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	**1RV17IS048**
NAME	**SONAL S R**
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 Ms. SONAL S R (USN : 1RV17IS048) 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**.** **(1M)**	Unable to explain the concepts. **(0M)**	CO 1
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

Program 1

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	imdl
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt0114
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt0113

5 rows × 24 columns

In [3]:

```
1 df.shape#size of the file in rows and columns
```

Out[3]:

```
(45466, 24)
```

Type of the attributes

In [4]:

```
1 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': '/7G9915LfUQ2l
VfwMEEhDsn3kT4B.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 br
ings Buzz Lightyear onto the scene. Afraid of losing his place in Andy's hea
rt, Woody plots against Buzz. But when circumstances separate Buzz and Woody
from 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	
freq	45454	29	36573	5000	12	3	

4 rows × 24 columns

In [11]:

```
1 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]:

```
1 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]:

```
1 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['budget']!=(
```

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_profi
budget	1.000000	-0.101671	0.768776	0.134733	0.073494	0.676642	0.61433
id	-0.101671	1.000000	-0.071263	-0.121399	-0.167573	-0.064903	-0.05395
revenue	0.768776	-0.071263	1.000000	0.103917	0.083868	0.812022	0.97689
runtime	0.134733	-0.121399	0.103917	1.000000	0.158146	0.113539	0.08318
vote_average	0.073494	-0.167573	0.083868	0.158146	1.000000	0.123607	0.07891
vote_count	0.676642	-0.064903	0.812022	0.113539	0.123607	1.000000	0.77575
actual_profit	0.614339	-0.053950	0.976896	0.083189	0.078916	0.775756	1.00000
profit_ratio	-0.002132	-0.006758	0.000794	0.000901	0.005585	0.003602	0.00169

In [24]:

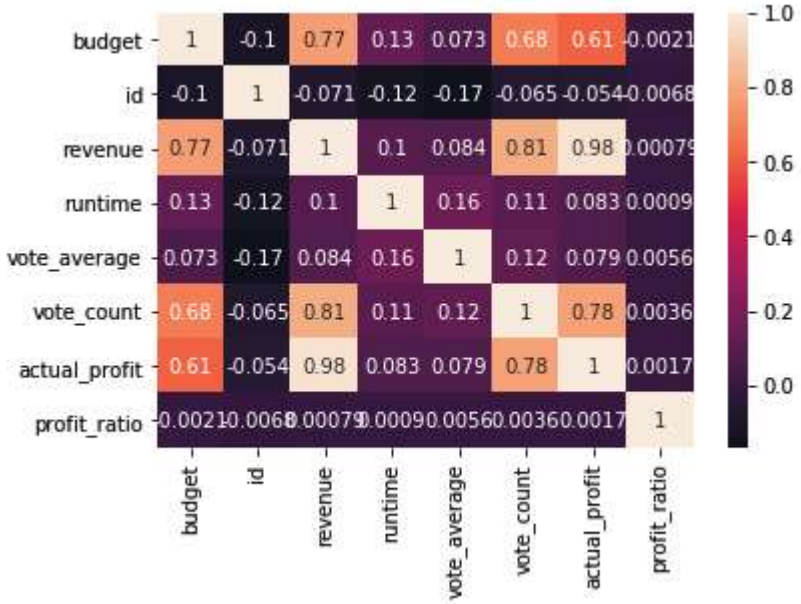
```
#so actual_profit is more corelated than profiy_ratio
```

In [25]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a992488b70>
```



Data Visualization

In [26]:

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

In [27]:

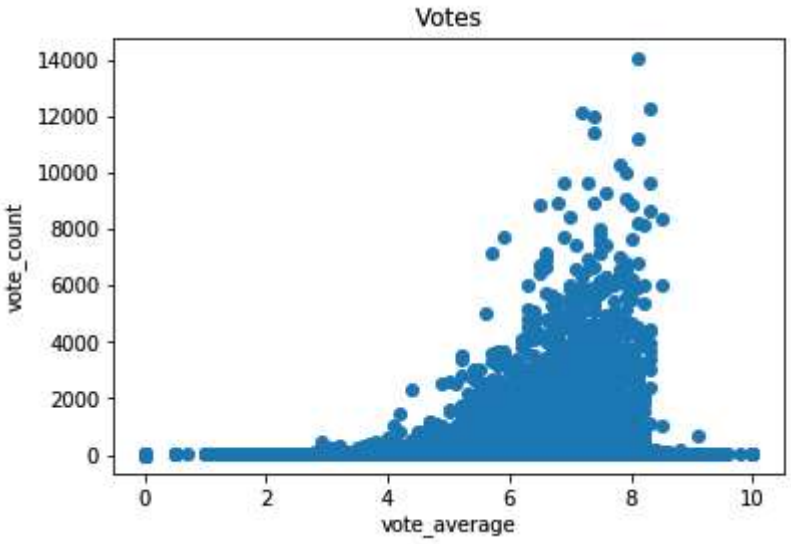
```
x=np.array(df['vote_average'])
y=np.array(df['vote_count'])
```

In [28]:

```
plt.xlabel('vote_average')
plt.ylabel('vote_count')
plt.title('Votes')
plt.scatter(x,y)
```

Out[28]:

```
<matplotlib.collections.PathCollection at 0x1a992687128>
```

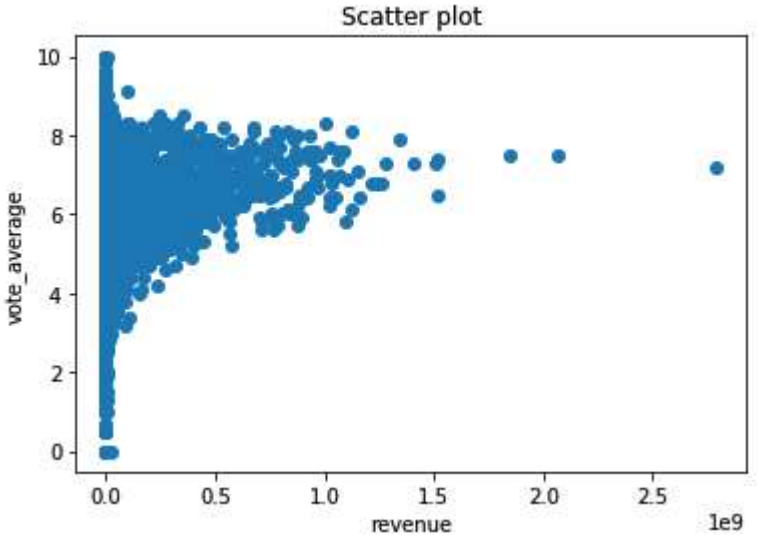


In [29]:

```
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 0x1a9927172e8>
```

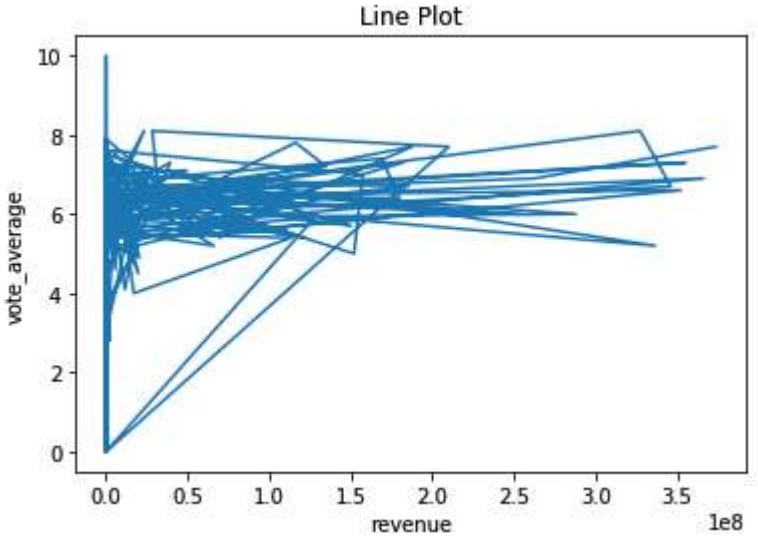


In [30]:

```
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[30]:

```
[<matplotlib.lines.Line2D at 0x1a99272be80>]
```



In [31]:

```
1 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:

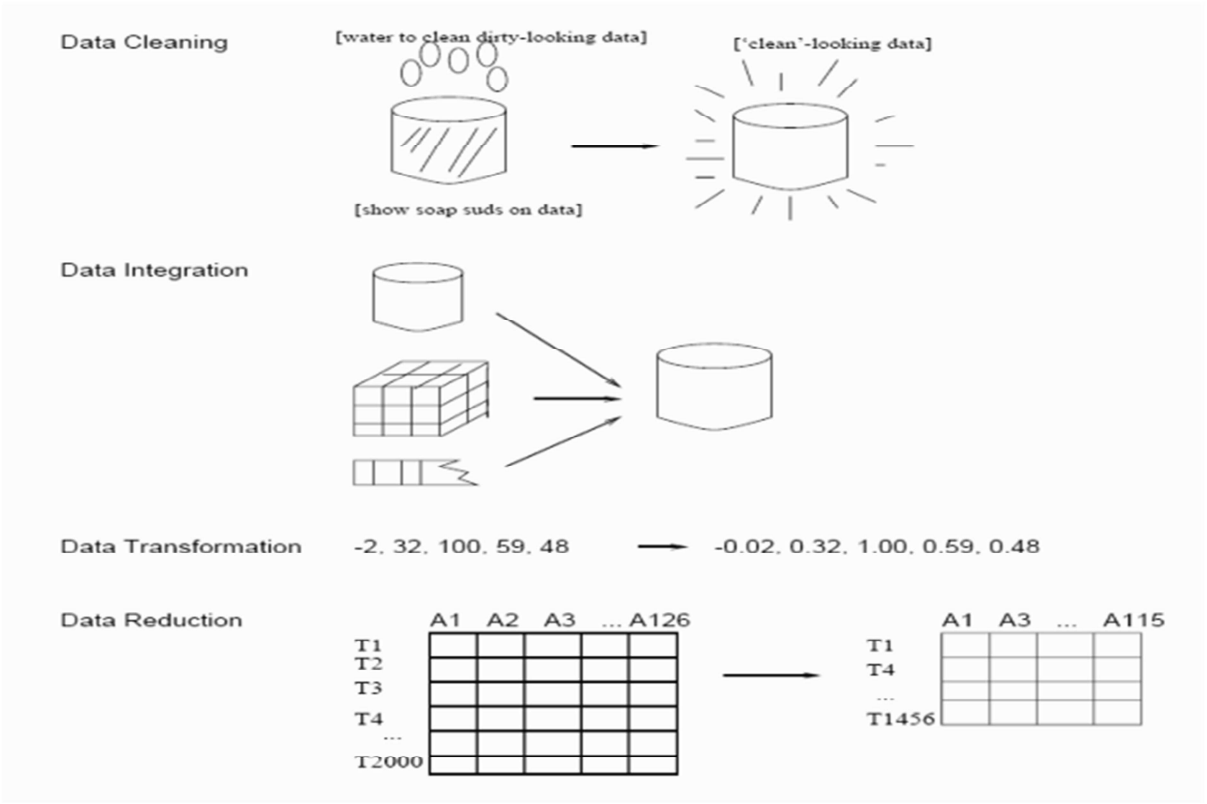
Data Cleaning

- 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

Program 2

In [1]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import os
```

In [2]:

```
os.stat('P2_mov.csv')
```

Out[2]:

```
os.stat_result(st_mode=33206, st_ino=53480245575045934, st_dev=3357596530, s
t_nlink=1, st_uid=0, st_gid=0, st_size=5698602, st_atime=1606720635, st_mtim
e=1571231928, st_ctime=1603862583)
```

In [3]:

```
df=pd.read_csv("P2_mov.csv")
```

In [4]:

```
df.shape
```

Out[4]:

```
(4803, 20)
```

In [ ]:

```

```

In [5]:

```
1 df.head()
```

Out[5]:

	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 [6]:

```
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 [7]:

```
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 [8]:

```
1 df.describe()
```

Out[8]:

	budget	id	popularity	revenue	runtime	vote_average	v
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	
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	13

In [9]:

```
1 df.columns
```

Out[9]:

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

In [10]:

```
1 df.index
```

Out[10]:

```
RangeIndex(start=0, stop=4803, step=1)
```

DATA Cleaning

Handling missing values

In [11]:

```
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 [12]:

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

In [13]:

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

In [14]:

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

```
homepage    0
runtime    0
```

Data Integration

In [15]:

```
df1=pd.read_csv("res.csv")
```

```
C:\Users\Sonal\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:
2785: DtypeWarning: Columns (11) have mixed types.Specify dtype option on im
port or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [16]:

```
1 df1.head()
```

Out[16]:

	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 [17]:

```
1 df_new=pd.merge(df,df1,on="original_title")
```

In [18]:

```
1 df_new.head()
```

Out[18]:

	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 [19]:

```
1 df=df_new
```

In [20]:

```
1 df.columns
```

Out[20]:

```
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 [21]:

```
1 print(df["runtime_y"].min(),df["runtime_y"].max())
```

```
0.0 877.0
```

In [22]:

```
1 df["runtime_y"]
```

Out[22]:

```
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 [23]:

```
1 df["runtime_y"]=((df["runtime_y"]-df["runtime_y"].min())/(df["runtime_y"].max()-df["ru
```

In [24]:

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

```
0.0 10.0
```

In [25]:

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

```
0 380000000
```

In [26]:

```
df['budget_x']=((df['budget_x']-df['budget_x'].min())/(df['budget_x'].max()-df['budget_
```

In [27]:

```
df['budget_x']
```

Out[27]:

```
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 [28]:

