simple Neural Network without hidden layer with Flattened Data

```
model = keras.Sequential([
        keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')])
model.compile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
model.fit(X_train_flattened, y_train, epochs=5)
```

```
...
Epoch 5/5 1875/1875 [==============================] - 3s
1ms/step - loss: 0.2668 - accuracy: 0.9252
```

HANDWRITTEN DIGIT RECOGNITION PROBLEM

Simple Neural Network without hidden layer with Flattened Data

model.evaluate(X_test_flattened, y_test)

```
313/313 [==============================] - 1s 1ms/step - loss:
0.2667 - accuracy: 0.9251 [0.2667323350906372, 0.9251000285148621]
```

y_predicted = model.predict(X_test_flattened)

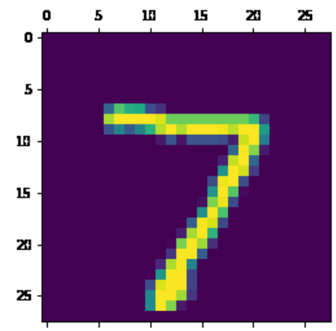
Testing Our 1st Test Data

y_predicted[0]

```
array([2.27106214e-02, 5.61349339e-07, 8.58167410e-02, 9.66806769e-01,
3.78498435e-03, 1.18708253e-01, 2.98759551e-06, 9.99864399e-01,
1.03265375e-01, 6.57766342e-01], dtype=float32)
```

Simple Neural Network without hidden layer with Flattened Data

plt.matshow(X_test[0])



np.argmax(y_predicted[0])
(This gives the index of the output layer where the maximum value occurs)

7

So our model gives accurate prediction for the 1st test data.

We shall check the labels for the first 5 test data

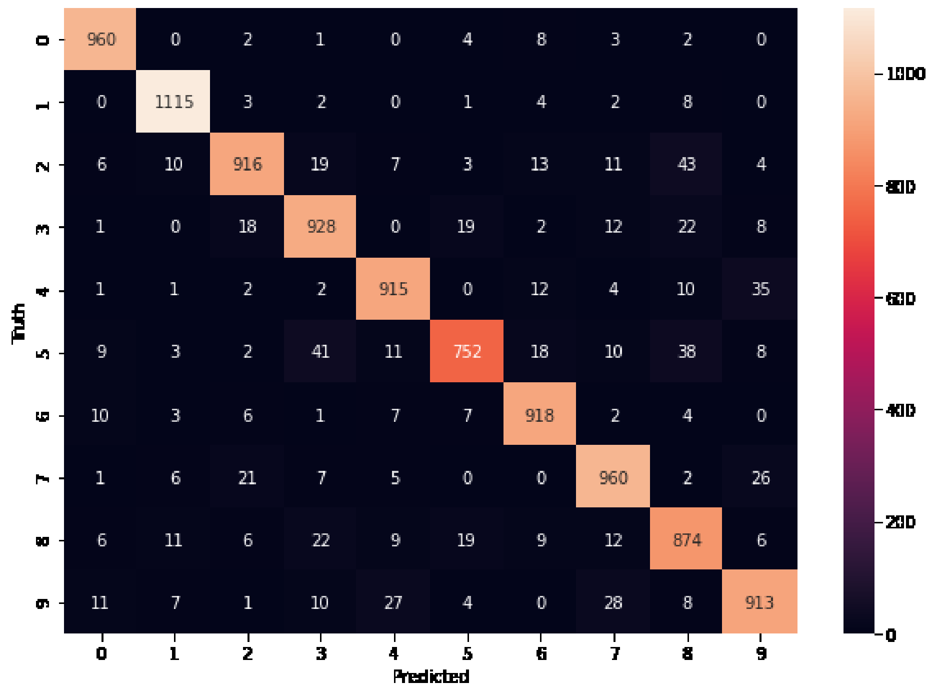
```
y_predicted_labels = [np.argmax(i) for i in y_predicted]
y_predicted_labels[:5]
```

