



**THIRD YEAR
INFORMATION TECHNOLOGY
(2019 COURSE)**

**LABORATORY MANUAL
FOR**

**LABORATORY PRACTICE-I
(ML, DAA/ADBMS)**

SEMESTER - V

[Subject code: 314448]

[Prepared By]

Mr. R. B. Murumkar Mrs. S. A. Jakhete

Mr. J. K. Kamble Dr. S. C. Dharmadhikari

Mr. A. C. Karve Mrs. S. T. Kembhavi

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE VISION AND MISSION

VISION

Pune Institute of Computer Technology aspires to be the leader in higher technical education and research of international repute.

MISSION

To be leading and most sought after Institute of education and research in emerging engineering and technology disciplines that attracts, retains and sustains gifted individuals of significant potential.

DEPARTMENT VISION AND MISSION

VISION

The department endeavors to be recognized globally as a center of academic excellence & research in Information Technology.

MISSION

To inculcate research culture among students by imparting information technology related fundamental knowledge, recent technological trends and ethics to get recognized as globally acceptable and socially responsible professionals.

Savitribai Phule Pune University, Pune
Third Year Information Technology (2019 Course)
314448 : Laboratory Practice-I (Machine Learning)

Teaching Scheme:

Practical (PR) : 4 hrs/week

Credit Scheme:

02 Credits

Examination Scheme:

PR : 25 Marks

TW: 25 Marks

Prerequisites:

1. Python programming language

Course Objectives:

1. The objective of this course is to provide students with the fundamental elements of machine learning for classification, regression, clustering.
2. Design and evaluate the performance of a different machine learning models.

Course Outcomes:

On completion of the course, students will be able to—

CO1: Implement different supervised and unsupervised learning algorithms.

CO2: Evaluate performance of machine learning algorithms for real-world applications.

Guidelines for Instructor's Manual

The faculty member should prepare the laboratory manual for all the experiments and it should be made available to students and laboratory instructor/Assistant.

Guidelines for Student's Lab Journal

1. Students should submit term work in the form of a handwritten journal based on a specified list of assignments.
2. Practical Examination will be based on the term work.
3. Students are expected to know the theory involved in the experiment.
4. The practical examination should be conducted if and only if the journal of the candidate is complete in all respects.

Guidelines for Lab /TW Assessment

1. Examiners will assess the term work based on performance of students considering the parameters such as timely conduction of practical assignment, methodology adopted for implementation of practical assignment, timely submission of assignment in the form of handwritten write-up along with results of implemented assignment, attendance etc.
2. Examiners will judge the understanding of the practical performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.
3. Appropriate knowledge of usage of software and hardware related to respective laboratories should be as a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers of the program in a journal may be avoided. There must be hand-written write-ups for every assignment in the journal. The DVD/CD containing student programs should be attached to the journal by every student and the same to be maintained by the department/lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Guidelines for Laboratory Conduction

1. All the assignments should be implemented using python programming language
2. **Implement any 4 assignments out of 6**
3. **Assignment clustering with K-Means is compulsory**
4. The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic.
5. The instructor may frame multiple sets of assignments and distribute them among batches of students.
6. All the assignments should be conducted on multicore hardware and 64-bit open-sources software

Guidelines for Practical Examination

1. Both internal and external examiners should jointly set problem statements for practical examination. During practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement.
2. The supplementary and relevant questions may be asked at the time of evaluation to judge the student's understanding of the fundamentals, effective and efficient implementation.
3. The evaluation should be done by both external and internal examiners.

List of Laboratory Assignments

Group A

1. Data preparation:

Download heart dataset from following link.

<https://www.kaggle.com/zhaoyingzhu/heartcsv>

Perform following operation on given dataset.

- a) Find Shape of Data
- b) Find Missing Values
- c) Find data type of each column
- d) Finding out Zero's
- e) Find Mean age of patients
- f) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide dataset in training (75%) and testing (25%).

Through the diagnosis test I predicted 100 report as COVID positive, but only 45 of those were actually positive. Total 50 people in my sample were actually COVID positive. I have total 500 samples.