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

Changing data types

In [29]:

```
df['release_date_x'] = pd.to_datetime(df['release_date_x'], format = '%Y-%m-%d', error
type(df['release_date_x'][0])
```

Out[29]:

```
pandas._libs.tslibs.timestamps.Timestamp
```

In [30]:

```
df['release_date_y'] = pd.to_datetime(df['release_date_y'], format = '%Y-%m-%d', error
type(df['release_date_y'][0])
```

Out[30]:

```
pandas._libs.tslibs.timestamps.Timestamp
```

Data Reduction

In [31]:

```
df.columns
```

Out[31]:

```
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 [32]:

```
df.shape[1]
```

Out[32]:

```
47
```

In [33]:

```
1 df.head(2)
```

Out[33]:

	budget_x	genres_x	homepage_x	id_x	keywords	original_lar
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 [34]:

```
1 df.corr()
```

Out[34]:

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

In [35]:

```
df.drop(['id_x','popularity_x','runtime_x','vote_average_x','vote_count_x','actual_pro
         'budget_y','id_y','revenue_y','vote_average_y','runtime_y','production_countr
         'spoken_languages_x','spoken_languages_y','budget_y','original_language_y','r
```

Out[35]:

	budget_x	genres_x	homepage_x	keyw
0	62.368421	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	[{"id": 1 "name": "cu clash"}, {"
1	78.947368	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	[{"id": 270, "na "ocean"}, {"id":
2	64.473684	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	[{"id": 470, "na "spy"}, {"id": "na
3	65.789474	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	[{"id": 849, "na "dc comics"}, 8
4	68.421053	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	[{"id": 818, "na "based on no {"
...	...	...	...	
5235	0.057895	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	source_unknown	[{"id": 5 "name": "u states\u2013m
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, "na "date"}, {"id": "n
5238	0.000000	[]	http://shanghaicalling.com/	

	budget_x	genres_x	homepage_x	keyw
5239	0.000000	[{"id": 99, "name": "Documentary"}]	source_unknown	[{"id": "na "obsession"},

5240 rows × 30 columns

In [36]:

```
1 df.columns
```

Out[36]:

```
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 [37]:

```
1 df.shape[1]
```

Out[37]:

```
47
```

Data Discretization

In [38]:

```
1 df['budget_x'].tail()
```

Out[38]:

```
5235    0.057895
5236    0.002368
5237    0.000000
5238    0.000000
5239    0.000000
Name: budget_x, dtype: float64
```

In [39]:

```
df['budget_range'] = pd.cut(x=df['budget_x'], bins=[-1,10,1000,10000,100000000000], lal
```

In [40]:

```
df['budget_range'].tail()
```

Out[40]:

```
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 [41]:

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

In [42]:

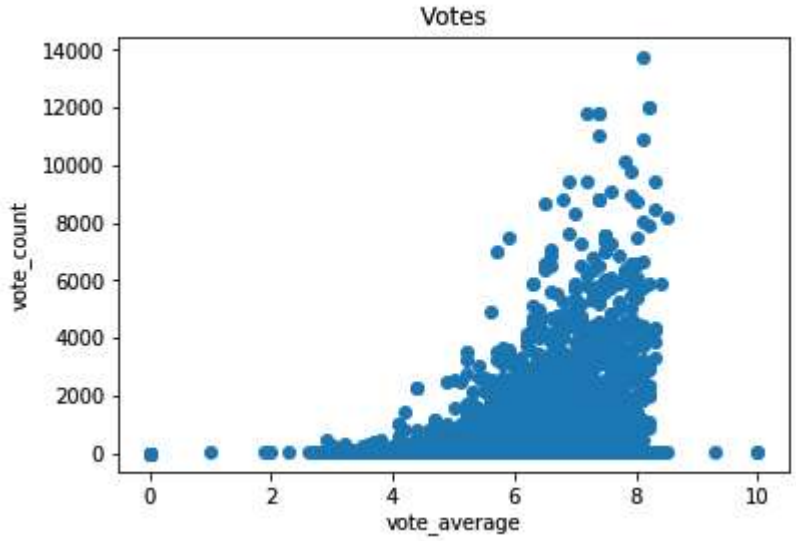
```
x=np.array(df['vote_average_x'])
y=np.array(df['vote_count_x'])
```

In [43]:

```
plt.xlabel('vote_average')
plt.ylabel('vote_count')
plt.title('Votes')
plt.scatter(x,y)
```

Out[43]:

```
<matplotlib.collections.PathCollection at 0x29416bd8be0>
```

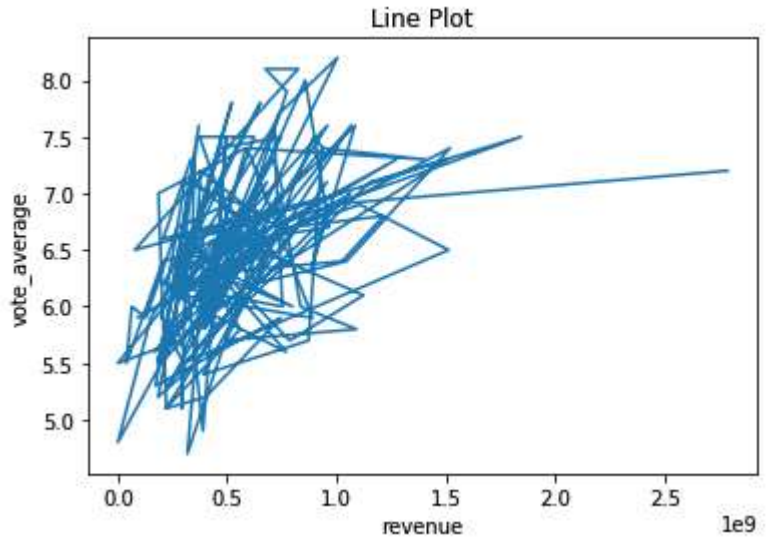


In [44]:

```
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[44]:

```
[<matplotlib.lines.Line2D at 0x29416c3a5f8>]
```



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

Program 3

In [67]:

```
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 [68]:

```
df=pd.read_csv('P3_Weather.csv')
```

```
C:\Users\Sonal\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:
2785: DtypeWarning: Columns (7,8,18,25) have mixed types.Specify dtype optio
n on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [69]:

```
df.head()
```

Out[69]:

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

5 rows × 31 columns

In [70]:

```
1 df.dtypes
```

Out[70]:

```
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 [71]:

```
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 [72]:

```
df.shape
```

Out[72]:

```
(119040, 31)
```

In [73]:

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

In [74]:

```
df.shape
```

Out[74]:

```
(119040, 15)
```

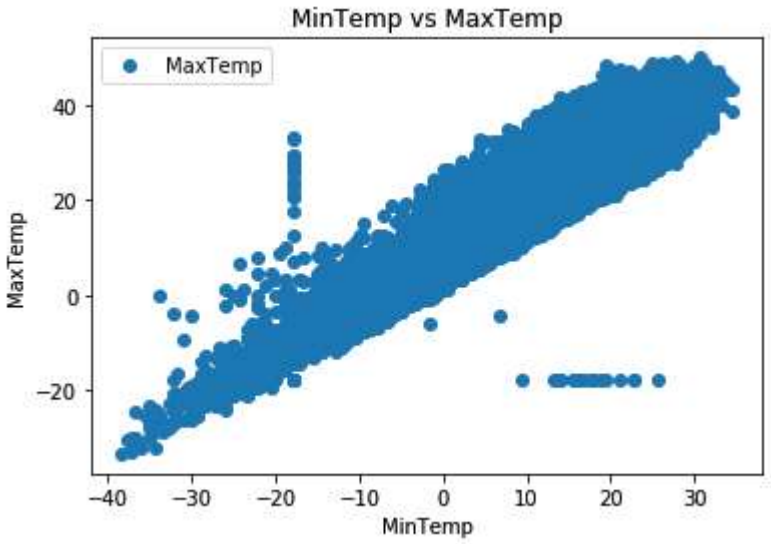
In [75]:

```
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 [76]:

```
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 [77]:

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

In [78]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [79]:

```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[79]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [80]:

```
print(lr.intercept_)
print(lr.coef_)
```

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

In [81]:

```
#Equation will be max_temp=0.92033997* min_temp + 10.66185201
#y=mx+c
```

In [82]:

```
y_pred=lr.predict(X_test)
```

In [83]:

```
y_test,y_pred
```

Out[83]:

```
(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 [84]:

```
lr.predict([[20]])
```

Out[84]:

```
array([[29.06865134]])
```

In [85]:

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

In [86]:

```
m_df
```

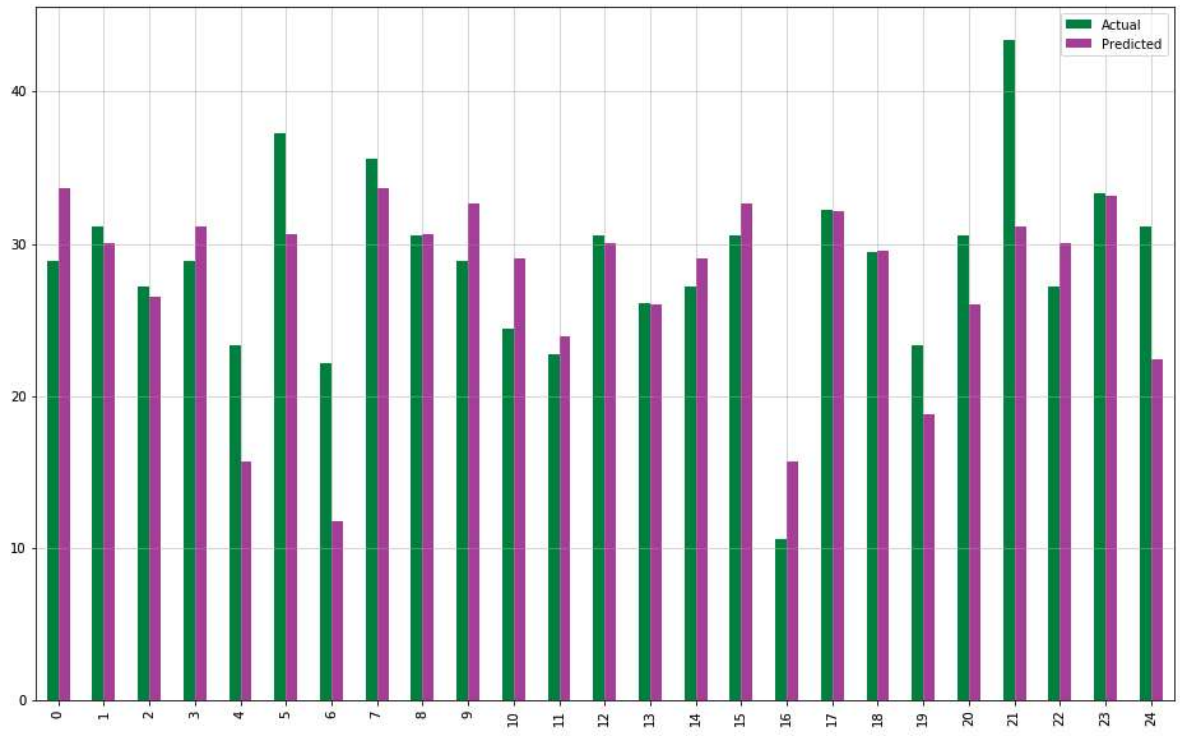
Out[86]:

	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

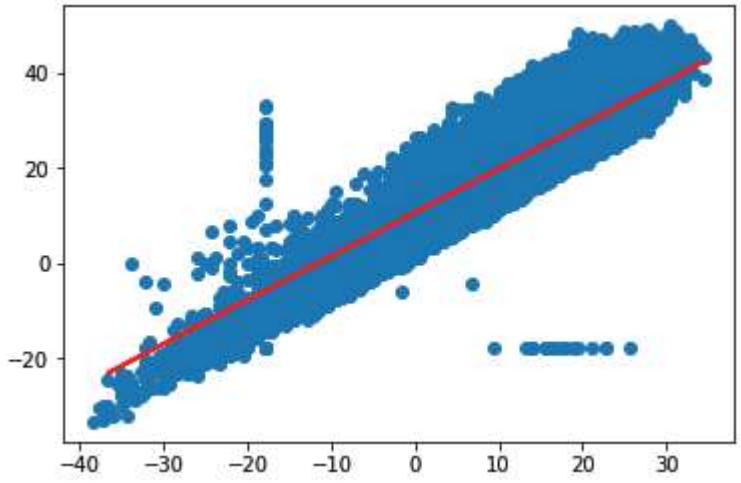
In [87]:

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



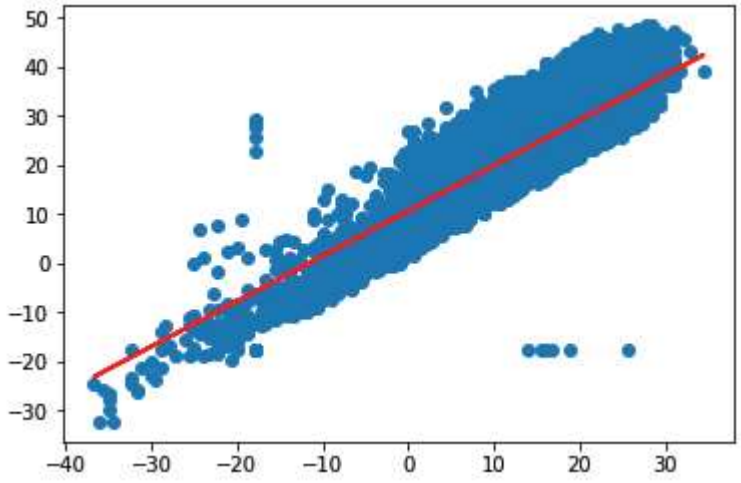
In [88]:

```
plt.scatter(X, y)
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()
```



