INTELLECUTUAL DATA VISUALISATION

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering

in

Electronics and Telecommunication

by

Tanveer Ahmed(17DET65)
Bhabhe Masih(15DET87)
Nagdade Asif (16DET107)
Qureshi Aatif (17DET55)

Under the guidance of

Prof.Rizwan Alvi

Department of Electronics and Telecommunication Engineering

Anjuman-I-Islam's Kalsekar Technical Campus
Sector 16, New Panvel , Navi Mumbai
University of Mumbai

2019-20

CERTIFICATE

Department of Electronics and Telecommunication Engineering
Anjuman-I-Islam's Kalsekar Technical Campus
Sector 16, New Panvel , Navi Mumbai
University of Mumbai

This is to certify that the project entitled **Intellectual Data Visualisation** is a bonafide work of**Tanveer Ahmed (17DET65),Bhabhe Masih (15DET87),Nagdade Asif (16DET107), Qureshi Aatif4(17DET55)**submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Department of Electronics and Telecommunication Engineering.

Supervisor	Examiner
Head of Department	Director

Project Report Approval for Bachelor of Engineering

This project entitled **Intellectual Data Visualisation"** by **Tanveer Ahmed ,Bhabhe Masih, Nagdade Asif,Qureshi Aatif**is approved for the degree of **Bachelor of Engineeringin Electronics and Telecommunication** .

Examiner

.....

Supervisor

.....

Date;

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

…
Tanveer Ahmed…
(17DET65)

Bhabhe Masih…
(15DET87)

Nagdade Asif…
(16DET107)

Qureshi Aatif…
(17DET65)

Date:

Acknowledgments

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals . I would like to extend my sincere thanks to all of them.

We am highly indebted to (Name of your Guide) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express my gratitude towards my parents& Staff of Anjuman-I-Islam's Kalsekar Technical Campus for their kind co-operation and encouragement which help me in completion of this project.

.

We My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Tanveer Ahmed(17DET65)
Bhabhe Masih(15DET87)
Nagdade Asif(16DET107)
Qureshi Aatif(17DET55)

Abstract

Globalization and technological advances has created an extremely competitive market. This also hasan impact on the banks. In recent years, banking and direct database marketing have become animportant strategy for understanding customer needs. The success rate of banking marketing dependson the achieved results and decisions. In order to make more accurate predictions, statistical tools andmethods are been used. This report examines how to use machine learning techniques to analyze and make predictions inbanking marketing using existing dataset. The purpose of building the models is to predict whether theclient will subscribe for a term deposit. This report presents the different stage of data analysis such asdata preparation and cleaning, building the models and model testing. Finally, the results of machinelearning techniques are evaluated and analysed. Although there is no significant difference in thedecision tree algorithm's accuracy, C5.0 achieved a higher percentage.Linear regression modelpresents the relationship between quantitative features.

Intellectual DataVisualisation:

NUMPY

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

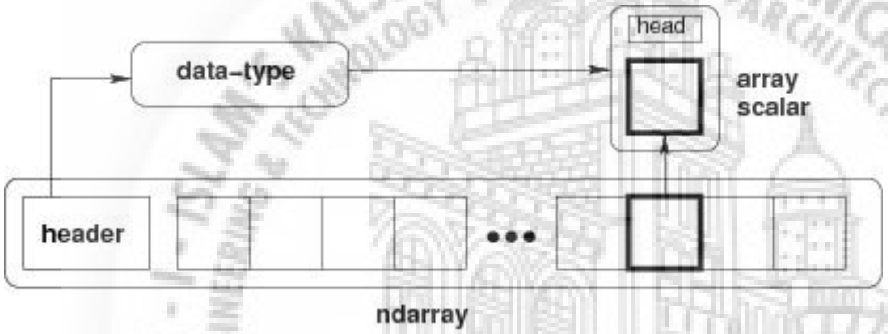
- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy - Ndarray Object

The most important object defined in NumPy is an N-dimensional array type called **ndarray**. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called **dtype**).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type −



An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows −

numpy.array

It creates an ndarray from any object exposing array interface, or from any method that returns an array.

numpy.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)

The above constructor takes the following parameters −

Sr.No.	Parameter & Description
1	**object** Any object exposing the array interface method returns an array, or any (nested) sequence.

2	**dtype** Desired data type of array, optional
3	**copy** Optional. By default (true), the object is copied
4	**order** C (row major) or F (column major) or A (any) (default)
5	**subok** By default, returned array forced to be a base class array. If true, sub-classes passed through
6	**ndmin** Specifies minimum dimensions of resultant array

Take a look at the following examples to understand better.

Example 1

```
import numpy as np
a = np.array([1,2,3])
printa
```

The output is as follows −

[1, 2, 3]

Example 2

```
# more than one dimensions
import numpy as np
a = np.array([[1,2],[3,4]])
print a
```

The output is as follows −

[[1, 2]
 [3, 4]]

Example 3

```
# minimum dimensions
import numpy as np
a = np.array([1,2,3,4,5], ndmin =2)
print a
```

The output is as follows −

```
[[1, 2, 3, 4, 5]]
```

Example 4

```
# dtype parameter
import numpy as np
a = np.array([1,2,3], dtype = complex)
printa
```

The output is as follows −

```
[ 1.+0.j,  2.+0.j,  3.+0.j]
```

The **ndarray** object consists of contiguous one-dimensional segment of computer memory, combined with an indexing scheme that maps each item to a location in the memory block. The memory block holds the elements in a row-major order (C style) or a column-major order (FORTRAN or MatLab style).

Data Type Objects (dtype)

A data type object describes interpretation of fixed block of memory corresponding to an array, depending on the following aspects −

- Type of data (integer, float or Python object)
- Size of data
- Byte order (little-endian or big-endian)
- In case of structured type, the names of fields, data type of each field and part of the memory block taken by each field.
- If data type is a subarray, its shape and data type

The byte order is decided by prefixing '<' or '>' to data type. '<' means that encoding is little-endian (least significant is stored in smallest address). '>' means that encoding is big-endian (most significant byte is stored in smallest address).

A dtype object is constructed using the following syntax −

```
numpy.dtype(object, align, copy)
```

The parameters are −

- Object − To be converted to data type object

- Align − If true, adds padding to the field to make it similar to C-struct
- Copy − Makes a new copy of dtype object. If false, the result is reference to builtin data type object

Example 1

```
# using array-scalar type
import numpy as np
dt = np.dtype(np.int32)
print dt
```

The output is as follows −

```
int32
```

Example 2

```
#int8, int16, int32, int64 can be replaced by equivalent string 'i1', 'i2','i4', etc.
import numpy as np

dt = np.dtype('i4')
print dt
```

The output is as follows −

```
int32
```

Example 3

```
# using endian notation
import numpy as np
dt = np.dtype('>i4')
print dt
```

The output is as follows −

```
>i4
```

The following examples show the use of structured data type. Here, the field name and the corresponding scalar data type is to be declared.