[7, 2, 1, 0, 4]

Simple Neural Network without hidden layer with Flattened Data

We shall visualize the Confusion Matrix:

```
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```



HANDWRITTEN DIGIT RECOGNITION PROBLEM

Neural Network with hidden layer having 100 neuron with Flattened Data

```
model = keras.Sequential([
    keras.layers.Dense(100, input_shape=(784,), activation='relu'),
    keras.layers.Dense(10, activation='sigmoid')])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train_flattened, y_train, epochs=5)
```

```
...
Epoch 5/5 1875/1875 [==============================] - 4s
2ms/step - loss: 0.0516 - accuracy: 0.9843
```

HANDWRITTEN DIGIT RECOGNITION PROBLEM

Neural Network with hidden layer having 100 neuron with Flattened Data

```
model.evaluate(X_test_flattened, y_test)
```

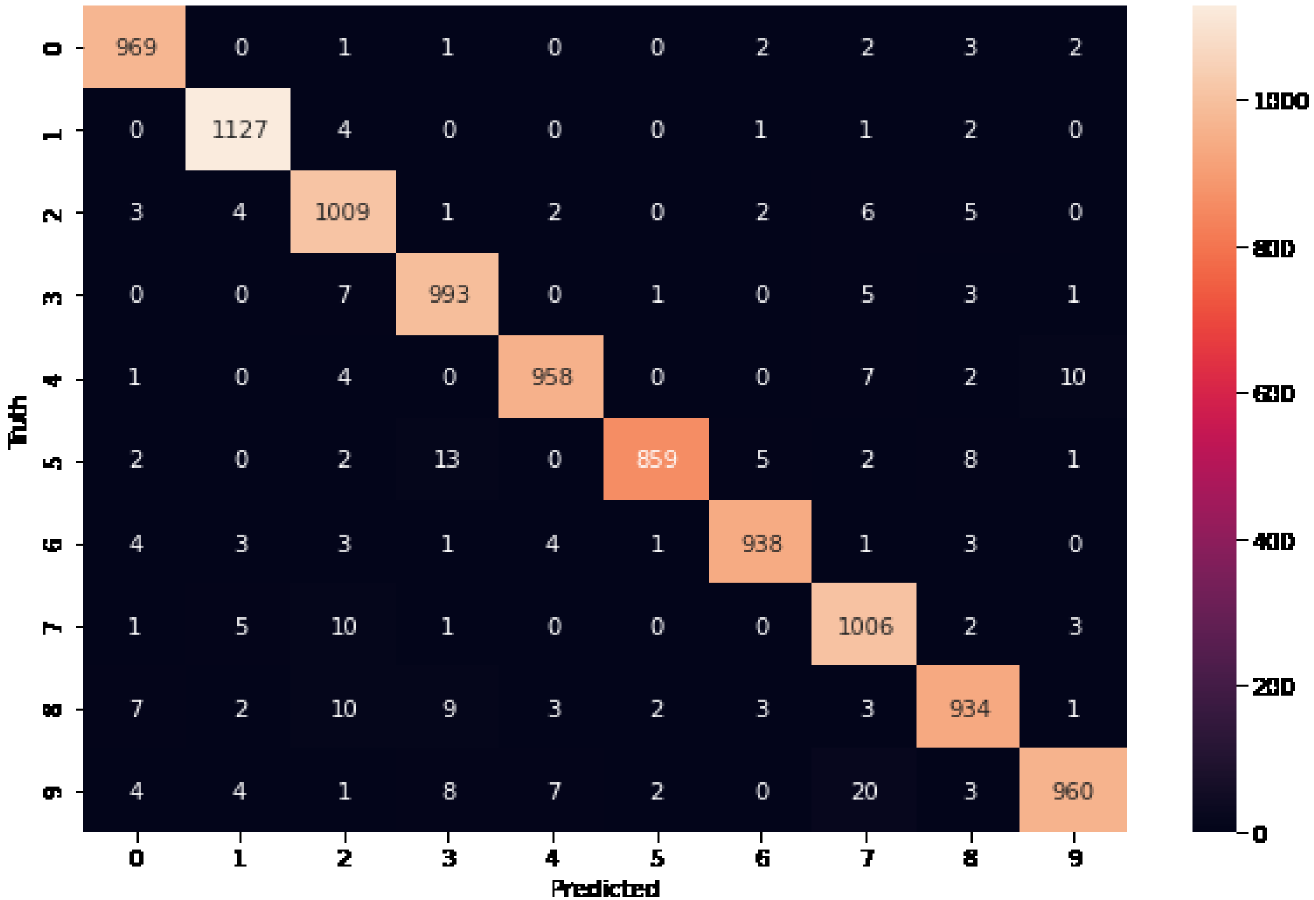
```
313/313 [==============================] - 1s 1ms/step - loss: 0.0785 -
accuracy: 0.9753 [0.07848526537418365, 0.9753000140190125]
```

Thus by adding a hidden layer the accuracy of the neural network has increased from 92.51% to 97.53%.

