RV COLLEGE OF ENGINEERING®

(*Autonomous Institution, Affiliated to VTU, Belagavi*)

Bengaluru, Karnataka– 560059

Lab Manual for 7th Semester

DATA SCIENCE AND ENGINEERING

(16IS72)

Department of Information Science & Engineering

Faculty In-charge

Prof. Smitha G R

USN	**1RV17IS058**
NAME	**VISMITHA N**

ACADEMIC YEAR	2020-21

Vision, Mission, PEO, PO and PSO of the department

Vision

To be the hub for innovation in Information Science & Engineering through Teaching, Research, Development and Consultancy; thus make the department a well known resource center in advanced sustainable and inclusive technology.

Mission

ISE1: To enable students to become responsible professionals, strong in fundamentals of information science and engineering through experiential learning.

ISE2: To bring research and entrepreneurship into class rooms by continuous design of innovative solutions through research publications and dynamic development oriented curriculum.

ISE3: To facilitate continuous interaction with the outside world through student internship, faculty consultancy, workshops, faculty development programmes, industry collaboration and association with the professional societies.

ISE4: To create a new generation of entrepreneurial problem solvers for a sustainable future through green technology with an emphasis on ethical practices, inclusive societal concerns and environment.

ISE5: To promote team work through inter-disciplinary projects, co-curricular and social activities.

Program Educational Objectives (PEOs)

PEO1: To provide adaptive and agile skills in Information Science and Engineering needed for professional excellence / higher studies /Employment, in rapidly changing scenarios.

PEO2: To provide students a strong foundation in basic sciences and its applications to technology.

PEO3: To train students in core areas of Information science and Engineering, enabling them to analyze, design and create products and solutions for the real world problems, in the context of changing technical, financial, managerial and legal issues.

PEO4: To inculcate leadership, professional ethics, effective communication, team spirit, multi-disciplinary approach in students and an ability to relate Information Engineering issues to social and environmental context.

PEO5: To motivate students to develop passion for lifelong learning, innovation, career growth and professional achievement.

Program Outcomes (PO)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to

assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the Engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcome (PSO)

PSO-1

Recognize and appreciate the principles of theoretical foundations, data organization, data communication, security and data analytical methods in the evolving technology

PSO-2

Learn the applicability of various system softwares for the development of quality products in solving real-world problems with a focus on performance optimization

PSO-3

Demonstrate the ability of team work, professional ethics, communication and documentation skills in designing and implementation of software products using the SDLC principles

(Autonomous Institution Affiliated to VTU, Belagavi)

Department of Information Science & Engineering

C E R T I F I C A T E

This is to certify that Mr./Ms……VISMITHA N…… USN ……1RV17IS058……. of 7th semester Information Science and Engineering has satisfactorily completed the course of experiments in practical of Data Science and Engineering (16IS72) during the academic year 2020-21.

LAB MARKS	
Max Marks	**Obtained**
50	

Signature of the Student **Signature of the Faculty In–Charge**

Signature of HOD

General Laboratory Instructions

PART – A **Program Execution**	
1.	Student has to do all the experiments in the collage lab only
2.	Students can solve the experiments by using Python or R tool.
	PART – B **Case Study implementation**
	Student shall finalize the topic in consultation with the teacher

General Guidelines

- The lab manual should contain duly signed data sheets with programs and results in the respective program slot given.
- Each program in PART A is evaluated for 10 marks (7 marks for execution + 3 marks for viva voce) and the average of 09 programs is considered for 30 marks.

Guidelines for Case Study

- Case Study is carried out in ISE lab with maximum of two members in a team.
- Complexity of the casestudy will be discussed with the faculty – In charge of the lab and

then allotment of the same would be confirmed.

- Students are required to strict to the schedule without fail.

Acknowledgement: Prof. Poornima K, ISE, RVCE

Evaluation distribution table for lab record

PART A	PART B	Internals	Total Marks
30	10	10	50

Scheme of Continuous Internal Evaluation for Practical

CIE consists of 50 marks out of which 30 marks for PART-A, 10 marks for PART-B (40) and 10 marks for test conducted at the end of semester.

Scheme of Semester End Examination for Practical

SEE is evaluated for 50 marks which include writing correct program, execution and viva voce.

1. In the examination, ONE program has to be asked from Part-A for a total of 50 marks.
2. The Case Study impleted under Part-B not to be evaluated for SEE

V COLLEGE OF ENGINEERING®, BENGALURU – 560059

(Autonomous Institution Affiliated to VTU, Belagavi)

Data Science and Engineering – 16IS72

Laboratory Evaluation

Week	Program No.	Program Evaluation	Date of Execution	Marks (10)	Signature of Staff
1	1	Process the Movie dataset and visualize the correlations			
2	2	Implement data preprocessing techniques			
3	3	Implement simple linear regression and multiple linear regression using relevant datasets for prediction.			
4	4	Implement k- nearest neighbour algorithm using relevant datasets.			
5	5	Implement decision tree algorithm for classification using relevant datasets.			
6	6	Implement Naïve bayes classification using relevant datasets.			
7	7	Implement support vector machine using relevant datasets.			
8	8	Implement Association rule process using Apriori algorithm using relevant datasets.			
9	9	Implement K- means clustering to classify the clusters in a given data set			
Total for program execution: /90 **CIE for program execution : /30**					
Case Study Evaluation					
Activity No.		**Activity**	**Date of Submission**	**Marks (10 marks)**	**Signature of Staff**
1		Case study topic finalization		-	

2	Finalization of Data Source and model planning based on empirical studies		**2**	
3	Building the model (Ensemble model)		2	
4	Evaluation of the model		2	
5	Submission of Report and other deliverables		2	
6				
Total for casestudy execution: /10 **CIE for casestudy execution : /10**				
Internal Conduction (10 marks) **1. Program execution -05 marks -** **(Writeup-1m ,Execution- 3m ,Viva- 1m) -** **2. Case Study Demonstration – 05 marks –**			**Final CIE marks –** **/50**	

Rubrics for Data Science and Engineering

Each program is evaluated for 10 marks – Write up & Execution = 7 Marks, Viva Voce = 3 Marks

Lab Write-up and Execution rubrics (Max: 7 marks)						
Sl no	**Criteria**	**Measuring methods**	**Excellent**	**Good**	**Poor**	**CO**
1	**Understanding of problem and requirements** **(2 Marks)**	Observations	Student exhibits thorough understanding of program requirements and applies the concepts of Data Science to develop a model **(2M)**	Student has sufficient understanding of program requirements and applies concepts of Data Science to develop a model **(1M)**	Student does not have clear understanding of program requirements and is unable to apply Data science concepts learnt for development. **(0M)**	CO 1
2	**Execution** **(3Marks)**	Observations	Student demonstrates the execution of the program with optimized code with all the necessary conditions **(3M)**	Student demonstrates the execution of the program without optimization of the code **(2M-1M)**	Student has not executed the program. **(0 M)**	CO 4
3	**Results and Documentation** **(2Marks)**	Observations	Documentation with appropriate comments and output is covered in data sheets and manual. **(2M)**	Documentation with only few comments and only few output cases is covered in data sheets and manual. **(1M)**	Documentation with no comments and no output cases is covered in data sheets and manual. **(0 M)**	CO 2
Viva Voce rubrics (Max: 3 marks)						
1	**Conceptual Understanding and Communication of concepts** **(2 Marks)**	Viva Voce	Explains Data Science concepts with Python/R and related concepts involved. **(2M)**	Adequately explains the Data Science concepts with Python/R and related concepts involved**.**	Unable to explain the concepts. **(0M)**	CO 1

				(1M)		
2	**Use of appropriate Design Techniques** **(1 Mark)**	Viva Voce	Insightful explanation of appropriate design techniques for the given problem to derive solution. **(1 M)**	Sufficiently explains the use of appropriate design techniques for the given problem to derive solution. **(0.5 M)**	Unable to explain the design techniques for the given problem. **(0 M)**	CO 3

Program No. 1

Title:

- **Process the Movie dataset and visualize the correlations using R**

Objective:

- To write a program to visualize the correlations in a dataset

Reference:

- Data Mining Concept and Technique By Han & Kamber

Theory:

Step 1: Importing The Data

In order to access the movies data set and put it to use, use the read.csv() function to import your data into a data frame and store it in the variable with the stunningly original name movies.

Step 2: Basic Inspection of Your Data

It's a good idea, once a data frame has been imported, to get an idea about your data. First, check out the structure of the data that is being examined. Use function str():

The console lists each variable by name, the class of each variable, and a few instances of each variable. This gives good idea of what is in the data frame, the understanding of which is crucial to our analytic endeavors.

Another great function to help perform a quick, high-level overview of our data frame is summary(). Note the similarities and differences between the output produced by running str().

Step 3:

In reviewing the variables available to you, it appears that some of the numeric variables can be manipulated to provide new insights into our data frame.

For instance, to get gross and budget variables. Calculate profit by using the formula profit = gross - budget.

Step 4: Correlation

Now that profit has been added as a new column in our data frame, it's time to take a closer look at the relationships between the variables of your data set. Let's check out how profit fluctuates

relative to each movie's rating.For this, you can use R's built in plot and abline functions, where plot will result in a scatter plot and abline will result in a regression line or "line of best fit" due to our inclusion of the linear model argument

Step 5: Calculating Correlation in R

To identify the types of relationships exist between the variables in movies, and how can you evaluate those relationships quantitatively?

The first way is to produce correlations and correlation matrices with cor(): Visually Exploring Correlation: The R Correlation Matrix Plot a correlation matrix using the variables available in your movies data frame. This simple plot will enable to quickly visualize which variables have a negative, positive, weak, or strong correlation to the other variables. To pull this wizardry off, use a nifty package called GGally and a function called ggcorr(). The form of this function call will be ggcorr(df), where df is the name of the data frame you're calling the function on. The output of this function will be a triangular-shaped, color-coded matrix labelled with our variable names. The correlation coefficient of each variable relative to the other variables can be found by reading across and / or down the matrix, depending on the variable's location in the matrix.

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
1							

Program No. 1

Paste your DATA SHEET here

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

Reading data

In [2]:
```
df=pd.read_csv("P1_movies_metadata.csv",dtype='unicode')
df.head()
```

Out[2]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt01147
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt01134
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt01132
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt01148
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt01130

5 rows × 24 columns

In [3]:
```
df.shape#size of the file in rows and columns
```

Out[3]:
```
(45466, 24)
```

Type of the attributes

In [4]:
```
df.dtypes
```

Out[4]:
```
adult                    object
belongs_to_collection    object
budget                   object
genres                   object
homepage                 object
id                       object
imdb_id                  object
original_language        object
original_title           object
overview                 object
popularity               object
poster_path              object
production_companies     object
production_countries     object
release_date             object
revenue                  object
runtime                  object
spoken_languages         object
status                   object
tagline                  object
title                    object
video                    object
vote_average             object
vote_count               object
dtype: object
```

In [5]:
```
for i in df.columns:
    print(i," ",type(df[i][0]))
```

```
adult   <class 'str'>
belongs_to_collection   <class 'str'>
budget   <class 'str'>
genres   <class 'str'>
homepage   <class 'str'>
id   <class 'str'>
imdb_id   <class 'str'>
original_language   <class 'str'>
original_title   <class 'str'>
overview   <class 'str'>
popularity   <class 'str'>
poster_path   <class 'str'>
production_companies   <class 'str'>
production_countries   <class 'str'>
release_date   <class 'str'>
revenue   <class 'str'>
runtime   <class 'str'>
spoken_languages   <class 'str'>
status   <class 'str'>
tagline   <class 'float'>
title   <class 'str'>
video   <class 'str'>
vote_average   <class 'str'>
vote_count   <class 'str'>
```

Attributes which are in json string format

In [6]:
```
df['belongs_to_collection'][0]
```

Out[6]:
```
"{'id': 10194, 'name': 'Toy Story Collection', 'poster_path': '/7G9915LfUQ2lV
fwMEEhDsn3kT4B.jpg', 'backdrop_path': '/9FBwqcd9IRruEDUrTdcaafOMKUq.jpg'}"
```

In [7]:
```
df['genres'][0]
```

Out[7]:
```
"[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 1075
1, 'name': 'Family'}]"
```

In [8]:
```
df['overview'][0]
```

Out[8]:
```
"Led by Woody, Andy's toys live happily in his room until Andy's birthday bri
ngs Buzz Lightyear onto the scene. Afraid of losing his place in Andy's hear
t, Woody plots against Buzz. But when circumstances separate Buzz and Woody f
rom their owner, the duo eventually learns to put aside their differences."
```

In [9]:
```
df['spoken_languages'][0]
```

Out[9]:
```
"[{'iso_639_1': 'en', 'name': 'English'}]"
```

In [10]:
```
df.describe()
```

Out[10]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb
count	45466	4494	45466	45466	7782	45466	45
unique	5	1698	1226	4069	7673	45436	45
top	False	{'id': 415931, 'name': 'The Bowery Boys', 'pos...	0	[{'id': 18, 'name': 'Drama'}]	http://www.georgecarlin.com	141971	tt1180
freq	45454	29	36573	5000	12	3	