Example 4

```
# first create structured data type
import numpy as np
dt = np.dtype([('age',np.int8)])
print dt
```

The output is as follows −

[('age', 'i1')]

Example 5

```
# now apply it to ndarray object
import numpy as np

dt = np.dtype([('age',np.int8)])
a = np.array([(10,),(20,),(30,)], dtype = dt)
print a
```

The output is as follows −

[(10,) (20,) (30,)]

Example 6

```
# file name can be used to access content of age column
import numpy as np

dt = np.dtype([('age',np.int8)])
a = np.array([(10,),(20,),(30,)], dtype = dt)
print a['age']
```

The output is as follows −

[10 20 30]

Example 7

The following examples define a structured data type called student with a string field 'name', an integer field 'age' and a float field 'marks'. This dtype is applied to ndarray object.

```
import numpy as np
student = np.dtype([('name','S20'),('age','i1'),('marks','f4')])
print student
```

The output is as follows −

[('name', 'S20'), ('age', 'i1'), ('marks', '<f4')])

Example 8

```
import numpy asnp

student = np.dtype([('name','S20'),('age','i1'),('marks','f4')])
a = np.array([('abc',21,50),('xyz',18,75)], dtype = student)
```

```
printa
```

The output is as follows −

[('abc', 21, 50.0), ('xyz', 18, 75.0)]

Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Introduction to Data Structures

Pandas deals with the following three data structures −

- Series
- DataFrame
- Panel

These data structures are built on top of Numpy array, which means they are fast.

Dimension & Description

Data Structure	Dimensions	Description
Series	1	1D labeled homogeneous array, sizeimmutable.
Data Frames	2	General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns.
Panel	3	General 3D labeled, size-mutable array.

Series

Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56, …

10	23	56	17	52	61	73	90	26	72

DataFrame

DataFrame is a two-dimensional array with heterogeneous data. For example,

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

The table represents the data of a sales team of an organization with their overall performance rating. The data is represented in rows and columns. Each column represents an attribute and each row represents a person.

Data Type of Columns

The data types of the four columns are as follows −

Column	Type
Name	String
Age	Integer
Gender	String
Rating	Float

Panel

Panel is a three-dimensional data structure with heterogeneous data. It is hard to represent the panel in graphical representation. But a panel can be illustrated as a container of DataFrame.

Create an Empty Series

A basic series, which can be created is an Empty Series

Example

```
#import the pandas library and aliasing as pd
import pandas as pd
s = pd.Series()
print s
```

Its output is as follows −

```
Series([], dtype: float64)
```

Create a Series from ndarray

If data is an ndarray, then index passed must be of the same length. If no index is passed, then by default index will be **range(n)** where **n** is array length, i.e., [0,1,2,3…. **range(len(array))-1].**

Example 1

```
#import the pandas library and aliasing as pd
importpandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print s
```

Its output is as follows −

```
0  a
1  b
2  c
3  d
dtype: object
```

We did not pass any index, so by default, it assigned the indexes ranging from 0 to **len(data)-1**, i.e., 0 to 3.

Example 2

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data,index=[100,101,102,103])
prints
```

Its output is as follows −

```
100  a
101  b
102  c
103  d
dtype: object
```

We passed the index values here. Now we can see the customized indexed values in the output.

Create a Series from dict

A dict can be passed as input and if no index is specified, then the dictionary keys are taken in a sorted order to construct index. If **index** is passed, the values in data corresponding to the labels in the index will be pulled out.

Example 1

```
#import the pandas library and aliasing as pd
import pandas aspd
import numpy as np
data ={'a':0.,'b':1.,'c':2.}
s = pd.Series(data)
print s
```

Its output is as follows −

```
a 0.0
b 1.0
c 2.0
dtype: float64
```

Example 2

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data ={'a':0.,'b':1.,'c':2.}
s = pd.Series(data,index=['b','c','d','a'])
print s
```

Its output is as follows −

```
b 1.0
c 2.0
d NaN
a 0.0
dtype: float64
```

Create a Series from Scalar

If data is a scalar value, an index must be provided. The value will be repeated to match the length of index

```
#import the pandas library and aliasing as pd
importpandas as pd
import numpy asnp
s = pd.Series(5, index=[0,1,2,3])
print s
```

Its output is as follows −

```
0  5
1  5
```

```
2  5
3  5
dtype: int64
```

Accessing Data from Series with Position

Data in the series can be accessed similar to that in an **ndarray.**

Example 1

Retrieve the first element. As we already know, the counting starts from zero for the array, which means the first element is stored at zeroth position and so on.

```
#import pandas as pd
s = pd.Series([1,2,3,4,5],index =['a','b','c','d','e'])

#retrieve the first element
print s[0]
```

Its output is as follows −

```
1
```

Example 2

Retrieve the first three elements in the Series. If a : is inserted in front of it, all items from that index onwards will be extracted. If two parameters (with : between them) is used, items between the two indexes (not including the stop index).

```
#import pandas as pd
s = pd.Series([1,2,3,4,5],index =['a','b','c','d','e'])

#retrieve the first three element
print s[:3]
```

Its output is as follows −

```
a  1
b  2
c  3
dtype: int64
```

Example 3

Retrieve the last three elements.

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index =['a','b','c','d','e'])

#retrieve the last three element
print s[-3:]
```

Its output is as follows −

```
c  3
d  4
e  5
dtype: int64
```

Retrieve Data Using Label (Index)

A Series is like a fixed-size **dict** in that you can get and set values by index label.

Example 1

Retrieve a single element using index label value.

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index =['a','b','c','d','e'])

#retrieve a single element
print s['a']
```

Its output is as follows −

```
1
```

Example 2

Retrieve multiple elements using a list of index label values.

```
importpandas as pd
s = pd.Series([1,2,3,4,5],index =['a','b','c','d','e'])

#retrieve multiple elements
print s[['a','c','d']]
```

Its output is as follows −

```
a  1
c  3
d  4
dtype: int64
```

Example 3

If a label is not contained, an exception is raised.

```
importpandas as pd
s = pd.Series([1,2,3,4,5],index =['a','b','c','d','e'])

#retrieve multiple elements
print s['f']
```

Its output is as follows −

```
…
```

KeyError: 'f'

MATPLOTLIB

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays.

matplotlib.pyplot is a collection of command style functions that make Matplotlib work like matlab. Each Pyplot function makes some change to a figure. For example, a function creates a figure, a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

Types of Plots

Sr.No	Function & Description
1	**Bar** Make a bar plot.
2	**Barh** Make a horizontal bar plot.
3	**Boxplot** Make a box and whisker plot.
4	**Hist** Plot a histogram.
5	**hist2d** Make a 2D histogram plot.
6	**Pie** Plot a pie chart.
7	**Plot** Plot lines and/or markers to the Axes.

8	**Polar** Make a polar plot..
9	**Scatter** Make a scatter plot of x vs y.
10	**Stackplot** Draws a stacked area plot.
11	**Stem** Create a stem plot.
12	**Step** Make a step plot.
13	**Quiver** Plot a 2-D field of arrows.