In [89]:

```
plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()
```



In [ ]:

```

```

In [90]:

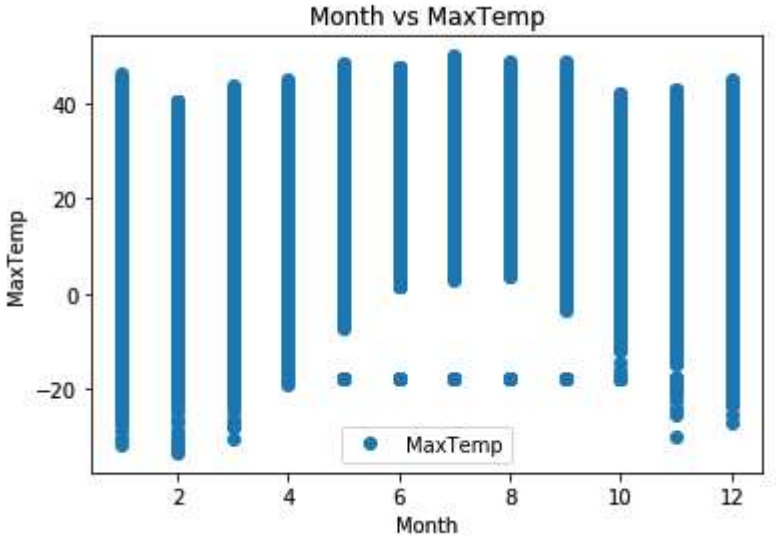
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

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

Multiple Linear Regression

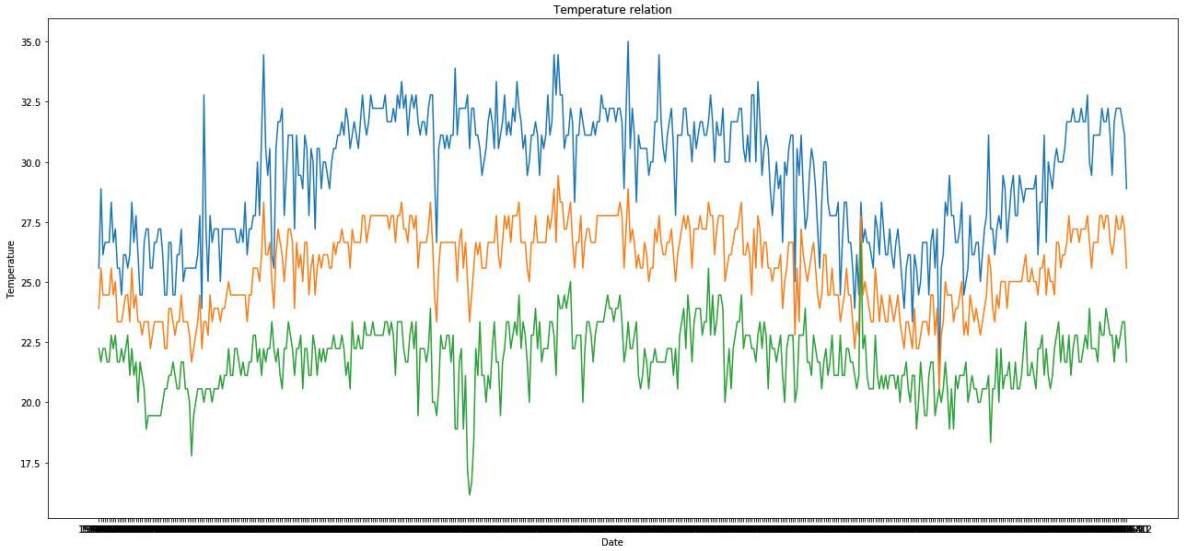
In [91]:

```
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 [92]:

```
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 [93]:

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

In [94]:

```
df['Precip']
```

Out[94]:

```
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 [95]:

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

In [96]:

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

In [97]:

```
nX_train, nX_test, ny_train, ny_test = train_test_split(n_X, n_y, test_size=0.2, rando
```

In [98]:

```
n_lr = LinearRegression()
n_lr.fit(nX_train, ny_train)
```

Out[98]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [99]:

```
print(n_lr.intercept_)
print(n_lr.coef_)
```

```
0.9008561086174041
[-0.86378141 -0.00216864  1.85445532 -0.00508113]
```

In [100]:

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

Out[100]:

```
array([29.95665841])
```

In [101]:

```
#y=intercept+a*x1+b*x2+c*x3+d*x4
```

In [102]:

```
ny_pred=n_lr.predict(nX_test)
```

In [103]:

```
ny_pred,ny_test
```

Out[103]:

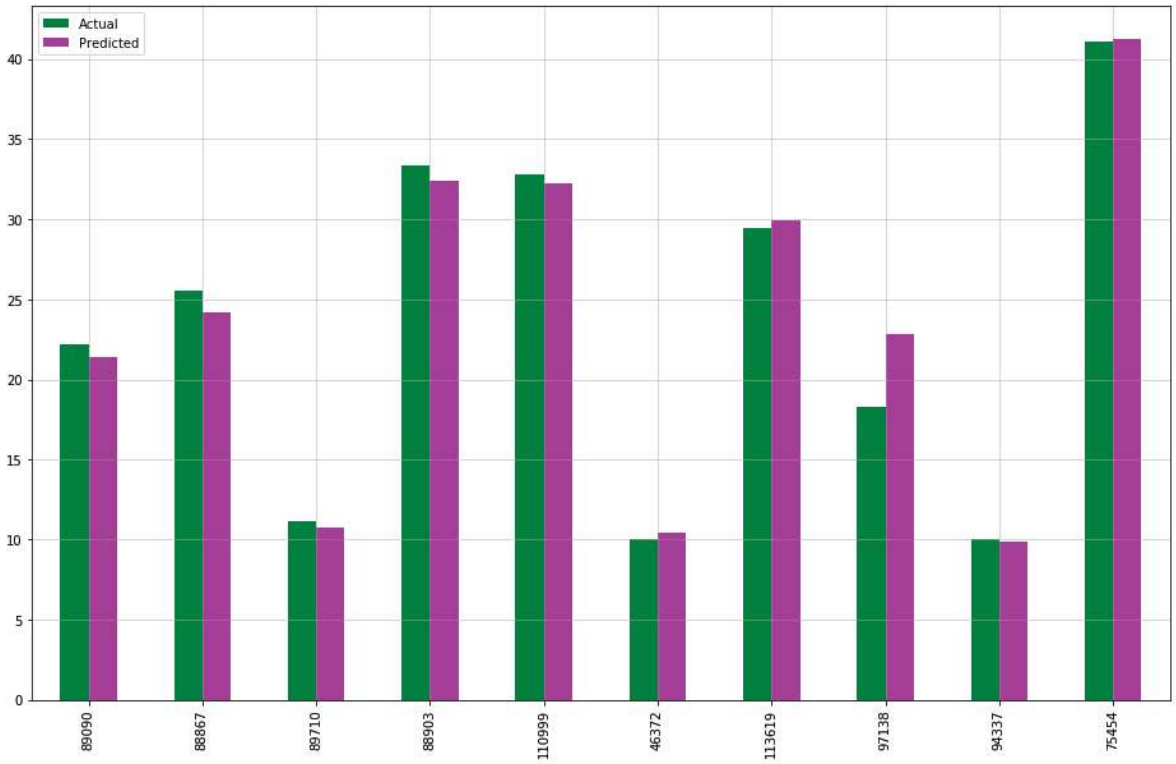
```
(array([21.44250569, 24.17644814, 10.69748345, ..., 31.2084209 ,
        21.42298795, 27.02166311]), 89090     22.222222
 88867     25.555556
 89710     11.111111
 88903     33.333333
 110999    32.777778
             ...
 81656     25.555556
 99765     26.111111
 25238     31.111111
 82545     21.111111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

In [104]:

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

In [105]:

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



In [106]:

```
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: 0.41657876630727664
Mean Squared Error: 1.1047876587504348
Root Mean Squared Error: 1.0510887967961768
```

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

Program 4

In [1]:

```
import numpy as np
import pandas as pd

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler
from sklearn import metrics

import matplotlib.pyplot as plt
```

In [2]:

```
df=pd.read_csv('P4_winequality.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9

In [4]:

```
df.shape
```

Out[4]:

```
(1599, 12)
```

In [5]:

```
1 df.dtypes
```

Out[5]:

```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

In [6]:

```
1 df.isna().sum()
```

Out[6]:

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [7]:

```
1 scaler=StandardScaler()
```

In [ ]:

```
1
```

In [8]:

```
1 X=df.drop(['quality'],axis=1)
2 y=df['quality']
```

In [9]:

```
1 X.head(1)
```

Out[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	7.4	0.7	0.0	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9

In [10]:

```
1 y.head(1)
```

Out[10]:

```
0    5
Name: quality, dtype: int64
```

In [ ]:

```
1
```

In [11]:

```
1 X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 1)
```

In [12]:

```
1 scaler.fit(X_train)
2 X_train = scaler.transform(X_train)
3 X_test = scaler.transform(X_test)
```

In [13]:

```
1 X_train
```

Out[13]:

```
array([[-0.73307913,  0.6648928 , -1.25704443, ...,  0.98846046,
         0.0630946 , -0.87223395],
       [ 1.06774091, -0.62346154,  1.52314768, ..., -1.7535127 ,
        -0.17390392, -0.77978452],
       [-1.74604041, -1.07158479, -1.35814232, ...,  2.32756363,
         0.77409018,  3.28799021],
       ...,
       [-0.95818164,  1.08500835, -0.90320179, ...,  0.22325865,
        -1.00339876,  0.32960859],
       [-0.62052788,  0.55286199, -1.35814232, ...,  0.35079228,
        -0.47015208, -1.33448108],
       [ 0.44870902, -0.73549236,  1.16930505, ..., -0.6694768 ,
         0.18159387,  1.90124882]])
```

In [14]:

```
1 X_test
```

Out[14]:

```
array([[ 0.27988214, -0.67947695,  1.87699031, ...,  1.43482818,
         0.00384497,  0.05226031],
       [ 0.22360652,  0.55286199,  0.05722821, ..., -0.35064271,
        -0.17390392, -0.22508797],
       [ 1.18029216, -1.07158479,  1.57369663, ..., -0.79701044,
         0.2408435 ,  0.79185571],
       ...,
       [-0.67680351,  0.77692361, -1.35814232, ...,  0.79716001,
        -0.7071506 , -0.96468337],
       [-1.18328414, -0.79150776,  0.81546242, ...,  0.86092682,
         1.24808723,  0.69940629],
       [ 1.4616703 , -1.18361561,  1.27040294, ..., -0.22310908,
         0.65559092,  0.97675456]])
```

In [15]:

```
1 acc_score = []
2
3 for i in range(1,40):
4     knn = KNeighborsClassifier(n_neighbors=i)
5     knn.fit(X_train,y_train)
6     y_pred = knn.predict(X_test)
7     acc_score.append(metrics.accuracy_score(y_test,y_pred))
```

In [16]:

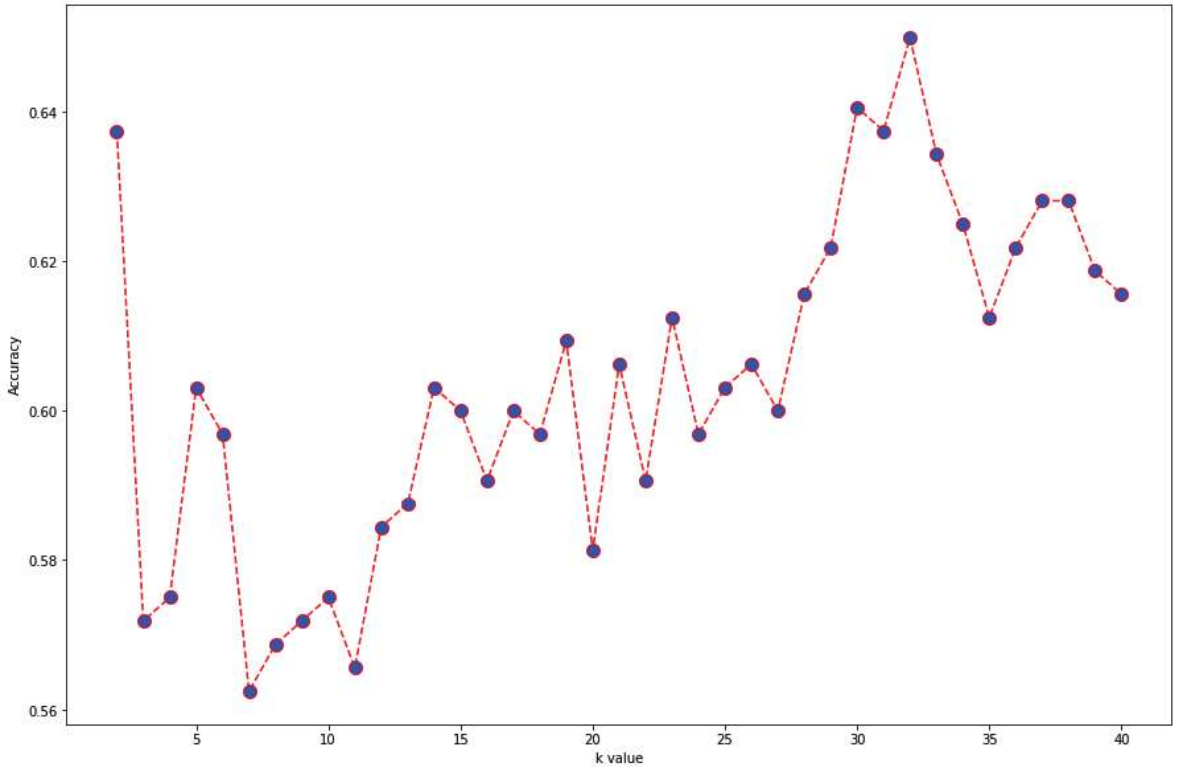
```
1 k=range(2,41)
```

In [17]:

```
plt.figure(figsize =(15,10))
plt.plot(k,acc_score,markersize = 10,color = 'red',linestyle = 'dashed',marker = 'o',ma
plt.xlabel('k value')
plt.ylabel('Accuracy')
```