```
y_predicted = model.predict(X_test_flattened)
y_predicted_labels = [np.argmax(i) for i in y_predicted]
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

HANDWRITTEN DIGIT RECOGNITION PROBLEM

Neural Network with hidden layer having 100 neuron with Flattened Data



Neural Network with hidden layers having 100 neuron & using Flatten layer

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(100, activation='relu'),
    keras.layers.Dense(10, activation='sigmoid')])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

model.fit(X_train, y_train, epochs=5)
```

```
...
Epoch 5/5 1875/1875 [==============================] - 4s
2ms/step - loss: 0.0517 - accuracy: 0.9839
```

Neural Network with hidden layers having 100 neuron & using Flatten layer

```
model.evaluate(X_test, y_test)
```

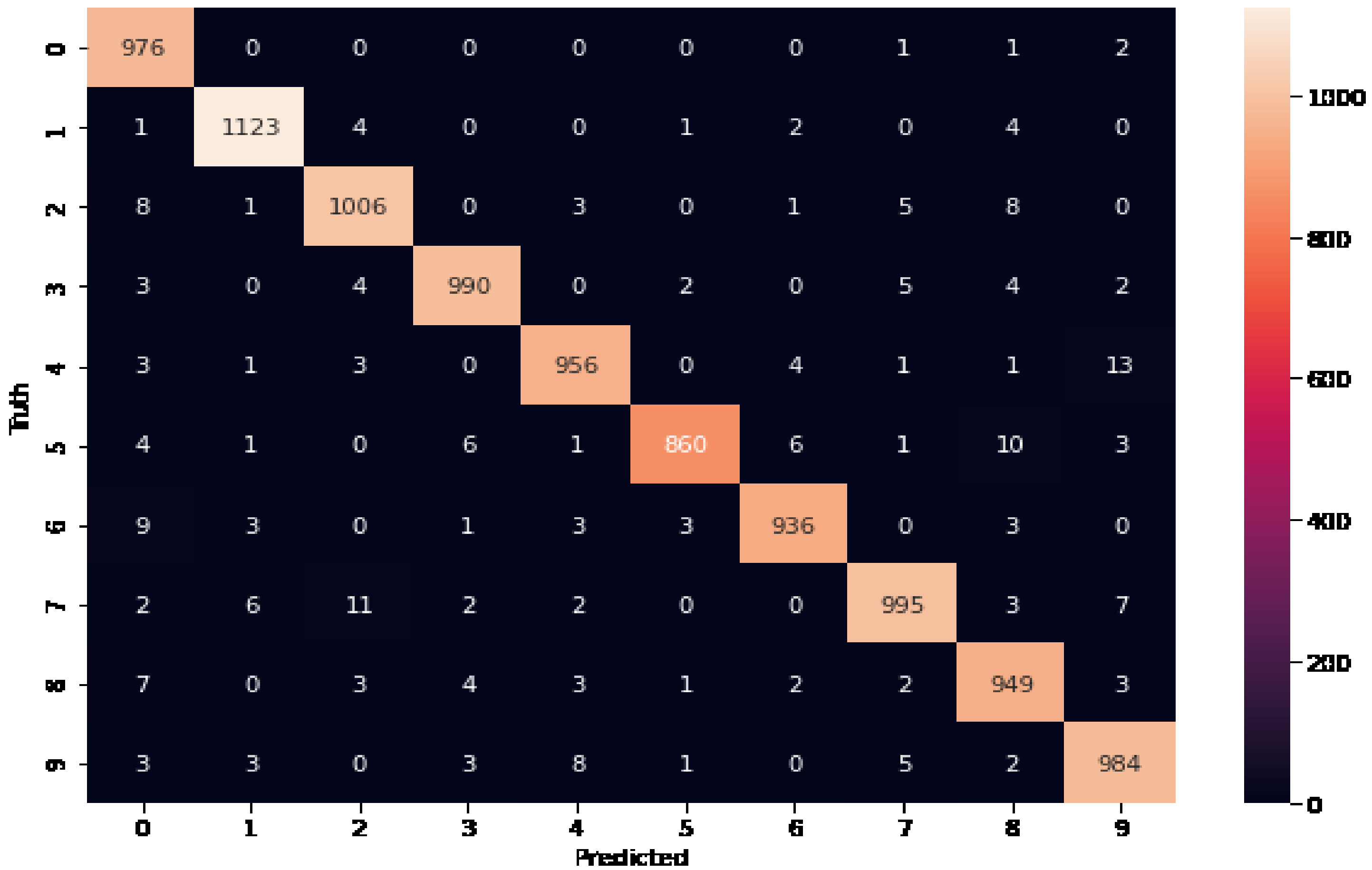
```
313/313 [==============================] - 0s 1ms/step - loss: 0.0794 - accuracy: 0.9775  [0.07937731593847275, 0.9775000214576721]
```

There is not a significant change in the accuracy as compared to the previous model but the advantage is that we need not flatten the data outside the neural network.