Create confusion matrix based on above data and find

- I. Accuracy
- II. Precision
- III. Recall
- IV. F-1 score

2. Assignment on Regression technique

Download temperature data from below link. <https://www.kaggle.com/venky73/temperatures-of-india?select=temperatures.csv>

This data consists of temperatures of INDIA averaging the temperatures of all places month wise. Temperatures values are recorded in CELSIUS

- a. Apply Linear Regression using suitable library function and predict the Month-wise

- temperature.
- b. Assess the performance of regression models using MSE, MAE and R-Square metrics
 - c. Visualize simple regression model.

3. Assignment on Classification technique

Every year many students give the GRE exam to get admission in foreign Universities. The data set contains GRE Scores (out of 340), TOEFL Scores (out of 120), University Rating (out of 5), Statement of Purpose strength (out of 5), Letter of Recommendation strength (out of 5), Undergraduate GPA (out of 10), Research Experience (0=no, 1=yes), Admitted (0=no, 1=yes). Admitted is the target variable.

Data Set Available on kaggle (The last column of the dataset needs to be changed to 0 or 1) Data Set : <https://www.kaggle.com/mohansacharya/graduate-admissions>

The counselor of the firm is supposed check whether the student will get an admission or not based on his/her GRE score and Academic Score. So to help the counselor to take appropriate decisions build a machine learning model classifier using Decision tree to predict whether a student will get admission or not.

Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if necessary.

Perform data-preparation (Train-Test Split)

C. Apply Machine Learning Algorithm

D. Evaluate Model.

4. Assignment on Improving Performance of Classifier Models

A SMS unsolicited mail (every now and then known as cell smartphone junk mail) is any junk message brought to a cellular phone as textual content messaging via the Short Message Service (SMS). Use probabilistic approach (Naive Bayes Classifier / Bayesian Network) to implement SMS Spam Filtering system. SMS messages are categorized as SPAM or HAM using features like length of message, word depend, unique keywords etc.

Download Data -Set from : <http://archive.ics.uci.edu/ml/datasets/sms+spam+collection>

This dataset is composed by just one text file, where each line has the correct class followed by the raw message.

- a. Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if necessary
- b. Perform data-preparation (Train-Test Split)
- c. Apply at least two Machine Learning Algorithms and Evaluate Models
- d. Apply Cross-Validation and Evaluate Models and compare performance.
- e. Apply Hyper parameter tuning and evaluate models and compare performance.

5. Assignment on Clustering Techniques

Download the following customer dataset from below link:

Data Set: <https://www.kaggle.com/shwetabh123/mall-customers>

This dataset gives the data of Income and money spent by the customers visiting a Shopping Mall. The data set contains Customer ID, Gender, Age, Annual Income, Spending Score. Therefore, as a mall owner you need to find the group of people who are the profitable customers for the mall owner. Apply at least two clustering algorithms (based on Spending Score) to find the group of customers.

- a. Apply Data pre-processing (Label Encoding , Data Transformation....) techniques if necessary.
- b. Perform data-preparation(Train-Test Split)

- c. Apply Machine Learning Algorithm
- d. Evaluate Model.
- e. Apply Cross-Validation and Evaluate Model

6. Assignment on Association Rule Learning

Download Market Basket Optimization dataset from below link.

Data Set: <https://www.kaggle.com/hemanthkumar05/market-basket-optimization>

This dataset comprises the list of transactions of a retail company over the period of one week. It contains a total of 7501 transaction records where each record consists of the list of items sold in one transaction. Using this record of transactions and items in each transaction, find the association rules between items.

There is no header in the dataset and the first row contains the first transaction, so mentioned header = None here while loading dataset.

- a. Follow following steps :
 - b. Data Preprocessing
 - c. Generate the list of transactions from the dataset
 - d. Train Apriori algorithm on the dataset
 - e. Visualize the list of rules
- F. Generated rules depend on the values of hyper parameters. By increasing the minimum confidence value and find the rules accordingly

7. Assignment on Multilayer Neural Network Model

Download the dataset of National Institute of Diabetes and Digestive and Kidney Diseases from below link :

Data Set: <https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv>

The dataset has total 9 attributes where the last attribute is “Class attribute” having values 0 and 1. (1=“Positive for Diabetes”, 0=“Negative”)

- a. Load the dataset in the program. Define the ANN Model with Keras. Define at least two hidden layers. Specify the ReLU function as activation function for the hidden layer and Sigmoid for the output layer.
- b. Compile the model with necessary parameters. Set the number of epochs and batch size and fit the model.
- c. Evaluate the performance of the model for different values of epochs and batch sizes.
- d. Evaluate model performance using different activation functions Visualize the model using ANN Visualizer.