Out[17]:

```
Text(0,0.5,'Accuracy')
```



In [18]:

```
for i in range(30,35):
    print(i+1,acc_score[i])
```

```
31 0.65
32 0.634375
33 0.625
34 0.6125
35 0.621875
```

In [19]:

```
for i in range(0,10):
    print(i+1,acc_score[i])
```

```
1 0.6375
2 0.571875
3 0.575
4 0.603125
5 0.596875
6 0.5625
7 0.56875
8 0.571875
9 0.575
10 0.565625
```

In [20]:

```
knn = KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)
```

In [21]:

```
df_new=df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
```

In [22]:

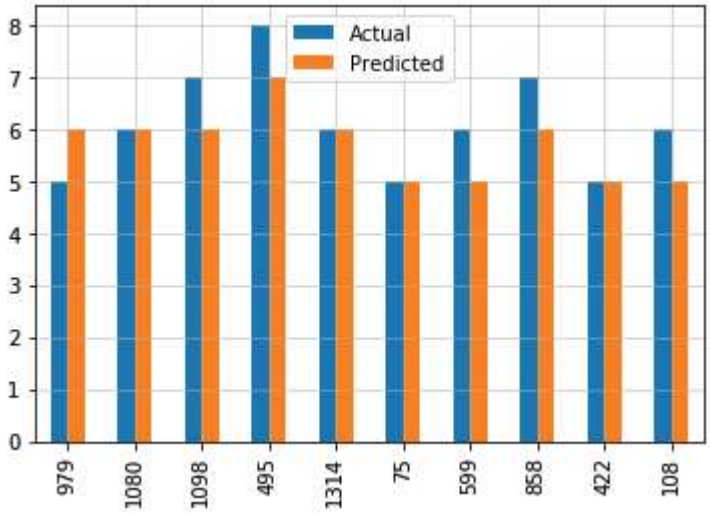
```
df_new.head()
```

Out[22]:

	Actual	Predicted
75	5	5
1283	6	5
408	6	6
1281	6	5
1118	6	6

In [23]:

```
df_new.sample(10).plot(kind='bar')
plt.grid(which='major',linewidth='0.5')
plt.grid(which='minor',linewidth='0.5')
plt.show()
```



In [ ]:

```

```

In [24]:

```
metrics.accuracy_score(y_test,y_pred)
```

Out[24]:

```
0.603125
```

In [25]:

```
knn.score(X_test,y_test)
```

Out[25]:

```
0.603125
```

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:

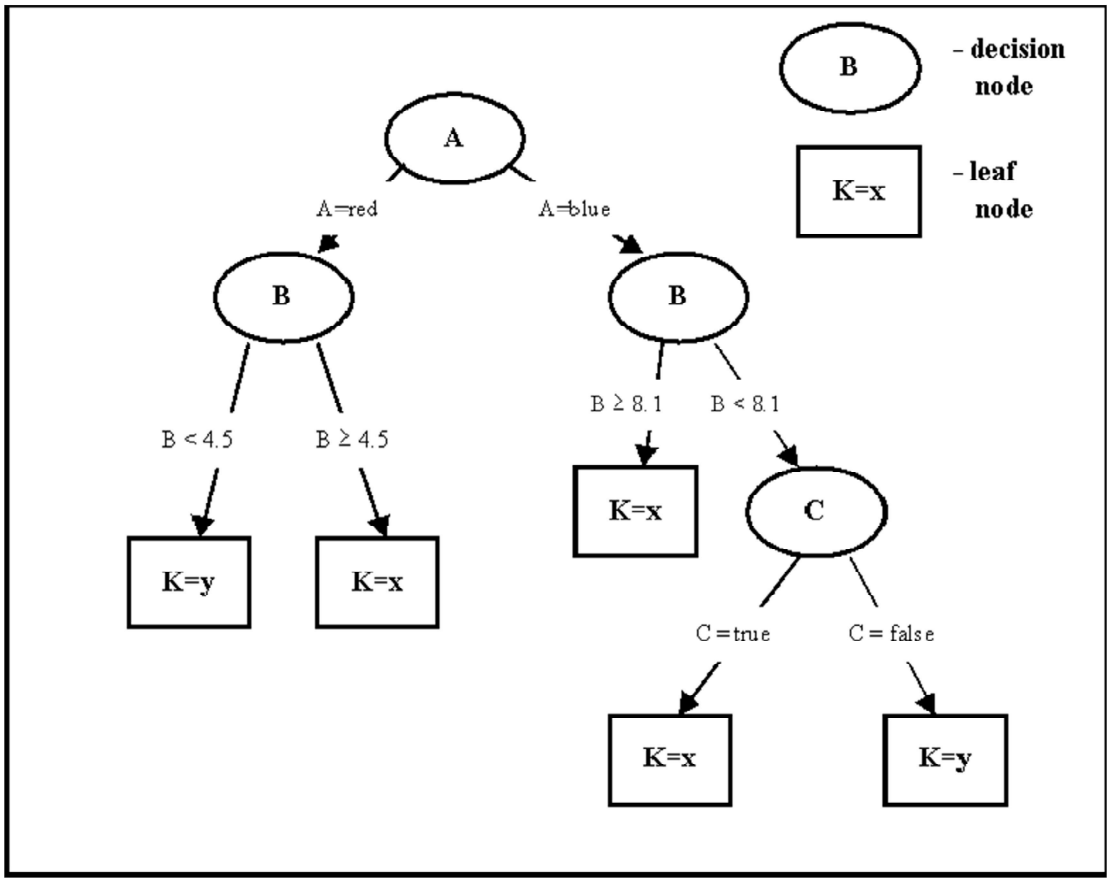
calculate entropy for all categorical values

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

Program 5

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
```

In [2]:

```
df=pd.read_csv('P5_diabetes.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [ ]:

```

```

In [6]:

```
#Splitting Dependent and independent Variables
```

In [7]:

```
X=df.drop(['Outcome'],axis=1)
```

In [8]:

```
y=df['Outcome'].values
```

In [9]:

```
#Formation of train and test Data
```

In [10]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

In [11]:

```
clf=DecisionTreeClassifier(criterion='entropy',max_depth=4)
```

In [12]:

```
clf.fit(X_train,y_train)
```

Out[12]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

In [13]:

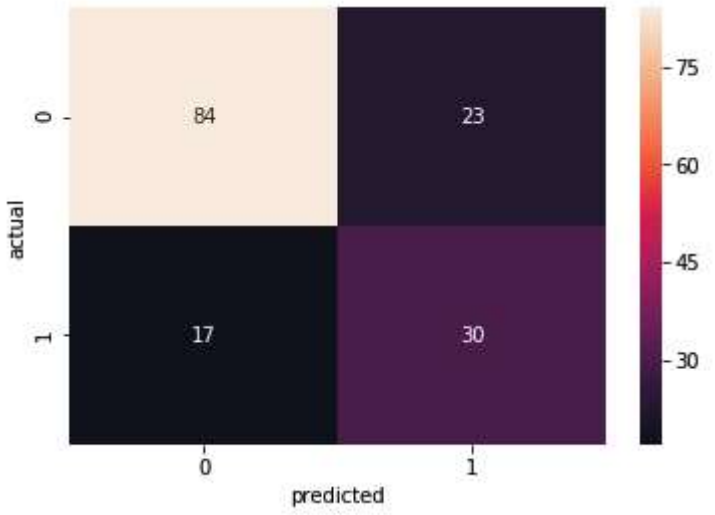
```
y_pred=clf.predict(X_test)
```

In [14]:

```
#checking the number of true or false positives/negatives
```

In [15]:

```
confusion_matrix = pd.crosstab(y_test, y_pred,rownames=['actual'],colnames=['predicted
sns.heatmap(confusion_matrix, annot=True)
plt.show()
```



In [16]:

```
#Evaluation metrics
```

In [17]:

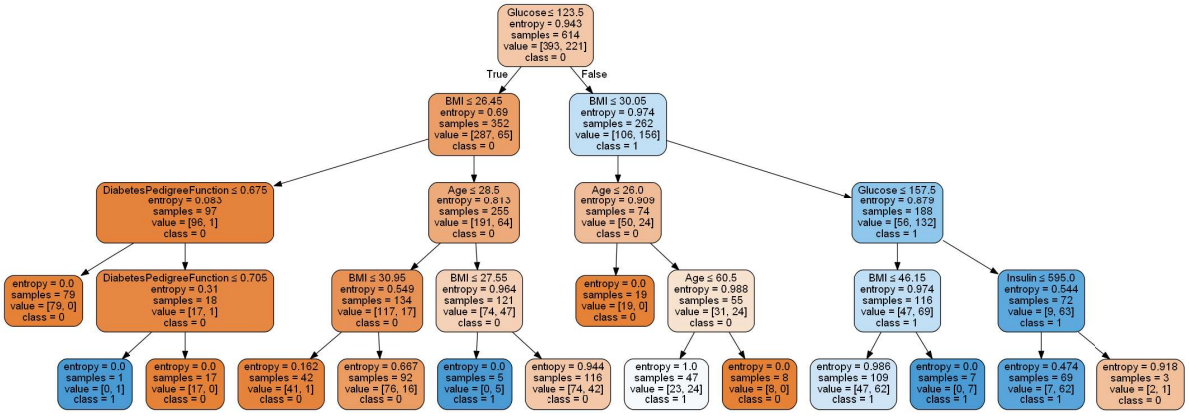
```
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.7402597402597403
```

In [18]:

```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = X.columns,class_names=['0','1'
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Out[18]:



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

Program 6

In [1]:

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

import string

import nltk
from nltk.corpus import stopwords

from sklearn.naive_bayes import MultinomialNB

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

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

In [2]:

```
df=pd.read_csv('P6_P7_spam.csv',encoding='latin-1')
```

In [3]:

```
df.isna().sum()
```

Out[3]:

```
v1              0
v2              0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

In [4]:

```
df=df[['v1','v2']]
```

In [5]:

```
1 df
```

Out[5]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

In [6]:

```
1 df.isna().sum()
```

Out[6]:

```
v1    0
v2    0
dtype: int64
```

In [7]:

```
1 df.columns=['label','message']
```

In [8]:

```
df
```

Out[8]:

	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

In [9]:

```
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 [10]:

```
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[10]:

```
True
```

In [11]:

```
X_train,X_test,y_train,y_test=train_test_split(df['message'],df['label'],test_size=0.2
```

In [12]:

```
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 [13]:

```
mess_bow1=bow_transformer.transform(X_test)
tfidf_transformer1=TfidfTransformer(use_idf=False)
mess_tfidf1=tfidf_transformer1.transform(mess_bow1)
```

In [14]:

```
mess_tfidf.shape
```

Out[14]:

```
(4457, 9832)
```

In [15]:

```
mnb=MultinomialNB()
mnb.fit(mess_tfidf,y_train)
```

Out[15]:

```
MultinomialNB()
```

In [16]:

```
y_pred=mnb.predict(mess_tfidf1)
```

In [17]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         ham       0.94      1.00      0.97       949
        spam       1.00      0.64      0.78       166

    accuracy                           0.95      1115
   macro avg       0.97      0.82      0.88      1115
weighted avg       0.95      0.95      0.94      1115
```

In [18]:

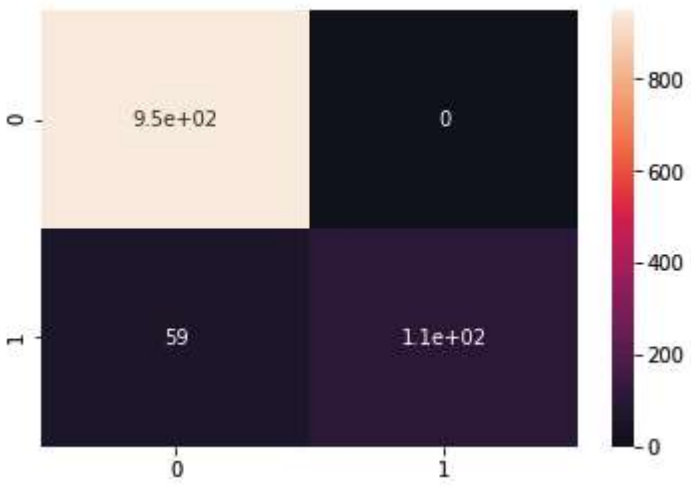
```
cf_matrix=confusion_matrix(y_test,y_pred)
```

In [19]:

```
1 sns.heatmap(cf_matrix,annot=True)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24aeda232b0>
```



In [20]:

```
1 print("Accuracy of the Naive Bayes Classifier is : ",accuracy_score(y_test,y_pred))
```

```
Accuracy of the Naive Bayes Classifier is :  0.947085201793722
```

In [ ]:

```
1
```

Program No. 7

Title:

Implementation of Support Vector Machine

Objective:

To understand the dynamics of SVM

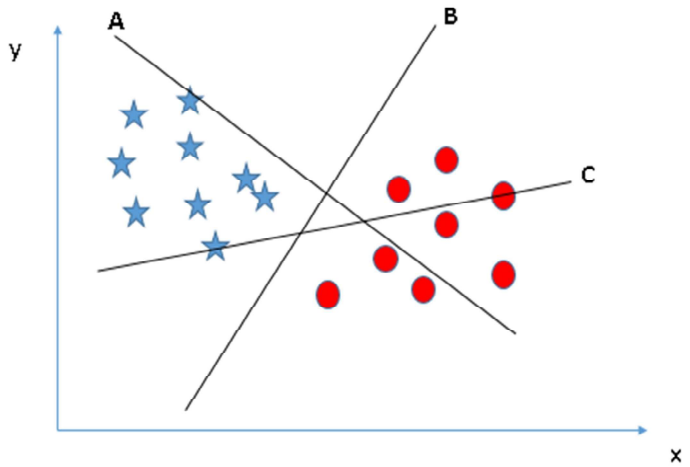
Reference:

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 .