```
y_predicted = model.predict(X_test)
y_predicted_labels = [np.argmax(i) for i in y_predicted]
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

HANDWRITTEN DIGIT RECOGNITION PROBLEM

Neural Network with hidden layers having 100 neuron & using Flatten layer



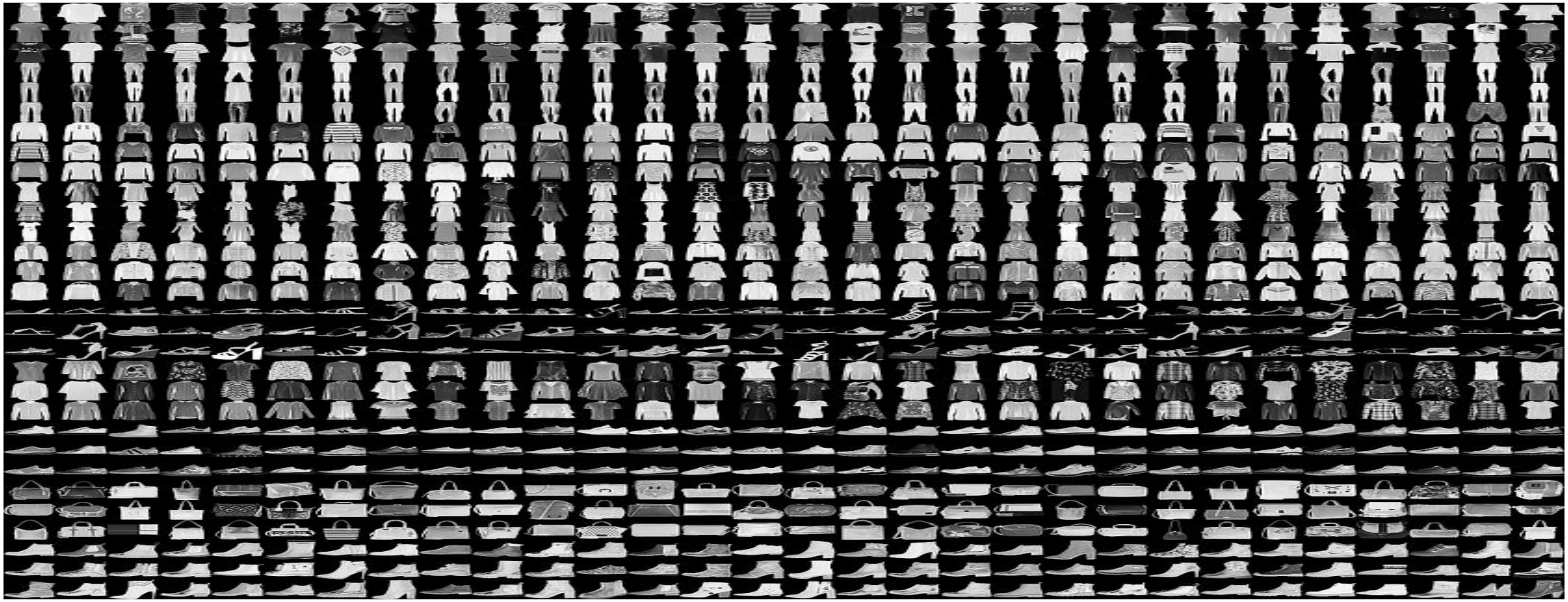
FASHION MNIST PROBLEM

- The digit recognition problem is one of the classic problems that has been used in the Machine Learning world for quite sometime is the MNIST problem. The objective is to identify the digit based on image. But MNIST is not very great problem because we come up with great accuracy even if we are looking at few pixels in the image. So, another common example problem against which we test algorithms is Fashion-MNIST.