Image Functions

Sr.No	Function & Description
1	**Imread** Read an image from a file into an array.
2	**Imsave** Save an array as in image file.
3	**Imshow** Display an image on the axes.

Axis Functions

Sr.No	Function & Description
1	**Axes** Add axes to the figure.
2	**Text** Add text to the axes.
3	**Title** Set a title of the current axes.
4	**Xlabel** Set the x axis label of the current axis.
5	**Xlim** Get or set the x limits of the current axes.
6	**Xscale** .
7	**Xticks** Get or set the x-limits of the current tick locations and labels.
8	**Ylabel** Set the y axis label of the current axis.
9	**Ylim** Get or set the y-limits of the current axes.
10	**Yscale** Set the scaling of the y-axis.

11	**Yticks** Get or set the y-limits of the current tick locations and labels.

Figure Functions

Sr.No	Function & Description
1	**Figtext** Add text to figure.
2	**Figure** Creates a new figure.
3	**Show** Display a figure.
4	**Savefig** Save the current figure.
5	**Close** Close a figure window.

Basic Plotting function in matplotlib

```
import matplotlib.pyplot as plt
```

Next we need an array of numbers to plot. Various array functions are defined in the NumPy library which is imported with the np alias.

```
import numpy as np
```

We now obtain the ndarray object of angles between 0 and 2π using the arange() function from the NumPy library.

```
x = np.arange(0, math.pi*2, 0.05)
```

The ndarray object serves as values on x axis of the graph. The corresponding sine values of angles in x to be displayed on y axis are obtained by the following statement −

```
y = np.sin(x)
```

The values from two arrays are plotted using the plot() function.

```
plt.plot(x,y)
```

You can set the plot title, and labels for x and y axes.

```
You can set the plot title, and labels for x and y axes.
plt.xlabel("angle")
plt.ylabel("sine")
plt.title('sine wave')
```

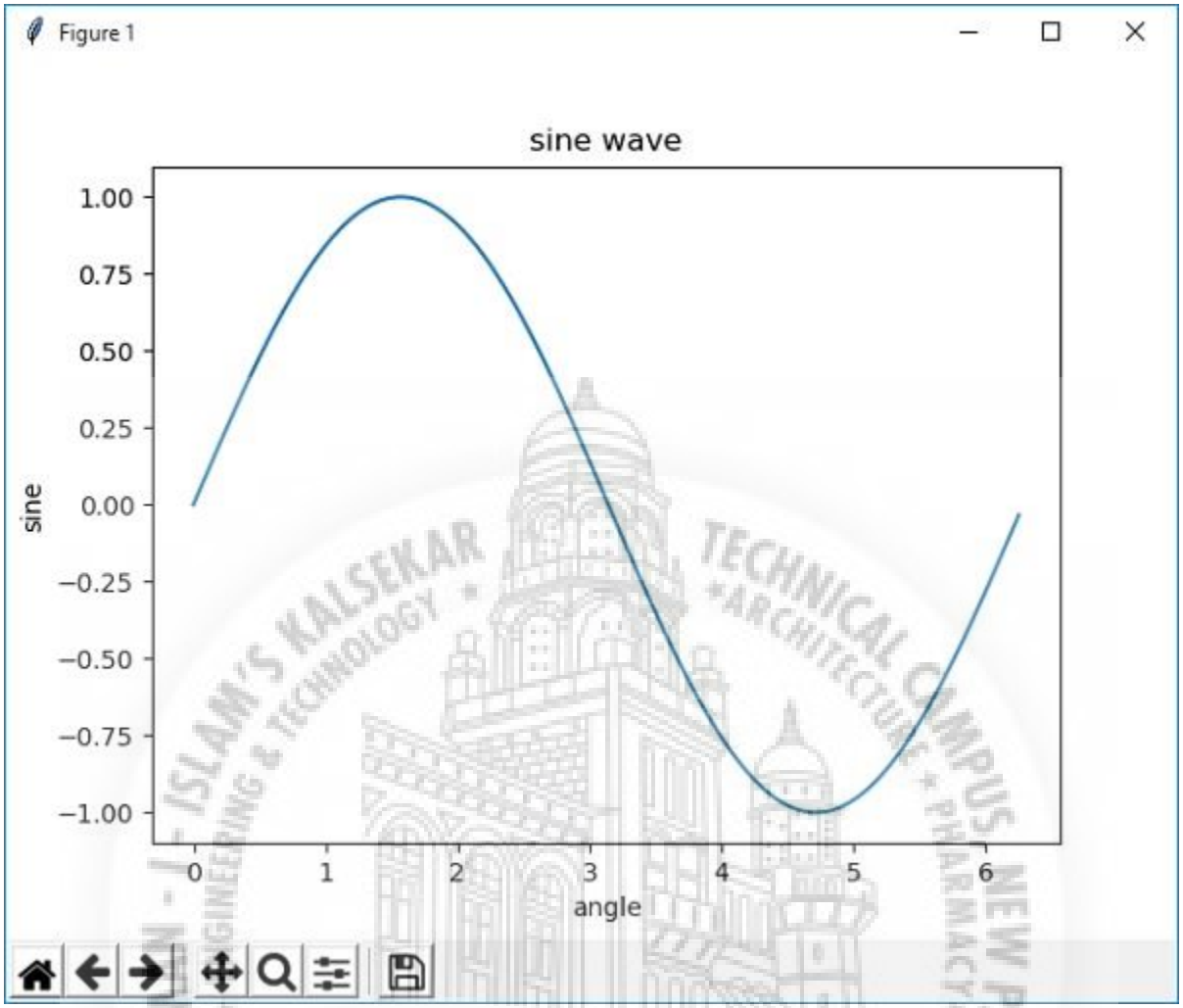
The Plot viewer window is invoked by the show() function −

```
plt.show()
```

The complete program is as follows −

```
from matplotlib import pyplot as plt
import numpy as np
import math #needed for definition of pi
x = np.arange(0, math.pi*2, 0.05)
y = np.sin(x)
plt.plot(x,y)
plt.xlabel("angle")
plt.ylabel("sine")
plt.title('sine wave')
plt.show()
```

When the above line of code is executed, the following graph is displayed −



Now, use the Jupyter notebook with Matplotlib.