Reference Books:

1. Ethem Alpaydin, Introduction to Machine Learning, PHI 2nd Edition-2013
2. Peter Flach: Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Cambridge University Press, Edition 2012.
3. Hastie, Tibshirani, Friedman: Introduction to Statistical Machine Learning with Applications in R, Springer, 2nd Edition 2012
4. Tom M. Mitchell , Machine Learning, 1997, McGraw-Hill, First EditionC. M. Bishop: Pattern Recognition and Machine Learning, Springer 1st Edition-2013.
5. Ian H Witten, Eibe Frank, Mark A Hall: Data Mining, Practical Machine Learning Tools and Techniques, Elsevier, 3rd Edition
6. Hastie, Tibshirani, Friedman: Introduction to Statistical Machine Learning with Applications in R, Springer, 2nd Edition 2012.

- 7. Kevin P Murphy: Machine Learning – A Probabilistic Perspective, MIT Press, August 2012.
- 8. Shalev-Shwartz S., Ben-David S., Understanding Machine Learning: From Theory to Algorithms, CUP, 2014
- 9. Jack Zurada: Introduction to Artificial Neural Systems, PWS Publishing Co. Boston, 2002

Virtual Laboratory:

- 1. http://vlabs.iitb.ac.in/vlabs-dev/labs/machine_learning/labs/index.php

Savitribai Phule Pune University, Pune
Third Year Information Technology (2019 Course)
314448 (A) : Laboratory Practice-I (Design and Analysis of Algorithm)

Teaching Scheme:	Credit Scheme	Examination Scheme:
Practical (PR) : 4 hrs/week	02 Credits	PR: 25 Marks TW: 25 Marks

Prerequisites:

1. Data Structures and Algorithms.
2. Discrete Structures.
3. C/C++ programming

Course Objectives:

1. To learn the various algorithmic design strategies.
2. To apply efficiently in problem solving.

Course Outcomes:

On completion of the course, students will be able to—

CO1: Implement the various algorithmic design strategies and use it to solve real time problems/applications

CO2: Apply Divide & Conquer as well as Greedy approach to design algorithms.

CO3: Analyze optimization problems using dynamic programming.

Guidelines for Instructor's Manual

The faculty member should prepare the laboratory manual for all the experiments and it should be made available to students and laboratory instructor/Assistant.

Guidelines for Student's Lab Journal

1. Students should submit term work in the form of a handwritten journal based on a specified list of assignments.
2. Practical Examination will be based on the term work.
3. Candidate is expected to know the theory involved in the experiment.
4. The practical examination should be conducted if and only if the journal of the candidate is complete in all respects.

Guidelines for Lab /TW Assessment

1. Examiners will assess the term work based on performance of students considering the parameters such as timely conduction of practical assignment, methodology adopted for implementation of practical assignment, timely submission of assignment in the form of handwritten write-up along with results of implemented assignment, attendance etc.
2. Examiners will judge the understanding of the practical performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.
3. Appropriate knowledge of usage of software and hardware related to respective laboratories should be as a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers of the program in a journal may be avoided. There must be handwritten write-ups for every assignment in the journal. The DVD/CD containing student programs

should be attached to the journal by every student and the same to be maintained by the department/lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Guidelines for Laboratory Conduction

1. The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic.
2. The instructor may set multiple sets of assignments and distribute them among batches of students. It is appreciated if the assignments are based on real world problems/applications.
3. All the assignments should be conducted on multicore hardware and 64-bit open-source software

Guidelines for Practical Examination

1. Both internal and external examiners should jointly set problem statements for practical examination. During practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement.
2. The supplementary and relevant questions may be asked at the time of evaluation to judge the student's understanding of the fundamentals, effective and efficient implementation.

The evaluation should be done by both external and internal examiners.