- Fashion-MNIST is a dataset of Zalando's fashion article images —consisting of a **training set of 60,000** examples and **a test set of 10,000** examples. Each instance is a 28×28 greyscale image, associated with a label.
- The objective is to identify (predict) different fashion products from the given images using various best possible Machine Learning Models (Algorithms) and compare their results (performance measures/scores) to arrive at the best ML model

FASHION MNIST PROBLEM

CATEGORIES OF PRODUCTS

0	1	2	3	4	5	6	7	8	9
T-shirt/ top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle Boot



FASHION MNIST PROBLEM

LOAD THE DATA

```
fm = tf.keras.datasets.fashion_mnist
(trainX, trainy), (testX, testy) = fm.load_data()
```

DATA SIZE

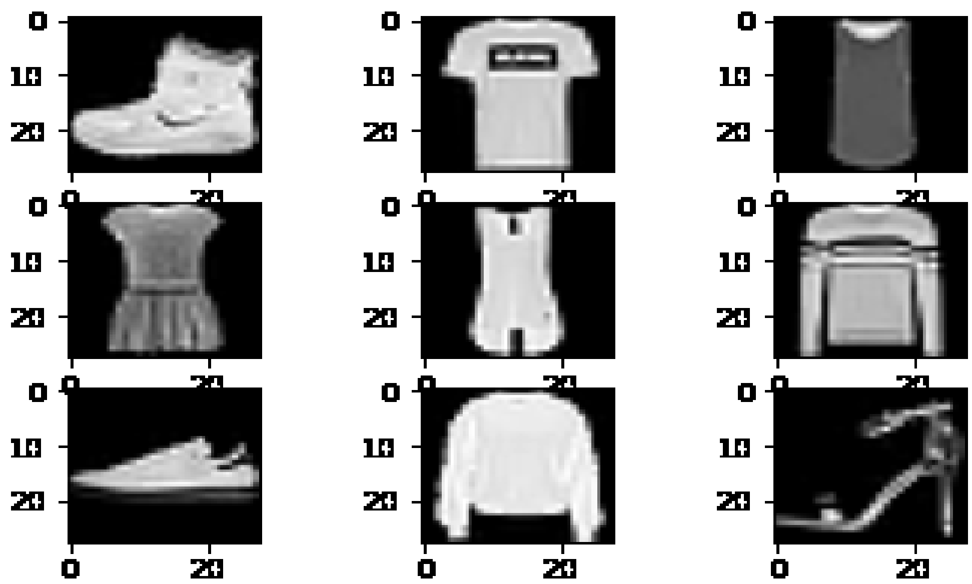
```
print('Train: X=%s, y=%s' % (trainX.shape, trainy.shape))
print('Test: X=%s, y=%s' % (testX.shape, testy.shape))
```