Launch the Jupyter notebook from Anaconda navigator.

```
from matplotlib import pyplot as plt
import numpy as np
```

To display plot outputs inside the notebook itself (and not in the separate viewer), enter the following magic statement −

```
%matplotlib inline
```

Obtain x as the ndarray object containing angles in radians between 0 to 2π, and y as sine value of each angle −

```
import math
x = np.arange(0, math.pi*2, 0.05)
y = np.sin(x)
```

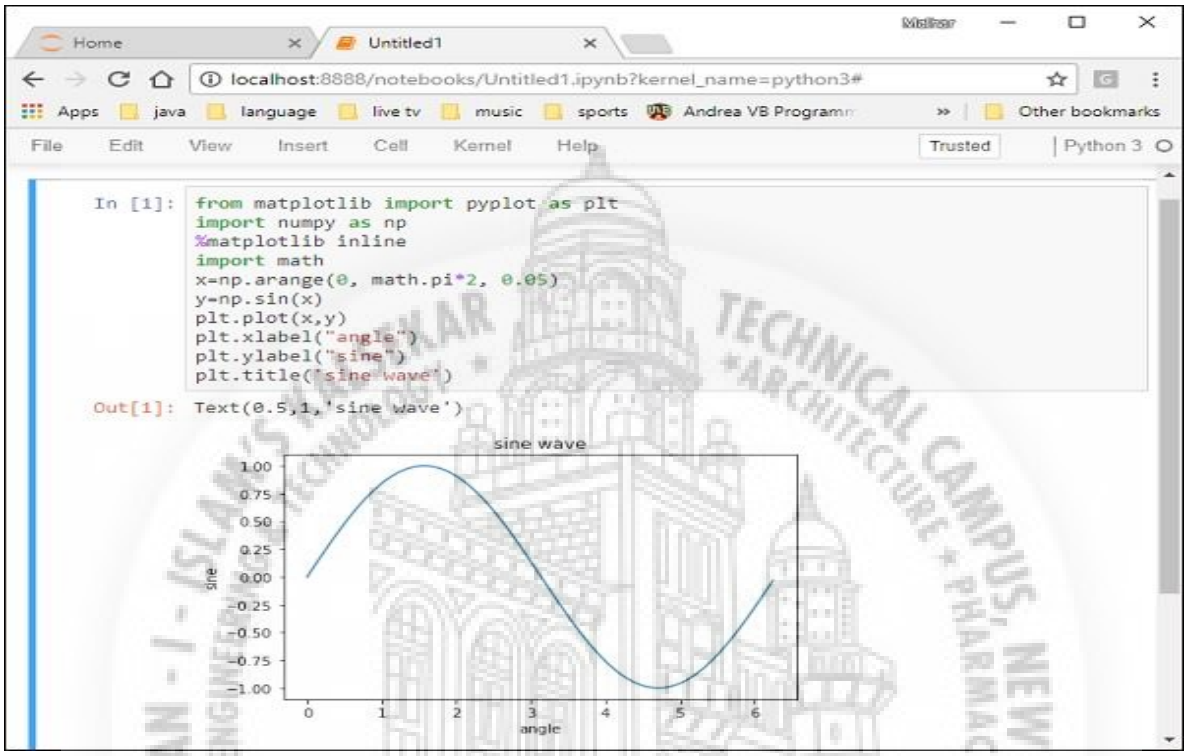
Set labels for x and y axes as well as the plot title −

```
plt.xlabel("angle")
plt.ylabel("sine")
plt.title('sine wave')
```

Finally execute the plot() function to generate the sine wave display in the notebook (no need to run the show() function) −

```
plt.plot(x,y)
```

After the execution of the final line of code, the following output is displayed −



Object-oriented Interface

To begin with, we create a figure instance which provides an empty canvas.

```
fig = plt.figure()
```

Now add axes to figure. The add_axes() method requires a list object of 4 elements corresponding to left, bottom, width and height of the figure. Each number must be between 0 and 1 −

```
ax=fig.add_axes([0,0,1,1])
```

Set labels for x and y axis as well as title −

```
ax.set_title("sine wave")
ax.set_xlabel('angle')
ax.set_ylabel('sine')
```

Invoke the plot() method of the axes object.

```
ax.plot(x,y)
```

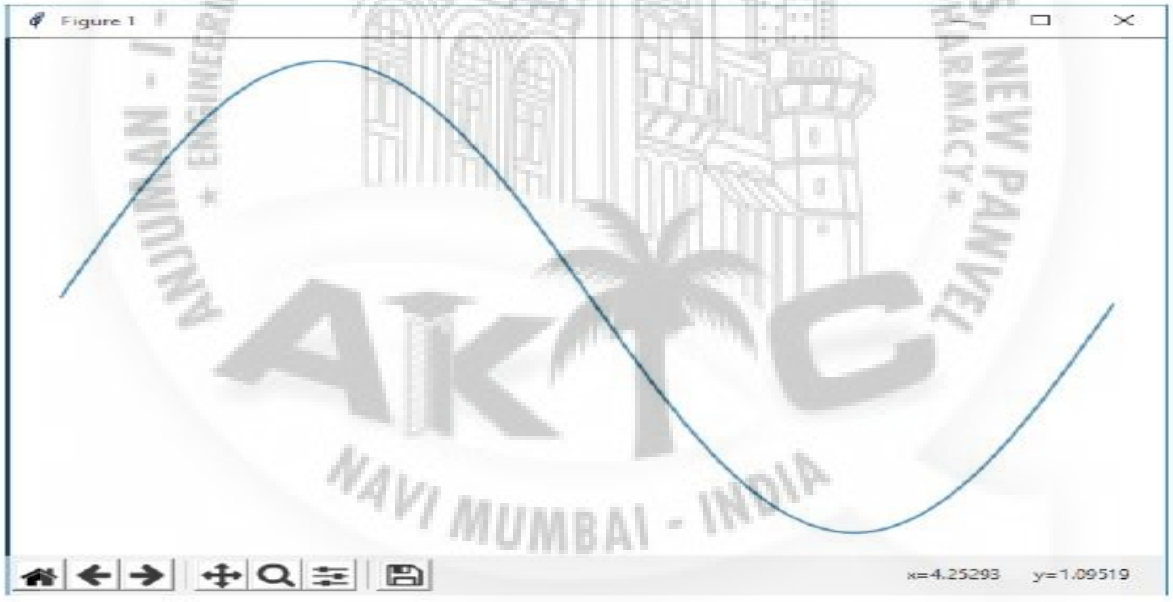
If you are using Jupyter notebook, the %matplotlib inline directive has to be issued; the otherwistshow() function of pyplot module displays the plot.