List of Laboratory Assignments

1. Write a program to implement Fractional knapsack using Greedy algorithm and 0/1 knapsack using dynamic programming. Show that Greedy strategy does not necessarily yield an optimal solution over a dynamic programming approach.
2. Write a program to implement Bellman-Ford Algorithm using Dynamic Programming and verify the time complexity
3. Write a recursive program to find the solution of placing n queens on the chessboard so that no two queens attack each other using Backtracking.
4. Write a program to solve the travelling salesman problem and to print the path and the cost using LC Branch and Bound.

Reference Books

1. Horowitz and Sahani, Fundamentals of computer Algorithms, Universities Press, ISBN : 9788173716126

<p style="text-align: center;">Savitribai Phule Pune University, Pune Third Year Information Technology (2019 Course) 314448 (B) : Laboratory Practice-I (ADBMS)</p>		
Teaching Scheme:	Credit Scheme	Examination Scheme:
Practical (PR) :4 hrs/week	02 Credits	PR : 25 Marks TW : 25 Marks
<p>Prerequisites:</p> <p>1. Database Management System</p>		
<p>Course Objectives:</p> <ol style="list-style-type: none">1. To learn and understand Database Modeling, Architectures.2. To learn and understand Advanced Database Programming Frameworks.3. To learn NoSQL Databases (Open source) such as MongoDB.4. To design and develop application using NoSQL Database.5. To design data warehouse schema for given system.		
<p>Course Outcomes:</p> <p>On completion of the course, students will be able to</p> <p>CO1: Apply advanced Database Programming Languages.</p> <p>CO2: Apply the concepts of NoSQL Databases.</p> <p>CO3: Install and configure database systems.</p> <p>CO4: Populate and query a database using MongoDB commands.</p> <p>CO5: Design data warehouse schema of any one real-time: CASE STUDY</p> <p>CO6: Develop small application with NoSQL Database for back-end.</p>		
<p>Guidelines for Instructor's Manual</p> <p>The faculty member should prepare the laboratory manual for all the experiments and it should be made available to students and laboratory instructor/Assistant.</p>		
<p>Guidelines for Student's Lab Journal</p> <ol style="list-style-type: none">1. Student should submit term work in the form of handwritten journal based on specified list of assignments.2. Practical Examination will be based on all the assignments in the lab manual3. Candidate is expected to know the theory involved in the experiment.4. The practical examination should be conducted if and only if the journal of the candidate is complete in all respects.		

Guidelines for Lab /TW Assessment

1. Examiners will assess the student based on performance of students considering the parameters such as timely conduction of practical assignment, methodology adopted for implementation of practical assignment, timely submission of assignment in the form of handwritten write-up along with results of implemented assignment, attendance etc.
2. Appropriate knowledge of usage of software and hardware related to respective laboratory should be checked by the concerned faculty member.
3. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers of the program in journal may be avoided. There must be hand-written write-ups for every assignment in the journal. The DVD/CD containing student's programs should be attached to the journal by every student and same to be maintained by department/lab In- charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Guidelines for Laboratory Conduction

1. Group A assignments are compulsory and should be performed by individual student.
2. Group B case study may be performed in group of 3/4.
3. Mini project of Group C can be implemented using any suitable front-end. But back-end must be MongoDB.

Guidelines for Practical Examination

1. Practical Examination will be based on the all topics covered.
2. Examiners will judge the understanding of the practical performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.

List of Laboratory Assignments

Group A : MongoDB

1. Create a database with suitable example using MongoDB and implement
 - Inserting and saving document (batch insert, insert validation)
 - Removing document
 - Updating document (document replacement, using modifiers, up inserts, updating multiple documents, returning updated documents)
 - Execute at least 10 queries on any suitable MongoDB database that demonstrates following:
 - a. Find and find One (specific values)
 - b. Query criteria (Query conditionals, OR queries, \$not, Conditional semantics) Type-specific queries (Null, Regular expression, Querying arrays)
 - c. \$ where queries
 - d. Cursors (Limit, skip, sort, advanced query options)

2. Implement Map-reduce and aggregation, indexing with suitable example in MongoDB.

Demonstrate the following:

- Aggregation framework
- Create and drop different types of indexes and explain () to show the advantage of the indexes.

3. Case Study: Design conceptual model using Star and Snowflake schema for any one database.

4. Mini Project

Pre-requisite: Build the mini project based on the requirement document and design prepared as a part of Database Management Lab in second year.