4 rows × 24 columns

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   adult                  45466 non-null  object
 1   belongs_to_collection  4494 non-null   object
 2   budget                 45466 non-null  object
 3   genres                 45466 non-null  object
 4   homepage               7782 non-null   object
 5   id                     45466 non-null  object
 6   imdb_id                45449 non-null  object
 7   original_language      45455 non-null  object
 8   original_title         45466 non-null  object
 9   overview               44512 non-null  object
 10  popularity             45461 non-null  object
 11  poster_path            45080 non-null  object
 12  production_companies   45463 non-null  object
 13  production_countries   45463 non-null  object
 14  release_date           45379 non-null  object
 15  revenue                45460 non-null  object
 16  runtime                45203 non-null  object
 17  spoken_languages       45460 non-null  object
 18  status                 45379 non-null  object
 19  tagline                20412 non-null  object
 20  title                  45460 non-null  object
 21  video                  45460 non-null  object
 22  vote_average           45460 non-null  object
 23  vote_count             45460 non-null  object
dtypes: object(24)
memory usage: 8.3+ MB
```

In [12]:
```
df.columns
```

Out[12]:
```
Index(['adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'i
d',
       'imdb_id', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'video',
       'vote_average', 'vote_count'],
      dtype='object')
```

In [13]:
```
df.index
```

Out[13]:
```
RangeIndex(start=0, stop=45466, step=1)
```

Changing the formats of the data

In [14]:
```
df['budget']=pd.to_numeric(df['budget'],errors='coerce')
```

In [15]:
```
df['id']=pd.to_numeric(df['id'],errors='coerce')
```

In [16]:
```
df['revenue']=pd.to_numeric(df['revenue'],errors='coerce')
```

In [17]:
```
df['runtime']=pd.to_numeric(df['runtime'],errors='coerce')
```

In [18]:
```
df['vote_average']=pd.to_numeric(df['vote_average'],errors='coerce')
```

In [19]:
```
df['vote_count']=pd.to_numeric(df['vote_count'],errors='coerce')
```

In [20]:
```
df['actual_profit']=df['revenue']-df['budget']
```

In [21]:
```
df['profit_ratio']=df.apply(lambda row:row['revenue']/row['budget'] if row['bu
dget']!=0 else 0,axis=1)
```

In [22]:
```
df['actual_profit']
```

Out[22]:
```
0        343554033.0
1        197797249.0
2                0.0
3         65452156.0
4         76578911.0
             ...
45461            0.0
45462            0.0
45463            0.0
45464            0.0
45465            0.0
Name: actual_profit, Length: 45466, dtype: float64
```

Corelation matrix

In [23]:
```
cor=df.corr()
cor.style.background_gradient(cmap='coolwarm')
```

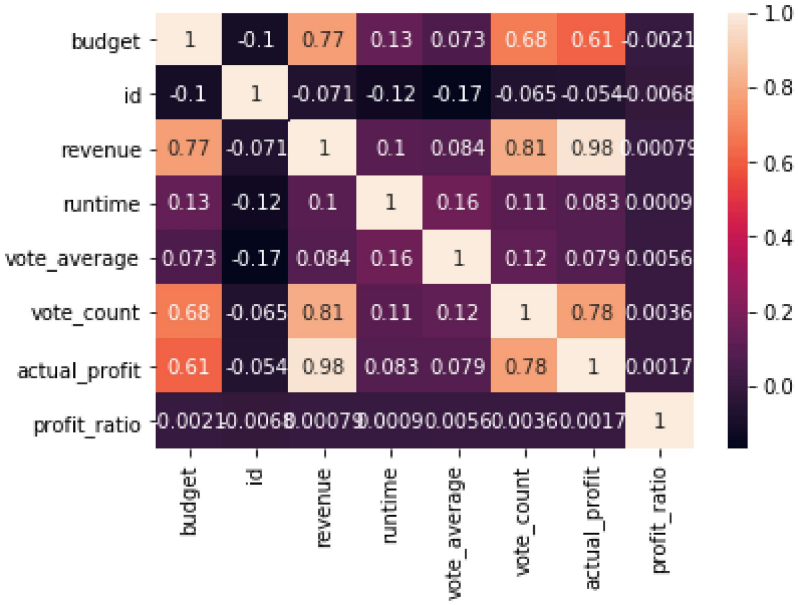
Out[23]:

	budget	id	revenue	runtime	vote_average	vote_count	actual_profit
budget	1.000000	-0.101671	0.768776	0.134733	0.073494	0.676642	0.614339
id	-0.101671	1.000000	-0.071263	-0.121399	-0.167573	-0.064903	-0.053950
revenue	0.768776	-0.071263	1.000000	0.103917	0.083868	0.812022	0.976896
runtime	0.134733	-0.121399	0.103917	1.000000	0.158146	0.113539	0.083189
vote_average	0.073494	-0.167573	0.083868	0.158146	1.000000	0.123607	0.078916
vote_count	0.676642	-0.064903	0.812022	0.113539	0.123607	1.000000	0.775756
actual_profit	0.614339	-0.053950	0.976896	0.083189	0.078916	0.775756	1.000000
profit_ratio	-0.002132	-0.006758	0.000794	0.000901	0.005585	0.003602	0.001692

In [24]:
```
#so actual_profit is more corelated than profiy_ratio
```

In [25]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x27337699e10>



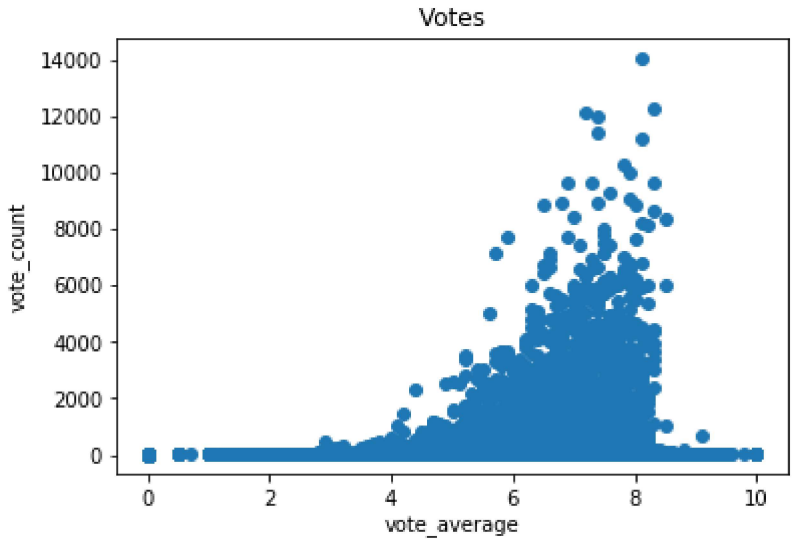
Data Visualization

In [26]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [27]:
```
x=np.array(df['vote_average'])
y=np.array(df['vote_count'])
```

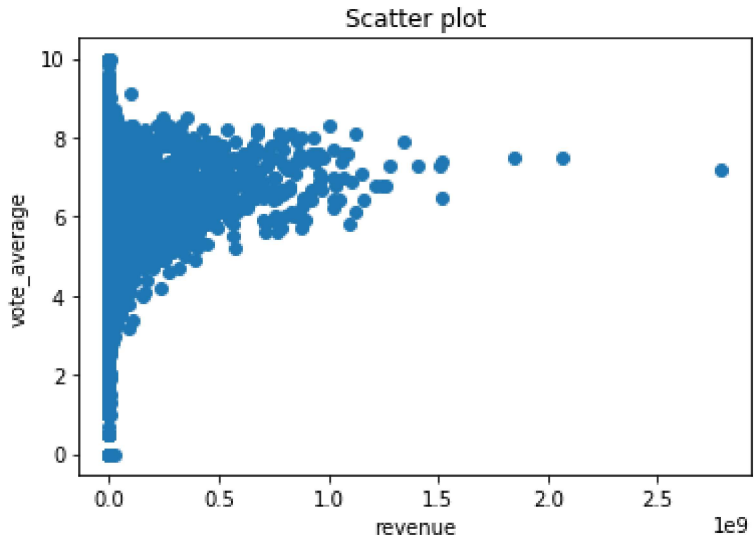
In [28]:
```
plt.xlabel('vote_average')
plt.ylabel('vote_count')
plt.title('Votes')
plt.scatter(x,y)
```

Out[28]: <matplotlib.collections.PathCollection at 0x2733783a7f0>



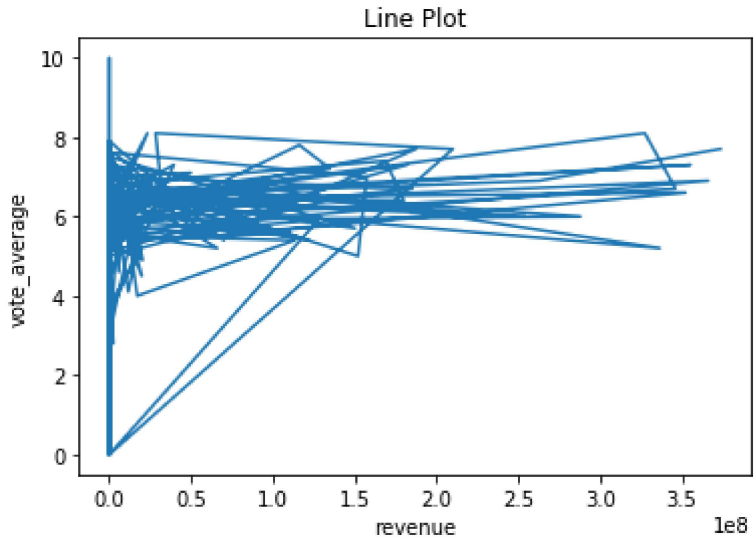
In [29]:
```
x1=np.array(df['revenue'])
y1=np.array(df['vote_average'])
plt.xlabel('revenue')
plt.ylabel('vote_average')
plt.title('Scatter plot')
plt.scatter(x1,y1)
```

Out[29]: <matplotlib.collections.PathCollection at 0x273378979b0>



In [36]:
```
x2=np.array(df['revenue'])[:200]
y2=np.array(df['vote_average'])[:200]
plt.xlabel('revenue')
plt.ylabel('vote_average')
plt.title('Line Plot')
plt.plot(x2,y2)
```

Out[36]: [<matplotlib.lines.Line2D at 0x27337a410b8>]



In [35]:
```
df.to_csv("res.csv")##writing clean data back
```

Program No. 2

Title:

Data Preprocessing Techniques for Data Mining

Objective:

To write a program to preprocess the data for building a model

Reference:

Data Mining Concept and Technique By Han & Kamber

Theory:

Data Pre-processing Methods

Raw data is highly susceptible to noise, missing values, and inconsistency. The quality of data affects the data mining results. In order to help improve the quality of the data and, consequently, of the mining results raw data is pre-processed so as to improve the efficiency and ease of the mining process. Data preprocessing is one of the most critical steps in a data mining process which deals with the preparation and transformation of the initial dataset. Data preprocessing methods are divided into following categories:

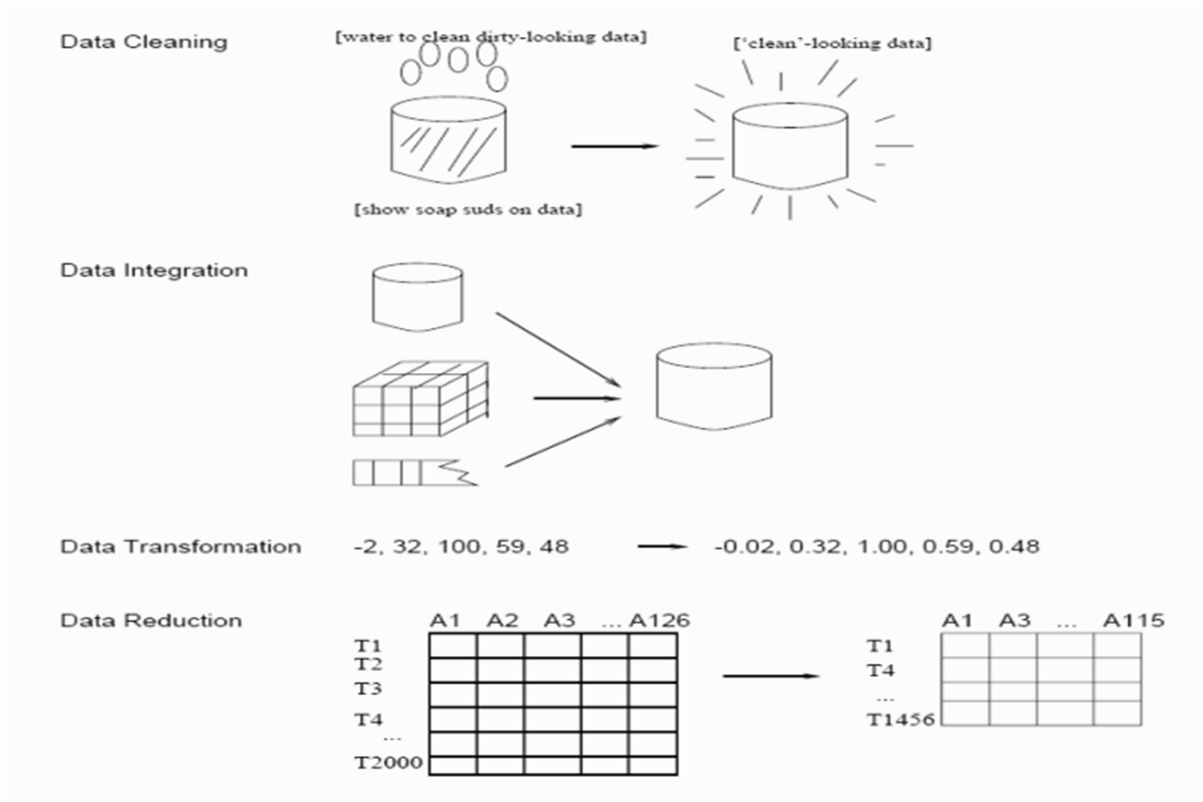
Data Cleaning

- Data Integration
- Data Transformation
- Data Reduction

Data Cleaning

Data that is to be analyze by data mining techniques can be incomplete (lacking attribute values or certain attributes of interest, or containing only aggregate data), noisy (containing errors, or outlier values which deviate from the expected), and inconsistent (e.g., containing discrepancies in the department codes used to categorize items).Incomplete, noisy, and inconsistent data are commonplace properties of large, real-world databases anddata warehouses. Incomplete data can occur for a number of reasons. Attributes of interest may not always be availab le, such as customer information for sales transaction data. Other data may not be included simply because it was not considered important at the time of entry. Relevant data may not be recorded due to a misunderstanding, or because of

equipment malfunctions. Data can be noisy, having incorrect attribute values, owing to the following. The data collection instruments used may be faulty. There may have been human or computer errors occurring at data entry. Errors in data transmission can also occur. There may be technology lim itations, such as limited buffer size for coordinating synchronized data transfer and consumption. Dirty data can cause confusion for the mining procedure. Although most mining routines have some procedures for dealing with incomplete or noisy data, they are not always robust. Instead, they may concentrate on avoiding over fitt ing the data to the function being modelled. Therefore, a useful preprocessing step is to run data through some data cleaning routines.



Forms of Data Preprocessing

Missing Values: If it is noted that there are many tuples that have no recorded value for several attributes, then the missing values can be filled in for the attribute by various methods described below:

1. Ignore the tuple: This is usually done when the class label is missing (assuming the mining task involves c lass if ication or description). This method is not very

effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.

2. Fill in the missing value manually: In general, this approach is time-consuming and

may not be feasible given a large data set with many missing values.

3. Values are replaced by, say, Unknown", then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common that of Unknown". Hence, although this method is simple, it is not recommended.

4. Use the attribute mean to fi ll in the missing value

5. Use the attribute mean for all samples belonging to the same class as the given tuple.

6. Use the most probable value to fill in the missing value: This may be determined with inference-based tools using a Bayesian formalism or decision tree induction.

Methods 3 to 6 bias the data. The filled-in value may not be correct. Method 6, however, is a popular strategy. In comparison to the other methods, it uses the most information from the present data to predict missing values.

Noisy Data: Noise is a random error or variance in a measured variable. Given a numeric attribute such as, say, price, how can the data be “smoothed" to remove the noise? The following data smoothing techniques describes this.

1. Binning methods: Binning methods smooth a sorted data value by consulting the neighborhood", or values around it. The sorted values are distributed into a number of 'buckets', or bins. Because binning methods consult the neighborhood of values, they perform local smoothing values around it. The sorted values are distributed

into a number of 'buckets', or bins. Because binning methods consult the neighborhood of values, they perform local smoothing.

2. Clustering: Outliers may be detected by clustering, where similar values are organized into groups or \clusters".

3. Combined computer and human inspection: Outliers may be identified through a

combination of computer and human inspection. In one application, for example, an information-theoretic measure was used to help identify outlier patterns in a handwritten character database for classification. The measure's value reflected the surprise" content of the predicted character label with respect to the known label. Outlier patterns may be informative (e.g., identifying useful data exceptions, such as different versions of the characters \0" or \7"), or \garbage" (e.g., mislabeled characters). Patterns whose surprise content is above a threshold are output to a list. A human can then sort through the patterns in the list to identify the actual garbage ones.

This is much faster than having to manually search through the entire database. The garbage patterns can then be removed from the (training) database.

4. Regression: Data can be smoothed by fitting the data to a function, such as with regression. Linear regression involves finding the \best" line to fit two variables, so that one variable can be used to predict the other. Multiple linear regression is an extension of linear regression, where more than two variables are involved and the data are fit to a multidimensional surface. Using regression to find a mathematical equation to fit the data helps smooth out the noise.

Inconsistent data: There may be inconsistencies in the data recorded for some transactions. Some data inconsistencies may be corrected manually using external references. For example, errors made at data entry may be corrected by performing a paper trace. This may be coupled with routines designed to help correct the inconsistent use of codes. Knowledge engineering tools may also be used to detect the violation of known data constraints. For example, known functional dependencies between attributes can be used to find values contradicting the functional constraints.

Data Integration

It is likely that your data analysis task will involve data integration, which combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files. There are a number of issues to consider during data integration. Schema integration can be tricky. How can like rea l world entities from multiple data sources be 'matched up'? This is referred to as the entity identification problem. For example, how can

the data analyst or the computer be sure that customer id in one database, and cust_number in another refer to the same entity? Databases and data warehouses typically have metadata - that is, data about the data. Such metadata can be used to help avoid errors in schema integration. Redundancy is another important issue. An attribute may be redundant if it can be \derived" from another table, such as annual revenue. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.

Data Trans formation

In data transformation, the data are transformed or consolidated into forms appropriate for mining. Data transformation can involve the following:

1. Normalization, where the attribute data are scaled so as to fall within a small specified
range, such as -1.0 to 1.0, or 0 to 1.0.
2. Smoothing works to remove the noise from data. Such techniques include binning, clustering, and regression.
3. Aggregation, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.
4. Generalization of the data, where low level or 'primitive' (raw) data are replaced by higher level concepts through the use of concept hierarchies. For example, categorical attributes, like street, can be generalized to higher level concepts, like city or county. Similarly, values for numeric attributes, like age, may be mapped to higher level concepts, like young, middle-aged, and senior.

Data Reduction

Complex data analysis and mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible. Data reduction techniques have been helpful in analyzing reduced representation of the dataset without compromising the integrity of the original data and yet producing the quality knowledge. The concept of data reduction is commonly understood as either reducing the volume or reducing the dimensions (number of attributes). There are

a number of methods that have facilitated in analyzing a reduced volume or dimension of data and yet yield useful knowledge. Certain partition based methods work on partition of data tuples.That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results

Strategies for data reduction include the following.

1. Data cube aggregation, where aggregation operations are applied to the data in the construction of a data cube.

2. Dimension reduction, where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.

3. Data compression, where encoding mechanisms are used to reduce the data set size. The methods used for data compression are wavelet transform and Principal Component Analysis.

4. Numerosity reduction, where the data are replaced or estimated by alternative, smaller data representations such as parametric models (which need store only the model parameters instead of the actual data e.g. regression and log-linear models), or nonparametric methods such as clustering, sampling, and the use of histograms.

5.Discretization and concept hierarchy generation, where raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction, and are a powerful tool for data mining.

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
2							

Program No. 2

Paste your DATA SHEET here

In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import os
```

In [4]:
```
os.stat(r'C:\Users\Admin\Documents\7th sem\ds\datasets\P2_mov.csv')
```

Out[4]:
```
os.stat_result(st_mode=33206, st_ino=4785074604091624, st_dev=1455008834, st_
nlink=1, st_uid=0, st_gid=0, st_size=5698602, st_atime=1604458970, st_mtime=1
604458970, st_ctime=1571231928)
```

In [3]:
```
df=pd.read_csv(r"C:\Users\Admin\Documents\7th sem\ds\datasets\P2_mov.csv")
```

In [5]:
```
df.shape
```

Out[5]:
```
(4803, 20)
```

In [ ]:
```

```

In [6]:
```
df.head()
```

Out[6]:

	budget	genres	homepage	id	keywords	original_
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4803 non-null   int64  
 1   genres                4803 non-null   object 
 2   homepage              1712 non-null   object 
 3   id                    4803 non-null   int64  
 4   keywords              4803 non-null   object 
 5   original_language     4803 non-null   object 
 6   original_title        4803 non-null   object 
 7   overview              4800 non-null   object 
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object 
 10  production_countries  4803 non-null   object 
 11  release_date          4802 non-null   object 
 12  revenue               4803 non-null   int64  
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object 
 15  status                4803 non-null   object 
 16  tagline               3959 non-null   object 
 17  title                 4803 non-null   object 
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64  
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

In [8]:
```
for i in df.columns:
    print(i," ",type(df[i][0]))
```

```
budget   <class 'numpy.int64'>
genres   <class 'str'>
homepage   <class 'str'>
id   <class 'numpy.int64'>
keywords   <class 'str'>
original_language   <class 'str'>
original_title   <class 'str'>
overview   <class 'str'>
popularity   <class 'numpy.float64'>
production_companies   <class 'str'>
production_countries   <class 'str'>
release_date   <class 'str'>
revenue   <class 'numpy.int64'>
runtime   <class 'numpy.float64'>
spoken_languages   <class 'str'>
status   <class 'str'>
tagline   <class 'str'>
title   <class 'str'>
vote_average   <class 'numpy.float64'>
vote_count   <class 'numpy.int64'>
```

In [9]:
```
df.describe()
```

Out[9]:

	budget	id	popularity	revenue	runtime	vote_average	vc
count	4.803000e+03	4803.000000	4803.000000	4.803000e+03	4801.000000	4803.000000	48(
mean	2.904504e+07	57165.484281	21.492301	8.226064e+07	106.875859	6.092172	6(
std	4.072239e+07	88694.614033	31.816650	1.628571e+08	22.611935	1.194612	12:
min	0.000000e+00	5.000000	0.000000	0.000000e+00	0.000000	0.000000	
25%	7.900000e+05	9014.500000	4.668070	0.000000e+00	94.000000	5.600000	5
50%	1.500000e+07	14629.000000	12.921594	1.917000e+07	103.000000	6.200000	2:
75%	4.000000e+07	58610.500000	28.313505	9.291719e+07	118.000000	6.800000	7:
max	3.800000e+08	459488.000000	875.581305	2.787965e+09	338.000000	10.000000	137(

In [10]:
```
df.columns
```

Out[10]:
```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count'],
      dtype='object')
```

In [11]:
```
df.index
```

Out[11]:
```
RangeIndex(start=0, stop=4803, step=1)
```

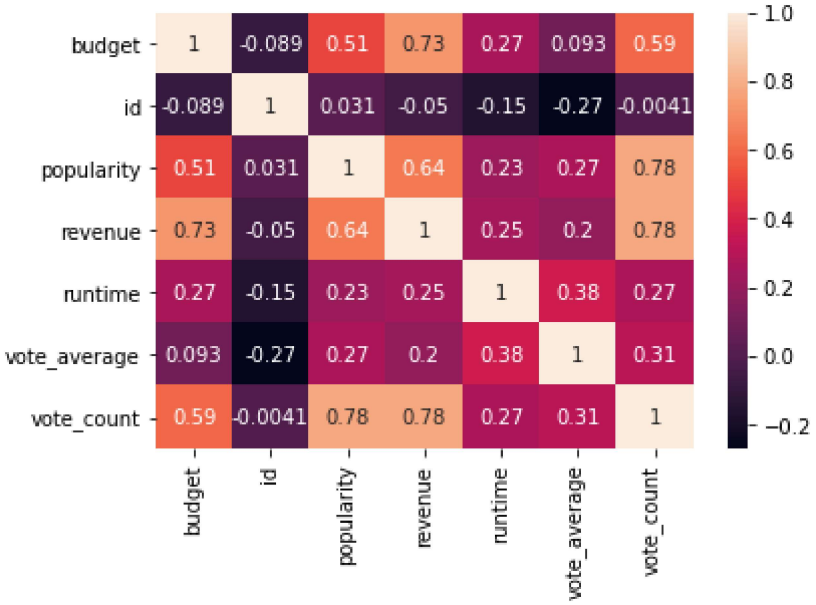
In [12]:
```
cor=df.corr()
cor.style.background_gradient(cmap='coolwarm')
```

Out[12]:

	budget	id	popularity	revenue	runtime	vote_average	vote_count
budget	1.000000	-0.089377	0.505414	0.730823	0.269851	0.093146	0.593180
id	-0.089377	1.000000	0.031202	-0.050425	-0.153536	-0.270595	-0.004128
popularity	0.505414	0.031202	1.000000	0.644724	0.225502	0.273952	0.778130
revenue	0.730823	-0.050425	0.644724	1.000000	0.251093	0.197150	0.781487
runtime	0.269851	-0.153536	0.225502	0.251093	1.000000	0.375046	0.271944
vote_average	0.093146	-0.270595	0.273952	0.197150	0.375046	1.000000	0.312997
vote_count	0.593180	-0.004128	0.778130	0.781487	0.271944	0.312997	1.000000

In [13]:
```
sns.heatmap(df.corr(),annot=True)
```

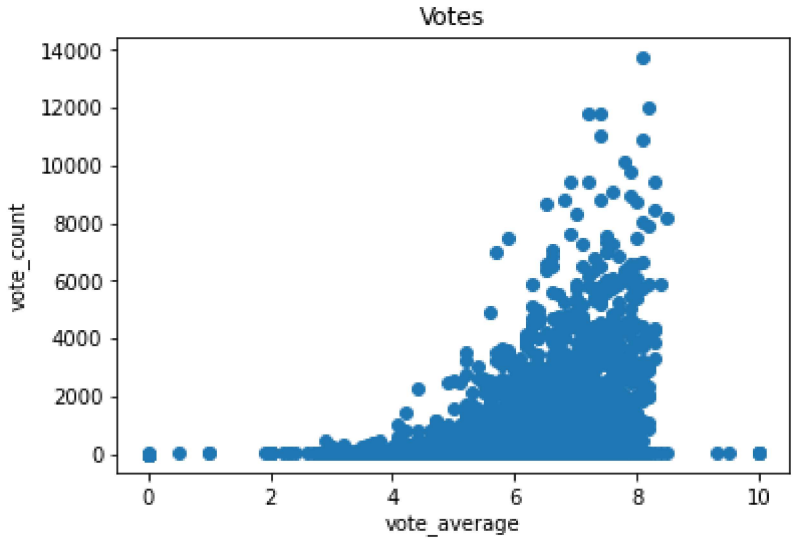
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1cd0ab785b0>



In [14]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

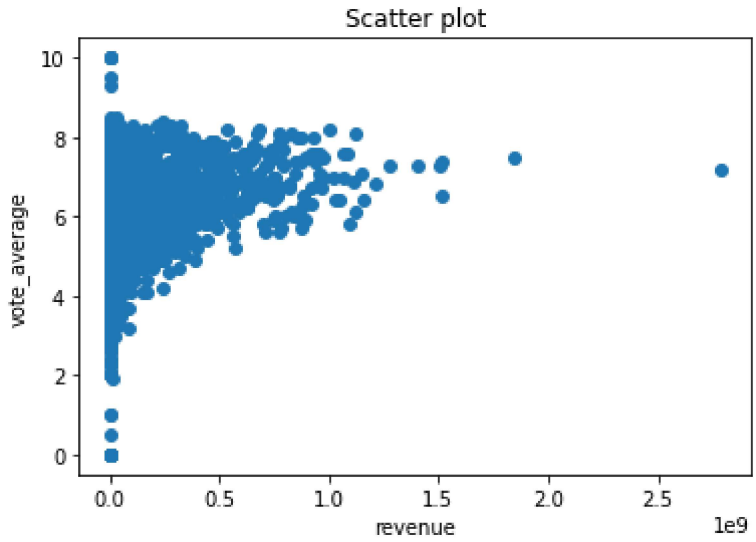
In [15]:
```
x=np.array(df['vote_average'])
y=np.array(df['vote_count'])
plt.xlabel('vote_average')
plt.ylabel('vote_count')
plt.title('Votes')
plt.scatter(x,y)
```

Out[15]: <matplotlib.collections.PathCollection at 0x1cd0b5ea430>



In [16]:
```
x1=np.array(df['revenue'])
y1=np.array(df['vote_average'])
plt.xlabel('revenue')
plt.ylabel('vote_average')
plt.title('Scatter plot')
plt.scatter(x1,y1)
```

Out[16]: <matplotlib.collections.PathCollection at 0x1cd0b6495b0>



DATA Cleaning

Handling missing values

In [121]:
```
for col in df.columns:
    print(col,"  ",df[col].isna().sum())
```

```
budget    0
genres    0
homepage    3091
id    0
keywords    0
original_language    0
original_title    0
overview    3
popularity    0
production_companies    0
production_countries    0
release_date    1
revenue    0
runtime    2
spoken_languages    0
status    0
tagline    844
title    0
vote_average    0
vote_count    0
```

In [122]:
```
df['runtime'].fillna(df['runtime'].mean(),inplace=True)
```

In [123]:
```
df['homepage'].fillna('source_unknown',inplace=True)
```

In [124]:
```
for c in ['homepage','runtime']:
    print(c,"  ",df[col].isna().sum())
```

```
homepage    0
runtime    0
```

Data Integration

In [125]:
```
df1=pd.read_csv("res.csv")
```

```
C:\Users\Sonal\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2
785: DtypeWarning: Columns (11) have mixed types.Specify dtype option on impo
rt or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [126]:
```
df1.head()
```

Out[126]:

	Unnamed: 0	adult	belongs_to_collection	budget	genres	homepage
0	0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000.0	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story
1	1	False	NaN	65000000.0	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN
2	2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0.0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN
3	3	False	NaN	16000000.0	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN
4	4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0.0	[{'id': 35, 'name': 'Comedy'}]	NaN

5 rows × 27 columns

In [127]:
```
df_new=pd.merge(df,df1,on="original_title")
```

In [128]:
```
df_new.head()
```

Out[128]:

	budget_x	genres_x	homepage_x	id_x	keywords	original_
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	

5 rows × 46 columns

In [129]:
```
df=df_new
```

In [130]:
```
df.columns
```

Out[130]:
```
Index(['budget_x', 'genres_x', 'homepage_x', 'id_x', 'keywords',
       'original_language_x', 'original_title', 'overview_x', 'popularity_x',
       'production_companies_x', 'production_countries_x', 'release_date_x',
       'revenue_x', 'runtime_x', 'spoken_languages_x', 'status_x', 'tagline_
x',
       'title_x', 'vote_average_x', 'vote_count_x', 'Unnamed: 0', 'adult',
       'belongs_to_collection', 'budget_y', 'genres_y', 'homepage_y', 'id_y',
       'imdb_id', 'original_language_y', 'overview_y', 'popularity_y',
       'poster_path', 'production_companies_y', 'production_countries_y',
       'release_date_y', 'revenue_y', 'runtime_y', 'spoken_languages_y',
       'status_y', 'tagline_y', 'title_y', 'video', 'vote_average_y',
       'vote_count_y', 'actual_profit', 'profit_ratio'],
      dtype='object')
```

Data Transformation

Normalization

In [131]:
```
print(df["runtime_y"].min(),df["runtime_y"].max())
```

```
0.0 877.0
```

In [132]:
```
df["runtime_y"]
```

Out[132]:
```
0       162.0
1       169.0
2       148.0
3       165.0
4       132.0
        ...
5235     81.0
5236     85.0
5237    120.0
5238     98.0
5239     90.0
Name: runtime_y, Length: 5240, dtype: float64
```

In [133]:
```
df["runtime_y"]=((df["runtime_y"]-df["runtime_y"].min())/(df["runtime_y"].max
()-df["runtime_y"].min()))*10
```

In [134]:
```
print(df["runtime_y"].min(),df["runtime_y"].max())
```