Train: X=(60000, 28, 28), y=(60000,) Test: X=(10000, 28, 28), y=(10000,)

FASHION MNIST PROBLEM

VISUALIZATION OF FEW INPUT DATA

```
class_labels = ["T-shirt/ top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt", “Sneaker", "Bag",                                  "Ankle boot"]
for i in range(9):
     pyplot.subplot(330 + 1 + i)
     pyplot.imshow(trainX[i], cmap=pyplot.get_cmap('gray'))
 pyplot.show()
```



FASHION MNIST PROBLEM

BUILDING THE SEQUENTIAL MODEL AND ADD LAYERS INTO IT

```
from keras.models import Sequential
from keras.layers import Flatten, Dense, Activation

model = Sequential()
model.add(Flatten(input_shape=[28, 28]))
model.add(Dense(100, activation="relu"))
model.add(Dense(10, activation="softmax"))
model.compile(loss="sparse_categorical_crossentropy",
            optimizer="adam",
            metrics=["accuracy"])
model.fit(trainX, trainy, epochs = 10)
```

```
…
Epoch 10/10 1875/1875 [==============================] - 4s 2ms/step - loss: 0.5266 -
accuracy: 0.8237
```

FASHION MNIST PROBLEM

TESTING MODEL ACCURACY

model.evaluate(testX, testy)

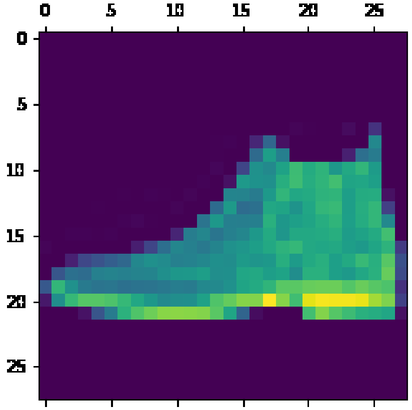
313/313 [==============================] - 1s 1ms/step - loss: 0.5724 - accuracy: 0.8132 [0.5723907947540283, 0.8131999969482422]

Above shows **accuracy score** of **81.31%.**

plt.matshow(testX[0])
yp = model.predict(testX)
yp_labels = [np.argmax(i) for i in yp]

np.argmax(yp[0]) **9**

class_labels[np.argmax(yp[0])]