1. Form teams of around 3 to 4 people.

2. Develop the application:

Build a suitable GUI by using forms and placing the controls on it for any application. Proper data entry validations are expected.

Add the database connection with front end. Implement the basic CRUD operations.

3. Prepare and submit report to include: Title of the Project, Abstract, List the hardware and software requirements at the backend and at the front end, Source Code , Graphical User Interface, Conclusion.

Reference Books:

1. Silberschatz A., Korth H., Sudarshan S., "Database System Concepts", 6th Edition, McGraw Hill Publishers, ISBN 0-07-120413-X.
2. Kristina Chodorow, MongoDB The definitive guide, O'Reilly Publications, ISBN:978-93-5110-269-4, 2nd Edition.
3. Jiawei Han, Micheline Kamber, Jian Pei "Data Mining: concepts and techniques", 2nd Edition, Publisher: Elsevier/Morgan Kaufmann.
4. <http://nosql-database.org/>.

PART A : Machine Learning

Assignment No -1

Title: Data preparation

Problem Statement:

Perform following operation on given dataset:

- a) Find Shape of Data
- b) Find Missing Values
- c) Find data type of each column
- d) Finding out Zero's
- e) Find Mean age of patients
- f) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide dataset in training (75%) and testing (25%).
- g) Through the diagnosis test I predicted 100 report as COVID positive, but only 45 of those were actually positive. Total 50 people in my sample were actually COVID positive. I have total 500 samples.

Create confusion matrix based on above data and find

- i. Accuracy
- ii. Precision
- iii. Recall
- iv. F-1 score

Objective: This assignment will help the students to realize what is need of data preparation

S/W Packages and H/W apparatus used:

Linux OS: Ubuntu/Windows , Jupyter notebook.

PC with the configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15"Color Monitor, Keyboard, Mouse

References:

1. Ethem Alpaydin, Introduction to Machine Learning, PHI 2nd Edition,2013.
2. Peter Flach: Machine Learning: The Art and Science of Algorithms that Make Sense of Data,Cambridge University Press, Edition 2012.

Theory:

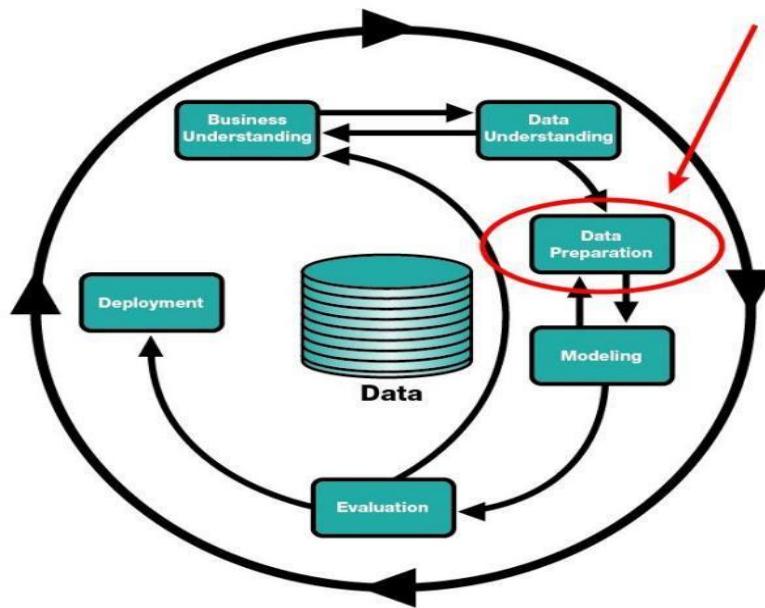
Data Preparation

Data preparation (also referred to as “data preprocessing”) is the process of transforming raw data so that data scientists and analysts can run it through machine learning algorithms to uncover insights or make predictions.

Why is Data Preparation Important?