```
0.0 10.0
```

In [135]:
```
print(df["budget_x"].min(),df["budget_x"].max())
```

```
0 380000000
```

In [136]:
```
df['budget_x']=((df['budget_x']-df['budget_x'].min())/(df['budget_x'].max()-df
['budget_x'].min()))*100
```

In [137]:
```
df['budget_x']
```

Out[137]:
```
0       62.368421
1       78.947368
2       64.473684
3       65.789474
4       68.421053
           ...
5235     0.057895
5236     0.002368
5237     0.000000
5238     0.000000
5239     0.000000
Name: budget_x, Length: 5240, dtype: float64
```

Adding extra column

In [138]:
```
df['profit']=df.apply(lambda row:row['revenue_x']-row['budget_x'] ,axis=1)
```

Changing data types

In [139]:
```
df['release_date_x'] = pd.to_datetime(df['release_date_x'], format = '%Y-%m-%d
', errors='coerce')
type(df['release_date_x'][0])
```

Out[139]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

In [140]:
```
df['release_date_y'] = pd.to_datetime(df['release_date_y'], format = '%Y-%m-%d
', errors='coerce')
type(df['release_date_y'][0])
```

Out[140]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

Data Reduction

In [141]:
```
df.columns
```

Out[141]:
```
Index(['budget_x', 'genres_x', 'homepage_x', 'id_x', 'keywords',
       'original_language_x', 'original_title', 'overview_x', 'popularity_x',
       'production_companies_x', 'production_countries_x', 'release_date_x',
       'revenue_x', 'runtime_x', 'spoken_languages_x', 'status_x', 'tagline_
x',
       'title_x', 'vote_average_x', 'vote_count_x', 'Unnamed: 0', 'adult',
       'belongs_to_collection', 'budget_y', 'genres_y', 'homepage_y', 'id_y',
       'imdb_id', 'original_language_y', 'overview_y', 'popularity_y',
       'poster_path', 'production_companies_y', 'production_countries_y',
       'release_date_y', 'revenue_y', 'runtime_y', 'spoken_languages_y',
       'status_y', 'tagline_y', 'title_y', 'video', 'vote_average_y',
       'vote_count_y', 'actual_profit', 'profit_ratio', 'profit'],
      dtype='object')
```

In [142]:
```
df.shape[1]
```

Out[142]:
```
47
```

In [143]:
```
df.head(2)
```

Out[143]:

	budget_x	genres_x	homepage_x	id_x	keywords	original_lang
0	62.368421	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	78.947368	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	

2 rows × 47 columns

In [144]:
```
df.corr()
```

Out[144]:

	budget_x	id_x	popularity_x	revenue_x	runtime_x	vote_average_x	vote_
budget_x	1.000000	-0.048746	0.505823	0.730486	0.235954	0.050369	0
id_x	-0.048746	1.000000	0.095342	-0.017321	-0.091962	-0.132884	0
popularity_x	0.505823	0.095342	1.000000	0.644270	0.192562	0.284232	0
revenue_x	0.730486	-0.017321	0.644270	1.000000	0.228948	0.200137	0
runtime_x	0.235954	-0.091962	0.192562	0.228948	1.000000	0.361488	0
vote_average_x	0.050369	-0.132884	0.284232	0.200137	0.361488	1.000000	0
vote_count_x	0.592242	0.042433	0.782670	0.783269	0.243595	0.336574	1
Unnamed: 0	0.042138	0.658779	0.107172	0.041964	-0.073098	-0.115316	0
budget_y	0.865506	-0.063301	0.446036	0.621478	0.211275	0.038852	0
id_y	-0.052737	0.852166	0.077118	-0.025621	-0.073714	-0.113076	0
revenue_y	0.642080	-0.029752	0.591254	0.876451	0.206834	0.179955	0
runtime_y	0.188313	-0.094368	0.151543	0.172748	0.697825	0.276005	0
vote_average_y	0.032470	-0.114801	0.212568	0.143477	0.238398	0.755853	0
vote_count_y	0.514372	0.022486	0.714547	0.679250	0.222546	0.310873	0
actual_profit	0.508824	-0.016557	0.574773	0.864039	0.184349	0.204269	0
profit_ratio	-0.011361	-0.007803	0.001608	-0.007137	-0.013921	0.024994	0
profit	0.730486	-0.017321	0.644270	1.000000	0.228948	0.200137	0

In [145]:
```
df.drop(['id_x','popularity_x','runtime_x','vote_average_x','vote_count_x','actual_profit','Unnamed: 0',
         'budget_y','id_y','revenue_y','vote_average_y','runtime_y','production_countries_x',
         'spoken_languages_x','spoken_languages_y','budget_y','original_language_y','release_date_y'],axis=1)
```

Out[145]:

	budget_x	genres_x	homepage_x	keywords
0	62.368421	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	[{"id": 1463, "name": "culture clash"}, {"id":...
1	78.947368	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	[{"id": 270, "name": "ocean"}, {"id": 726, "na...
2	64.473684	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	[{"id": 470, "name": "spy"}, {"id": 818, "name...
3	65.789474	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	[{"id": 849, "name": "dc comics"}, {"id": 853,...
4	68.421053	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	[{"id": 818, "name": "based on novel"}, {"id":...
...	...	...	...	...
5235	0.057895	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	source_unknown	[{"id": 5616, "name": "united states\u2013mexi...
5236	0.002368	[{"id": 35, "name": "Comedy"}, {"id": 10749, "...	source_unknown	[]
5237	0.000000	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	http://www.hallmarkchannel.com/signedsealeddel...	[{"id": 248, "name": "date"}, {"id": 699, "nam...
5238	0.000000	[]	http://shanghaicalling.com/	[]
5239	0.000000	[{"id": 99, "name": "Documentary"}]	source_unknown	[{"id": 1523, "name": "obsession"}, {"id": 224...

5240 rows × 30 columns

In [110]:
```
df.columns
```

Out[110]:
```
Index(['budget_x', 'genres_x', 'homepage_x', 'id_x', 'keywords',
       'original_language_x', 'original_title', 'overview_x', 'popularity_x',
       'production_companies_x', 'production_countries_x', 'release_date_x',
       'revenue_x', 'runtime_x', 'spoken_languages_x', 'status_x', 'tagline_
x',
       'title_x', 'vote_average_x', 'vote_count_x', 'Unnamed: 0', 'adult',
       'belongs_to_collection', 'budget_y', 'genres_y', 'homepage_y', 'id_y',
       'imdb_id', 'original_language_y', 'overview_y', 'popularity_y',
       'poster_path', 'production_companies_y', 'production_countries_y',
       'release_date_y', 'revenue_y', 'runtime_y', 'spoken_languages_y',
       'status_y', 'tagline_y', 'title_y', 'video', 'vote_average_y',
       'vote_count_y', 'actual_profit', 'profit_ratio'],
      dtype='object')
```

In [111]:
```
df.shape[1]
```

Out[111]:
```
46
```

Data Discretization

In [146]:
```
df['budget_x'].tail()
```

Out[146]:
```
5235    0.057895
5236    0.002368
5237    0.000000
5238    0.000000
5239    0.000000
Name: budget_x, dtype: float64
```

In [147]:
```
df['budget_range'] = pd.cut(x=df['budget_x'], bins=[-1,10,1000,10000,100000000
000], labels=['No budget','Low', 'Medium', 'High'])
```

In [148]:
```
df['budget_range'].tail()
```

Out[148]:
```
5235    No budget
5236    No budget
5237    No budget
5238    No budget
5239    No budget
Name: budget_range, dtype: category
Categories (4, object): [No budget < Low < Medium < High]
```

In [ ]:
```

```

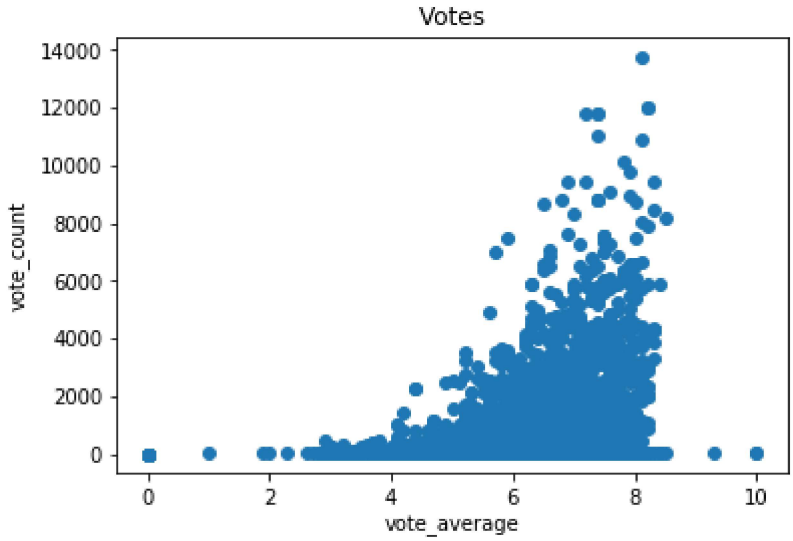
Data Visualization

In [103]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [107]:
```
x=np.array(df['vote_average_x'])
y=np.array(df['vote_count_x'])
```

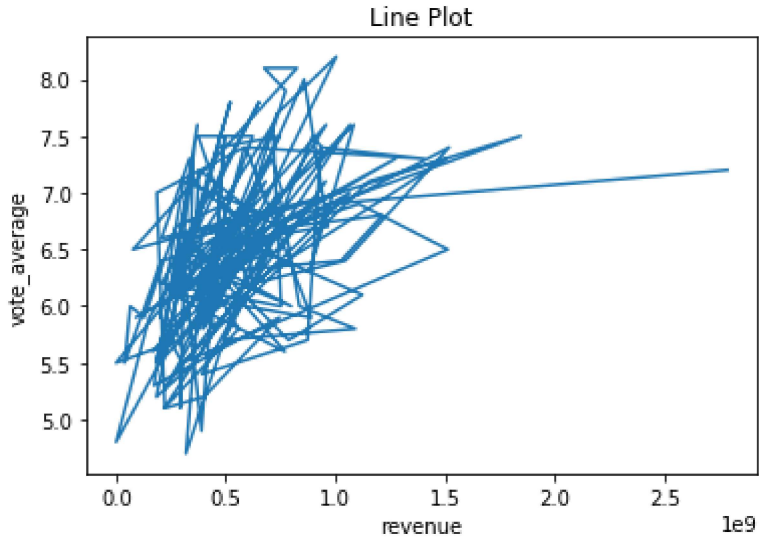
In [108]:
```
plt.xlabel('vote_average')
plt.ylabel('vote_count')
plt.title('Votes')
plt.scatter(x,y)
```

Out[108]:
```
<matplotlib.collections.PathCollection at 0x26ca020f710>
```



In [109]:
```
x2=np.array(df['revenue_x'])[:200]
y2=np.array(df['vote_average_x'])[:200]
plt.xlabel('revenue')
plt.ylabel('vote_average')
plt.title('Line Plot')
plt.plot(x2,y2)
```

Out[109]: [<matplotlib.lines.Line2D at 0x26ca01f4f60>]



In [ ]:

In [ ]:

Program No. 3

Title:

Linear Regression

Objective:

To write a program to classify the tuples using linear regression.

Reference:

Data Mining Introductory & Advanced Topic by Margaret H. Dunham

Data Mining Concept and Technique By Han & Kamber

Pre-requisite:

Knowledge of regression techniques

Theory:

Regression problem deals with estimation of output values based on input values. In the method we estimate the formula of straight line, which partitions data into 2 classes, by defining the regression coefficient C, the relation between output parameter Y and input parameter X1, X2, X3 ….. Xn can be estimated

Input : Training data set

Name	Gender	Height
Christina	F	1.6m
Jim	M	1.9m
Maggie	F	1.9m
Martha	F	1.88m
Stephony	F	1.7m
Bob	M	1.85m
Dave	M	1.7m
Steven	M	2.1m
Amey	F	1.8m

Output

The tuple is being classified using linear regression technique. Having value > 0.5 is classified as medium else < 0.5 then tuple is classified as short.

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
3							

Program No. 3

Paste your DATA SHEET here

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import numpy as np
```

In [2]:
```
df=pd.read_csv(r"C:\Users\Admin\Documents\7th sem\ds\datasets\P3_Weather.csv")
```

```
C:\Users\Admin\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3
071: DtypeWarning: Columns (7,8,18,25) have mixed types.Specify dtype option
on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [3]:
```
df.head()
```

Out[3]:

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	PoorWeather
0	10001	1942-7-1	1.016	NaN	25.555556	22.222222	23.888889	0	NaN
1	10001	1942-7-2	0	NaN	28.888889	21.666667	25.555556	0	NaN
2	10001	1942-7-3	2.54	NaN	26.111111	22.222222	24.444444	0	NaN
3	10001	1942-7-4	2.54	NaN	26.666667	22.222222	24.444444	0	NaN
4	10001	1942-7-5	0	NaN	26.666667	21.666667	24.444444	0	NaN

5 rows × 31 columns

In [4]:
```
df.dtypes
```

Out[4]:
```
STA              int64
Date            object
Precip          object
WindGustSpd    float64
MaxTemp        float64
MinTemp        float64
MeanTemp       float64
Snowfall        object
PoorWeather     object
YR               int64
MO               int64
DA               int64
PRCP            object
DR             float64
SPD            float64
MAX            float64
MIN            float64
MEA            float64
SNF             object
SND            float64
FT             float64
FB             float64
FTI            float64
ITH            float64
PGT            float64
TSHDSBRSGF      object
SD3            float64
RHX            float64
RHN            float64
RVG            float64
WTE            float64
dtype: object
```

In [5]:
```
for c in df.columns:
    print(c,'--',df[c].isna().sum())
```

```
STA -- 0
Date -- 0
Precip -- 0
WindGustSpd -- 118508
MaxTemp -- 0
MinTemp -- 0
MeanTemp -- 0
Snowfall -- 1163
PoorWeather -- 84803
YR -- 0
MO -- 0
DA -- 0
PRCP -- 1932
DR -- 118507
SPD -- 118508
MAX -- 474
MIN -- 468
MEA -- 498
SNF -- 1163
SND -- 113477
FT -- 119040
FB -- 119040
FTI -- 119040
ITH -- 119040
PGT -- 118515
TSHDSBRSGF -- 84803
SD3 -- 119040
RHX -- 119040
RHN -- 119040
RVG -- 119040
WTE -- 119040
```

In [6]:
```
df.shape
```

Out[6]:
```
(119040, 31)
```

In [7]:
```
df= df.drop(['WindGustSpd', 'PoorWeather', 'DR', 'SPD', 'SND','FT','FB','FTI',
'ITH','PGT','TSHDSBRSGF',
                         'SD3','RHX','RHN','RVG','WTE'],axis=1)
```

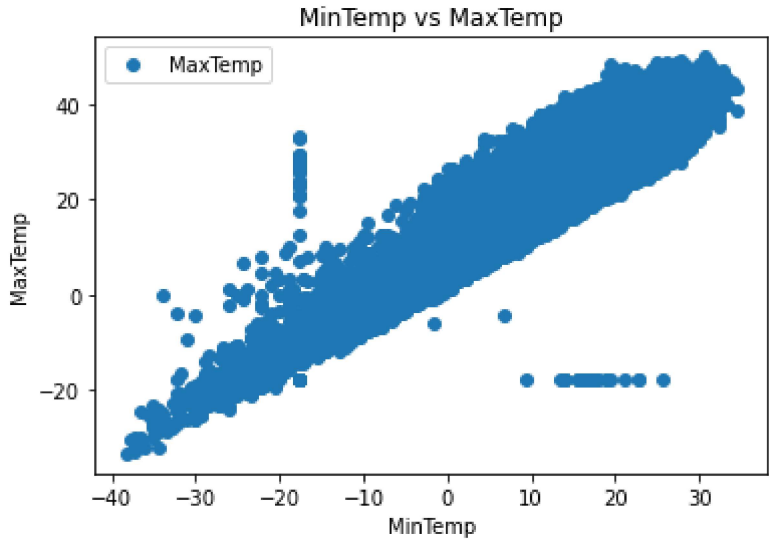
In [8]:
```
df.shape
```

Out[8]:
```
(119040, 15)
```

In [9]:
```
for c in df.columns:
    print(c,'--',df[c].isna().sum())
```

```
STA -- 0
Date -- 0
Precip -- 0
MaxTemp -- 0
MinTemp -- 0
MeanTemp -- 0
Snowfall -- 1163
YR -- 0
MO -- 0
DA -- 0
PRCP -- 1932
MAX -- 474
MIN -- 468
MEA -- 498
SNF -- 1163
```

In [10]:
```
import matplotlib.pyplot as plt
df.plot(x='MinTemp', y='MaxTemp', style='o')
plt.title('MinTemp vs MaxTemp')
plt.xlabel('MinTemp')
plt.ylabel('MaxTemp')
plt.show()
```



Simple Linear Regression

In [11]:
```
X = df['MinTemp'].values.reshape(-1,1)
y = df['MaxTemp'].values.reshape(-1,1)
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=0)
```

In [13]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
print(lr.coef_)
```

```
[10.66185201]
[[0.92033997]]
```

In [15]:
```
#Equation will be max_temp=0.92033997* min_temp + 10.66185201
#y=mx+c
```

In [16]:
```
y_pred=lr.predict(X_test)
```

In [17]:
```
y_test,y_pred
```

Out[17]:
```
(array([[28.88888889],
        [31.11111111],
        [27.22222222],
        ...,
        [31.11111111],
        [31.11111111],
        [36.66666667]]),
 array([[33.67035117],
        [30.0912513 ],
        [26.51215143],
        ...,
        [32.64775121],
        [30.60255128],
        [31.62515124]]))
```

In [18]:
```
lr.predict([[20]])
```

Out[18]:
```
array([[29.06865134]])
```

In [19]:
```
m_df=pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
```

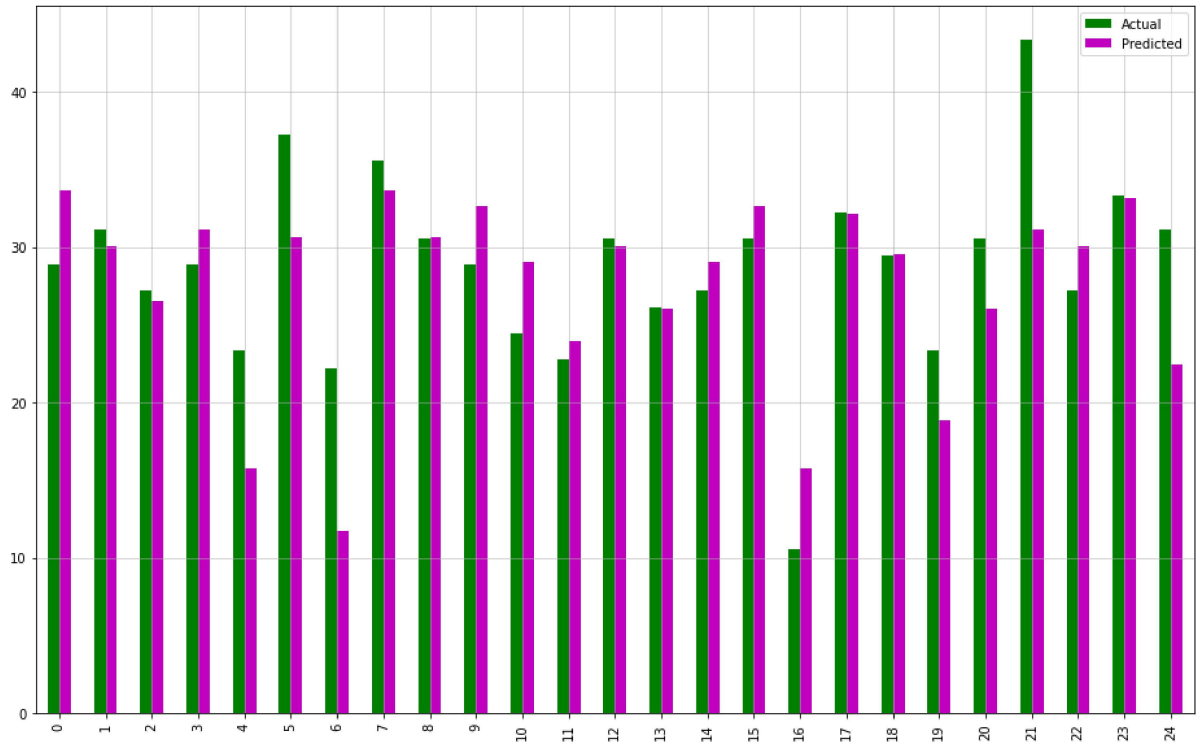
In [20]:
```
m_df
```

Out[20]:

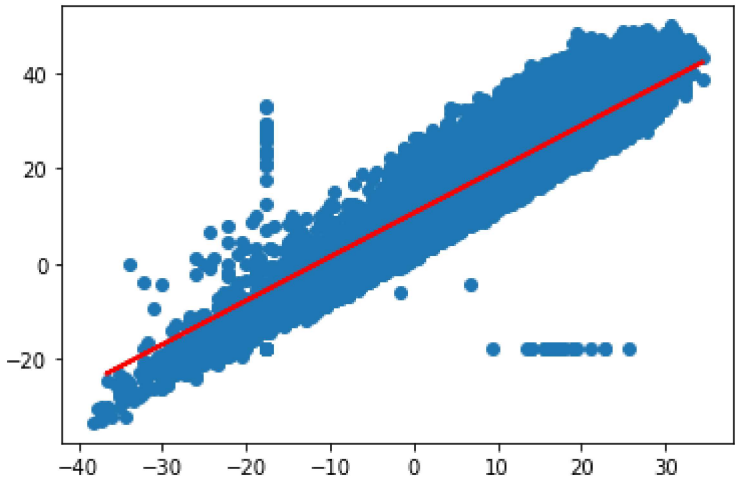
	Actual	Predicted
0	28.888889	33.670351
1	31.111111	30.091251
2	27.222222	26.512151
3	28.888889	31.113851
4	23.333333	15.774852
...	...	...
23803	32.777778	32.136451
23804	32.222222	29.068651
23805	31.111111	32.647751
23806	31.111111	30.602551
23807	36.666667	31.625151