Consider executing the following code −

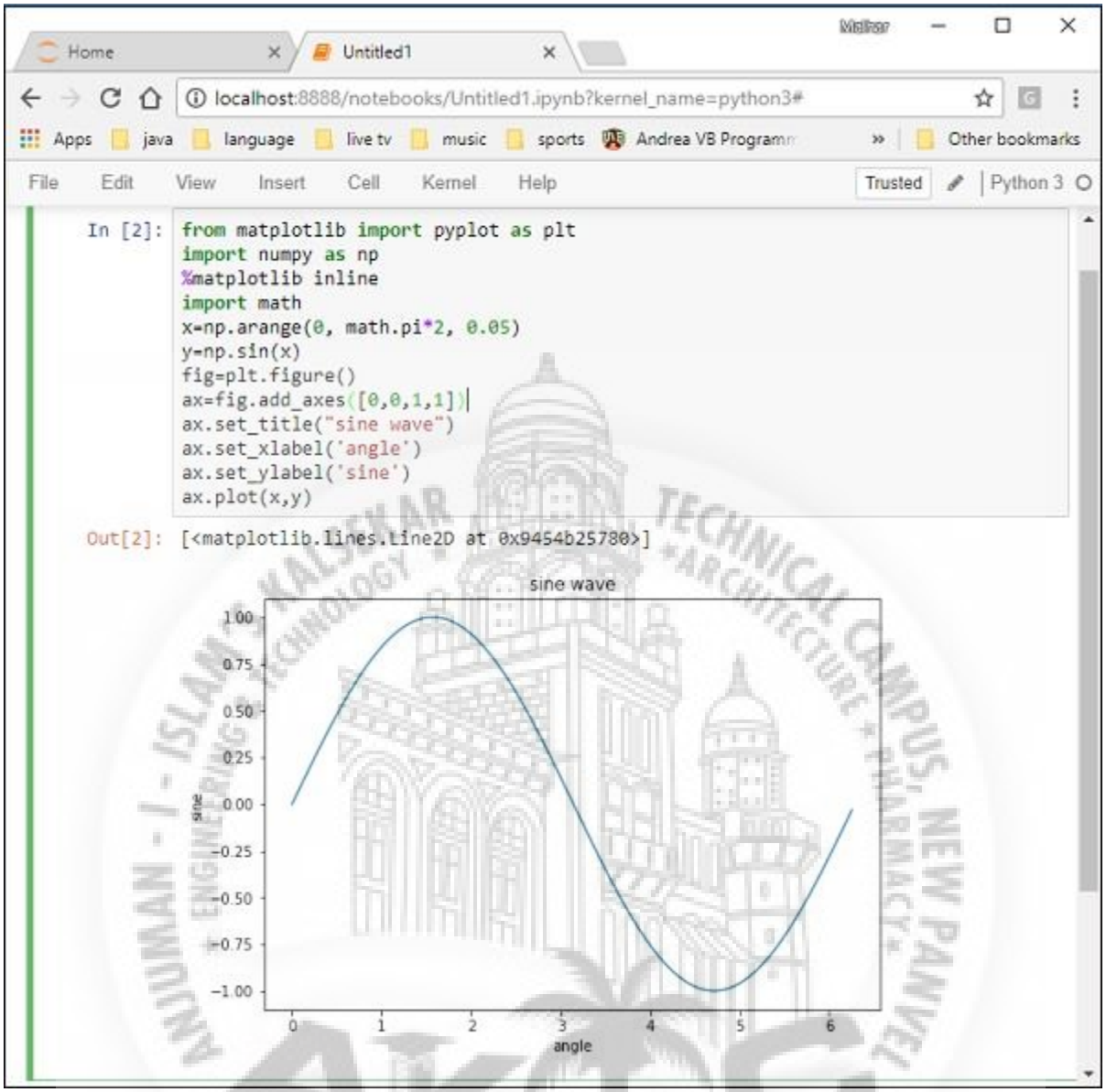
```
from matplotlib import pyplot as plt
import numpy as np
import math
x = np.arange(0, math.pi*2,0.05)
y = np.sin(x)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(x,y)
ax.set_title("sine wave")
ax.set_xlabel('angle')
ax.set_ylabel('sine')
plt.show()
```

Output

The above line of code generates the following output −



The same code when run in Jupyter notebook shows the output as shown below −



Bar Plot Using Matplotlib

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.

A bar graph shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value.

Matplotlib API provides the bar() function that can be used in the MATLAB style use as well as object oriented API. The signature of bar() function to be used with axes object is as follows −

```
ax.bar(x, height, width, bottom, align)
```

The function makes a bar plot with the bound rectangle of size (x −width = 2; x + width=2; bottom; bottom + height).

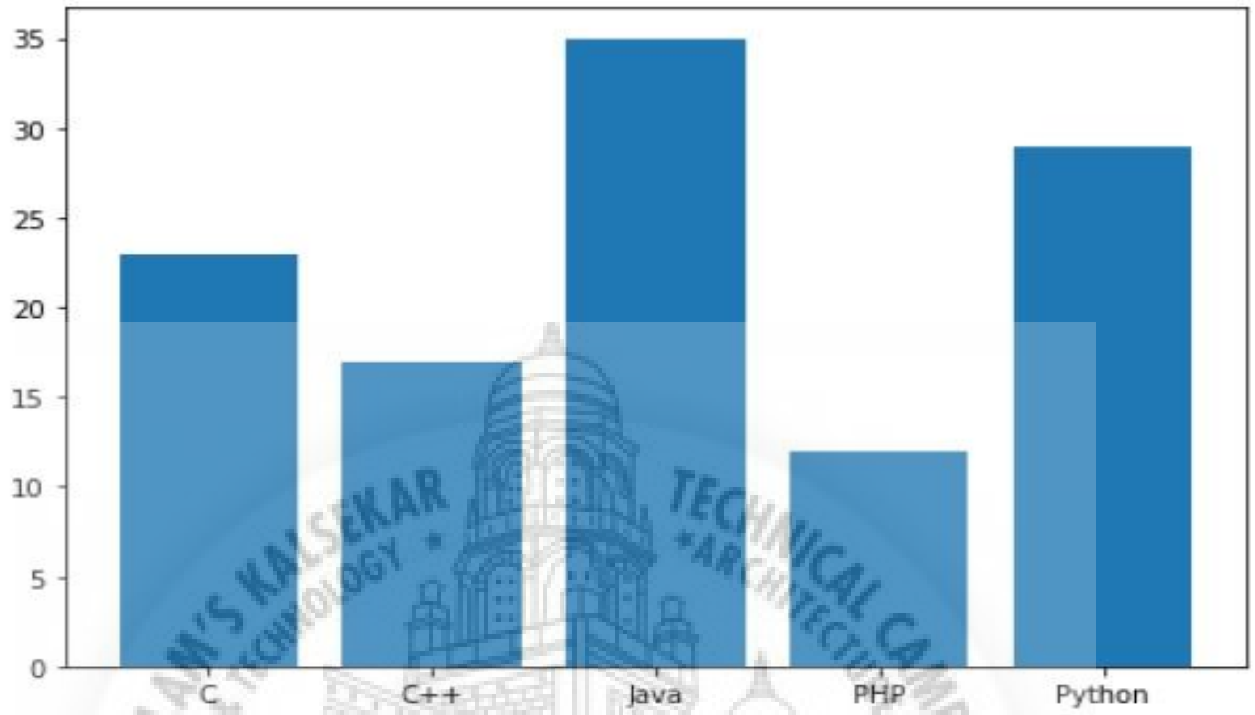
The parameters to the function are −

x	sequence of scalars representing the x coordinates of the bars. align controls if x is the bar center (default) or left edge.
height	scalar or sequence of scalars representing the height(s) of the bars.
width	scalar or array-like, optional. the width(s) of the bars default 0.8
bottom	scalar or array-like, optional. the y coordinate(s) of the bars default None.
align	{‘center’, ‘edge’}, optional, default ‘center’

The function returns a Matplotlib container object with all bars.