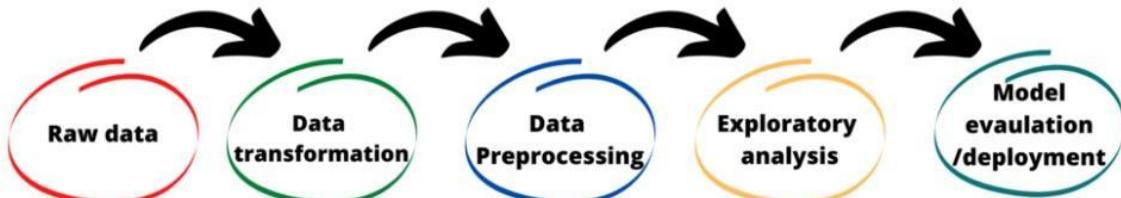
Most machine learning algorithms require data to be formatted in a very specific way, so datasets generally require some amount of preparation before they can yield useful insights. Some datasets have values that are missing, invalid, or otherwise difficult for an algorithm to process. If data is missing, the algorithm can't use it. If data is invalid, the algorithm produces less accurate or even misleading outcomes. Some datasets are relatively clean but need to be shaped (e.g., aggregated or pivoted) and many datasets are just lacking useful business context (e.g., poorly defined ID values), hence the need for feature enrichment. Good data preparation produces clean and well-curated data which leads to more practical, accurate model outcomes.

It is the most required process before feeding the data into the machine learning model. The reason behind that the data set needs to be different and specific according to the model so that we have to find out the required features of that data. The data preparation process offers a method via which we can prepare the data for defining the project and also for the project evaluation of ML algorithms. Different many predicting machine learning models are there with a different process but some of the processes are common that are performed in every model, and also it allows us to find out the actual business problem and their solutions. Some of the data preparation processes are:



1. Determine the problems
2. Data cleaning
3. Feature selection

4. Data transformation
5. feature engineering
6. Dimensionality reduction



1. Determine the problems:

This step tells us about the learning method of the project to find out the results for future prediction or forecasting. For example, which ML model suitable for the data set regression or classification or clustering algorithms. This includes data collection that is useful for predicting the result and also involving the communication to project stakeholders and domain expertise. We use classification and regression models for categorical and numerical data respectively.

It includes determining the relevant attributes with the stored data in form of .csv, .html, .json, .doc, and many, also for unstructured data in a form for audio, video, text, images, etc for scanning and detect the patterns of data with searching and identifying the data that have taken from external repositories.

2. Data cleaning:

After collecting the data, it is very necessary to clean that data and make it proper for the ML model. It includes solving problems like outliers, inconsistency, missing values, incorrect, skewed, and trends. Cleaning the data is very important as the model learning from that data only, so if we feed inconsistent, appropriate data to model it will return garbage only, so it is required to make sure that the data does not contain any unseen problem. For example, if we have a data set of sales, it might be possible that it contains some features like height, age, that cannot help in the model building so we can remove it. We generally remove the null values columns, fill the missing values, make the data set consistent, and remove the outliers and skewed data in data cleaning.

3. Feature selection:

Sometimes we face the problem of identifying the related features from the set of data and deleting the irrelevant and less important data without touching the target variables to get the better accuracy of the model. Features selection plays a wide role in building a machine learning model that impacts the performance and accuracy of the model. It is that process which contributes mostly to the predictions or output that we need by selecting the features automatically or manually. If we have irrelevant data that would cause the model with overfitting and underfitting.

The benefits of feature selection:

1. Reduce the overfitting/underfitting
2. Improves the accuracy
3. Reduced training/testing time
4. Improves performance

4. Data transformation:

Data transformation is the process that converts the data from one form to another. It is required for data integration and data management. In data transformation, we can change the types of data, clear the data removing the null values or duplicate values, and get enrich data that depends on the requirements of the model. It allows us to perform data mapping that determines how individual features are mapped, modified, filtered, aggregated, and joined. Data transformation is needed for both structured and unstructured data but it is time consuming, costly, slow.

5. Feature engineering:

Every ML algorithms use some input data for giving required output and this input required some features which are in a structured form. To get the proper result the algorithms required features with some specific characteristics which we find out with feature engineering. we need to perform different feature engineering on different datasets and we can observe their effect on model performance. Here I am listing out the techniques of feature engineering.