23808 rows × 2 columns

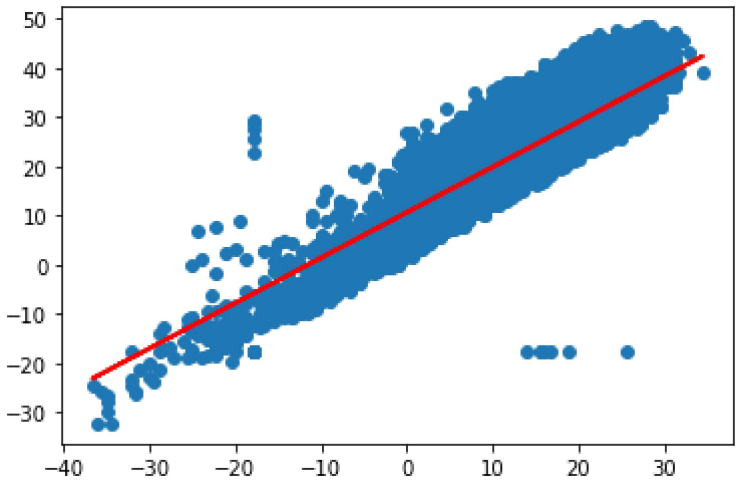
In [21]:
```
m_df.head(25).plot(kind='bar',color='gm',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5')
plt.show()
```



In [22]:
```
plt.scatter(X, y)
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()
```



In [23]:
```
plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()
```



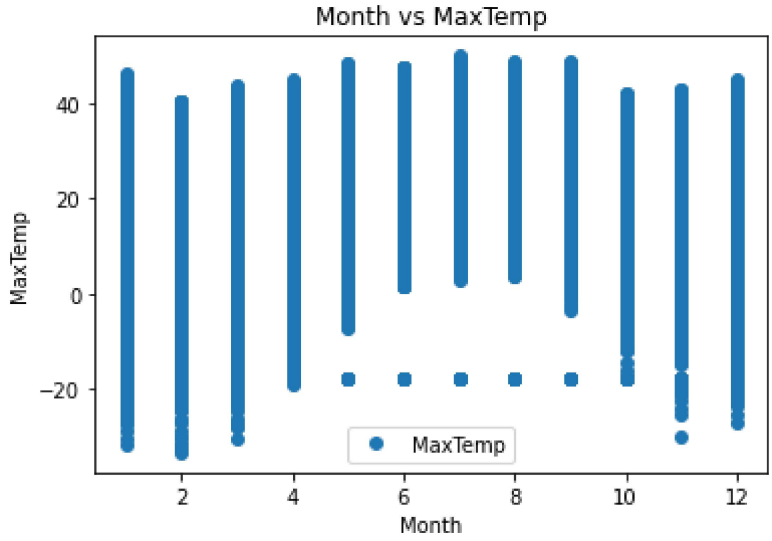
In [ ]:
```
```

In [24]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y
_pred)))
```

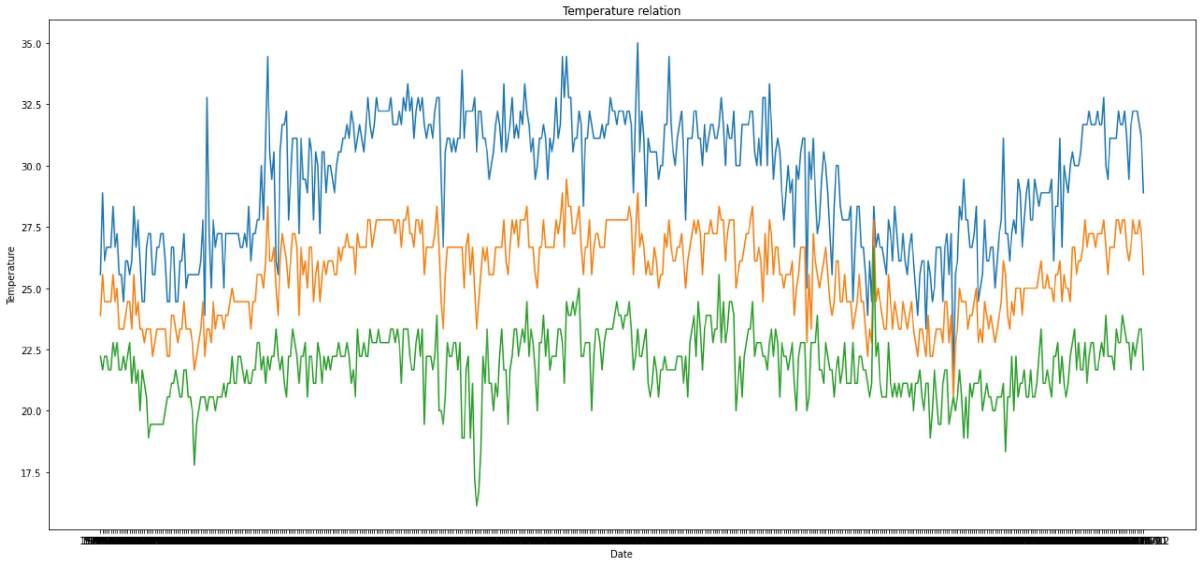
```
Mean Absolute Error: 3.1993291783785285
Mean Squared Error: 17.631568097568444
Root Mean Squared Error: 4.198996082109204
```

Multiple Linear Regression

In [25]:
```
import matplotlib.pyplot as plt
df.plot(x='MO', y='MaxTemp', style='o')
plt.title('Month vs MaxTemp')
plt.xlabel('Month')
plt.ylabel('MaxTemp')
plt.show()
```



In [26]:
```
plt.figure(figsize=(22,10))
plt.plot(df["Date"][:500],df["MaxTemp"][:500])
plt.plot(df["Date"][:500],df["MeanTemp"][:500])
plt.plot(df["Date"][:500],df["MinTemp"][:500])
plt.title("Temperature relation")
plt.xlabel("Date")
plt.ylabel("Temperature")
plt.show()
```



In [27]:
```
df['Precip']=pd.to_numeric(df['Precip'],errors='coerce')
```

In [28]:
```
df['Precip']
```

Out[28]:
```
0         1.016
1         0.000
2         2.540
3         2.540
4         0.000
          ...
119035    0.000
119036    9.906
119037    0.000
119038    0.000
119039    0.000
Name: Precip, Length: 119040, dtype: float64
```

In [29]:
```
df.dropna(subset=['MinTemp','MO','MeanTemp','Precip'],inplace=True)
```

In [30]:
```
#Multiple Linear Regression
n_X=df[['MinTemp','MO','MeanTemp','Precip']]
n_y=df['MaxTemp']
```

In [31]:
```
nX_train, nX_test, ny_train, ny_test = train_test_split(n_X, n_y, test_size=0.
2, random_state=0)
```

In [32]:
```
n_lr = LinearRegression()
n_lr.fit(nX_train, ny_train)
```

Out[32]:
```
LinearRegression()
```

In [33]:
```
print(n_lr.intercept_)
print(n_lr.coef_)
```

```
0.9008561086174076
[-0.86378141 -0.00216864  1.85445532 -0.00508113]
```

In [34]:
```
n_lr.predict([[20,11,25,1.2]])
```

Out[34]:
```
array([29.95665841])
```

In [35]:
```
#y=intercept+a*x1+b*x2+c*x3+d*x4
```

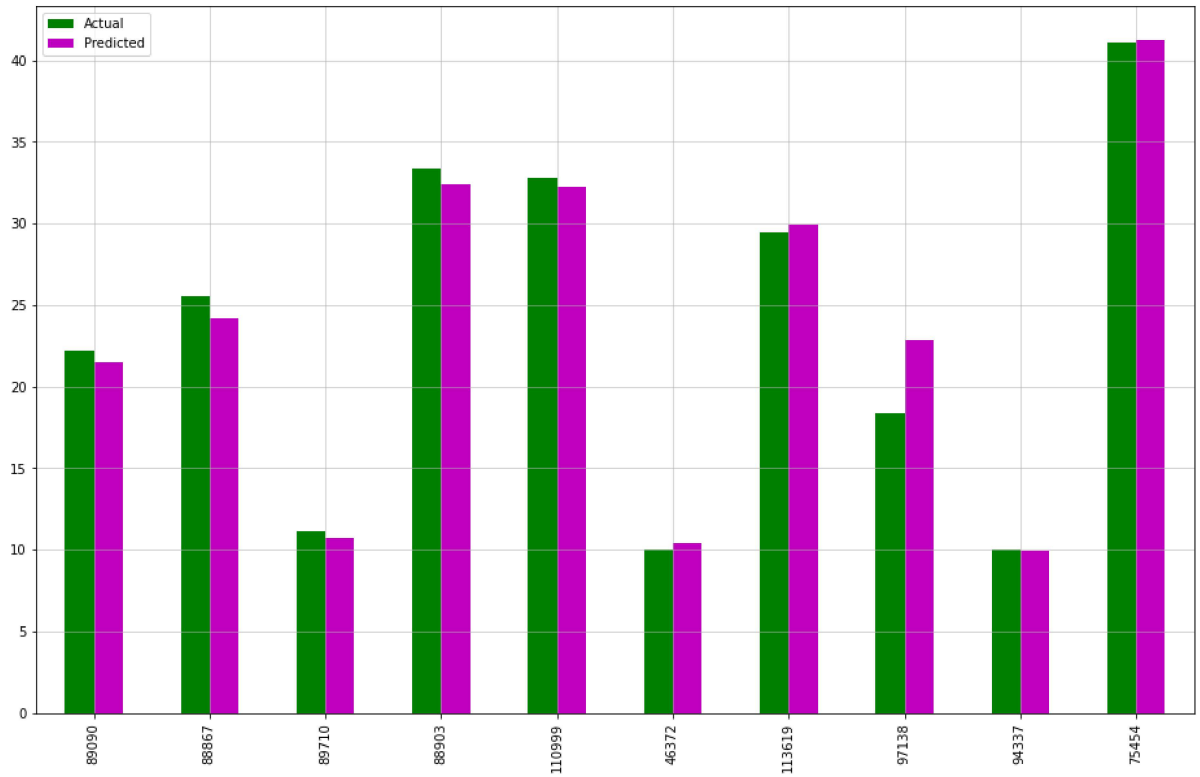
In [36]:
```
ny_pred=n_lr.predict(nX_test)
```

In [37]:
```
ny_pred,ny_test
```

Out[37]:
```
(array([21.44250569, 24.17644814, 10.69748345, ..., 31.2084209 ,
        21.42298795, 27.02166311]),
 89090     22.222222
 88867     25.555556
 89710     11.111111
 88903     33.333333
 110999    32.777778
             ...
 81656     25.555556
 99765     26.111111
 25238     31.111111
 82545     21.111111
 111950    27.222222
 Name: MaxTemp, Length: 20458, dtype: float64)
```

In [38]:
```
m_df=pd.DataFrame({'Actual': ny_test, 'Predicted': ny_pred})
```

In [39]:
```
m_df.head(10).plot(kind='bar',color='gm',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5')
plt.show()
```



In [40]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(ny_test, ny_pred))
print('Mean Squared Error:', metrics.mean_squared_error(ny_test, ny_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(ny_test,
ny_pred)))
```

```
Mean Absolute Error: 0.4165787663072768
Mean Squared Error: 1.1047876587504348
Root Mean Squared Error: 1.0510887967961768
```

In [ ]:

Program No. 4

Title:

Implement classification using K nearest neighbor classification

Objective:

To learn how to classify data by K nearest neighbor algorithm for classification

Reference:

Data Mining Introductory & Advanced Topic by Margaret H. Dunham

Data Mining Concept and Technique By Han & Kamber

Theory:

In k-nearest-neighbor classification, the training dataset is used to classify each member of a "target" dataset.

The structure of the data is that there is a classification (categorical) variable of interest ("buyer," or "non-buyer," for example), and a number of additional predictor variables (age, income, location so on.)

Algorithm:

For each row (case) in the target dataset (the set to be classified), locate the k closest members (the k nearest neighbors) of the training dataset. A Euclidean Distance measure is used to calculate how close each member of the training set is to the target row that is being examined.

Examine the k nearest neighbors - which classification (category) do most of them belong to? Assign this category to the row being examined.

Repeat this procedure for the remaining rows (cases) in the target set.

This algorithm lets the user select a maximum value for k, builds models parallelly on all values of k upto the maximum specified value and scoring is done on the best of these models.

The computing time goes up as k goes up, but the advantage is that higher values of k provide smoothing that reduces vulnerability to noise in the training data.

In practical applications, typically, k is in units or tens rather than in hundreds or thousands.

Name	Gender	Height(m)
Kristina	F	1.6
Jim	M	2
Maggie	F	1.9
Bob	M	1.85
Dave	F	1.7
Kimm	M	1.9
Todd	M	1.9
Amy	F	1.85
Kathy	F	1.6

2m<=Tall, 1.7m< H<2m Medium, H<=1.7m Short

OutPut:

New Tuple <Pat,F,1.6> , suppose K=5 is given than K nearest neighbors to input tuple

{(Kristina,F,1.6) , (Kathy,F,1.6),(Dave,F,1.7)}

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
4							

Program No. 4

Paste your DATA SHEET here

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
```

In [2]:
```
df=pd.read_csv(r"C:\Users\Admin\Documents\7th sem\ds\datasets\P4_winequality.csv")
```

In [91]:
```
df.shape
```

Out[91]:
```
(1599, 12)
```

In [92]:
```
print(df.columns)
df.info()
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [93]:
```
labels=[]
cnt=0
for i in range(0, len(df['quality'])):

    if df['quality'][i] not in labels:

        labels.append(df['quality'][i])

        cnt += 1

print(n,labels)
```

```
6 [5, 6, 7, 4, 8, 3]
```

In [94]:
```
df.describe().head(5)
```

Out[94]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sul dioxi
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000(
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.4677
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.8953
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000(
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000(

In [95]:
```
df.count()
```

Out[95]:
```
fixed acidity           1599
volatile acidity        1599
citric acid             1599
residual sugar          1599
chlorides               1599
free sulfur dioxide     1599
total sulfur dioxide    1599
density                 1599
pH                      1599
sulphates               1599
alcohol                 1599
quality                 1599
dtype: int64
```

In [96]:
```
X=df.drop(['quality'],axis=1)
```

In [97]:
```
X.head()
```

Out[97]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoho
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

In [30]:
```
y=df['quality'].values
```

In [31]:
```

```

Out[31]:
```
array([5, 5, 5, ..., 6, 5, 6], dtype=int64)
```

In [32]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

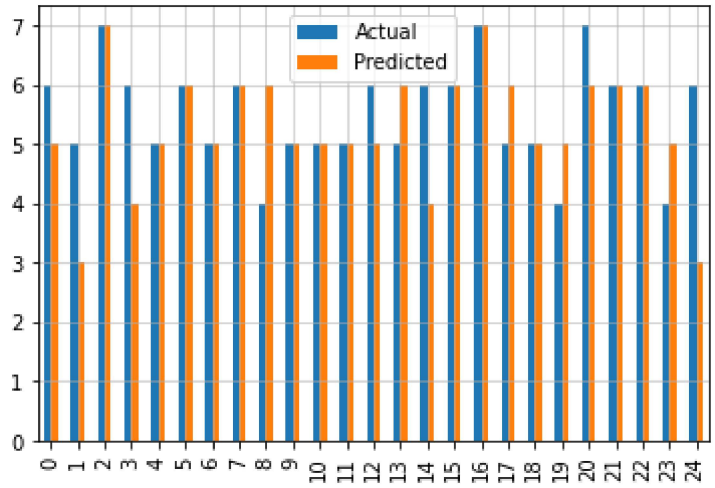
k=3

In [56]:
```
knn3=KNeighborsClassifier(n_neighbors=3)
knn3.fit(X_train,y_train)
```

Out[56]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [57]:
```
y_pred3=knn3.predict(X_test)
```

In [58]:
```
df1=df.head(25)
df1.plot(kind='bar')

plt.grid(which='major',linewidth='0.5')
plt.grid(which='minor',linewidth='0.5')
plt.show()
```



In [59]:
```
metrics.mean_absolute_error(y_test,y_pred3)
```

Out[59]:
```
0.60625
```

In [60]:
```
df3=pd.DataFrame({'Actual':y_test.flatten(),'Predicted':y_pred3.flatten()})
print(df)
```

```
     Actual  Predicted
0         6          5
1         5          3
2         7          7
3         6          4
4         5          5
..      ...        ...
315       6          7
316       4          4
317       5          6
318       4          5
319       6          7

[320 rows x 2 columns]
```

In [61]:
```
orig=np.where(y_test>5,'Good','BAd')
pred=np.where(y_pred3>5,'Good','BAd')
```

In [62]:
```
s=pd.DataFrame({'actual':orig,'Pred':pred})
```

In [63]:
```
s
```

Out[63]:

	actual	Pred
0	Good	BAd
1	BAd	BAd
2	Good	Good
3	Good	BAd
4	BAd	BAd
...	...	...
315	Good	Good
316	BAd	BAd
317	BAd	Good
318	BAd	BAd
319	Good	Good

320 rows × 2 columns

In [64]:
```
knn3.score(X_test,y_test)
```

Out[64]: 0.5125

In [ ]:
```

```

k=4

In [65]:
```
knn4=KNeighborsClassifier(n_neighbors=4)
knn4.fit(X_train,y_train)
y_pred4=knn4.predict(X_test)
```

In [66]:
```
metrics.mean_absolute_error(y_test,y_pred4)
```

Out[66]: 0.53125

In [67]:
```
df4=pd.DataFrame({'Actual':y_test.flatten(),'Predicted':y_pred4.flatten()})
print(df4)
```

```
     Actual  Predicted
0         6          5
1         5          5
2         7          7
3         6          6
4         5          6
..      ...        ...
315       6          7
316       4          5
317       5          6
318       4          5
319       6          7

[320 rows x 2 columns]
```

In [68]:
```
knn4.score(X_test,y_test)
```

Out[68]: 0.521875

k=5

In [69]:
```
knn5=KNeighborsClassifier(n_neighbors=5)
knn5.fit(X_train,y_train)
y_pred5=knn5.predict(X_test)
```

In [70]:
```
metrics.mean_absolute_error(y_test,y_pred5)
```

Out[70]: 0.565625

In [71]:
```
knn5.score(X_test,y_test)
```

Out[71]: 0.48125

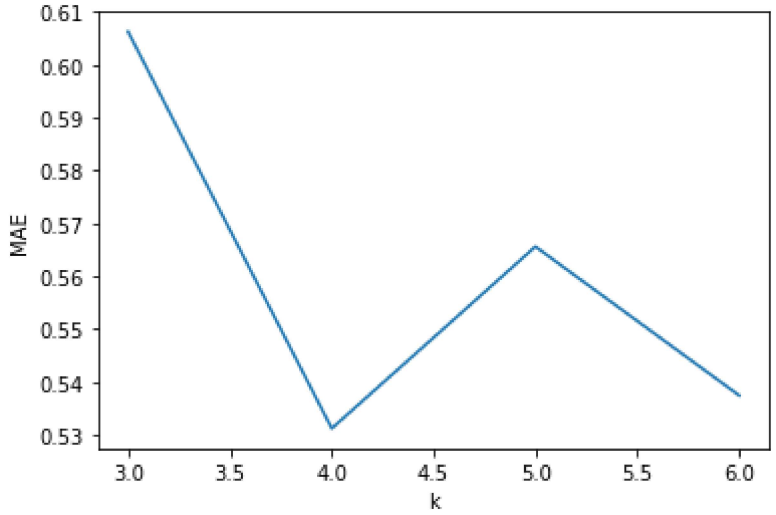
k=6

In [80]:
```
knn6=KNeighborsClassifier(n_neighbors=6)
knn6.fit(X_train,y_train)
y_pred6=knn6.predict(X_test)
print('Mean Abs Error',metrics.mean_absolute_error(y_test,y_pred6))
print('score',knn6.score(X_test,y_test))
```

```
Mean Abs Error 0.5375
score 0.509375
```

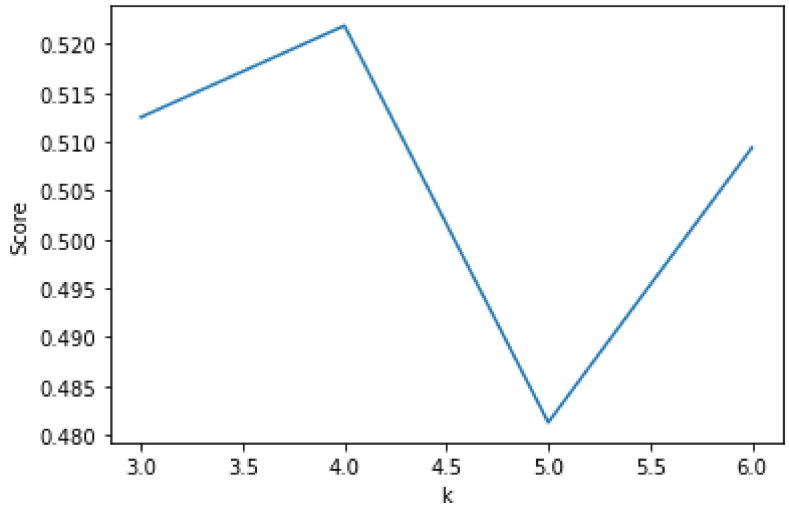
In [83]:
```
xe=[3,4,5,6]
ye=[0.60625,0.53125,0.565625,0.5375]
```

In [84]:
```
plt.plot(xe, ye)
plt.xlabel('k')

plt.ylabel('MAE')

plt.show()
```



In [85]:
```
xs=[3,4,5,6]
ys=[0.5125,0.521875,0.48125,0.509375]
```

In [86]:
```
plt.plot(xs, ys)
plt.xlabel('k')

plt.ylabel('Score')

plt.show()
```