Following is a simple example of the Matplotlib bar plot. It shows the number of students enrolled for various courses offered at an institute.

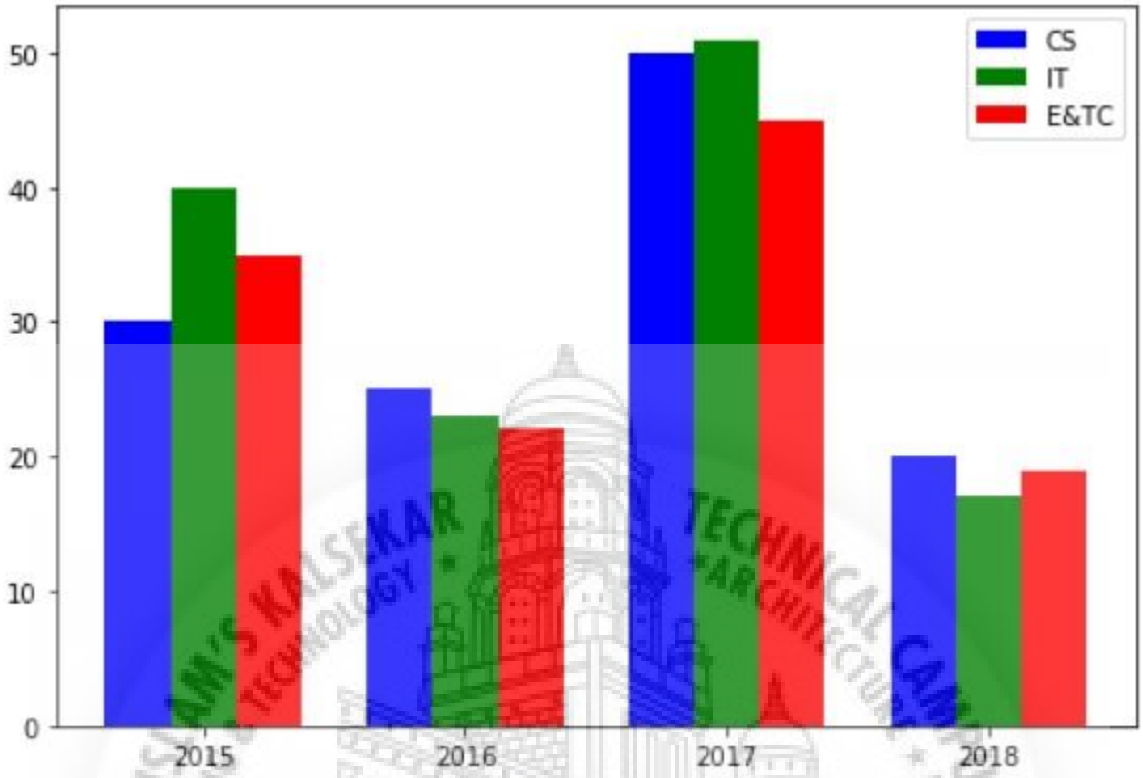
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
ax.bar(langs,students)
plt.show()
```



When comparing several quantities and when changing one variable, we might want a bar chart where we have bars of one color for one quantity value.

We can plot multiple bar charts by playing with the thickness and the positions of the bars. The data variable contains three series of four values. The following script will show three bar charts of four bars. The bars will have a thickness of 0.25 units. Each bar chart will be shifted 0.25 units from the previous one. The data object is a multidict containing number of students passed in three branches of an engineering college over the last four years.

```
import numpy as np
import matplotlib.pyplot as plt
data = [[30, 25, 50, 20],
[40, 23, 51, 17],
[35, 22, 45, 19]]
X = np.arange(4)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(X + 0.00, data[0], color = 'b', width = 0.25)
ax.bar(X + 0.25, data[1], color = 'g', width = 0.25)
ax.bar(X + 0.50, data[2], color = 'r', width = 0.25)
```

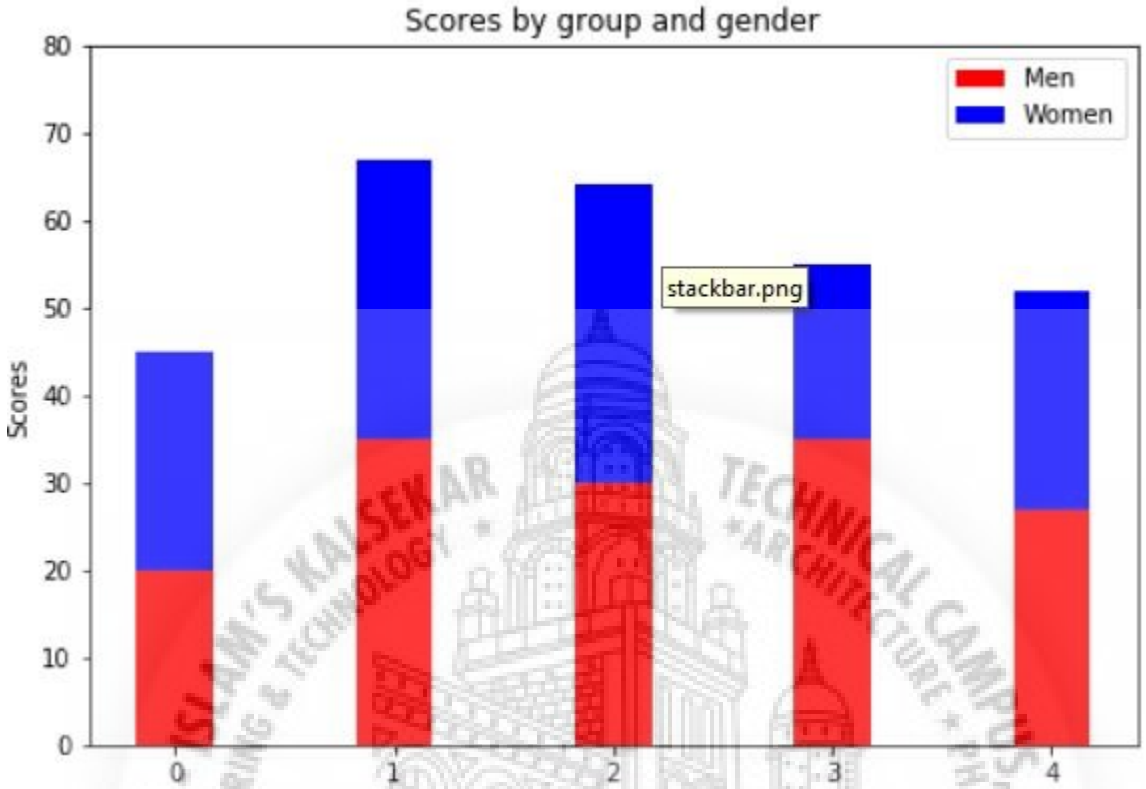


The stacked bar chart stacks bars that represent different groups on top of each other. The height of the resulting bar shows the combined result of the groups.

The optional bottom parameter of the pyplot.bar() function allows you to specify a starting value for a bar. Instead of running from zero to a value, it will go from the bottom to the value. The first call to pyplot.bar() plots the blue bars. The second call to pyplot.bar() plots the red bars, with the bottom of the blue bars being at the top of the red bars.