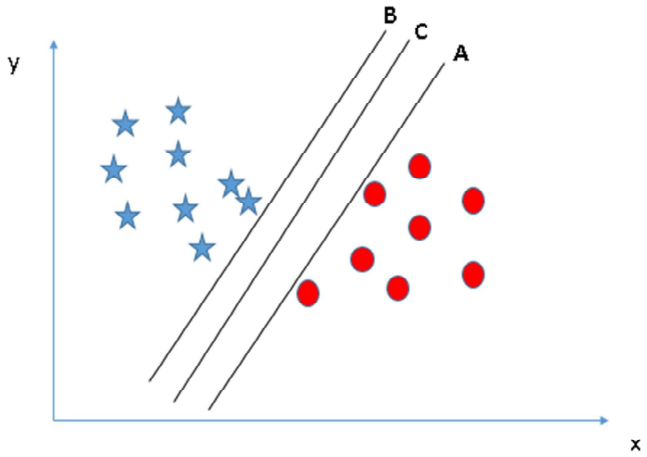
Let's understand: Identify the right hyper-plane (Scenario-1): Consider three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.



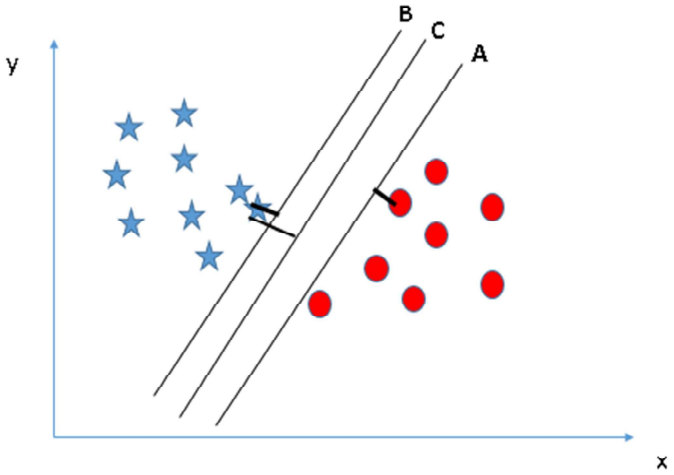
You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

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?

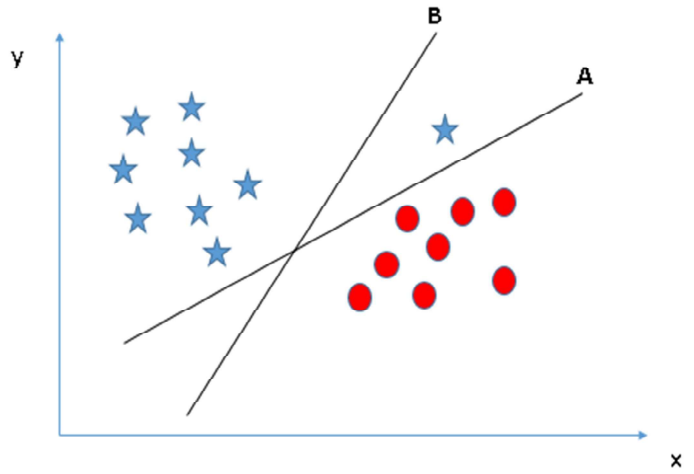


Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin. Let's look at the below snapshot:



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

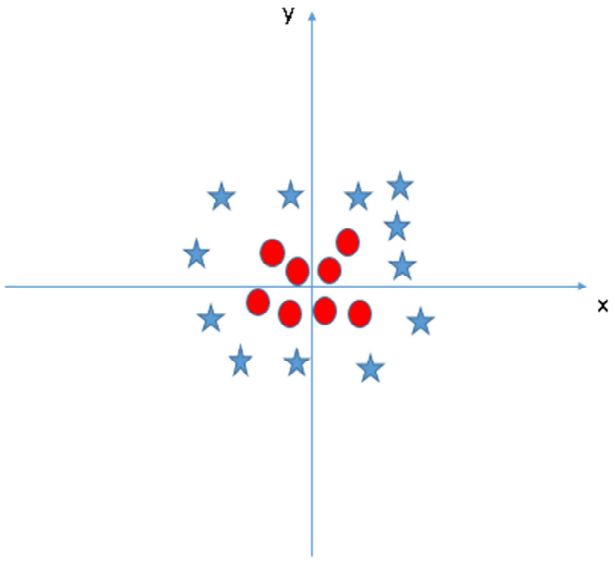


Some of you may have selected the hyper-plane B as it has higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore,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

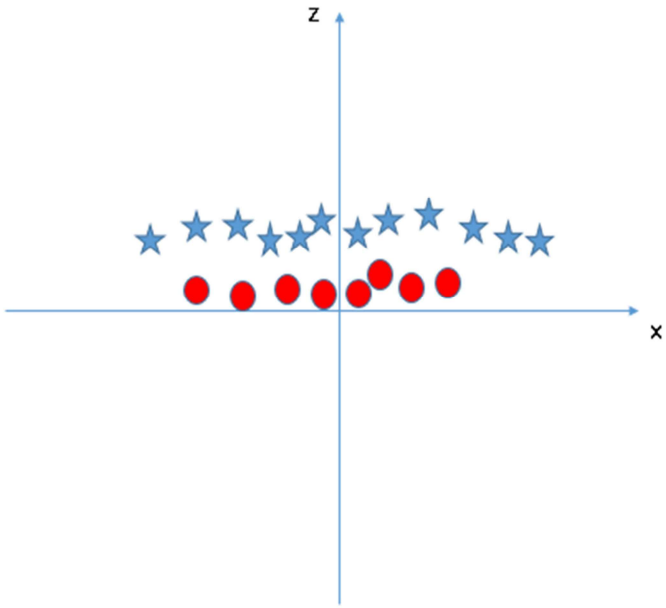
Can we classify two classes (Scenario-4)?: Below, I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.

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:

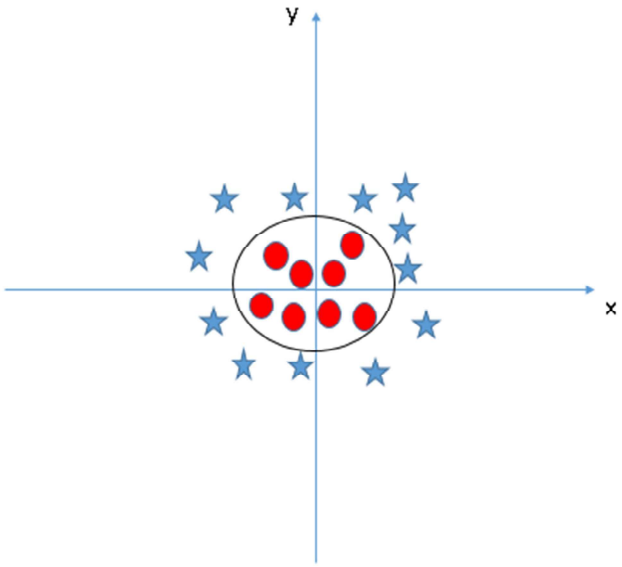
All values for z would be positive always because z is the squared sum of both x and y

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

Program 7

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
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 [2]:

```
df=pd.read_csv('P6_P7_spam.csv',encoding='latin-1')
```

In [3]:

```
df=df[['v1','v2']]
```

In [4]:

```
df.columns=['label','message']
```

In [5]:

```
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 [6]:

```
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[6]:

```
True
```

In [7]:

```
X_train,X_test,y_train,y_test=train_test_split(df['message'],df['label'],test_size=0.2
```

In [8]:

```
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 [9]:

```
mess_bow1=bow_transformer.transform(X_test)
tfidf_transformer1=TfidfTransformer(use_idf=False)
mess_tfidf1=tfidf_transformer1.transform(mess_bow1)
```

In [10]:

```
svc=LinearSVC()
svc.fit(mess_tfidf,y_train)
```

Out[10]:

```
LinearSVC()
```

In [11]:

```
y_pred=svc.predict(mess_tfidf1)
```

In [12]:

```
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 [13]:

```
accuracy_score(y_test,y_pred)
```

Out[13]:

```
0.9829596412556054
```

In [ ]:

```

```

In [14]:

```
svc=SVC()
svc.fit(mess_tfidf,y_train)
```

Out[14]:

```
SVC()
```

In [15]:

```
y_pred=svc.predict(mess_tfidf1)
```

In [16]:

```
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

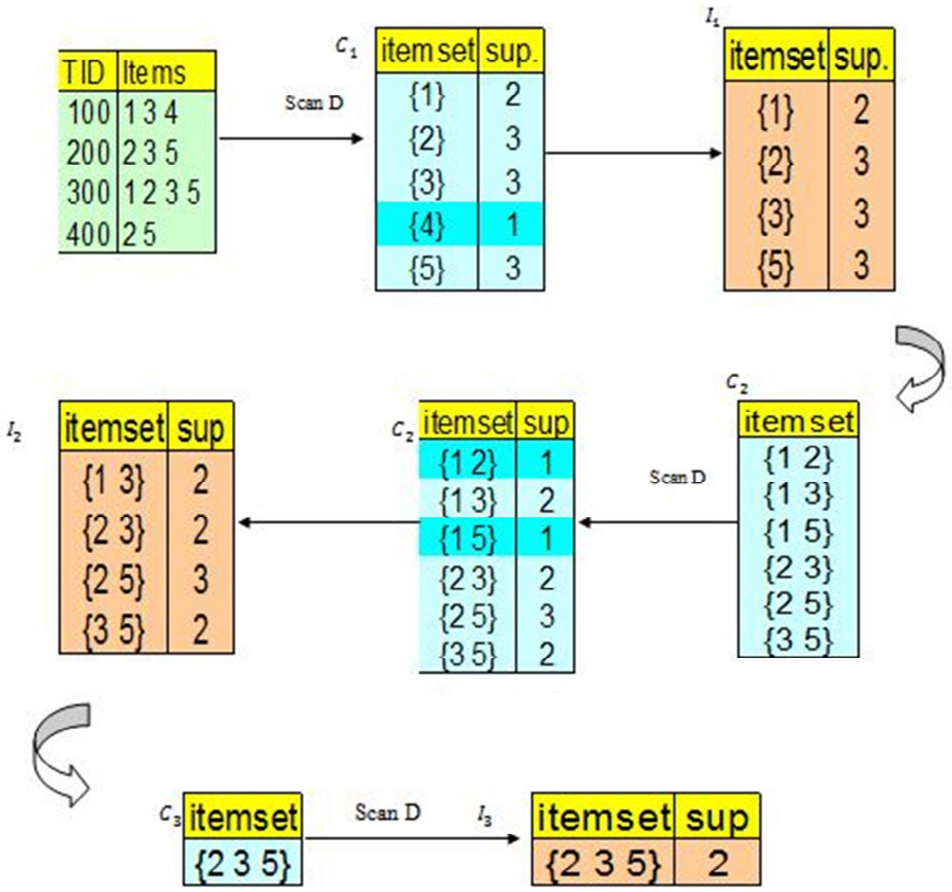
Inexplicable

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

Program 8

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df=pd.read_csv('P8_store_data.csv',header=None)
```

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.shape
```

Out[4]:

```
(7501, 20)
```

In [ ]:

```

```

In [5]:

```
data=[]
for i in range(0,7501):
    data.append([str(df.values[i,j]) for j in range(0,20) if(pd.isnull(df.values[i,j])
```

In [ ]:

```

```

In [6]:

```
data
```

```
 'salt'],
['champagne', 'low fat yogurt'],
['champagne'],
['burgers', 'almonds', 'eggs', 'french fries', 'cookies', 'green tea'],
['ham', 'soup', 'escalope', 'body spray'],
['turkey',
 'ham',
 'frozen vegetables',
 'pepper',
 'oil',
 'extra dark chocolate',
 'tea',
 'magazines'],
['muffins', 'eggs', 'cookies'],
['cookies'],
['frozen vegetables',
 'whole wheat pasta',
 'ground beef',
 'spaghetti',
 'chocolate',
```

In [7]:

```
from apyori import apriori
```

In [8]:

```
a_rules=apriori(data,min_support=0.003, min_confidence=0.2, min_lift=3, min_length=2)
```

In [9]:

```
result=list(a_rules)
```

In [10]:

```
len(result)
```

Out[10]:

```
80
```

In [11]:

```
result[0]
```

Out[11]:

```
RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532
728969470737, ordered_statistics=[OrderedStatistic(items_base=frozenset({'li
ght cream'}), items_add=frozenset({'chicken'}), confidence=0.290598290598290
57, lift=4.84395061728395)])
```

In [12]:

```
for item in result:
    shopping_pair=item[0]
    shop_items=[x for x in shopping_pair]
    print('Rule: ',shop_items[0],'-->',shop_items[1:])
    print('Support: ',item[1])
    print('Confidence: ',item[2][0][2])
    print('Lift: ',item[2][0][3])
    print("****************************")
```