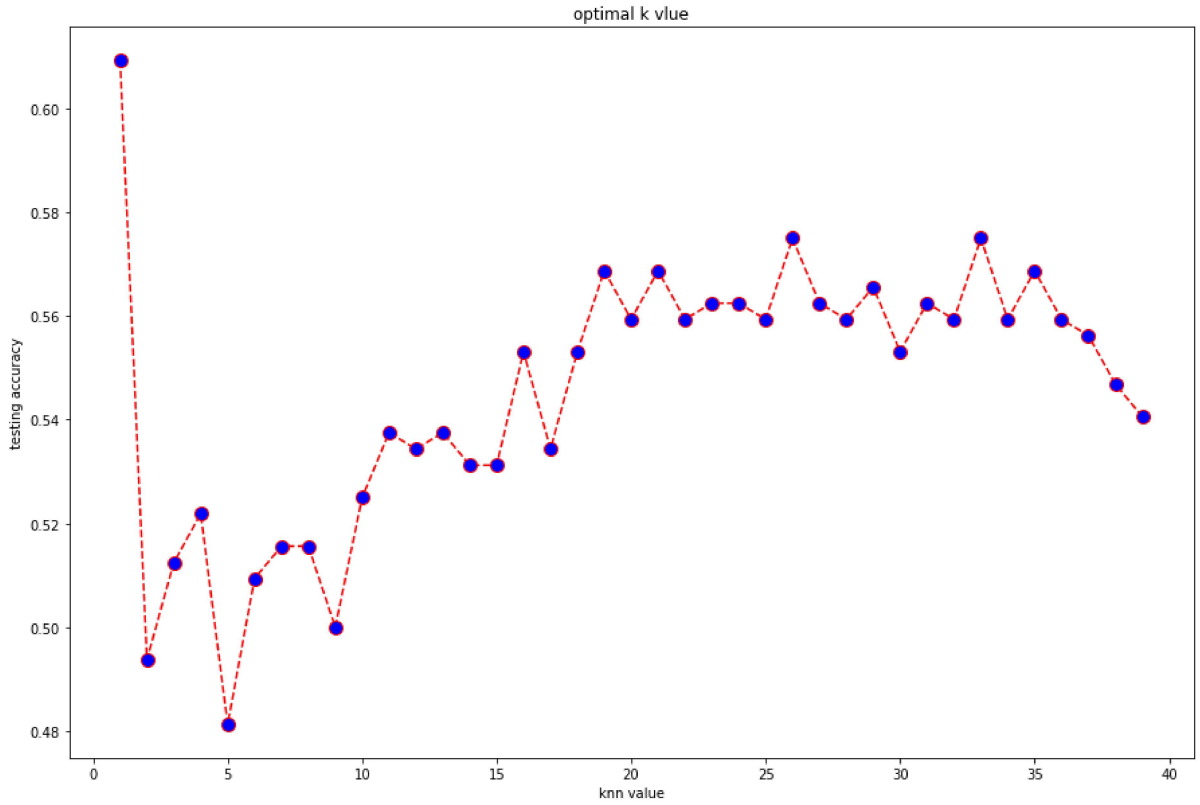
In [112]:
```
from sklearn.model_selection import cross_val_score
import numpy as np

knn_cv = KNeighborsClassifier(n_neighbors=4)
#train model with cv of 5
cv_scores = cross_val_score(knn_cv, X, y, cv=5)
#print each cv score (accuracy) and average them
print(cv_scores)
print(np.mean(cv_scores))
```

```
[0.45625    0.484375   0.475      0.390625   0.42946708]
0.44714341692789966
```

In [113]:
```
score = []
k = range(1,20)
for i in k:
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    y_pred = knn.predict(X_test)
    score.append(metrics.accuracy_score(y_test,y_pred))
    #print(classification_report(y_test,y_pred))
    #print(confusion_matrix(y_test,y_pred))
score
```

Out[113]:
```
[0.609375,
 0.49375,
 0.5125,
 0.521875,
 0.48125,
 0.509375,
 0.515625,
 0.515625,
 0.5,
 0.525,
 0.5375,
 0.534375,
 0.5375,
 0.53125,
 0.53125,
 0.553125,
 0.534375,
 0.553125,
 0.56875,
 0.559375,
 0.56875,
 0.559375,
 0.5625,
 0.5625,
 0.559375,
 0.575,
 0.5625,
 0.559375,
 0.565625,
 0.553125,
 0.5625,
 0.559375,
 0.575,
 0.559375,
 0.56875,
 0.559375,
 0.55625,
 0.546875,
 0.540625]
```

In [114]:
```
plt.figure(figsize =(15,10))
plt.plot(k,score,markersize = 10,color = 'red',linestyle = 'dashed',marker =
'o',markerfacecolor= 'blue')
plt.title('optimal k vlue')
plt.xlabel('knn value')

plt.ylabel('testing accuracy')
plt.show()
```



In [ ]:

Program No. 5

Title:

Implement decision tree based algorithm for classification

Objective:

To learn decision tree based algorithm for classification

Reference:

Data Mining Introductory & Advanced Topic by Margaret H. Dunham

Data Mining Concept and Technique By Han & Kamber

Theory:

Decision tree learning, used in data mining and machine learning, uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. More descriptive names for such tree models are classification trees or regression trees. In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making.

Basic steps in building tree

Applying the tree to database

Internal node-test on attribute Branch-outcome of test

Leaf node-class Topmost-root node

Algorithm

1.compute the entropy for data-set

2.for every attribute/feature:

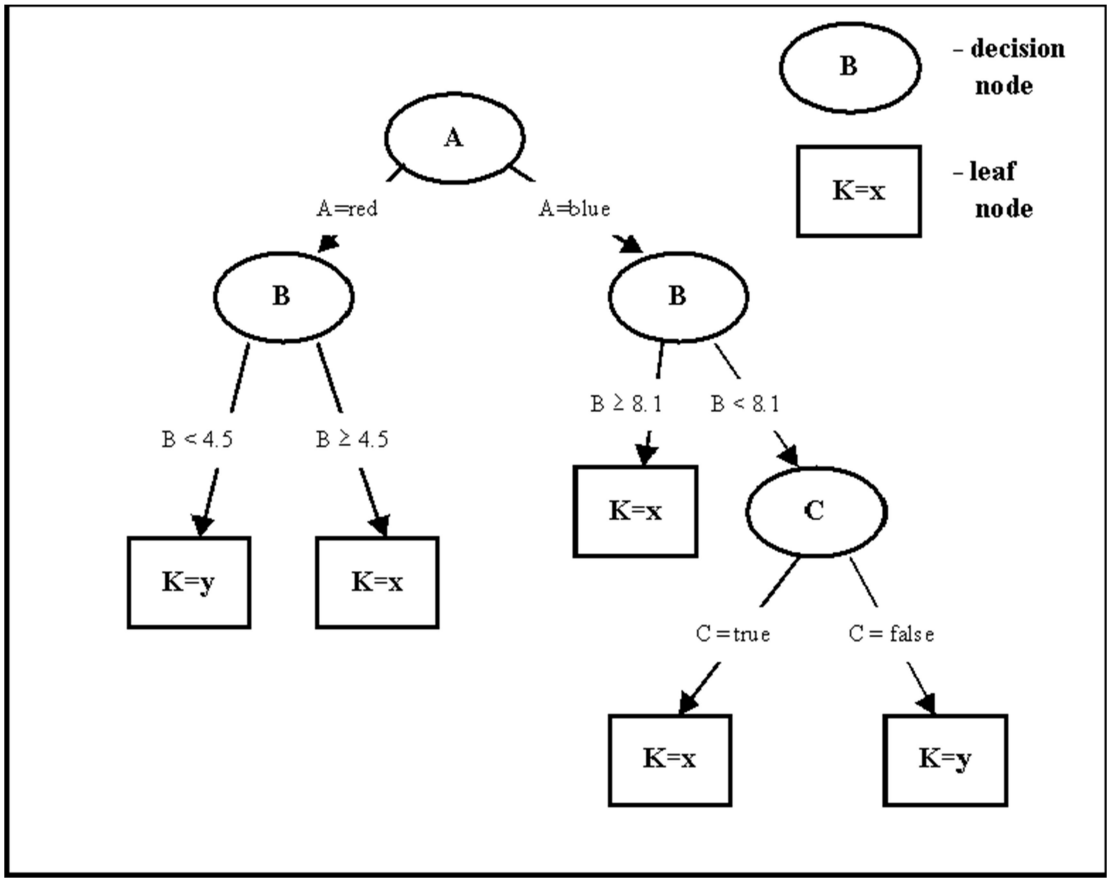
calculate entropy for all categorical values

take average information entropy for the current attribute

calculate gain for the current attribute

3. pick the highest gain attribute.

4. Repeat until the desired tree is constructed .



Final Decision Tree

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
5							

Program No. 5

Paste your DATA SHEET here

In [1]:
```
import pandas as pd
import numpy as numpy
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
```

In [2]:
```
df = pd.read_csv("diabetes.csv")
df.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [4]:
```
feature_cols = ['Pregnancies','Glucose','BloodPressure' ,'SkinThickness','Insu
lin' ,'BMI','DiabetesPedigreeFunction','Age']
```

In [5]:
```
df.corr()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.01
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.22
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.28
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.39
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.19
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.00
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.14
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.03
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.29

In [6]:
```
df.describe()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [7]:
```
df.shape
```

Out[7]:
```
(768, 9)
```

In [8]:
```
X = df.iloc[:,:-1].values
```

In [9]:
```
y = df.iloc[:,-1].values
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, rando
m_state = 1)
```

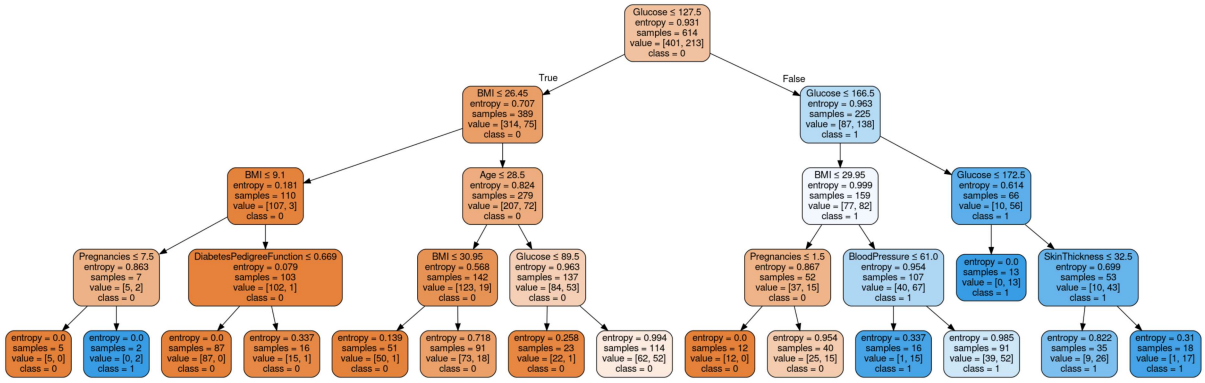
In [11]:
```
clf = DecisionTreeClassifier(criterion = 'entropy', max_depth = 4)
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
```

In [12]:
```
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.7987012987012987
```

In [13]:
```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_nam
es=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Out[13]:



In [ ]:
```
```

Program No. 6

Title:

Navie Bayesian Classification

Objective:

To implement classification using Bayes theorm.

Reference:

Data Mining Introductory & Advanced Topic by Margaret H. Dunham

Data Mining Concept and Technique By Han & Kamber

Theory:

The simple baysian classification assumes that the effect of an attribute value of a given class membership is independent of other attribute.

The Bayes theorem is as follows –

Let X be an unknown sample. Let it be hypothesis such that X belongs to particular class C. We need to determine P(H/X).

The probability that hypothesis it holds is given that all values of X are observed.

$$P(H/X)=(P(X/H).P(H))/P(X)$$

In this program, initially take the number of tuples in training data set in variable L. The string array's name, gender, height, output to store the details and output respectfully. Therefore, the tuple details are taken from user using 'for' loops. Bayesian classification has an expected classification. Now using the counter variables for various attributes i.e. (male/female) for gender and (short/medium/tall) for height. The tuples are scanned and the respective counter is incremented accordingly using if- else-if structure. Therefore variables pshort, pmed, plong are used to convert the counter variables to corresponding values.

Algorithm –

START

- Store the training data set
- Specify ranges for classifying the data
- Calculate the probability of being tall, medium, short
- Also, calculate the probabilities of tall, short, medium according to gender and

classification ranges

- Calculate the likelihood of short, medium and tall
- Calculate P(t) by summing up of probable likelihood
- Calculate actual probabilities

Input :

Training data set

Name	Gender	Height	Output
Christina	F	1.6m	Short
Jim	M	1.9m	Tall
Maggie	F	1.9m	Medium
Martha	F	1.88m	Medium
Stephony	F	1.7m	Medium
Bob	M	1.85m	Short
Dave	M	1.7m	Short
Steven	M	2.1m	Tall
Amey	F	1.8m	Medium

Output

The tuple belongs to the class having highest probability. Thus new tuple is classified.

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
6							

Program No. 6

Paste your DATA SHEET here

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt1
import seaborn as sns
import string
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.metrics import classification_report,confusion_matrix,accuracy_sc
ore
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC, NuSVC, LinearSVC
%matplotlib inline
```

In [2]:
```
df = pd.read_csv(r'C:\Users\Admin\Documents\7th sem\ds\datasets\P6_P7_spam.cs
v',encoding = 'latin-1')
```

In [3]:
```
df.head()
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [4]:
```
df.shape
```

Out[4]:
```
(5572, 5)
```

In [5]:
```
df.info()
df=df.dropna(axis=1)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

In [6]:
```
df.shape
```

Out[6]:
```
(5572, 2)
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [8]:
```
df.columns = ['label','message']
df = df[['message','label']]
```

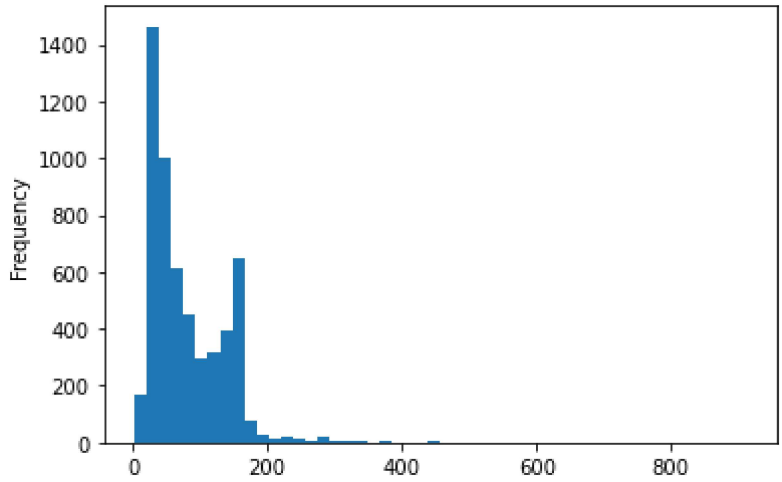
In [9]:
```
df['length']=df['message'].apply(len)
df.head()
```

Out[9]:

	message	label	length
0	Go until jurong point, crazy.. Available only ...	ham	111
1	Ok lar... Joking wif u oni...	ham	29
2	Free entry in 2 a wkly comp to win FA Cup fina...	spam	155
3	U dun say so early hor... U c already then say...	ham	49
4	Nah I don't think he goes to usf, he lives aro...	ham	61

In [10]:
```
df['length'].plot(bins=50,kind='hist')
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c6ea38e190>
```



In [11]:
```
df.length.describe()
```

Out[11]:
```
count    5572.000000
mean       80.118808
std        59.690841
min         2.000000
25%        36.000000
50%        61.000000
75%       121.000000
max       910.000000
Name: length, dtype: float64
```

In [12]:
```
df[df['length']==910]['message'].iloc[0]
```

Out[12]:
```
"For me the love should start with attraction.i should feel that I need her e
very time around me.she should be the first thing which comes in my thoughts.
I would start the day and end it with her.she should be there every time I dr
eam.love will be then when my every breath has her name.my life should happen
around her.my life will be named to her.I would cry for her.will give all my
happiness and take all her sorrows.I will be ready to fight with anyone for h
er.I will be in love when I will be doing the craziest things for her.love wi
ll be when I don't have to proove anyone that my girl is the most beautiful l
ady on the whole planet.I will always be singing praises for her.love will be
when I start up making chicken curry and end up makiing sambar.life will be t
he most beautiful then.will get every morning and thank god for the day becau
se she is with me.I would like to say a lot..will tell later.."
```

In [13]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[13]:
```
True
```

In [14]:
```
def text_process(mess):#remove punctuation
    nopunc =[char for char in mess if char not in string.punctuation]
    nopunc=''.join(nopunc)
    return [word for word in nopunc.split() if word.lower() not in stopwords.w
ords('english')] #remove stopwords
```

In [15]:
```
#nltk.download('stopwords')
df['message'].head(5).apply(text_process) #split words remove stopwords and pu
nctuation
```

Out[15]:
```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                        [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: message, dtype: object
```

In [17]:
```
msg_train,msg_test,label_train,label_test = train_test_split(df['message'],df[
'label'],test_size=0.2)

bow_transformer = CountVectorizer(analyzer=text_process).fit(msg_train)
```

In [18]:
```
print(len(bow_transformer.vocabulary_))
```

```
10008
```

In [19]:
```
#messages_bow = bow_transformer.transform(df['message'])
messages_bow = bow_transformer.transform(msg_train)
```

In [20]:
```
tfidf_transformer = TfidfTransformer(use_idf = False) #feature seletion
messages_tfidf=tfidf_transformer.transform(messages_bow)
print(messages_tfidf.shape)
```

```
(4457, 10008)
```

In [21]:
```
#spam_detect_model = MultinomialNB().fit(messages_tfidf,df['label'])
spam_detect_model = MultinomialNB().fit(messages_tfidf,label_train)
```

In [22]:
```
#bow_transformer1 = CountVectorizer(analyzer=text_process).fit(msg_test)
messages_bow1 = bow_transformer.transform(msg_test)
tfidf_transformer = TfidfTransformer(use_idf = False)
messages_tfidf1=tfidf_transformer.transform(messages_bow1)
all_predictions = spam_detect_model.predict(messages_tfidf1)
print(messages_tfidf1.shape)

print(all_predictions)
print(len(all_predictions))
```

```
(1115, 10008)
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
1115
```

In [23]:
```
print(classification_report(label_test,all_predictions))
print(confusion_matrix(label_test,all_predictions))
```

```
              precision    recall  f1-score   support

         ham       0.95      1.00      0.97       957
        spam       1.00      0.68      0.81       158

    accuracy                           0.95      1115
   macro avg       0.97      0.84      0.89      1115
weighted avg       0.96      0.95      0.95      1115

[[957   0]
 [ 51 107]]
```

In [24]:
```
accuracy_score(label_test,all_predictions)
```

Out[24]:
```
0.9542600896860987
```

In [ ]:
```

```

Program No. 7

Title:

Implementation of Support Vector Machine

Objective:

To understand the dynamics of SVM

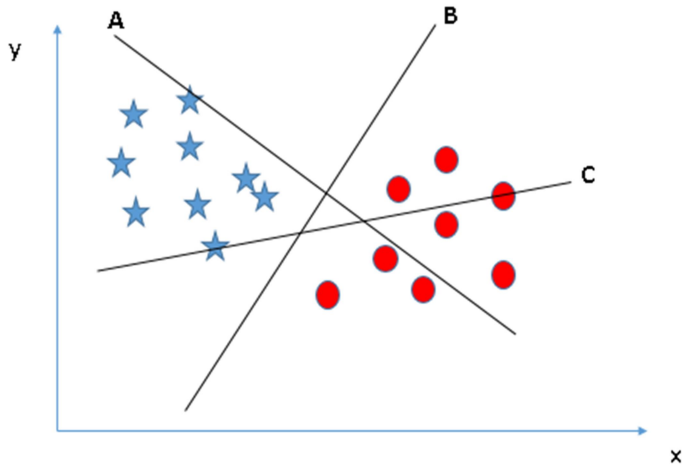
Reference:

Data Mining Concept and Technique By Han & Kamber

Theory:

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well .

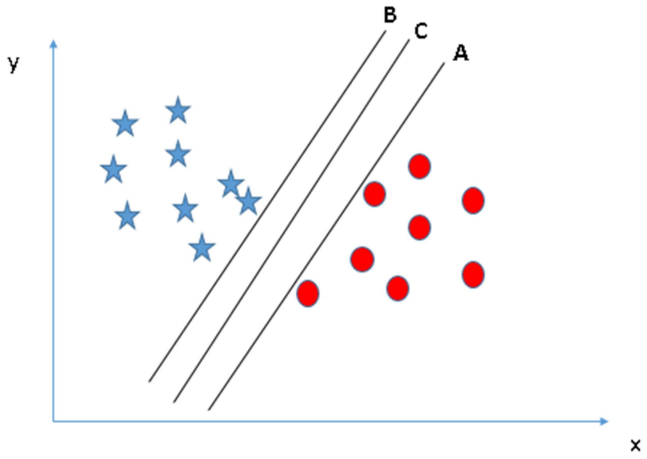
Let's understand: Identify the right hyper-plane (Scenario-1): Consider three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.



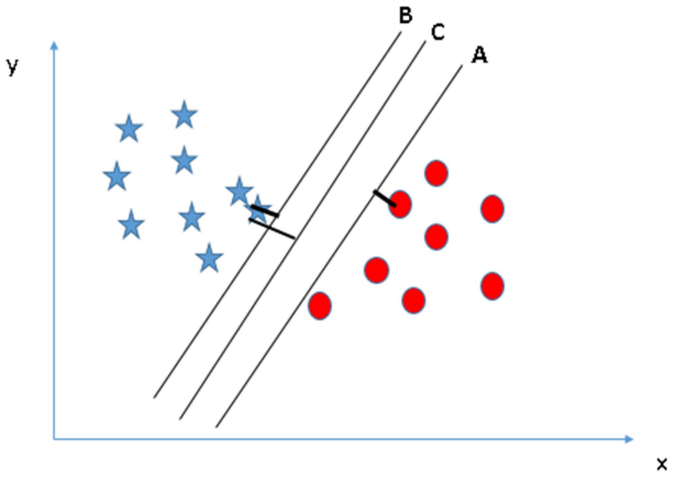
You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right

hyper-plane?

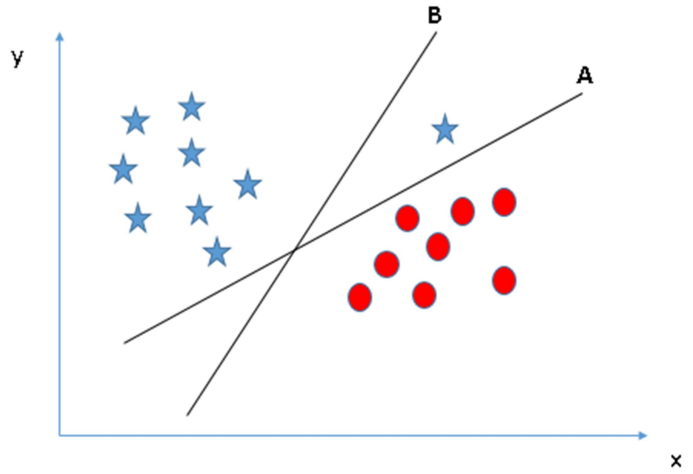


Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin. Let's look at the below snapshot:



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

Identify the right hyper-plane (Scenario-3):Hint: Use the rules as discussed in previous section to identify the right hyper-plane

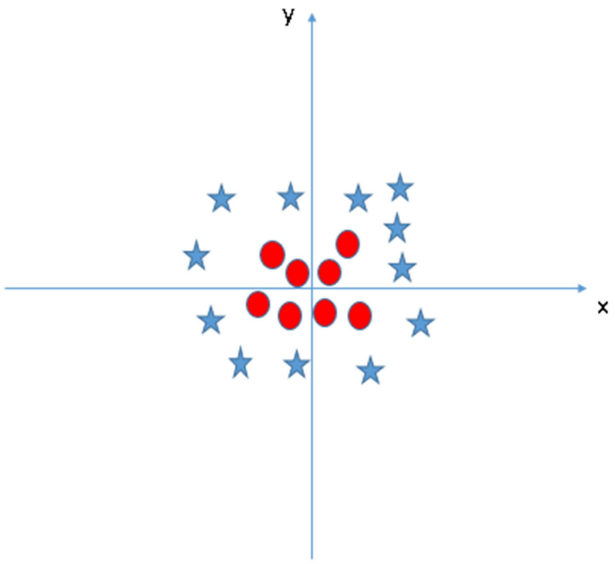


Some of you may have selected the hyper-plane B as it has higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

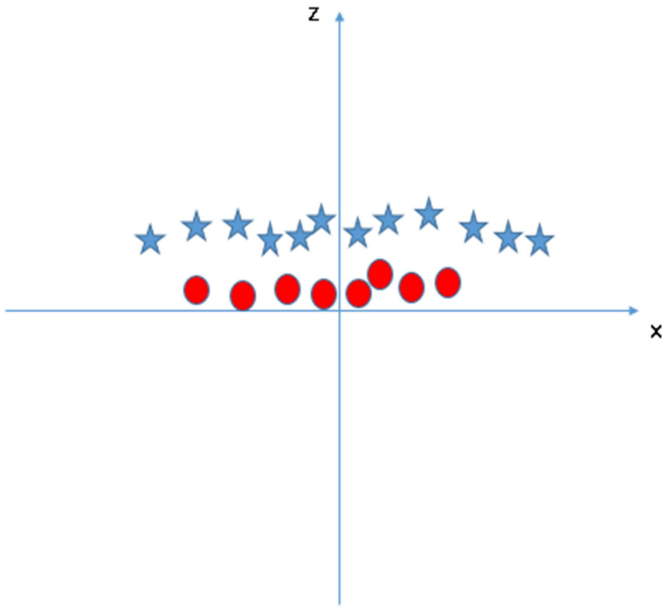
Can we classify two classes (Scenario-4)?: Below, I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.