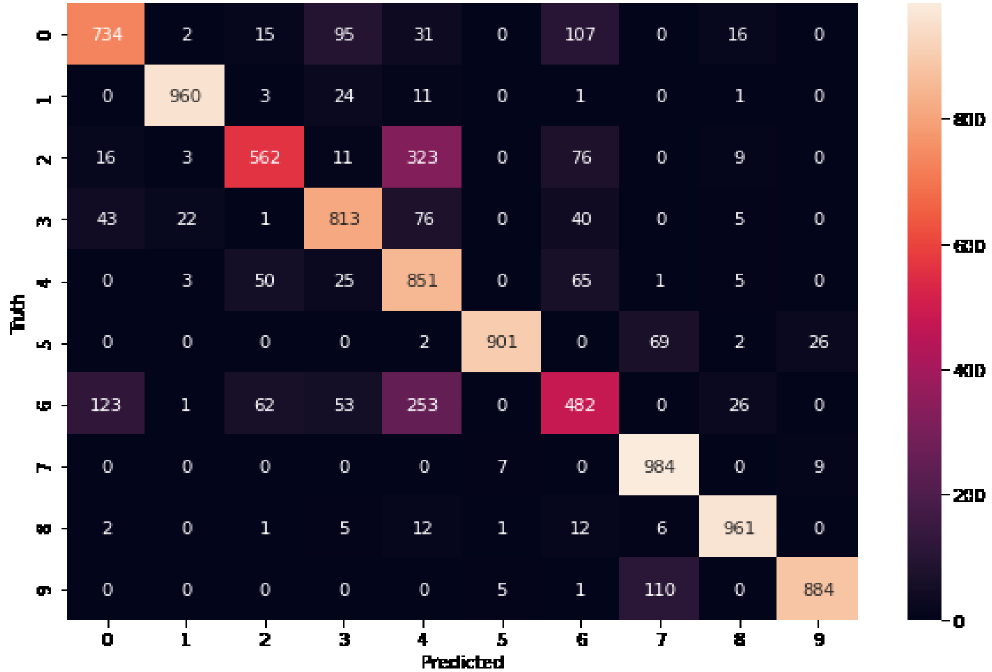
‘ Ankle Boot ’

So our model gives accurate prediction for the 1st test data.

FASHION MNIST PROBLEM

VISUALIZING THE CONFUSION METRIX

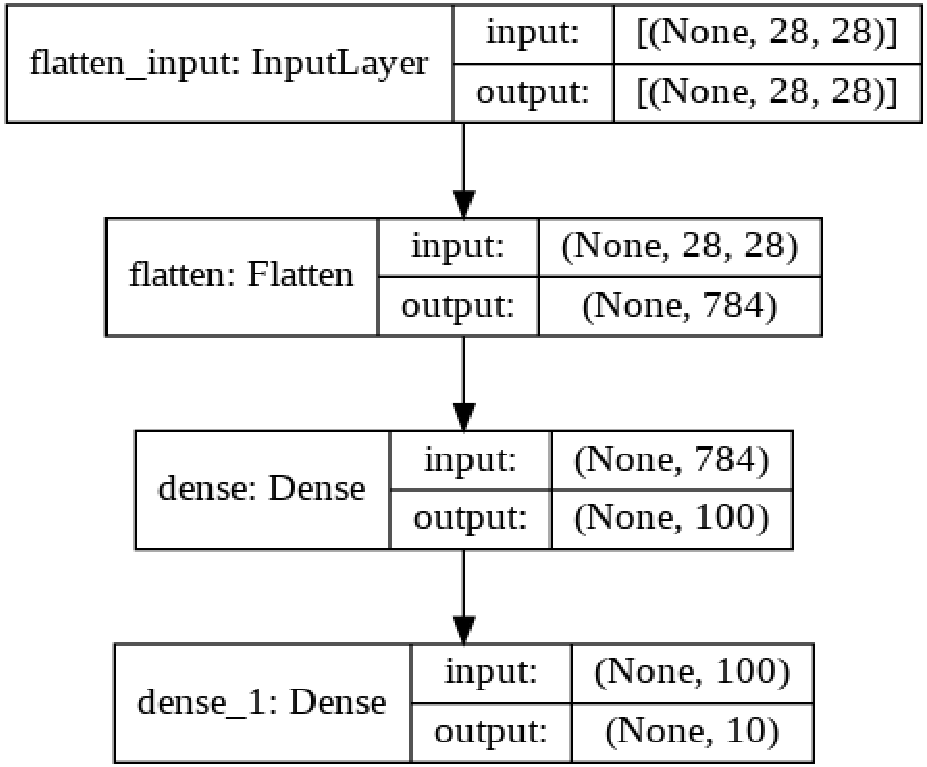
```
cm = tf.math.confusion_matrix(labels=testy,predictions=yp_labels)
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```



FASHION MNIST PROBLEM

VISUALIZING THE NEURAL NETWORK MODEL