```
import numpy as np
import matplotlib.pyplot as plt
N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
ind = np.arange(N) # the x locations for the groups
width = 0.35
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(ind, menMeans, width, color='r')
ax.bar(ind, womenMeans, width,bottom=menMeans, color='b')
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
ax.set_yticks(np.arange(0, 81, 10))
ax.legend(labels=['Men', 'Women'])
```

```
plt.show()
```


Scores by group and gender
Men
Women
stackbar.png
Scores
80
70
60
50
40
30
20
10
0
0
1
2
3
4

Linear Regression

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

The general mathematical equation for a linear regression is −

y = ax + b

Following is the description of the parameters used −

- **y** is the response variable.
- **x** is the predictor variable.
- **a** and **b** are constants which are called the coefficients.

Steps to Establish a Regression

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is −

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the **lm()** functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
- To predict the weight of new persons, use the predict() function in R.

Input Data

Below is the sample data representing the observations −

```
# Values of height
151, 174, 138, 186, 128, 136, 179, 163, 152, 131

# Values of weight.
63, 81, 56, 91, 47, 57, 76, 72, 62, 48
```

lm() Function

This function creates the relationship model between the predictor and the response variable.

Syntax

The basic syntax for **lm()** function in linear regression is −

```
lm(formula,data)
```

Following is the description of the parameters used −

- formula is a symbol presenting the relation between x and y.
- data is the vector on which the formula will be applied.

Create Relationship Model & get the Coefficients

```
x <- c(151,174,138,186,128,136,179,163,152,131)
y <- c(63,81,56,91,47,57,76,72,62,48)

# Apply the lm() function.
relation <- lm(y~x)

print(relation)
```

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

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   -38.4551        0.6746
```

Get the Summary of the Relationship

```
x <- c(151,174,138,186,128,136,179,163,152,131)
y <- c(63,81,56,91,47,57,76,72,62,48)

# Apply the lm() function.
relation <- lm(y~x)

print(summary(relation))
```

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

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-6.3002  -1.6629  0.0412  1.8944  3.9775

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -38.45509    8.04901  -4.778  0.00139 **
x             0.67461    0.05191  12.997 1.16e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.253 on 8 degrees of freedom
Multiple R-squared:  0.9548,    Adjusted R-squared:  0.9491
F-statistic: 168.9 on 1 and 8 DF,  p-value: 1.164e-06
```

predict() Function

Syntax

The basic syntax for predict() in linear regression is −

```
predict(object, newdata)
```

Following is the description of the parameters used −

- object is the formula which is already created using the lm() function.
- newdata is the vector containing the new value for predictor variable.

Predict the weight of new persons

```
# The predictor vector.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

# The resposne vector.
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

# Find weight of a person with height 170.
a <- data.frame(x = 170)
result <-  predict(relation,a)
```

```
print(result)
```

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

```
      1
76.22869
```

Visualize the Regression Graphically

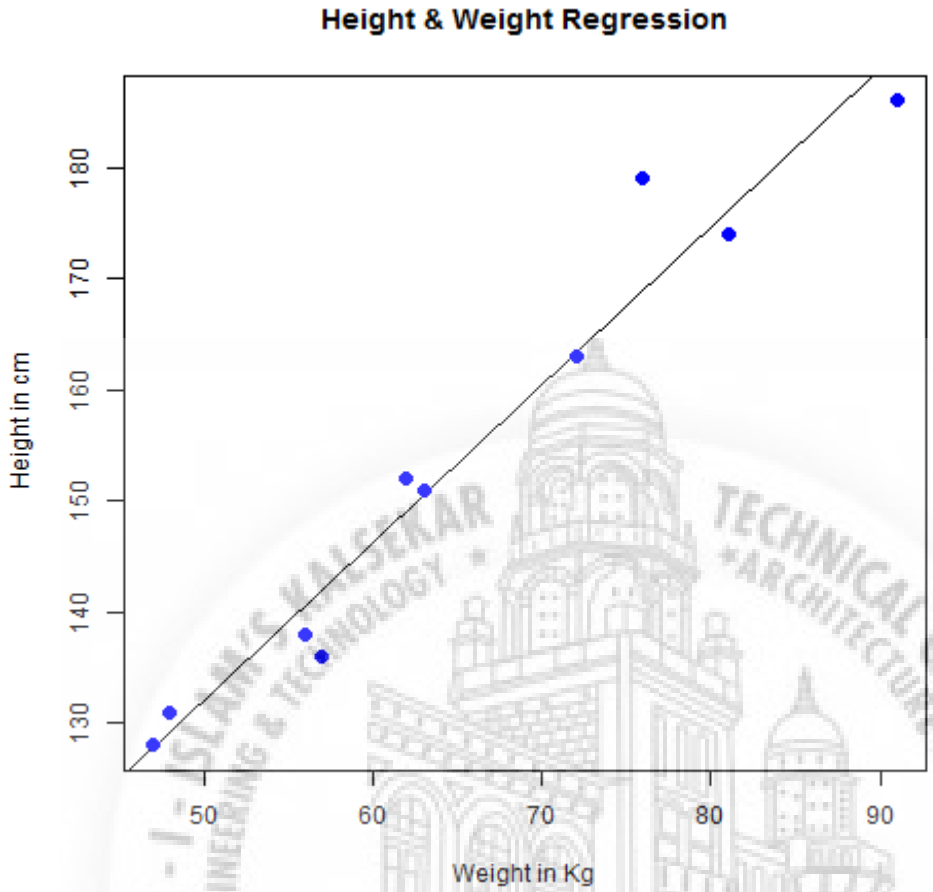
```
# Create the predictor and response variable.
x <- c(151,174,138,186,128,136,179,163,152,131)
y <- c(63,81,56,91,47,57,76,72,62,48)
relation <- lm(y~x)

# Give the chart file a name.
png(file ="linearregression.png")

# Plot the chart.
plot(y,x,col ="blue",main ="Height & Weight Regression",
abline(lm(x~y)),cex =1.3,pch =16,xlab ="Weight in Kg",ylab ="Height in cm")