As I have already mentioned, one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.

Find the hyper-plane to segregate to classes (Scenario-5): In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:



In above plot, points to consider are:

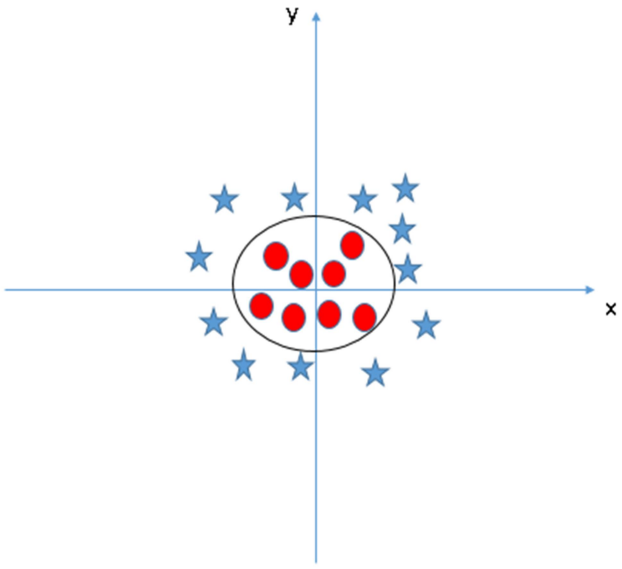
All values for z would be positive always because z is the squared sum of both x and y

In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the kernel trick. These are

functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:



Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
7							

Program No. 7

Paste your DATA SHEET here

In [11]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
ort
import seaborn as sns
import string
import nltk
from nltk.corpus import stopwords

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

from sklearn.svm import SVC,LinearSVC,NuSVC

import matplotlib.pyplot as plt
%matplotlib inline
```

In [5]:
```
df=pd.read_csv('P6_P7_spam.csv',encoding='latin-1')
```

In [6]:
```
df=df[['v1','v2']]
```

In [7]:
```
df.columns=['label','message']
```

In [8]:
```
def processing_text(message):
    msg_nopunc=[c for c in message if c not in string.punctuation]
    msg_nopunc=''.join(msg_nopunc)
    return [w for w in msg_nopunc.split() if w.lower() not in stopwords.words(
'english')]
```

In [9]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Sonal\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[9]: True

In [12]:
```
X_train,X_test,y_train,y_test=train_test_split(df['message'],df['label'],test_
size=0.2,random_state=0)
```

In [13]:
```
bow_transformer=CountVectorizer(analyzer=processing_text).fit(X_train)
mess_trans=bow_transformer.transform(X_train)
tfidf_transformer=TfidfTransformer(use_idf=False)
mess_tfidf=tfidf_transformer.transform(mess_trans)
```

In [14]:
```
mess_bow1=bow_transformer.transform(X_test)
tfidf_transformer1=TfidfTransformer(use_idf=False)
mess_tfidf1=tfidf_transformer1.transform(mess_bow1)
```

In [15]:
```
svc=LinearSVC()
svc.fit(mess_tfidf,y_train)
```

Out[15]:
```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
```

In [19]:
```
y_pred=svc.predict(mess_tfidf1)
```

In [20]:
```
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99       949
        spam       0.99      0.89      0.94       166

    accuracy                           0.98      1115
   macro avg       0.99      0.95      0.96      1115
weighted avg       0.98      0.98      0.98      1115

[[948   1]
 [ 18 148]]
```

In [22]:
```
accuracy_score(y_test,y_pred)
```

Out[22]:
```
0.9829596412556054
```

In [ ]:
```

```

In [23]:
```
svc=SVC()
svc.fit(mess_tfidf,y_train)
```

Out[23]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [24]:
```
y_pred=svc.predict(mess_tfidf1)
```

In [25]:
```
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98       949
        spam       0.99      0.83      0.90       166

    accuracy                           0.97      1115
   macro avg       0.98      0.92      0.94      1115
weighted avg       0.97      0.97      0.97      1115

[[948   1]
 [ 28 138]]
```

In [ ]:
```

```

Program No. 8

Title:

Implement Apriori algorithm for association rule

Objective:

To learn association rule for Apriori algorithm

Reference:

Data Mining Introductory & Advanced Topic by Margaret H. Dunham

Data Mining Concept and Technique By Han & Kamber

Theory:

Association rule mining is defined as:

Let I= { …} be a set of 'n' binary attributes called items. Let D= { ….} be set of transaction called database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. a rule is defined as implication of the form X→Y where X, Y⊆ I and X∩Y=Φ. The set of items X and Y are called antecedent and consequent of the rule respectively.

Useful Terms

To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best known constraints are minimum thresholds on support and confidence.

Support

The support supp(X) of an item set X can be defined as proportion of transactions in the data set which contain the item set.

Supp(X) = no. of transactions which contain the item set 'X' / total no. of transactions

Confidence

The confidence of a rule is defined as:

Conf (X→Y) = supp (XUY)/supp(X)

Definition of Apriori Algorithm

The Apriori Algorithm is an influential algorithm for mining frequent item sets for Boolean association rules. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation, and groups of candidates are tested against the data. Apriori is designed to operate on

database containing transactions (for example, collections of items bought by customers, or details of a website frequentation).

Key Concept

Frequent item sets: All the sets which contain the item with the minimum support (denoted as for item set.

Apriori Property: Any subset of frequent item set must be frequent.

Join operation: To find, a set of candidate k-item sets is generated by joining with itself.

Apriori Algorithm Steps

Below are the apriori algorithm steps:

- Scan the transaction data base to get the support 'S' each 1-itemset, compare 'S' with min_sup, and get a support of 1-itemsets,
- Use join to generate a set of candidate k-item set. Use apriori property to prune the unfrequented k-item sets from this set.
- Scan the transaction database to get the support 'S' of each candidate k-item set in the given set, compare 'S' with min_sup, and get a set of frequent k-item set
- If the candidate set is NULL, for each frequent item set 1, generate all nonempty subsets of 1.
- For every nonempty subsets of 1, output the rule "s=>(1-s)" if confidence C of the rule "s=>(1-s)" min_conf
- If the candidate set is not NULL, go to step 2.

Example for Apriori Algorithms

Market-Basket Analysis is one of the examples for Apriori.

Provides insight into which products tend to be purchased together and which are most amenable to promotion.

Actionable rules

Trivial rules

People who buy chalk-piece also buy duster

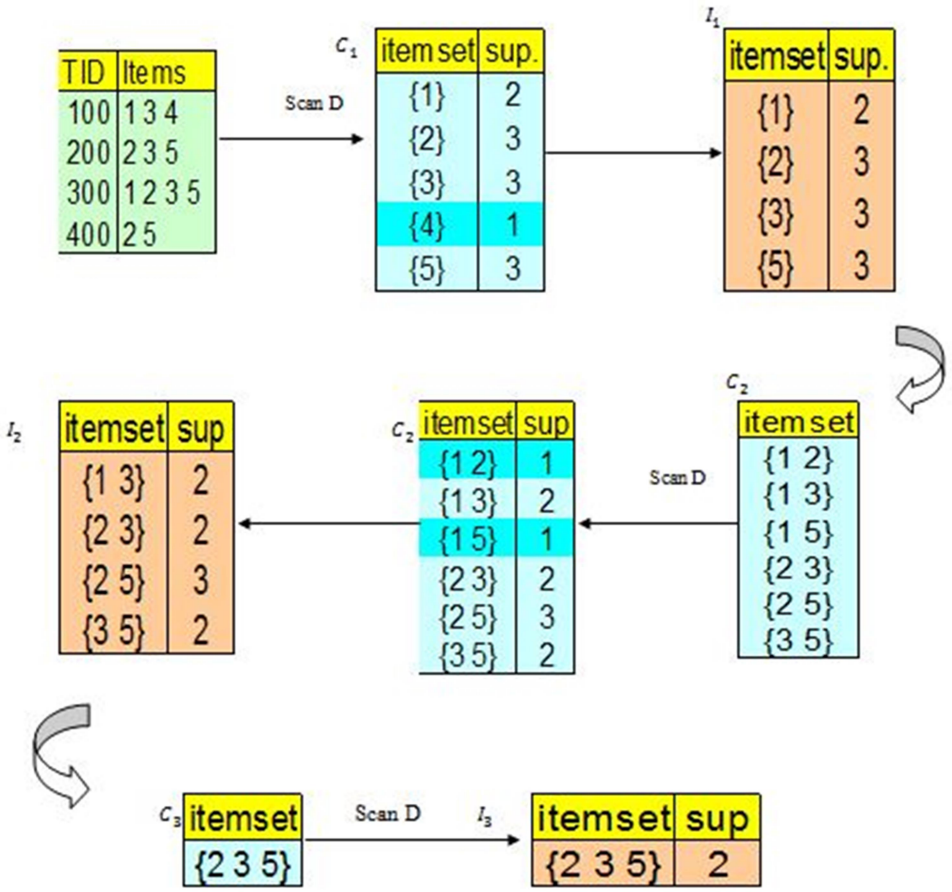
Inexplicable

People who buy mobile also buy bag

Database D

Minsup = 0.5

Example of Apriori algorithm



Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
8							

Program No. 8

Paste your DATA SHEET here

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
```

In [5]:
```
store_data = pd.read_csv(r'C:\Users\Admin\Documents\7th sem\ds\datasets\P8_store_data.csv',header=None)
```

In [6]:
```
store_data.head()
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	10
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN

In [7]:
```
store_data.shape
```

Out[7]:
```
(7501, 20)
```

In [8]:
```
records = []
for i in range(0, 7501):
    records.append([str(store_data.values[i,j]) for j in range(0, 20)])
```

In [9]:
```
association_rules = apriori(records, min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2)
association_results = list(association_rules)
```

In [10]:
```
print(len(association_results))
```

```
48
```

In [11]:
```
print(association_results[0])
```

```
RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.004532728969470737, ordered_statistics=[OrderedStatistic(items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=4.84395061728395)])
```

In [19]:
```
l1=[]
l2=[]
for item in association_results:
    pair = item[0]
    items = [x for x in pair]
    l1.append(item[1])
    l2.append(item[2][0][2])

print("Rule: " + items[0] + " -> " + items[1])
print("Support: " + str(item[1]))
print("Confidence: " + str(item[2][0][2]))
print("Lift: " + str(item[2][0][3]))
print("====================================")
```

```
Rule: mineral water -> spaghetti
Support: 0.004532728969470737
Confidence: 0.28813559322033894
Lift: 3.0228043143297376
====================================
```

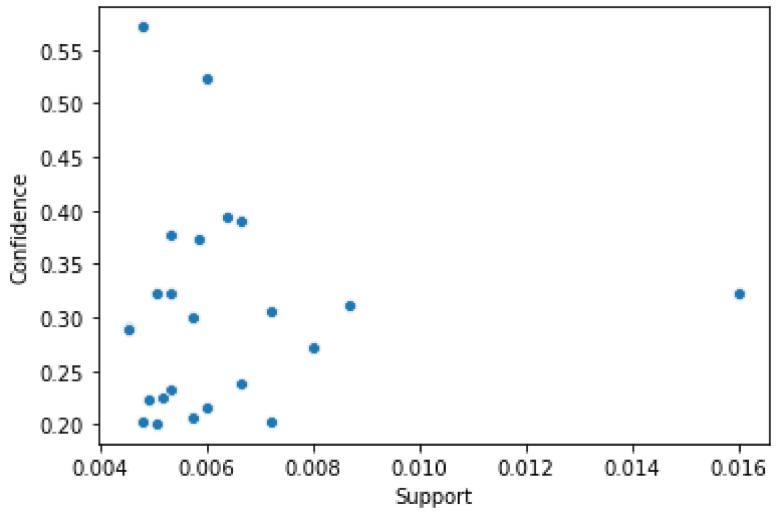
In [13]:
```
l1[:10]
```

Out[13]:
```
[0.004532728969470737,
 0.005732568990801226,
 0.005865884548726837,
 0.015997866951073192,
 0.005332622317024397,
 0.007998933475536596,
 0.005065991201173177,
 0.004532728969470737,
 0.005332622317024397,
 0.004799360085321957]
```

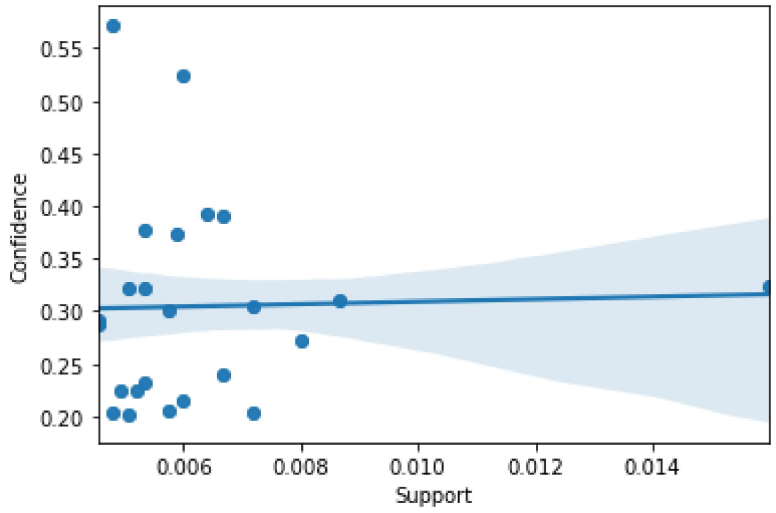
In [15]:
```
l2[:10]
```

Out[15]:
```
[0.29059829059829057,
 0.3006993006993007,
 0.3728813559322034,
 0.3234501347708895,
 0.3773584905660377,
 0.2714932126696833,
 0.3220338983050847,
 0.29059829059829057,
 0.23255813953488375,
 0.5714285714285714]
```

In [16]:
```
import seaborn as sns

data_plot = pd.DataFrame({"Support":l1, "Confidence":l2})


sns.scatterplot(x = "Support", y = "Confidence", data=data_plot)
plt.show()
```



In [17]:
```
ax = sns.regplot(x="Support", y="Confidence", data=data_plot)
```



In [ ]:
```

```

Program No. 9

Title:

Implement K means algorithm for clustering

Objective:

To learn K means algorithm for clustering

Reference:

Data Mining Introductory & Advanced Topic by Margaret H. Dunham

Data Mining Concept and Technique By Han & Kamber

Theory:

In statistics and machine learning, k-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.-Mean Clustering algorithm works?

Step by step explanation of K-means clustering algorithm:

Step 1. Begin with a decision on the value of k = number of clusters

Step 2. Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as the following:

Take the first k training sample as single-element clusters

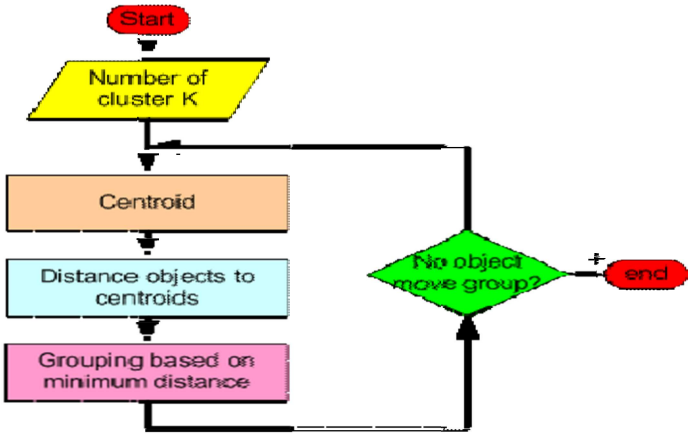
Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid. After each assignment, recomputed the centroid of the gaining cluster.

Step 3 . Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.

Step 4 . Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

If the number of data is less than the number of cluster then we assign each data as the centroid of the cluster. Each centroid will have a cluster number. If the number of data is bigger than the number of cluster, for each data, we calculate the distance to all centroid and get the minimum distance. This data is said belong to the cluster that has minimum distance from this data.

Flow chart for K-means Clustering



Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
9							

Program No. 9

Paste your DATA SHEET here

In [17]:
```
import numpy as np
import pandas as pd
import random
from base64 import b64encode
from json import loads
```

In [18]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [19]:
```
from mlxtend.data import loadlocal_mnist
```

In [20]:
```
X, y = loadlocal_mnist(images_path='train-images.idx3-ubyte',labels_path='trai
n-labels.idx1-ubyte')
```

In [21]:
```
X.shape
```

Out[21]:
```
(60000, 784)
```

In [48]:
```
X = X.astype(float) / 255
```

In [49]:
```
np.unique(y)
```

Out[49]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=uint8)
```

In [50]:
```
np.bincount(y)
```

Out[50]:
```
array([5923, 6742, 5958, 6131, 5842, 5421, 5918, 6265, 5851, 5949],
      dtype=int64)
```

In [51]:
```
#np.savetxt(fname='images.csv', X=X, delimiter=',', fmt='%d')
#np.savetxt(fname='labels.csv', X=y, delimiter=',', fmt='%d')
```

In [ ]:
```

```

In [52]:
```
from sklearn.cluster import MiniBatchKMeans
```

In [53]:
```
kmeans=MiniBatchKMeans(n_clusters = 10)
```

In [54]:
```
kmeans.fit(X)
```

Out[54]:
```
MiniBatchKMeans(batch_size=100, compute_labels=True, init='k-means++',
                init_size=None, max_iter=100, max_no_improvement=10,
                n_clusters=10, n_init=3, random_state=None,
                reassignment_ratio=0.01, tol=0.0, verbose=0)
```

In [55]:
```
len(kmeans.labels_)
```

Out[55]:
```
60000
```

In [56]:
```
kmeans.predict(X[1:10])
```

Out[56]:
```
array([7, 3, 6, 0, 5, 6, 2, 6, 4])
```

In [57]:
```
y[1:10]
```

Out[57]:
```
array([0, 4, 1, 9, 2, 1, 3, 1, 4], dtype=uint8)
```

In [ ]:
```

```

In [58]:
```
def infer_cluster_labels(kmeans, actual_labels):
    """
    Associates most probable label with each cluster in KMeans model
    returns: dictionary of clusters assigned to each label
    """

    inferred_labels = {}

    for i in range(kmeans.n_clusters):

        # find index of points in cluster
        labels = []
        index = np.where(kmeans.labels_ == i)

        # append actual labels for each point in cluster
        labels.append(actual_labels[index])

        # determine most common label
        if len(labels[0]) == 1:
            counts = np.bincount(labels[0])
        else:
            counts = np.bincount(np.squeeze(labels))

        # assign the cluster to a value in the inferred_labels dictionary
        if np.argmax(counts) in inferred_labels:
            # append the new number to the existing array at this slot
            inferred_labels[np.argmax(counts)].append(i)
        else:
            # create a new array in this slot
            inferred_labels[np.argmax(counts)] = [i]

        #print(labels)
        #print('Cluster: {}, label: {}'.format(i, np.argmax(counts)))

    return inferred_labels
```

In [59]:
```
def infer_data_labels(X_labels, cluster_labels):
    """
    Determines label for each array, depending on the cluster it has been assi
gned to.
    returns: predicted labels for each array
    """

    # empty array of len(X)
    predicted_labels = np.zeros(len(X_labels)).astype(np.uint8)

    for i, cluster in enumerate(X_labels):
        for key, value in cluster_labels.items():
            if cluster in value:
                predicted_labels[i] = key

    return predicted_labels
```

In [60]:
```
cluster_labels = infer_cluster_labels(kmeans, y)
```

In [61]:
```
X_clusters = kmeans.predict(X)
```

In [62]:
```
predicted_labels = infer_data_labels(X_clusters, cluster_labels)
```

In [63]:
```
print(predicted_labels[:20])
print(y[:20])
```

```
[3 0 4 1 7 2 1 8 1 4 3 1 3 6 1 4 2 8 6 4]
[5 0 4 1 9 2 1 3 1 4 3 5 3 6 1 7 2 8 6 9]
```

In [64]:
```
from sklearn import metrics

def calculate_metrics(estimator, data, labels):

    # Calculate and print metrics
    print('Number of Clusters: {}'.format(estimator.n_clusters))
    print('Inertia: {}'.format(estimator.inertia_))
    print('Homogeneity: {}'.format(metrics.homogeneity_score(labels, estimator
.labels_)))
```

In [65]:
```
clusters = [10, 16, 36, 64, 144, 256]

# test different numbers of clusters
for n_clusters in clusters:
    estimator = MiniBatchKMeans(n_clusters = n_clusters)
    estimator.fit(X)

    # print cluster metrics
    calculate_metrics(estimator, X, y)

    # determine predicted labels
    cluster_labels = infer_cluster_labels(estimator, y)
    predicted_y = infer_data_labels(estimator.labels_, cluster_labels)

    # calculate and print accuracy
    print('Accuracy: {}\n'.format(metrics.accuracy_score(y, predicted_y)))
```

```
Number of Clusters: 10
Inertia: 2384683.0282846894
Homogeneity: 0.4356882151121713
Accuracy: 0.54235

Number of Clusters: 16
Inertia: 2208670.7339694733
Homogeneity: 0.5698386688772752
Accuracy: 0.6547833333333334

Number of Clusters: 36
Inertia: 1963566.061065766
Homogeneity: 0.6680641791886999
Accuracy: 0.7564833333333333

Number of Clusters: 64
Inertia: 1816581.500515221
Homogeneity: 0.7500034344459123
Accuracy: 0.8331333333333333

Number of Clusters: 144
Inertia: 1632377.184340191
Homogeneity: 0.8011783262201534
Accuracy: 0.8615

Number of Clusters: 256
Inertia: 1516984.5593761946
Homogeneity: 0.8392151122141166
Accuracy: 0.89225
```