```
Rule:  chicken --> ['light cream']
Support:  0.004532728969470737
Confidence:  0.29059829059829057
Lift:  4.84395061728395
****************************
Rule:  mushroom cream sauce --> ['escalope']
Support:  0.005732568990801226
Confidence:  0.3006993006993007
Lift:  3.790832696715049
****************************
Rule:  escalope --> ['pasta']
Support:  0.005865884548726837
Confidence:  0.3728813559322034
Lift:  4.700811850163794
****************************
Rule:  fromage blanc --> ['honey']
Support:  0.003332888948140248
Confidence:  0.2450980392156863
Lift:  5.164270764485569
****************************
```

In [ ]:

```
1
```

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

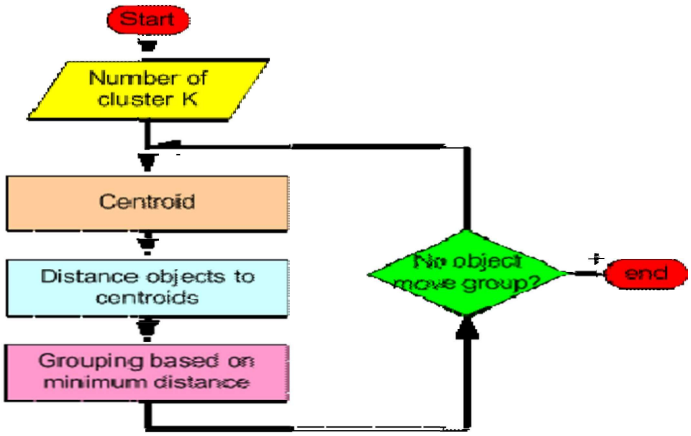
Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid. After each assignment, recomputed the centroid of the gaining cluster.

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

Program 9

In [1]:

```
import numpy as np
import pandas as pd
import random
from base64 import b64encode
from json import loads
```

In [2]:

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

In [3]:

```
from mlxtend.data import loadlocal_mnist
```

In [4]:

```
X, y = loadlocal_mnist(images_path='train-images.idx3-ubyte',labels_path='train-labels
```

In [5]:

```
X.shape
```

Out[5]:

```
(60000, 784)
```

In [6]:

```
X = X.astype(float) / 255
```

In [7]:

```
np.unique(y)
```

Out[7]:

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

In [8]:

```
np.bincount(y)
```

Out[8]:

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

In [9]:

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

In [ ]:

```

```

In [10]:

```
from sklearn.cluster import MiniBatchKMeans
```

In [11]:

```
kmeans=MiniBatchKMeans(n_clusters = 10)
```

In [12]:

```
kmeans.fit(X)
```

Out[12]:

```
MiniBatchKMeans(n_clusters=10)
```

In [13]:

```
len(kmeans.labels_)
```

Out[13]:

```
60000
```

In [14]:

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

Out[14]:

```
array([8, 9, 6, 1, 2, 6, 5, 6, 3])
```

In [15]:

```
y[1:10]
```

Out[15]:

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

In [ ]:

```

```

In [16]:

```
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 [17]:

```
def infer_data_labels(X_labels, cluster_labels):
    """
    Determines label for each array, depending on the cluster it has been assigned 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 [18]:

```
cluster_labels = infer_cluster_labels(kmeans, y)
```

In [19]:

```
X_clusters = kmeans.predict(X)
```

In [20]:

```
predicted_labels = infer_data_labels(X_clusters, cluster_labels)

```

In [21]:

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

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

In [22]:

```
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 [23]:

```
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: 2375291.679925211
Homogeneity: 0.49866254109816643
Accuracy: 0.6099333333333333

Number of Clusters: 16
Inertia: 2271087.568947321
Homogeneity: 0.5261581330509923
Accuracy: 0.6433166666666666

Number of Clusters: 36
Inertia: 1958383.8845497835
Homogeneity: 0.6818558321370068
Accuracy: 0.7738

Number of Clusters: 64
Inertia: 1816527.0042198256
Homogeneity: 0.7409626605182396
Accuracy: 0.81045

Number of Clusters: 144
Inertia: 1631586.7146993065
Homogeneity: 0.80445340172215
Accuracy: 0.8706333333333334

Number of Clusters: 256
Inertia: 1514643.988767254
Homogeneity: 0.8425231315234392
Accuracy: 0.89835
```

In [24]:

```
X_test,y_test=loadlocal_mnist(images_path='t10k-images.idx3-ubyte',labels_path='t10k-l
```

In [25]:

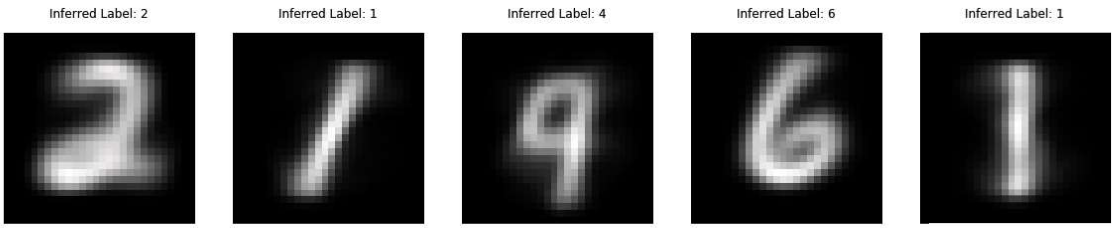
```
kmeans = MiniBatchKMeans(n_clusters = 256)
kmeans.fit(X)
cluster_labels = infer_cluster_labels(kmeans, y)

# 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)))
```

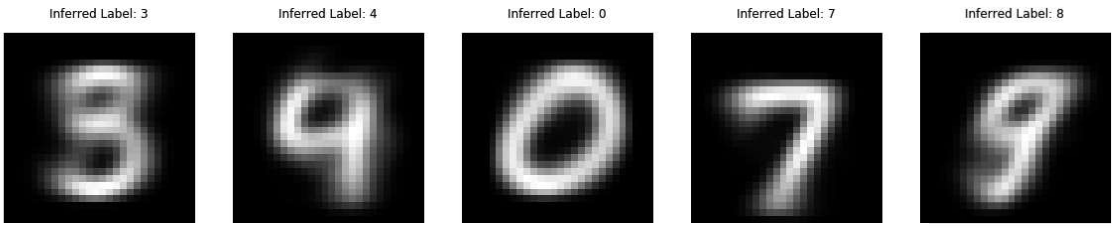
```
Accuracy: 0.6521
```

In [26]:

```
# 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: UserWar
ning: matplotlib is currently using a non-GUI backend, so cannot show the fi
gure
  "matplotlib is currently using a non-GUI backend, "
```


Inferred Label: 2
Inferred Label: 1
Inferred Label: 4
Inferred Label: 6
Inferred Label: 1


Inferred Label: 3
Inferred Label: 4
Inferred Label: 0
Inferred Label: 7
Inferred Label: 8

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							

Customer Segmentation on Online Retail Dataset

Description of the case study

Customer segmentation is the process of dividing customers into groups based on common characteristics so companies can market to each group effectively and appropriately. Segmentation allows marketers to better tailor their marketing efforts to various audience subsets.Those efforts can relate to both communications and product development. Using clustering techniques, companies can identify the several segments of customers allowing them to target the potential user base.

Segmentation helps a company to:

- Create and communicate targeted marketing messages that will resonate with specific groups of customers, but not with others (who will receive messages tailored to their needs and interests, instead).
- Select the best communication channel for the segment, which might be email, social media posts, radio advertising, or another approach, depending on the segment.
- Identify ways to improve products or new product or service opportunities.
- Establish better customer relationships.
- Test pricing options.
- Focus on the most profitable customers.
- Improve customer service.
- Upsell and cross-sell other products and services.

Customer segmentation requires a company to gather specific information – data – about customers and analyze it to identify patterns that can be used to create segments.

Common characteristics in customer segments can guide how a company markets to individual segments and what products or services it promotes to them.

Data set used-

- Kaggle_data_sales
- Kaggle_data_demographics
- UCI_data

Data set description

1. Kaggle_data_sales - Customer sales data which contains transactions between 1/12/2010 and 9/12/2011

```
In [3]:  1 df1.head(5)
```

Out[3]:

	Unnamed: 0	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice
0	0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55
1	1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39
2	2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75
3	3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39
4	4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39

2. Kaggle_data_demographics - Customer demographics data which contains customer demographics who purchased between 1/12/2010 and 9/12/2011

```
In [6]:  1 df2.head()
```

Out[6]:

	Unnamed: 0	InvoiceNo	CustomerID	Country
0	0	536365	17850.0	United Kingdom
1	7	536366	17850.0	United Kingdom
2	9	536367	13047.0	United Kingdom
3	21	536368	13047.0	United Kingdom
4	25	536369	13047.0	United Kingdom

```
In [7]:  1 df2.shape
```

Out[7]: (25900, 4)

3. UCI_data - Customer sales and demographics data which contains transactions occurring between 1/12/2009 and 9/12/2010

```
In [9]:  1 df3.head()
```

Out[9]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	01-12-2009 07:45	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	01-12-2009 07:45	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	01-12-2009 07:45	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	01-12-2009 07:45	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	01-12-2009 07:45	1.25	13085.0	United Kingdom

```
In [10]:  1 df3.shape
```

Out[10]: (525461, 8)

Data visualization techniques used and justification

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

Modules used:

- matplotlib.pyplot
- missingno
- seaborn

Types:

- Bar graphs
- Tables
- Scatter plot
- Histograms
- Distance plot
- Pie charts
- Line graphs
- Heat map
- 3D plots

Steps where Data visualizations are done:

- Analyzing missing values

To find the missing values in the dataset

- Dataset analysis by grouping it with respect to country ,customers, invoice numbers, month of the year etc to get the overall view of the data

Data Visualization

No of customers from each country

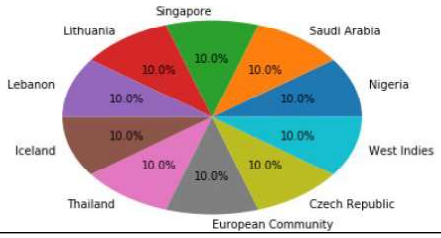
In [53]:
```
customer_country=df_cleaned.drop_duplicates(subset=['CustomerID'],keep='first')
customer_country=customer_country.groupby(['Country'])['CustomerID'].aggregate('count').reset_index().sort_values('CustomerID
customer_country.columns=['Country', 'NoOfCustomers']
```

In [54]:
```
customer_country.head(5)
```

Out[54]:

	Country	NoOfCustomers
38	United Kingdom	5352
14	Germany	106
13	France	95
32	Spain	38
3	Belgium	28

In [55]:
```
plt.pie(customer_country.tail(10)['NoOfCustomers'],labels=customer_country.tail(10)['Country'],autopct='%1.1f%%')
plt.show()
```



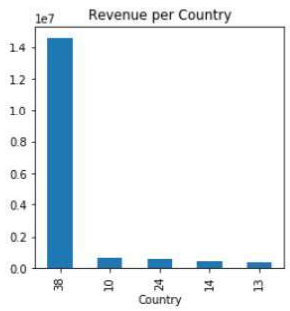
Revenue generated per countries

In [129]:
```
revenue_per_countries = df_cleaned.groupby(["Country"])["TotalPrice"].sum().sort_values(ascending=False)
revenue_per_countries.head(2)
```

Out[129]:
```
Country
38    1.461027e+07
10    6.187078e+05
Name: TotalPrice, dtype: float64
```

In [133]:
```
revenue_per_countries.head(5).plot(kind='bar', figsize=(4,4))
plt.title("Revenue per Country")
```

Out[133]:
```
Text(0.5,1,'Revenue per Country')
```



No of products purchased per country

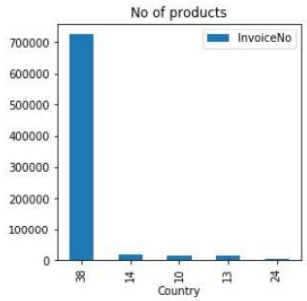
In [128]:
```
No_invoice_per_country = df_cleaned.groupby(["Country"])[["InvoiceNo"]].count().sort_values(by='InvoiceNo',ascending=False)
No_invoice_per_country.head(2)
#grouping by countries to check amount of sales in each country in descending order
```

Out[128]:

Country	InvoiceNo
38	725296
14	16703

In [134]:
```
No_invoice_per_country.head(5).plot(kind='bar', figsize=(4,4))
plt.title("No of products")
```

Out[134]: Text(0.5,1,'No of products')



Grouping data wrt Countries

Ranking customers with no of orders

In [135]:
```
#to check customers with highest no of orders
df_cleaned.groupby(['CustomerID']).count()[['InvoiceNo']].sort_values(by='InvoiceNo',ascending=False).head(4)
```

Out[135]:

CustomerID	InvoiceNo
17841.0	12890
14911.0	11247
12748.0	7230
14606.0	6566

Ranking customers with expenditure

In [136]:
```
#to rank customers with high expenditure
df_cleaned.groupby(['CustomerID']).sum()[['Quantity','UnitPrice']].sort_values(by='UnitPrice',ascending=False).head(4)
```

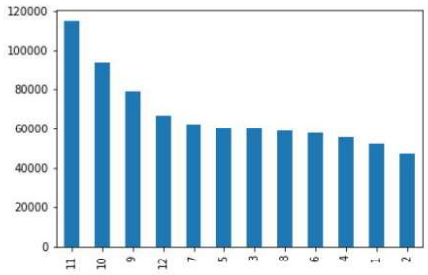
Out[136]:

CustomerID	Quantity	UnitPrice
14911.0	150237	57753.06
17841.0	37558	35064.34
14096.0	16352	33332.45
14156.0	165992	24053.55