# Save the file.
dev.off()
```

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


Height & Weight Regression
Height in cm
130
140
150
160
170
180
50
60
70
80
90
Weight in Kg

Contents

Chapter 1

Introduction

In the last years, machine learning (ML) has grown into one of the most significant IT and Artificialintelligence (AI) branches. This is a specific sub-group of AI based on the idea that the machine canlearn by identifying patterns and make predictions in various data problems with minimum humanintervention. Machine learning is a data analysis method that is widely used in various business andindustrial sectors. The main reason for that because ML can build predictive models to produce betterpredictions and achieve the desired level of accuracy, leading to better outcomes.The aim of the project is to find how to use machine learning techniques for analysis and makingpredictions using existing dataset in banking marketing. To find how they can be used together in aprocess of converting raw data to effective decision making knowledge. Building the predictive modelswill help to predict whether the client will subscribe for a term deposit.This report will describe the different stages of preparation and implementation of the predictivemodels, staring with a literature review on machine learning techniques; in particular, linear regressionand decision trees and how machine learning techniques are used in banking marking.Once the literature review of these techniques has been revised, a methodology will be composed onhow to pursue the investigation. The methodology is needed to establish how the implementation of themodels will continue.After the establishment of the methodology, the methods of data cleaning and preparation methods onthe raw data will be described and explained. The project will identify probabilities and visualize theresults in order to improve the solutions and achieve desired outcomes. At the same time, a goodunderstanding of banking marketing dataset will be provided so that the scope of the analysis can beclearly defined.The implementation will include building the linear regression and Decision trees machine learningtechniques using R. The dataset used in this project is downloaded from UCI Irvine machine learningrepository (At the same time, the practical development will try to test how thetheory works and find better solutions. The report will include visualizing the results, testing and evaluating the performance of the models and solving various data problems.Then, by analysis of the performance and achieved results, and summarise them at the conclusion, the aim and objectives of the project will be fulfilled.

Chapter 2

Literature Review

This project examines how to use machine learning techniques to analyze and make predictions inbanking marketing using existing dataset. In order to find how they are used in a process of convertingraw data to effective decision making knowledge, the literature review on machine learning methodswas carried out for existing research papers and books; in particular, linear regression and decision trees.By the same token, the use of machine learning techniques in the banking marketing sector isinvestigated based on last published articles and research papers. The main purpose of the literaturereview is to give a clear idea of what was written as a theory of these techniques and how they wereused to make predictions in banking marking.

PaperTitle

Relevant Papers:

S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimaraes, Portugal, October, 2011. EUROSIS. [bank.zip]

Weaknesses

A bank usually invests the customer's long-term deposits into riskier financial Assets.

How toOvercome

In this project we apply machine learning algorithms tu build a predictive model and attract the clients for their term deposit to enhance the business by using the shown chunk of codes select the randomsample of size 500.

Chapter 3

TechnicalDetails

Methodology

In order to find how to use machine learning techniques for analysis and making predictions usingexisting dataset in banking marketing are used two machine learning – Decision trees and Simple andMultiple Linear Regression, in particular – CART based on Gini method of attribute selection and C5.0– Entropy. Data cleaning and preparation methods are applied such as statistical methods to check formissing values and outliers, converting the datatype, visualization of variables using ggplot, method ofdividing the dataset into training and test data, confusion matrix to evaluate the accuracy of the modelsand Scatterplot matrix for visualization of the correlation between the quantitative features. The analysisis performed by R language and environment. R is free software for statistical computing. R language isa function based. To take advantages of R, R packages are installed and reloaded each time by librarieswith the same name.

in this project are used decision tree algorithms to predictwhether the client will subscribe for a term deposit. Although there is no significant difference in theresult accuracy, C5.0 achieved a higher percentage. At the same time, the linear model is applied torepresent the relationship between quantitative attributes and how significantly relevant they are

ProjectRequirements

In this project it is demonstrated that how to build a model predicting clients subscribing to a term deposit using the following steps –

These different methods are used for the processing and accessing of data

- Project definition
- Data exploration
- Feature engineering
- Building training/validation/test samples
- Model selection
- Model evaluation

SoftwareRequirements

Annaconda Navigator Shell

Jupyter Notebook (Annaconda Navigator Shell)

Microsoft Excel (for CSV data)

Chapter 4

Market Potential

Market Potential ofProject

- n today's world, data is the king. Use it in an effective way and it can create a huge impact on your business, don't leverage it and you will be left behind in this fast paced world in no time. And one of the ways that a organisation can improve it's performance in the market is to capture and analyse customer data in an efficient way to improve the customer experience.

- One of the industries that is being transformed the most by the recent Machine learning advances is the finance industry. Be it predicting stock prices, or in our case predicting if a customer will subscribe to a term deposit Machine learning can be an incredibly useful tool for providing better profitability.

Competitive Advantages ofProject

Predictive Analysis Using Multiple Machine Learning Tech- niques ... the project is on analyzing the customers for long-term deposits in the banks, and the ... The main advantage of using this algorithm is it increases the competitive accuracy and quick learning capacity of the process.

Chapter 5

Conclusion and FutureScope

Conclusion

- By applying logistic and ridge regression algorithms, classification and estimation model were successfully built. With these two models, the bank will be able to predict a customer's response to its telemarketing campaign before calling this customer. In this way, the bank can allocate more marketing efforts to the clients who are classified as highly likely to accept term deposits, and call less to those who are unlikely to make term deposits.

- In addition, predicting duration before calling and adjusting marketing plan benefit both the bank and its clients. On the one hand, it will increase the efficiency of the bank's telemarketing campaign, saving time and efforts. On the other hand, it prevents some clients from receiving undesirable advertisements, raising customer satisfaction. With the aid of logistic and ridge regression models, the bank can enter a virtuous cycle of effective marketing, more investments and happier customers.

  ○

FutureScope

1.More appropriate timing

When implementing a marketing strategy, external factors, such as the time of calling, should also be carefully considered. The previous analysis points out that March, September, October and December had the highest success rates. Nevertheless, more data should be collected and analyzed to make sure that this seasonal effect is constant over time. If the trend has the potential to continue in the future, the bank should consider initiating its telemarketing campaign in fall and spring.

2. Smarter marketing design

By targeting the right customers, the bank will have more and more positive responses, and the classification algorithms would ultimately eliminate the imbalance in the original dataset. Hence, more accurate information will be presented to the bank for improving the subscriptions. Meanwhile, to increase the likelihood of subscription, the bank should re-evaluate the content and design of its current campaign, making it more appealing to its target customers.

3. Better services provision

With a more granular understanding of its customer base, the bank has the ability to provide better banking services. For example, marital status and occupation reveal a customer's life stage while loan status indicates his/her overall risk profile. With this information, the bank can estimate when a customer might need to make an investment. In this way, the bank can better satisfy its customer demand by providing banking services for the right customer at the right time.

References

:

- https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/
- https://www.jejurtv.com/blank-6
- https://giphy.com/gifs/animation-animated-l378c04F2fjeZ7vH2
- http://algolytics.com/tutorial-how-to-determine-the-quality-and-correctness-of-classification-models-part-1-introduction/
- https://sendiancreations.com/our-marketing-services/best-motion-graphics/opening/
- https://www.agencialorean.com.br/conteudos/ineficaz-organico

INTELLECTUAL DATA VISUALIZATION

ANJUMAN - I - ISLAM'S KALSEKAR
ENGINEERING & TECHNOLOGY
TECHNICAL CAMPUS, NEW PANVEL
ARCHITECTURE
PHARMACY
AIKTC
NAVI MUMBAI - INDIA