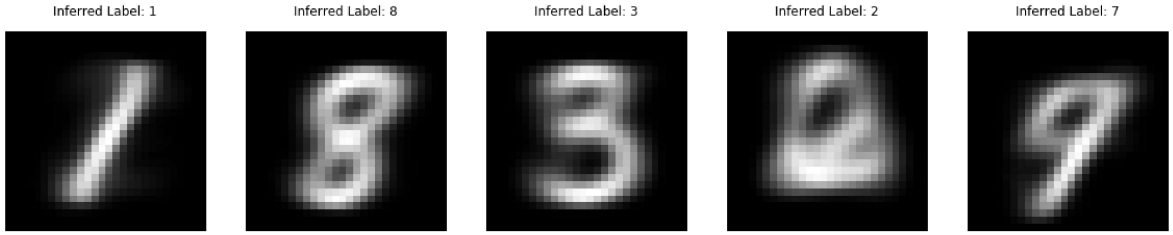
In [66]:
```
X_test,y_test=loadlocal_mnist(images_path='t10k-images.idx3-ubyte',labels_path
='t10k-labels.idx1-ubyte')
```

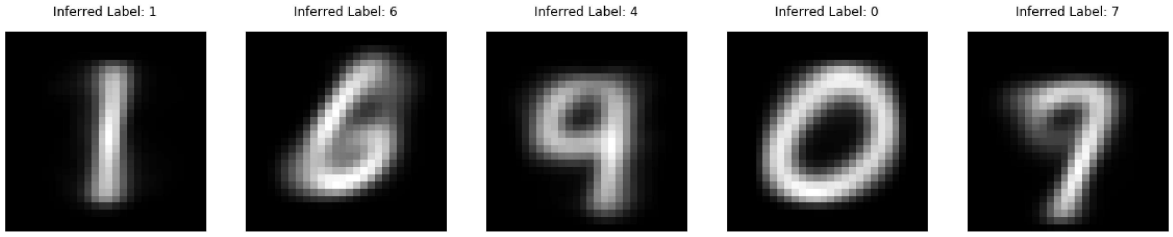
In [67]:
```
kmeans = MiniBatchKMeans(n_clusters = 256)
kmeans.fit(X)
cluster_labels = infer_cluster_labels(kmeans, y)

# predict labels for testing data
test_clusters = kmeans.predict(X_test)
predicted_labels = infer_data_labels(kmeans.predict(X_test), cluster_labels)

# calculate and print accuracy
print('Accuracy: {}\n'.format(metrics.accuracy_score(y_test, predicted_labels
)))
```

```
Accuracy: 0.5076
```

In [69]:
```
# Initialize and fit KMeans algorithm
kmeans = MiniBatchKMeans(n_clusters = 10)
kmeans.fit(X)

# record centroid values
centroids = kmeans.cluster_centers_

# reshape centroids into images
images = centroids.reshape(10, 28, 28)
images *= 255
images = images.astype(np.uint8)

# determine cluster labels
cluster_labels = infer_cluster_labels(kmeans, y)

# create figure with subplots using matplotlib.pyplot
fig, axs = plt.subplots(2, 5, figsize = (20, 20))
plt.gray()

# loop through subplots and add centroid images
for i, ax in enumerate(axs.flat):

    # determine inferred label using cluster_labels dictionary
    for key, value in cluster_labels.items():
        if i in value:
            ax.set_title('Inferred Label: {}'.format(key))

    # add image to subplot
    ax.matshow(images[i])
    ax.axis('off')

# display the figure
fig.show()
```

```
C:\Users\Sonal\Anaconda3\lib\site-packages\matplotlib\figure.py:459: UserWarn
ing: matplotlib is currently using a non-GUI backend, so cannot show the figu
re
  "matplotlib is currently using a non-GUI backend, "
```





In [ ]:

In [ ]:

Case Study:

Documents to be attached:

- Description of the case study
- Data visualization techniques used and justification
- Pre-processing techniques used and justification
- Algorithm to build the model
- Evaluation of the model
- Conclusion of the Experiment wrt dataset.

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
Case Study							

Case Study

Title : Detection of Phishing Websites

1.Description of the case study

A phishing website (or a "spoofed" site) tries to steal victims' account password or other confidential information by tricking the victim into believing that he is on a legitimate website. Hence Identifying whether a given website is Legitimate or a Phishing website helps prevent attacks.
This Problem Statement falls under Supervised Learning where we collected the data from sites listing phishing and legitimate websites.

This is a Binary Classification problem[0-Legitimate 1-Phishing]

Dataset sources-
1.https://www.hk.abchina.com/en/ebanking_4916/corporatebanking/201302/t20130204_316055.htm
2.https://raw.githubusercontent.com/mitchellkrogza/Phishing.Database/master/phishing-links-ACTIVE.txt
3.https://www.kaggle.com/taruntiwarihp/phishing-site-urls

2.Data visualization techniques used and justification

```
http_tokens                0.042659  0.019396     0.021286
label                      0.414884  0.073926     0.037264
statistical_report         1.000000 -0.081985    -0.045545
tiny_url                  -0.081985  1.000000    -0.029528
web_traffic               -0.045545 -0.029528     1.000000
```

Out[17]:

	Having_@_symbol	Having_IP	Prefix_suffix_separation	Redirection_//_symbol	Sub_domains	URL_Length	age_domain	dn
Having_@_symbol	1	-0.00884728	-0.0355011	-0.00884728	0.0536659	0.11468	-0.0350811	0.(
Having_IP	-0.00884728	1	-0.0352079	-0.00649351	0.0628867	-0.0181059	0.0365011	0
Prefix_suffix_separation	-0.0355011	-0.0352079	1	-0.0183125	0.15222	0.140197	0.0628712	
Redirection_//_symbol	-0.00884728	-0.00649351	-0.0183125	1	0.0211119	0.117818	-0.0253797	-0
Sub_domains	0.0536659	0.0628867	0.15222	0.0211119	1	0.274794	0.0361624	
URL_Length	0.11468	-0.0181059	0.140197	0.117818	0.274794	1	0.0542341	
age_domain	-0.0350811	0.0365011	0.0628712	-0.0253797	0.0361624	0.0542341	1	
dns_record	0.00850473	0.0595065	0.315312	-0.0214015	0.105737	0.117847	0.198327	
domain_registration_length	0.00205928	0.00784305	0.0374235	0.0195322	-0.0412673	0.0822264	0.671294	0
http_tokens	-0.00648242	-0.00475781	0.0431737	-0.00475781	0.0300442	0.0612181	0.00268638	0
label	0.110832	0.0813457	0.265201	0.0689479	0.0340643	0.324025	0.0410273	
statistical_report	0.0299141	-0.0472785	0.375889	-0.0188747	0.058812	0.136906	0.0938739	
tiny_url	0.00917418	-0.0208997	-0.0409562	0.00460781	-0.0403898	-0.0268674	-0.0601786	-0
web_traffic	-0.00852283	-0.00625537	-0.000726803	-0.0252668	0.0877377	0.0685669	-0.111361	0

Fig 1: Correlation Matrix

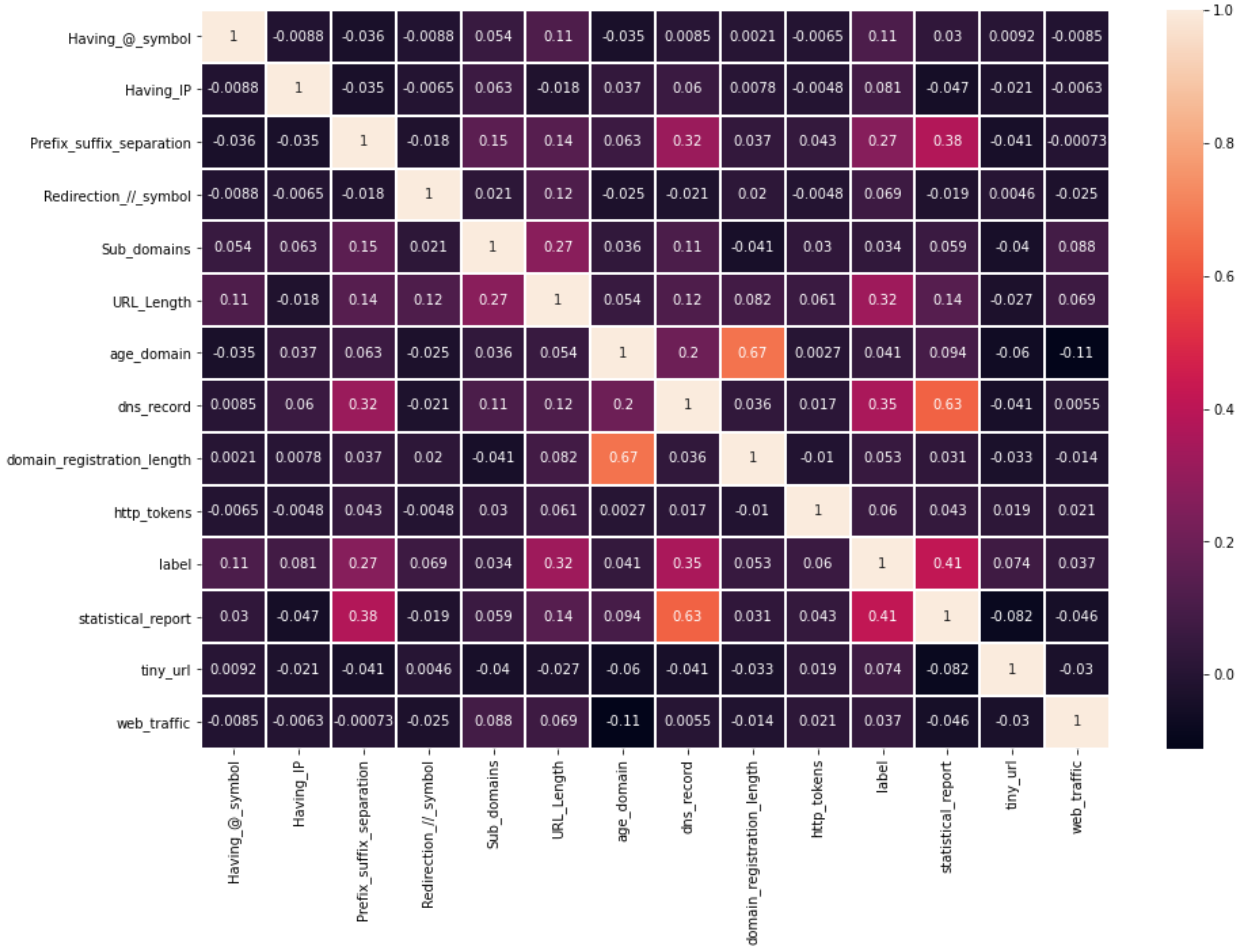


Fig 2: Heatmap of Features

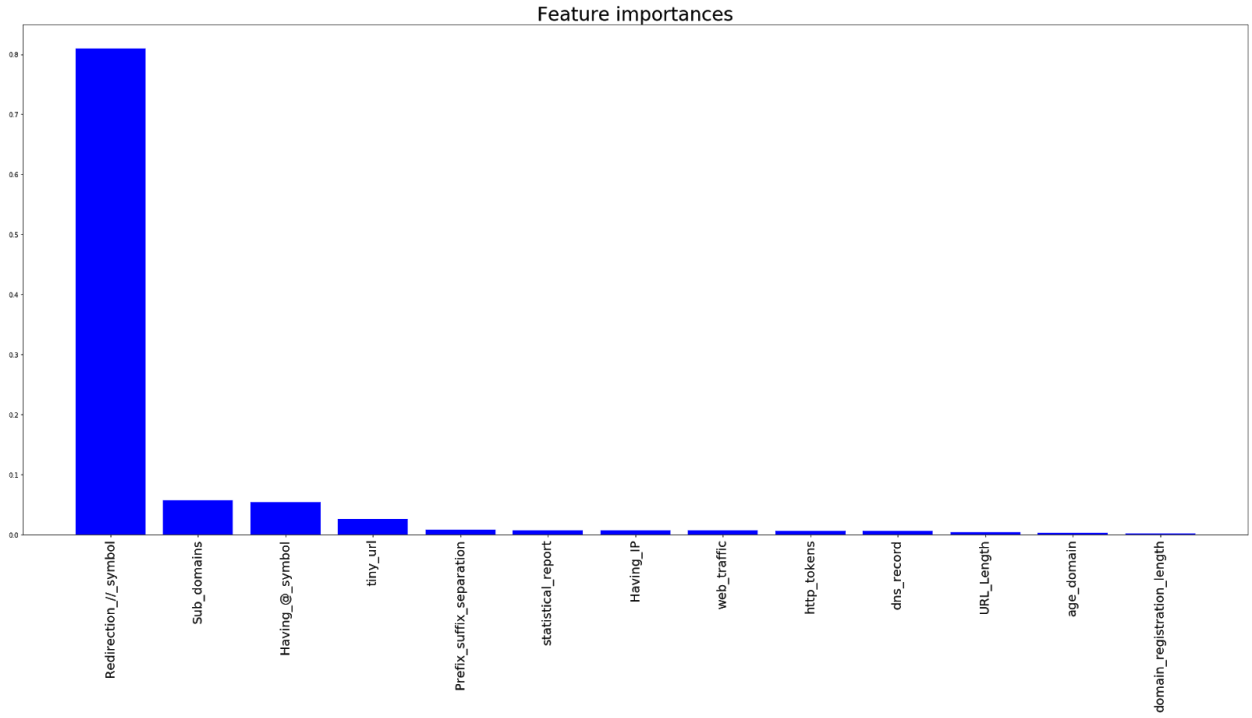


Fig 3: Feature Importance Plot using Decision Tree Classifier Algorithm

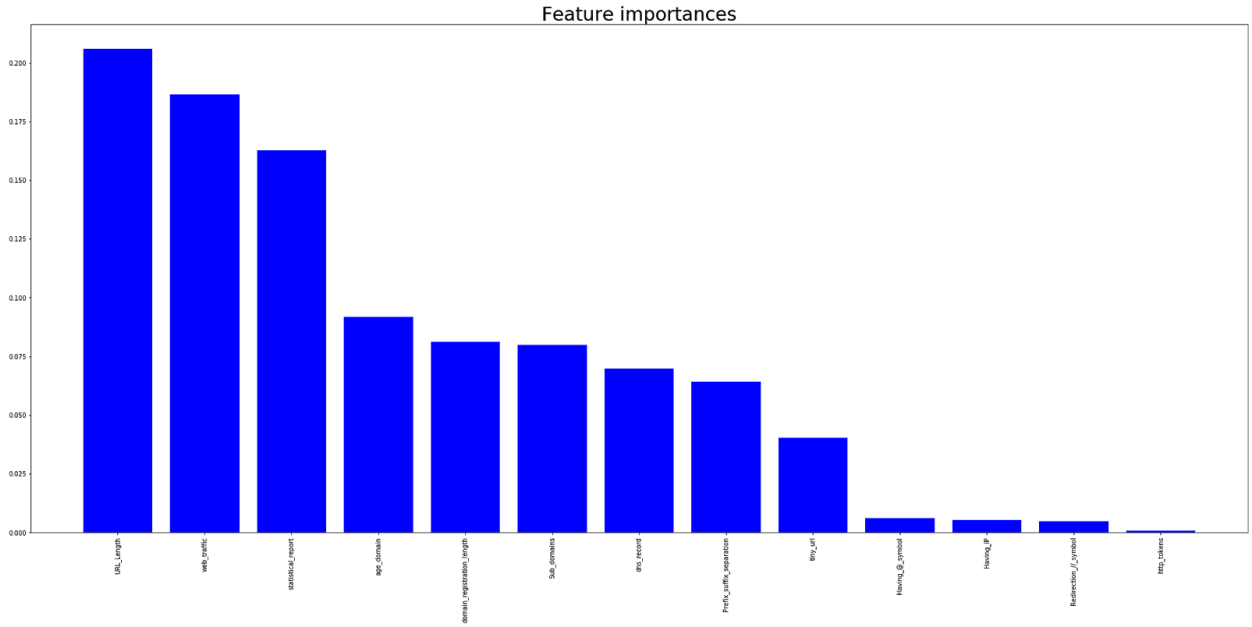
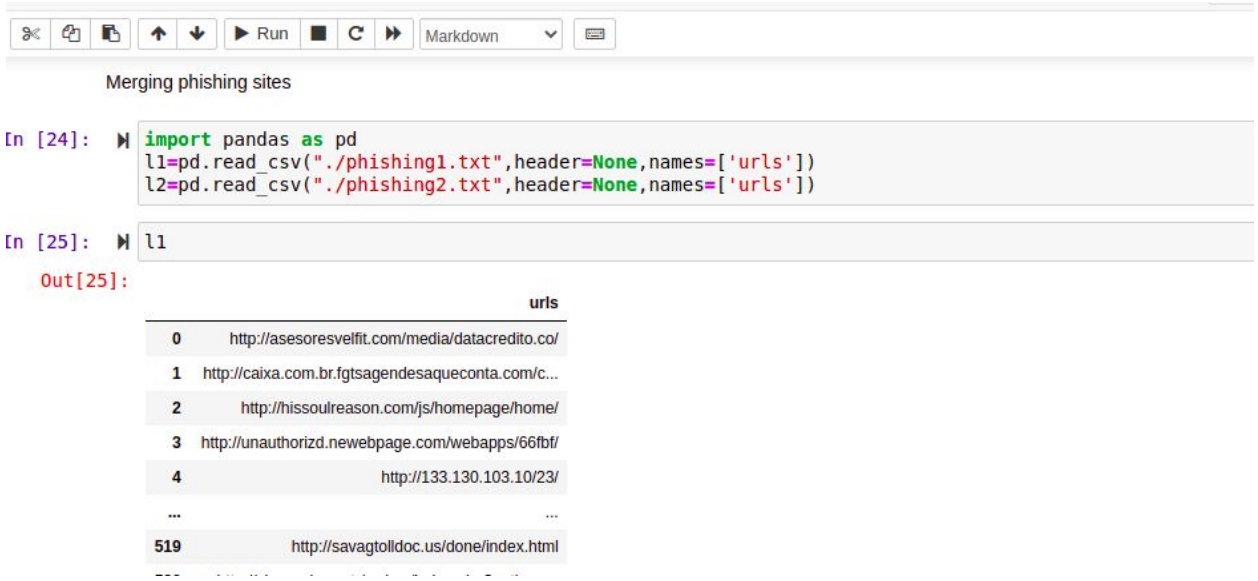


Fig 4: Feature Importance Plot using Random Forest Classifier Algorithm

3.Pre-processing techniques used and justification



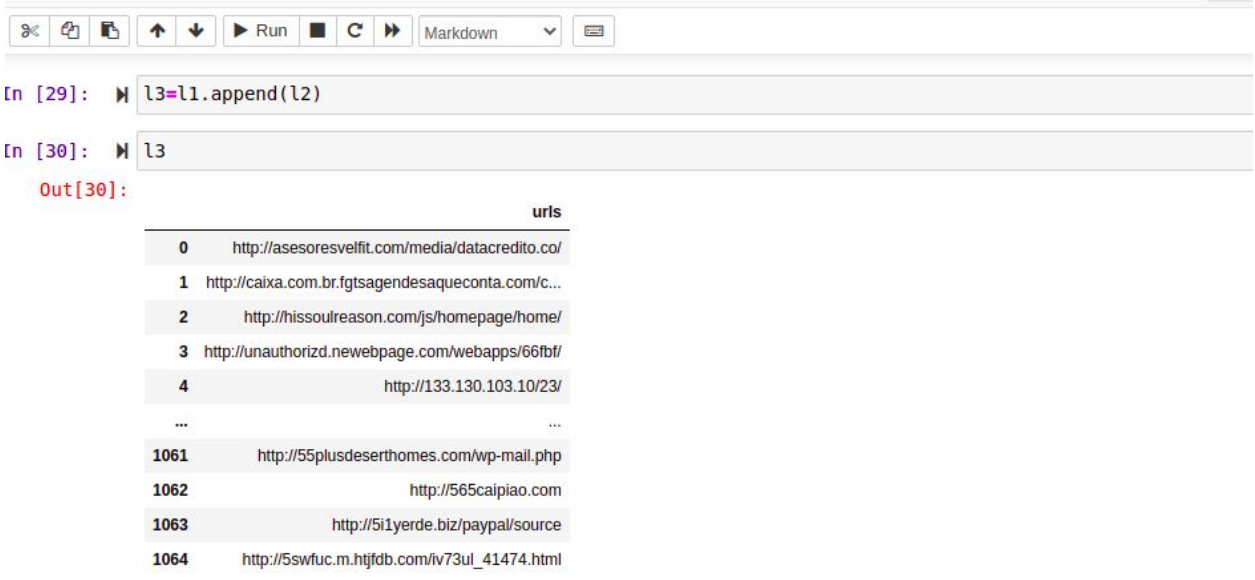
Merging phishing sites

```
In [24]: import pandas as pd
         l1=pd.read_csv("./phishing1.txt",header=None,names=['urls'])
         l2=pd.read_csv("./phishing2.txt",header=None,names=['urls'])

In [25]: l1
```

Out[25]:

	urls
0	http://asesoresvelfit.com/media/datacredito.co/
1	http://caixa.com.br.fgtsagendesaqueconta.com/c...
2	http://hissoulreason.com/js/homepage/home/
3	http://unauthorizd.newebpage.com/webapps/66fbf/
4	http://133.130.103.10/23/
...	...
519	http://savagtolldoc.us/done/index.html

Fig5: Merging



```
In [29]: l3=l1.append(l2)

In [30]: l3
```

Out[30]:

	urls
0	http://asesoresvelfit.com/media/datacredito.co/
1	http://caixa.com.br.fgtsagendesaqueconta.com/c...
2	http://hissoulreason.com/js/homepage/home/
3	http://unauthorizd.newebpage.com/webapps/66fbf/
4	http://133.130.103.10/23/
...	...
1061	http://55plusdeserthomes.com/wp-mail.php
1062	http://565caipiao.com
1063	http://5i1yerde.biz/paypal/source
1064	http://5swfuc.m.htjfdb.com/iv73ul_41474.html