Grouping data wrt Customers

Checking relation of month with amount of sales

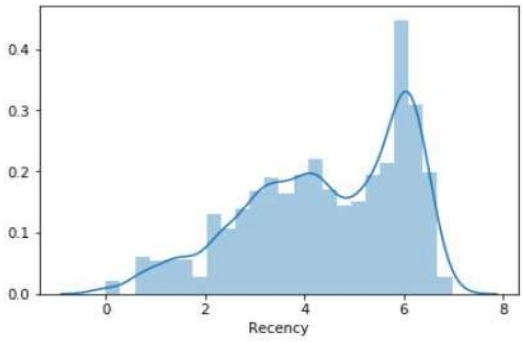
```
In [62]: df_cleaned['InvoiceDate']=pd.to_datetime(df_cleaned['InvoiceDate'],errors='coerce').dt.date

In [63]: df_cleaned['Month']=df_cleaned['InvoiceDate'].map(lambda x:x.month)

In [64]: h=df_cleaned['Month'].value_counts()

In [65]: h.plot(kind='bar')

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x266ca0729e8>
```



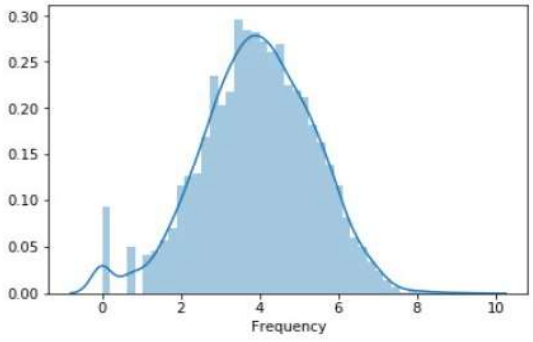

```
In [66]: #November has highest no of sales
```

Checking the sales as per months of the year

```
In [99]: a=sns.distplot(Log_Data['Recency'])
```

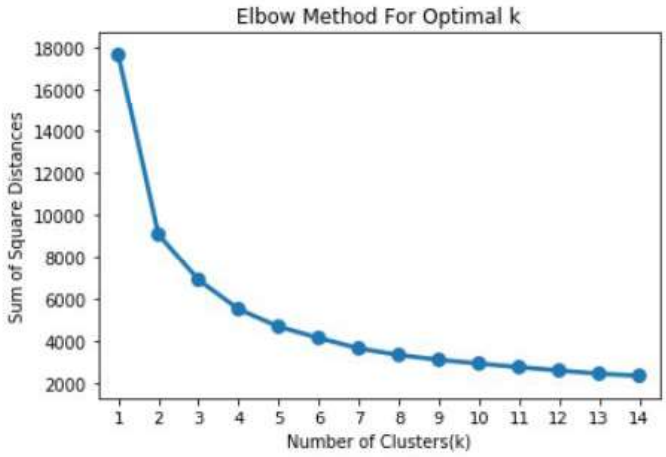



```
In [100]: Frequency_Plot = Log_Data.query('Frequency < 1000')['Frequency']
          b=sns.distplot(Frequency_Plot)
```



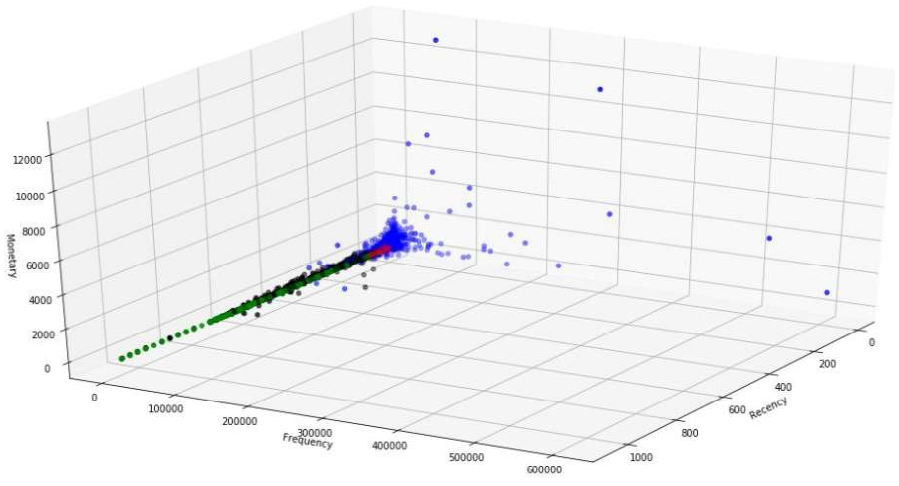
Distance plot after log transformation while fitting kmeans

```
#Plot the graph for the sum of square distance values and Number of Clusters
sns.pointplot(x = list(sum_of_sq_dist.keys()), y = list(sum_of_sq_dist.values()))
plt.xlabel('Number of Clusters(k)')
plt.ylabel('Sum of Square Distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



Elbow method to find the optimal k

```
In [111]:
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(20,10))
ax = plt.axes(projection='3d')
xline=r_table["Recency"]
yline=r_table["Frequency"]
zline=r_table['Monetary']
ax.scatter3D(xline, zline,yline,c = r_table['Color'])

ax.set_xlabel("Recency")
ax.set_ylabel("Frequency")
ax.set_zlabel('Monetary')
ax.view_init(30, 30)
```



Clusters formed by K-Means model (n=4)

Pre-processing techniques used and justification

Reading and Integrating Datasets

```
In [138]: df1=pd.read_csv('data_kaggle_sales.csv')
          df1.head(2)
```

Out[138]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39

```
In [4]: df1.shape
```

```
Out[4]: (541909, 6)
```

```
In [139]: df2=pd.read_csv('data_kaggle_demographic.csv')
          df2.head(2)
```

Out[139]:

	InvoiceNo	CustomerID	Country
0	536365	17850.0	United Kingdom
1	536366	17850.0	Unitcd Kingdom

```
In [ ]: df2.shape
```

```
In [140]: df3=pd.read_csv('data_UCI.csv',encoding='latin-1')
          df3.head(2)
```

Out[140]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	01-12-2009 07:45	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	01-12-2009 07:45	6.75	13085.0	United Kingdom

```
In [141]: df3.shape
```

```
Out[141]: (525461, 8)
```

Importing all three datasets using pandas library

Data Integration

```
In [11]: df1.columns
```

```
Out[11]: Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
                'UnitPrice'],
               dtype='object')
```

```
In [12]: df2.columns
```

```
Out[12]: Index(['InvoiceNo', 'CustomerID', 'Country'], dtype='object')
```

```
In [13]: df4=pd.merge(df1,df2,on='InvoiceNo',how='left') # since the attribute name is same in both no need of renaming
```

```
In [14]: df4.head()
```

Out[14]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

```
In [15]: df4.columns
```

```
Out[15]: Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
                'UnitPrice', 'CustomerID', 'Country'],
               dtype='object')
```

```
In [16]: df3.columns #attribute names are different hence renaming is required
```

```
Out[16]: Index(['Invoice', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
                'Price', 'Customer ID', 'Country'],
               dtype='object')
```

Renaming the columns

```
In [17]: df3.columns=['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate','UnitPrice', 'CustomerID', 'Country']
         #change the column name same as the first dataset so that they same attributes are not distinguished

In [18]: df=pd.concat([df4,df3],ignore_index=True)

In [19]: df.head()
```

Out[19]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

```
In [20]: df.shape
Out[20]: (1067370, 8)
```

Final dataset after integration

Data Cleaning

```
In [23]: nan_rows = df[df.isnull().T.any().T]
         nan_rows.head(5)
```

Out[23]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	12/1/2010 11:52	0.00	NaN	United Kingdom
1443	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	12/1/2010 14:32	2.51	NaN	United Kingdom
1444	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	12/1/2010 14:32	2.51	NaN	United Kingdom
1445	536544	21786	POLKADOT RAIN HAT	4	12/1/2010 14:32	0.85	NaN	United Kingdom
1446	536544	21787	RAIN PONCHO RETROSPOT	2	12/1/2010 14:32	1.66	NaN	United Kingdom

```
In [24]: #nan_rows basically gives the dataframe with NaN values in any column
```

Identifying the rows with NaN values

```
In [30]: g=df[df['InvoiceNo']=='581498']
         g[pd.isnull(g['CustomerID'])==False]
```

Out[30]:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country

```
In [31]: #customer ID cant be filled in any manner so the rows with no customer id are dropped

In [32]: df.dropna(subset=['CustomerID'],inplace=True)
```

As seen in the above figure data set was analysed to check if there is any row which has customer id filled so that the other rows with the same invoice number can be filled using Customer id .
But no such row was found , hence having no other way to fill the customer id , all such rows were dropped

```
In [34]: 1 df.isna().sum()
         2 #No missing values left

Out[34]: InvoiceNo      0
         StockCode      0
         Description    0
         Quantity       0
         InvoiceDate    0
         UnitPrice      0
         CustomerID     0
         Country        0
         dtype: int64
```

No missing values left

Statistical Analysis of the numeric data

```
In [25]: 1 df.describe()
```

Out[25]:

	Quantity	UnitPrice	CustomerID
count	1.067370e+06	1.067370e+06	824363.000000
mean	9.938907e+00	4.649375e+00	15324.641712
std	1.727059e+02	1.235531e+02	1697.462981
min	-8.099500e+04	-5.359436e+04	12346.000000
25%	1.000000e+00	1.250000e+00	13975.000000
50%	3.000000e+00	2.100000e+00	15255.000000
75%	1.000000e+01	4.150000e+00	16797.000000
max	8.099500e+04	3.897000e+04	18287.000000

Interpretation of the results:

- Quantity cannot be 0 or less while ordering
- price should be greater than 0
- Hence any such inconsistency should be taken into consideration to form a cleaned dataset

```
In [40]: 1 df[df['Quantity']<0]
```

Out[40]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
141	C536379	D	Discount	-1	2010-12-01	27.50	14527.0	United Kingdom
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	2010-12-01	4.65	15311.0	United Kingdom
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	2010-12-01	1.65	17548.0	United Kingdom
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	2010-12-01	0.29	17548.0	United Kingdom

Rows with negative quantity were either cancelled orders or were given on discount

In [42]:
```
# Constucting a basket for later use
temp=df.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['Quantity'].sum()
nb_products_per_basket = temp.rename(columns = {'Quantity':'Number of products'})
nb_products_per_basket
```

Out[42]:

	CustomerID	InvoiceNo	Number of products
0	12346.0	491725	10
1	12346.0	491742	5
2	12346.0	491744	5
3	12346.0	492718	5
4	12346.0	492722	1
...	...	...	...
44871	18287.0	534346	187
44872	18287.0	554065	488
44873	18287.0	570715	990
44874	18287.0	573167	108
44875	18287.0	C489592	-2

44876 rows × 3 columns

In [43]:
```
nb_products_per_basket['order_canceled'] = nb_products_per_basket['InvoiceNo'].apply(lambda x:int('C' in x))
```

In [44]:
```
len(nb_products_per_basket[nb_products_per_basket['order_canceled']==1])/len(nb_products_per_basket)*100
```

Out[44]: 17.606292895980033

About 17% of data rows are of cancelled orders

```
df_cleaned = df.copy(deep = True)
df_cleaned['QuantityCanceled'] = 0

entry_to_remove = [] ; doubtfull_entry = []

for index, col in  df_cleaned.iterrows():

    if (col['Quantity'] > 0): continue
    df_test = df_cleaned[(df_cleaned['CustomerID'] == col['CustomerID']) &
                         (df_cleaned['StockCode']  == col['StockCode']) &
                         (df_cleaned['InvoiceDate'] <= col['InvoiceDate']) &
                         (df_cleaned['Quantity']> -col['Quantity'])].copy()

    # Cancelation WITHOUT counterpart
    if (df_test.shape[0] == 0):
        doubtfull_entry.append(index)
        df_cleaned.drop(index,axis=0,inplace=True)

    # Cancelation WITH a counterpart
    elif (df_test.shape[0] > 0 ):
        index_order = df_test.index[0]
        df_cleaned.loc[index_order, 'QuantityCanceled'] = -col['Quantity']
        entry_to_remove.append(index)
        df_cleaned.drop(index,axis=0,inplace=True)
```

```
print("entry_to_remove: {}".format(len(entry_to_remove)))
print("doubtfull_entry: {}".format(len(doubtfull_entry)))
```

```
#Removing the cancellation orders once they are considered
remaining_entries = df_cleaned[df_cleaned['Quantity'] <= 0]
print("nb of entries to delete: {}".format(remaining_entries.shape[0]))
```

Cleaning the data:

- Create a new column called ‘QuantityCancelled’
- If the same product with same or more quantity(as quantity cancelled) was ordered before update the newly added column, remove the cancelled order row
- If no such data entry is found, then that row can be considered inconsistent and hence dropped.