1. Imputation
2. Handling outliers
3. Binning

4. Log transform
5. one-hot encoding
6. Grouping operations
7. Feature split
8. Scaling

6. Dimensionality reduction:

When we use the dataset for building an ML model, we need to work with 1000s of features that cause the curse of dimensionality, or we can say that it refers to the process to convert a set of data. For the ML model, we have to access a large amount of data and that large amount of data can lead us in a situation where we can take possible data that can be available to feed it into a forecasting model to predict and give the result of the target variable. It reduced the time that is required for training and testing our machine learning model and also helps to eliminate overfitting. It is kind of zipping the data for the model.

Implementation:

```
import numpy as nm
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

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

```
df.head(8)
```

	Unmed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes

2	3	67	1	asymptomatic	120	22 9	0	2	129	1	2.6	2	2. 0	reversible	Yes
3	4	37	1	nonanginal	130	25 0	0	0	187	0	3.5	3	0. 0	normal	No
4	5	41	0	nontypical	130	20 4	0	2	172	0	1.4	1	0. 0	normal	No
5	6	56	1	nontypical	120	23 6	0	0	178	0	0.8	1	0. 0	normal	No
6	7	62	0	asymptomatic	140	26 8	0	2	160	0	3.6	3	2. 0	normal	Yes
7	8	57	0	asymptomatic	120	35 4	0	0	163	1	0.6	1	0. 0	normal	No

In []:

df.shape

```
(303, 15)
df.isnull().sum()
```

Out[]:

Unnamed: 0	0
Age	0
Sex	0
ChestPain	0
RestBP	0
Chol	0
Fbs	0
RestECG	0
MaxHR	0
ExAng	0
Oldpeak	0
Slope	0
Ca	4
Thal	2

```

AHD          0
dtype: int64

In []:
print("Total missing values: ", df.isnull().sum().sum())
Total missing values:  6
In []:
df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Unnamed: 0    303 non-null    int64  
 1   Age         303 non-null    int64  
 2   Sex         303 non-null    int64  
 3   ChestPain   303 non-null    object  
 4   RestBP      303 non-null    int64  
 5   Chol        303 non-null    int64  
 6   Fbs         303 non-null    int64  
 7   RestECG     303 non-null    int64  
 8   MaxHR       303 non-null    int64  
 9   ExAng       303 non-null    int64  
 10  Oldpeak     303 non-null    float64 
 11  Slope        303 non-null    int64  
 12  Ca           299 non-null    float64 
 13  Thal         301 non-null    object  
 14  AHD          303 non-null    object  

dtypes: float64(2), int64(10), object(3)
memory usage: 32.0+ KB

In []:
df.dtypes
Out[]:
Unnamed: 0      int64
Age             int64

```

```

Sex           int64
ChestPain    object
RestBP        int64
Chol          int64
Fbs           int64
RestECG       int64
MaxHR         int64
ExAng         int64
Oldpeak       float64
Slope          int64
Ca             float64
Thal           object
AHD            object
dtype: object

```

In [:]

```
(df == 0).sum(axis=0)
```

Out[:]

```

Unnamed: 0      0
Age            0
Sex             97
ChestPain     0
RestBP         0
Chol           0
Fbs            258
RestECG        151
MaxHR          0
ExAng          204
Oldpeak        99
Slope          0
Ca              176
Thal            0
AHD             0
dtype: int64

```

```
In []:
mean_age = df['Age'].mean()

mean_age
```

```
Out []:
54.43894389438944
```

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

```
Out []:
Index(['Unnamed: 0', 'Age', 'Sex', 'ChestPain', 'RestBP', 'Chol',
       'Fbs',
       'RestECG', 'MaxHR', 'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal',
       'AHD'],
      dtype='object')
```

```
In []:
df2 = df.filter(['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol'])
```

```
In []:
df2
```

```
Out []:
   Age  Sex  ChestPain  RestBP  Chol
0    63     1    typical     145    233
1    67     1  asymptomatic     160    286
2    67     1  asymptomatic     120    229
3    37     1  nonanginal     130    250
4    41     0  nontypical     130    204
...
298   45     1    typical     110    264
299   68     1  asymptomatic     144    193
300   57     1  asymptomatic     130    131
```

```
301    57    0    nontypical    130    236
```

```
302    38    1    nonanginal    138    175
```

303 rows × 5 columns

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