Fig6: Merging

```
In [ ]:
# object creation
fe = FeatureExtraction()
rows = len(df["urls"])

for i in range(0,rows):
    url=df["urls"][i]
    #print(i ),print(url)
    protocol.append(fe.getProtocol(url))
    path.append(fe.getPath(url))
    domain.append(fe.getDomain(url))
    having_ip.append(fe.havingIP(url))
    len_url.append(fe.long_url(url))
    having_at_symbol.append(fe.have_at_symbol(url))
    redirection_symbol.append(fe.redirection(url))
    prefix_suffix_separation.append(fe.prefix_suffix_separation(url))
    sub_domains.append(fe.sub_domains(url))
    tiny_url.append(fe.shortening_service(url))
    web_traffic.append(fe.web_traffic(url))
    domain_registration_length.append(fe.domain_registration_length(url))
    dns_record.append(fe.dns_record(url))
    statistical_report.append(fe.statistical_report(url))
    age_domain.append(fe.age_domain(url))
    http_tokens.append(fe.https_token(url))
    #google_index.append(fe.google_index(url))
    #abnormal_url.append(fe.abnormal_url(url))
```

Fig7: Feature Extraction

```
In [ ]:
'Protocol':pd.Series(protocol),'Domain':pd.Series(domain),'Path':pd.Series(path),'Having_IP':pd.Series(having_ip
'URL_Length':pd.Series(len_url),'Having_@_symbol':pd.Series(having_at_symbol),
'Redirection_//_symbol':pd.Series(redirection_symbol),'Prefix_suffix_separation':pd.Series(prefix_suffix_separat
'Sub_domains':pd.Series(sub_domains),'tiny_url':pd.Series(tiny_url),'web_traffic' : pd.Series(web_traffic) ,
'domain_registration_length':pd.Series(domain_registration_length),'dns_record':pd.Series(dns_record),
'statistical_report':pd.Series(statistical_report),'age_domain':pd.Series(age_domain),'http_tokens':pd.Series(ht
'label':pd.Series(label)}
a=pd.DataFrame(d)
a

In [ ]:
#data.to_csv("legitimate-urls.csv",index=False,encoding='UTF-8')

In [ ]:
data.to_csv("phishing-urls.csv",index=False,encoding='UTF-8')
```

Fig8: Features

Data Reduction

Removing Unnecessary columns

In [14]:
```
urls = urls.drop(urls.columns[[0,3,5]],axis=1) # Path,Domain,Protocol
```

In [15]:
```
urls.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2015 entries, 0 to 997
Data columns (total 14 columns):
Having_@_symbol               2015 non-null int64
Having_IP                     2015 non-null int64
Prefix_suffix_separation      2015 non-null int64
Redirection_//_symbol         2015 non-null int64
Sub_domains                   2015 non-null int64
URL_Length                    2015 non-null int64
age_domain                    2015 non-null int64
dns_record                    2015 non-null int64
domain_registration_length    2015 non-null int64
http_tokens                   2015 non-null int64
label                         2015 non-null int64
statistical_report            2015 non-null int64
tiny_url                      2015 non-null int64
web_traffic                   2015 non-null int64
dtypes: int64(14)
memory usage: 236.1 KB
```

Fig9: Data Reduction

Data Shuffling

Since we merged two dataframes top 1000 rows will have legitimate urls and bottom 1000 rows will have phishing urls. So if we split the data now and create a model for it will overfit so we need to shuffle the rows before splitting the data into training set and test set

In [16]:
```
# shuffling the rows in the dataset so that when splitting the train and test set are equally distributed
urls = urls.sample(frac=1).reset_index(drop=True)
```

Fig10: Data Shuffling

Removing class variable from the dataset

In [19]:

```
urls_without_labels = urls.drop('label',axis=1)
print(urls_without_labels.info())
labels = urls['label']
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2015 entries, 0 to 2014
Data columns (total 13 columns):
Having_@_symbol               2015 non-null int64
Having_IP                     2015 non-null int64
Prefix_suffix_separation      2015 non-null int64
Redirection_//_symbol         2015 non-null int64
Sub_domains                   2015 non-null int64
URL_Length                    2015 non-null int64
age_domain                    2015 non-null int64
dns_record                    2015 non-null int64
domain_registration_length    2015 non-null int64
http_tokens                   2015 non-null int64
statistical_report            2015 non-null int64
tiny_url                      2015 non-null int64
web_traffic                   2015 non-null int64
dtypes: int64(13)
memory usage: 204.8 KB
None
```

Fig11: Data Reduction

4.Algorithm to build the model

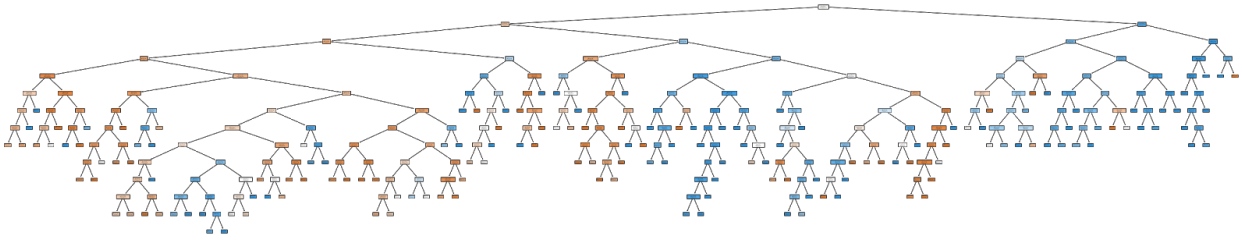


Fig12 : Tree Visualisation using Pyplot

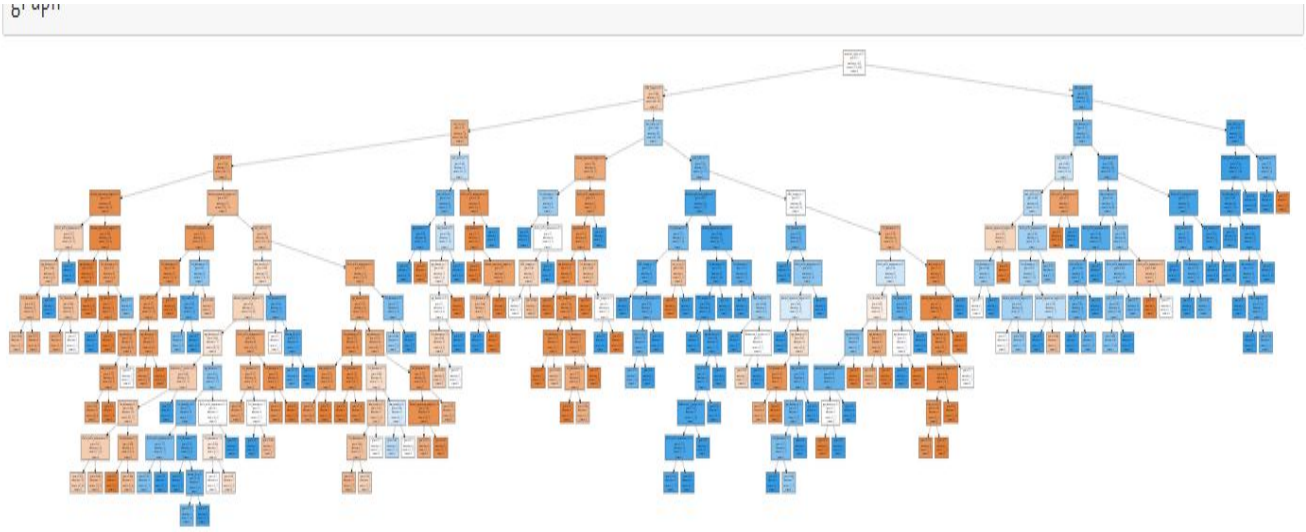


Fig13 : Tree Visualisation using graphviz

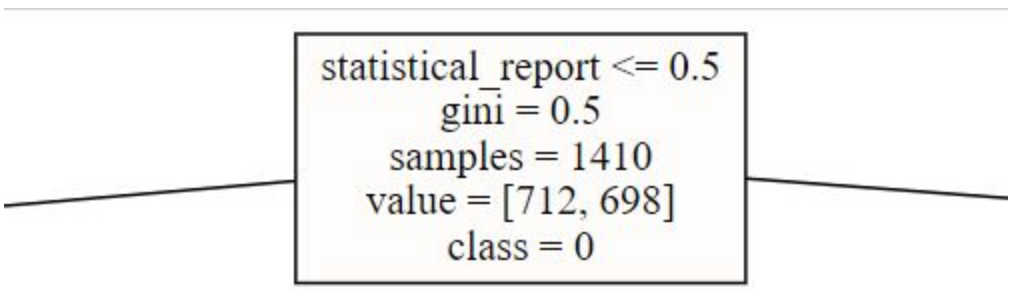


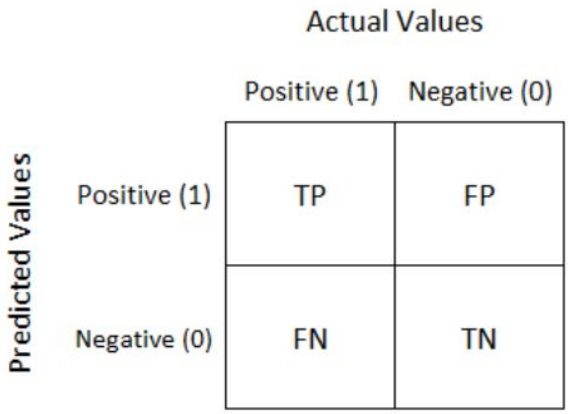
Fig14 : Tree Visualisation using Sklearn Tree

```
       |--- feature_10 <= 0.50
|   |--- feature_5 <= 0.50
|   |   |--- feature_11 <= 0.50
|   |   |   |--- feature_12 <= 0.50
|   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |--- feature_2 <= 0.50
|   |   |   |   |   |   |--- feature_6 <= 0.50
|   |   |   |   |   |   |   |--- feature_4 <= 1.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- feature_4 >  1.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_6 >  0.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_2 >  0.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_8 >  0.50
|   |   |   |   |   |--- feature_8 <= 1.50
|   |   |   |   |   |   |--- feature_6 <= 0.50
|   |   |   |   |   |   |   |--- feature_4 <= 1.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- feature_4 >  1.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_6 >  0.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- feature_8 >  1.50
|   |   |   |   |   |   |--- feature_4 <= 0.50
|   |   |   |   |   |   |   |--- feature_6 <= 1.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- feature_6 >  1.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_4 >  0.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- feature_12 >  0.50
|   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |--- feature_2 <= 0.50
|   |   |   |   |   |   |--- feature_4 <= 0.50
|   |   |   |   |   |   |   |--- feature_12 <= 1.50
|   |   |   |   |   |   |   |   |--- feature_6 <= 1.00
```

5. Evaluation of model

Evaluation of Model

1)confusion matrix ¶




```
from sklearn.metrics import confusion_matrix,accuracy_score
cm_gini = confusion_matrix(labels_test,pred_label_gini)
cm_gini
```

```
array([[270,  35],
       [ 71, 229]], dtype=int64)
```

```
cm_ent = confusion_matrix(labels_test,pred_label_ent)
cm_ent
```

```
array([[271,  34],
       [ 68, 232]], dtype=int64)
```

Fig15 :Confusion Matrix of Decision Tree Classifier

2)Accuracy score ¶

```
accuracy_score(labels_test,pred_label_gini)
```

```
0.824793388429752
```

```
accuracy_score(labels_test,pred_label_ent)
```

```
0.8314049586776859
```

3)Cross Validation Score

```
from sklearn.model_selection import cross_val_score
scores_gini = cross_val_score(model_gini,urls_without_labels, labels, cv=5)
scores_gini.mean()
```

```
0.8287841191066997
```

```
scores_ent = cross_val_score(model_ent,urls_without_labels, labels, cv=5)
scores_ent.mean()
```

```
0.8312655086848636
```

Fig16 :Accuracy and Cross validation score of Decision Tree Classifier

```
print(classification_report(labels_test,pred_label_gini))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       305
           1       0.87      0.76      0.81       300

    accuracy                           0.82       605
   macro avg       0.83      0.82      0.82       605
weighted avg       0.83      0.82      0.82       605
```

```
print(classification_report(labels_test,pred_label_ent))
```

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       305
           1       0.87      0.77      0.82       300

    accuracy                           0.83       605
   macro avg       0.84      0.83      0.83       605
weighted avg       0.84      0.83      0.83       605
```

Fig17 :Classification report of Decision Tree Classifier

Evaluation

```
from sklearn.metrics import confusion_matrix,accuracy_score
cpnfusionMatrix = confusion_matrix(labels_test,prediction_label)
print(cpnfusionMatrix)
accuracy_score(labels_test,prediction_label)
```

```
[[269  36]
 [ 63 237]]
```

```
0.8363636363636363
```

Cross validation Score

```
from sklearn.model_selection import cross_val_score
sc = cross_val_score(random_forest_classifier,urls_without_labels, labels, cv=5)
sc.mean()
```

```
0.8387096774193548
```

```
print(classification_report(labels_test,prediction_label))
```

```
              precision    recall  f1-score   support

           0       0.81      0.88      0.84       305
           1       0.87      0.79      0.83       300

    accuracy                           0.84       605
   macro avg       0.84      0.84      0.84       605
weighted avg       0.84      0.84      0.84       605
```

Fig18 :Confusion Matrix ,Cross Validation Score and Classification report of Random Forest Classifier

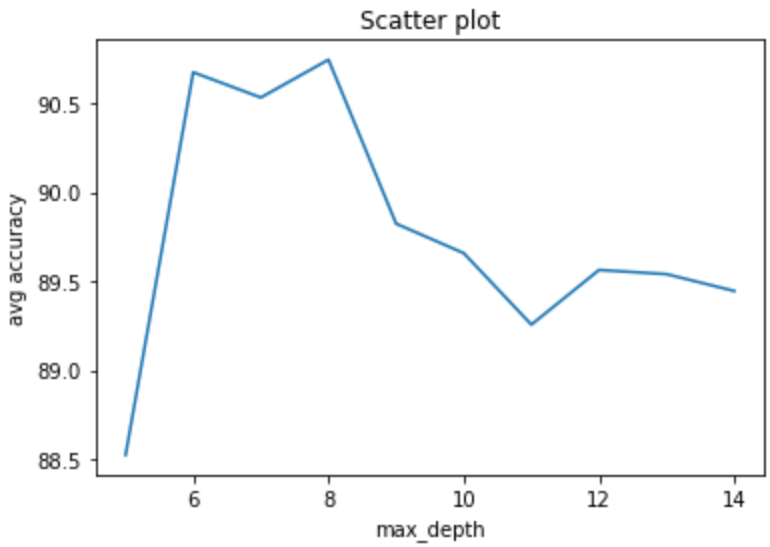


Fig19 : Scatter Plot of Avg Accuracy Vs Max_depth using Decision Tree Classifier

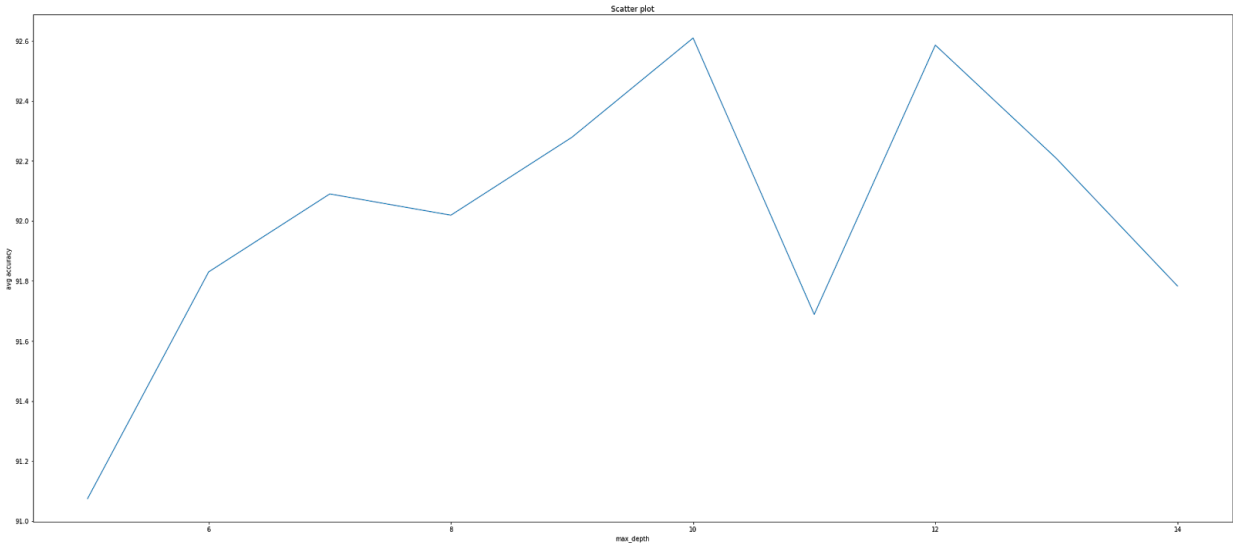


Fig20 : Scatter Plot of Avg Accuracy Vs Max_depth using Random Forest Classifier

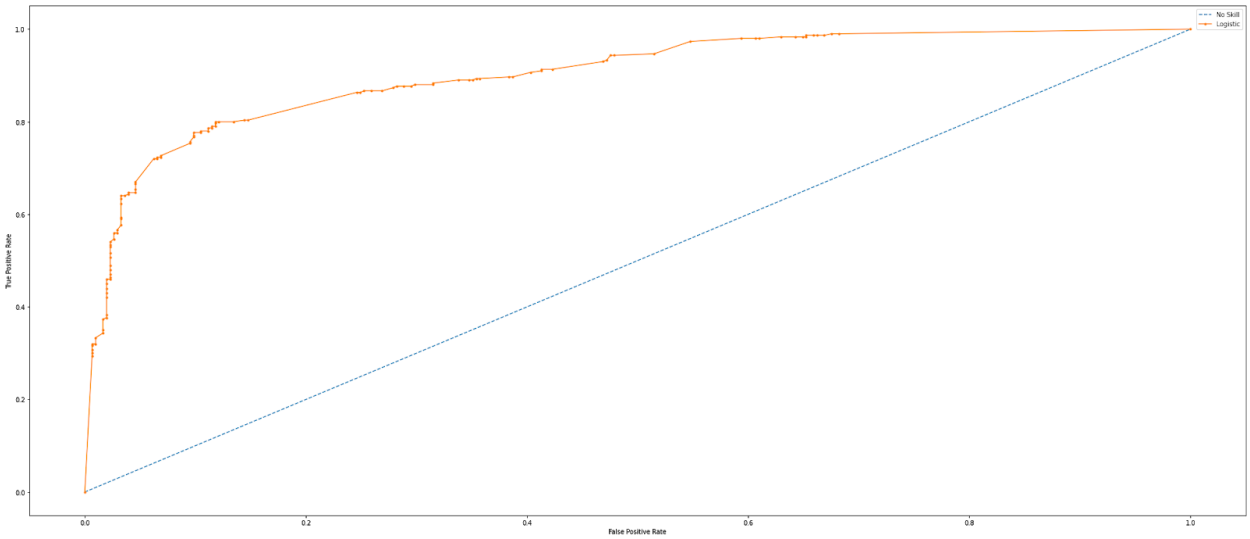


Fig21 : ROC Curve of Random Forest Classifier

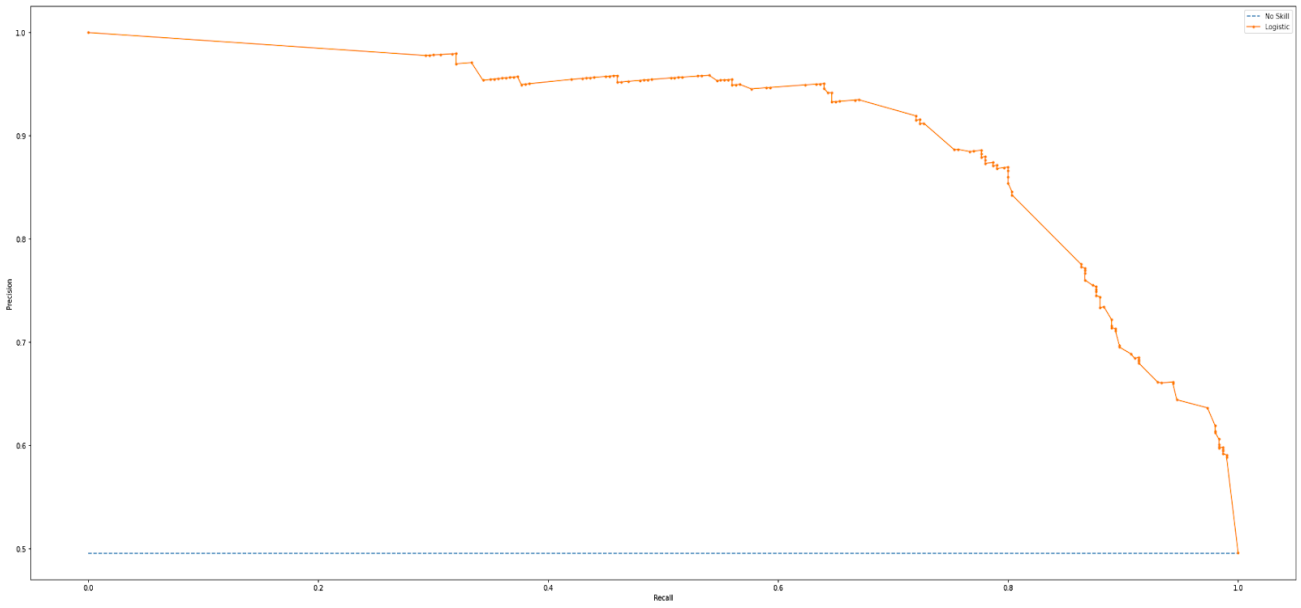


Fig22 : PR Curve of Random Forest Classifier

6. Conclusion

Real world data has a lot of noise and inconsistency and also has a lot of features which a model does not require. From this project we get to know the importance of Data preprocessing and the ambiguity in the prediction of the model gets reduced considerably.

Decision Tree Classifier resulted in Average Accuracy score of 82.5%.
After Integrating Feature Importance with the model, avg accuracy score rose to 91% with max_depth=8.

Random Forest Classifier,which is an ensemble model,gave Accuracy score of 83%.
After Integrating Feature Importance with the model, avg Accuracy score rose to 93% with max_depth=10.

Viva - Voce Questions

1. Name Data mining techniques?
2. Name areas of applications of data mining?
3. Name diffrent classification methods?
4. Name different methods in clustering?
5. What are the types of tasks that are carried out during data mining ?
6. What is Data cleaning ?
7. Explain Data reduction and transformation?
8. What is Discrete and Continuous data in Data mining world?\
9. What is Naïve Bayes Algorithm?
10. What is HDFS? Why is it important
11. Describe Hadoop ecosystem
12. How is pig different than hive in Hadoop?
13. Map parallel processing and Hadoop.
14. What are the limitations of R tool?
15. What is the importance of siket and pandas in python?
16. What is the limitation of Navie Bayes theorem ?
17. When to use SVM?
18. Give the application of Apriori algorithm.
19. What is the scope of regression in data mining?
20. What is a Decision Tree Algorithm?