```
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```



FASHION MNIST PROBLEM

APPLYING CONVOLUTIONAL NEURAL NETWORK ON OUR DATASET

IMPORT THE NECESSARY PACKAGES

```
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD
```

FASHION MNIST PROBLEM

BUILDING CNN

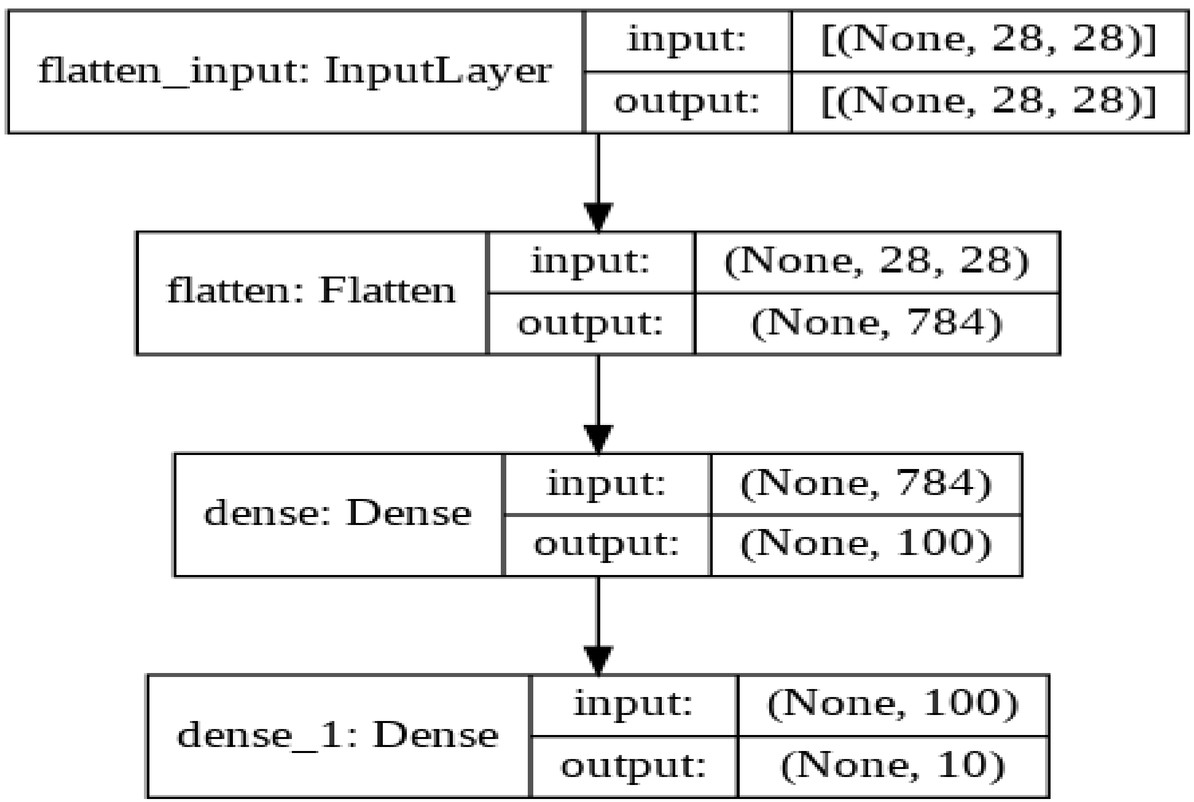
```
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape
              =(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    opt = SGD(lr=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model

model.fit(trainX, trainy, epochs = 10, batch_size=32, verbose=0)
model.evaluate(testX, testy)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.6233 - accuracy:
0.8119  [0.6233423352241516, 0.8119000196456909]
```

FASHION MNIST PROBLEM

VISUALIZING THE CONVOLUTIONAL NEURAL NETWORK MODEL

```
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```



Thank You