```
df_cleaned.drop(remaining_entries.index,axis=0,inplace=True)
```

```
df_cleaned.Quantity.describe()

count    805619.000000
mean         13.307680
std         144.306828
min           1.000000
25%           2.000000
50%           5.000000
75%          12.000000
max       80995.000000
Name: Quantity, dtype: float64
```

```
df_cleaned.to_csv('cleaned_final.csv',index=False)
```

No more rows with negative quantity

Data Reduction

```
In [30]: df_cleaned.columns

Out[30]: Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
                'UnitPrice', 'CustomerID', 'Country', 'QuantityCanceled', 'TotalPrice'],
               dtype='object')
```

As all the columns are important for the model, none of the columns are removed.

Data Transformation

Changing the format of data

```
In [35]: for i in df.columns:
             print(i,type(df[i][0]))

InvoiceNo <class 'str'>
StockCode <class 'str'>
Description <class 'str'>
Quantity <class 'numpy.int64'>
InvoiceDate <class 'str'>
UnitPrice <class 'numpy.float64'>
CustomerID <class 'numpy.float64'>
Country <class 'str'>

In [36]: df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'],errors='coerce').dt.date

In [37]: type(df['InvoiceDate'][0])

Out[37]: datetime.date
```

Calculating the total price paid per product

```
In [51]: df_cleaned['TotalPrice'] = df_cleaned['UnitPrice'] * (df_cleaned['Quantity'] - df_cleaned['QuantityCanceled'])

In [52]: df_cleaned.head()
```

Out[52]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	QuantityCanceled	TotalPrice
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01	2.55	17850.0	United Kingdom	0	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01	3.39	17850.0	United Kingdom	0	20.34

Label Encoding the countries:

```
In [67]:
le=LabelEncoder()
le.fit(df_cleaned['Country'])
l=[i for i in range(len(df_cleaned['Country'].unique()))]
dict(zip(list(le.classes_),l))

Out[67]: {'Australia': 0,
 'Austria': 1,
 'Bahrain': 2,
 'Belgium': 3,
 'Brazil': 4,
 'Canada': 5,
```

Algorithm to build the model

Customer segmentation has a model called RFM which stands for Recency, Frequency and Monetary values respectively which help in grouping the customers thus making the system more effective and customized for each user.

- **Recency**: How recently a customer has made a purchase
- **Frequency**: How often a customer makes a purchase
- **Monetary Value**: How much money a customer spends on purchases

RFM Analysis

```
In [71]: df_cleaned['InvoiceDate'].min() #year-month-day
Out[71]: datetime.date(2009, 1, 12)

In [72]: df_cleaned['InvoiceDate'].max()
         #we have 3 years of customer data
Out[72]: datetime.date(2011, 12, 9)

In [73]: today=dt.date(2011,12,10)

In [74]: cust_agg={
             'InvoiceDate':'max',
             'InvoiceNo':lambda x:len(x),
             'TotalPrice':'sum'
         }

In [75]: r_table=df_cleaned.groupby('CustomerID').agg(cust_agg)
         r_table.columns=['Recent_InvoiceDate','Frequency','Monetary']

In [76]: r_table['Recency']=(today-r_table['Recent_InvoiceDate']).dt.days

In [77]: r_table.drop(['Recent_InvoiceDate'],axis=1,inplace=True)

In [78]: r_table=r_table[['Recency','Frequency','Monetary']]
```

Adding RFM values to each customer

	CustomerID	Recency	Frequency	Monetary
0	12346.0	326	34	77551.96
1	12347.0	3	253	5633.32
2	12348.0	76	51	2019.40
3	12349.0	19	175	4428.69
4	12350.0	311	17	334.40
...	...	...	...	...
5876	18283.0	4	986	2736.65
5877	18284.0	609	28	461.68
5878	18285.0	661	12	427.00
5879	18286.0	477	67	1242.43
5880	18287.0	43	155	4182.99

5881 rows × 4 columns

Modified dataset

In [84]:
```
def RScoring(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4

def FMScoring(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1
```

In [85]:
```
r_table['R'] = r_table['Recency'].apply(RScoring, args=('Recency',quantiles,))
r_table['F'] = r_table['Frequency'].apply(FMScoring, args=('Frequency',quantiles,))
r_table['M'] = r_table['Monetary'].apply(FMScoring, args=('Monetary',quantiles,))
r_table.head()
```

Out[85]:

CustomerID	Recency	Frequency	Monetary	R	F	M
12346.0	326	34	77551.96	3	3	1
12347.0	3	253	5633.32	1	1	1
12348.0	76	51	2019.40	2	3	2
12349.0	19	175	4428.69	1	1	1
12350.0	311	17	334.40	3	4	4

- Giving RFM Score to the customer (with the range 4 to 12)
- 12(Loyal customer) - 4(customer at the verge of losing)

In [86]:
```
r_table['RFMGroup'] = r_table.R.map(str) + r_table.F.map(str) + r_table.M.map(str)
r_table['RFMScore'] = r_table[['R', 'F', 'M']].sum(axis = 1)
r_table.head()
```

Out[86]:

CustomerID	Recency	Frequency	Monetary	R	F	M	RFMGroup	RFMScore
12346.0	326	34	77551.96	3	3	1	331	7
12347.0	3	253	5633.32	1	1	1	111	3
12348.0	76	51	2019.40	2	3	2	232	7
12349.0	19	175	4428.69	1	1	1	111	3
12350.0	311	17	334.40	3	4	4	344	11

In [87]:
```
Loyalty_Level = ['Platinum', 'Gold', 'Silver', 'Bronze']
Score_cuts = pd.qcut(r_table.RFMScore, q = 4, labels = Loyalty_Level)
r_table['RFM_Loyalty_Level'] = Score_cuts.values
r_table.reset_index().head()
```

Out[87]:

	CustomerID	Recency	Frequency	Monetary	R	F	M	RFMGroup	RFMScore	RFM_Loyalty_Level
0	12346.0	326	34	77551.96	3	3	1	331	7	Gold
1	12347.0	3	253	5633.32	1	1	1	111	3	Platinum
2	12348.0	76	51	2019.40	2	3	2	232	7	Gold
3	12349.0	19	175	4428.69	1	1	1	111	3	Platinum
4	12350.0	311	17	334.40	3	4	4	344	11	Bronze

Dividing customers into 4 classes based on their RFM score

K Means:

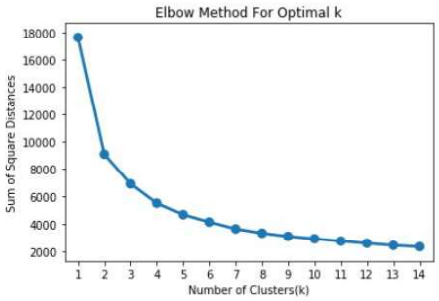
In [97]:
```
#Handle negative and zero values so as to handle infinite numbers during log transformation
def handle_neg_n_zero(num):
    if num <= 0:
        return 1
    else:
        return num
#Apply handle_neg_n_zero function to Recency and Monetary columns
r_table['Recency'] = [handle_neg_n_zero(x) for x in r_table.Recency]
r_table['Monetary'] = [handle_neg_n_zero(x) for x in r_table.Monetary]

Log_Data = r_table[['Recency', 'Frequency', 'Monetary']].apply(np.log, axis = 1).round(3)
```

Applying Log Transformation

In [102]:
```
from sklearn.preprocessing import StandardScaler
scaleobj = StandardScaler()
Scaled_Data = scaleobj.fit_transform(Log_Data)
Scaled_Data = pd.DataFrame(Scaled_Data, index = r_table.index, columns = Log_Data.columns)
```

Normalizing the data using Standard Scaler

In [103]:
```
from sklearn.cluster import KMeans

sum_of_sq_dist = {}
for k in range(1,15):
    km = KMeans(n_clusters= k, init= 'k-means++', max_iter= 1000)
    km = km.fit(Scaled_Data)
    sum_of_sq_dist[k] = km.inertia_

#Plot the graph for the sum of square distance values and Number of Clusters
sns.pointplot(x = list(sum_of_sq_dist.keys()), y = list(sum_of_sq_dist.values()))
plt.xlabel('Number of Clusters(k)')
plt.ylabel('Sum of Square Distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



In [104]:
```
from kneed import KneeLocator
kl = KneeLocator(list(sum_of_sq_dist.keys()),list(sum_of_sq_dist.values()),curve="convex", direction="decreasing")
kl.elbow
```

Out[104]: 4

Elbow method to find the optimal k

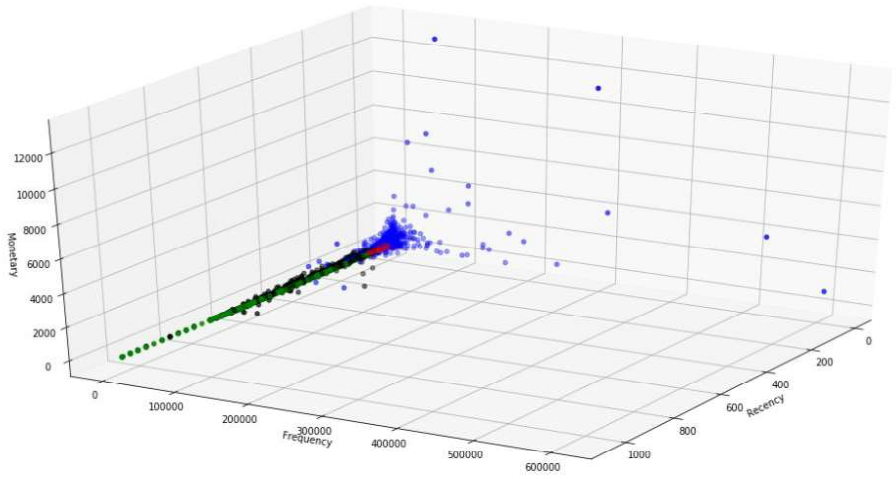
In [105]:
```
KMean_clust = KMeans(n_clusters= 4, init= 'k-means++', max_iter= 1000)
KMean_clust.fit(Scaled_Data)

#Find the clusters for the observation given in the dataset
r_table['Cluster'] = KMean_clust.labels_
r_table.head()
```

Out[105]:

CustomerID	Recency	Frequency	Monetary	R	F	M	RFMGroup	RFMScore	RFM_Loyalty_Level	s_color	Cluster
12346.0	326	34	77551.96	3	3	1	331	7	Gold	blue	3
12347.0	3	253	5633.32	1	1	1	111	3	Platinum	green	2
12348.0	76	51	2019.40	2	3	2	232	7	Gold	blue	3
12349.0	19	175	4428.69	1	1	1	111	3	Platinum	green	2
12350.0	311	17	334.40	3	4	4	344	11	Bronze	black	1

K means model for k=4 and updating the labels in a new column

In [111]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(20,10))
ax = plt.axes(projection='3d')
xline=r_table["Recency"]
yline=r_table["Frequency"]
zline=r_table['Monetary']
ax.scatter3D(xline, zline,yline,c = r_table['Color'])

ax.set_xlabel("Recency")
ax.set_ylabel("Frequency")
ax.set_zlabel('Monetary')
ax.view_init(30, 30)
```



3D plot of the clusters obtained from k means model

- Hence Loyal or Platinum users are the ones who buy frequently and spend more rather than being recent.

DBScan:

DBSCan

```
In [112]:
from sklearn.cluster import DBSCAN
from sklearn.metrics import adjusted_rand_score

In [113]:
dbscan = DBSCAN(eps=0.5,min_samples=3)

In [114]:
dbscan.fit(Scaled_Data)

Out[114]:
DBSCAN(min_samples=3)

In [115]:
r_table['Cluster1']=dbscan.labels_

In [116]:
r_table[r_table['Cluster1']==-1].head(5)
```

Out[116]:

CustomerID	Recency	Frequency	Monetary	R	F	M	RFMGroup	RFMScore	RFM_Loyalty_Level	s_color	Cluster	Color	Cluster1
12346.0	326	34	77551.96	3	3	1	331	7	Gold	blue	3	black	-1
12415.0	25	928	144151.29	1	1	1	111	3	Platinum	green	2	blue	-1
12482.0	370	204	23691.40	3	1	1	311	5	Platinum	green	2	blue	-1
12748.0	1	7230	56406.95	1	1	1	111	3	Platinum	green	2	blue	-1
12918.0	627	1	10953.50	4	4	1	441	9	Silver	red	1	green	-1

Evaluation of the model

- Using silhouette Score(When Actual cluster labels are not known)

```
In [121]:
#using silhoutte score since actual labels are not known

In [147]:
from sklearn.metrics import silhouette_score

In [148]:
dbscan_silhouette = silhouette_score(Scaled_Data, dbscan.labels_, metric='euclidean').round (2)
print("Silhoute Score for DBScan: ",dbscan_silhouette)

Silhoute Score for DBScan:  0.16

In [149]:
kmeans_silhouette = silhouette_score(Scaled_Data, KMean_clust.labels_, metric='euclidean').round (2)
print("Silhoute Score for Kmeans: ",kmeans_silhouette)

Silhoute Score for Kmeans:  0.33
```

- Using adjusted_rand_score (Considering RFM Classes as expected Labels)

```
In [150]:
#using labels from RFM Analysis

In [151]:
le.fit(r_table['RFM_Loyalty_Level'])
l=[i for i in range(len(r_table['RFM_Loyalty_Level'].unique()))]
dict(zip(list(le.classes_),l))

Out[151]: {0: 0, 1: 1, 2: 2, 3: 3}

In [152]:
r_table['RFM_Loyalty_Level'] = le.transform(r_table['RFM_Loyalty_Level'])

In [153]:
s1=adjusted_rand_score(r_table['RFM_Loyalty_Level'],r_table['Cluster'])
print(s1)

0.44709000563599444

In [145]:
s2=adjusted_rand_score(r_table['RFM_Loyalty_Level'],r_table['Cluster'])
print(s2)

0.44709000563599444
```

Conclusion of the Experiment wrt dataset.

We approached the customer segmentation problem from a behavioural aspect with the number of products ordered, date of purchase and total spending for each customer. Use of 3 features helped us with the understandability and visualization of the model- Recency, Frequency and Monetary value. The dataset was apt to perform an unsupervised machine learning problem.So 2 kinds of clustering i.e K means and DBscan are used and are compared with the RFM analysis.

- Kmeans had better accuracy score when compared to DBScan for this particular dataset
- K means segmented the dataset in the similar way as classes formed directly from RFM Model

With these customer groups or segments , retailers can customize and focus their marketing campaigns depending on the shopping behaviour of customers in the segment.
Hence the project plays a key role for retailers.

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?