```
mode = df['Thal'].mode().iloc[0]
df['Thal'].fillna(value=mode, inplace=True)
```

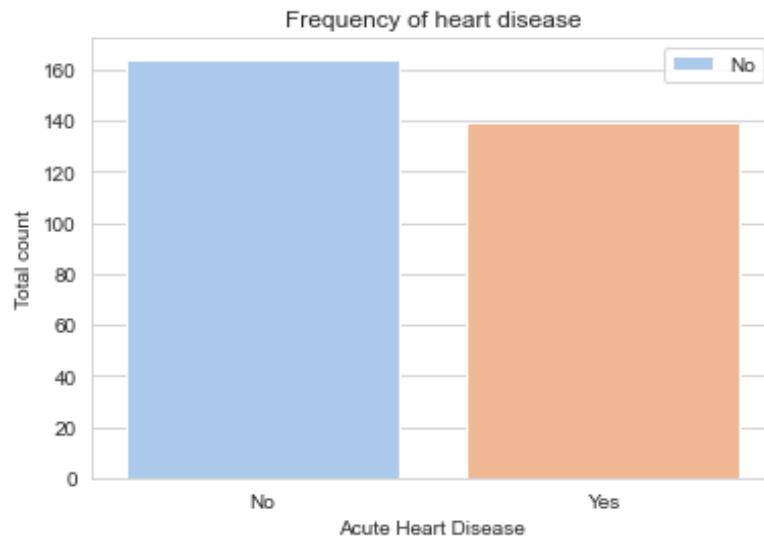
```
df.isnull().sum()
```

```
Unnamed: 0    0
Age          0
Sex          0
ChestPain   0
RestBP       0
Chol         0
Fbs          0
RestECG     0
MaxHR        0
ExAng        0
Oldpeak      0
Slope        0
Ca           0
Thal          0
AHD          0
dtype: int64
```

```
sns.countplot(x='AHD', data=df, palette='pastel')
sns.set_style("whitegrid")
```

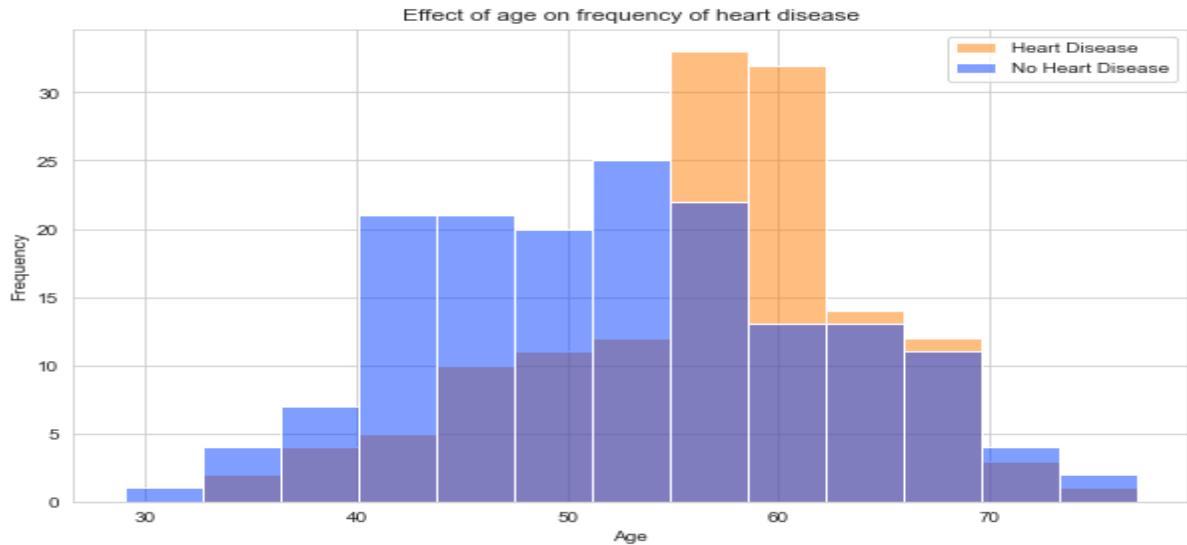
```
plt.xlabel("Acute Heart Disease")
plt.ylabel("Total count")
plt.title("Frequency of heart disease")
plt.legend(['No', 'Yes'], loc='upper right')
```

```
<matplotlib.legend.Legend at 0x6d758f8>
```



In [:]

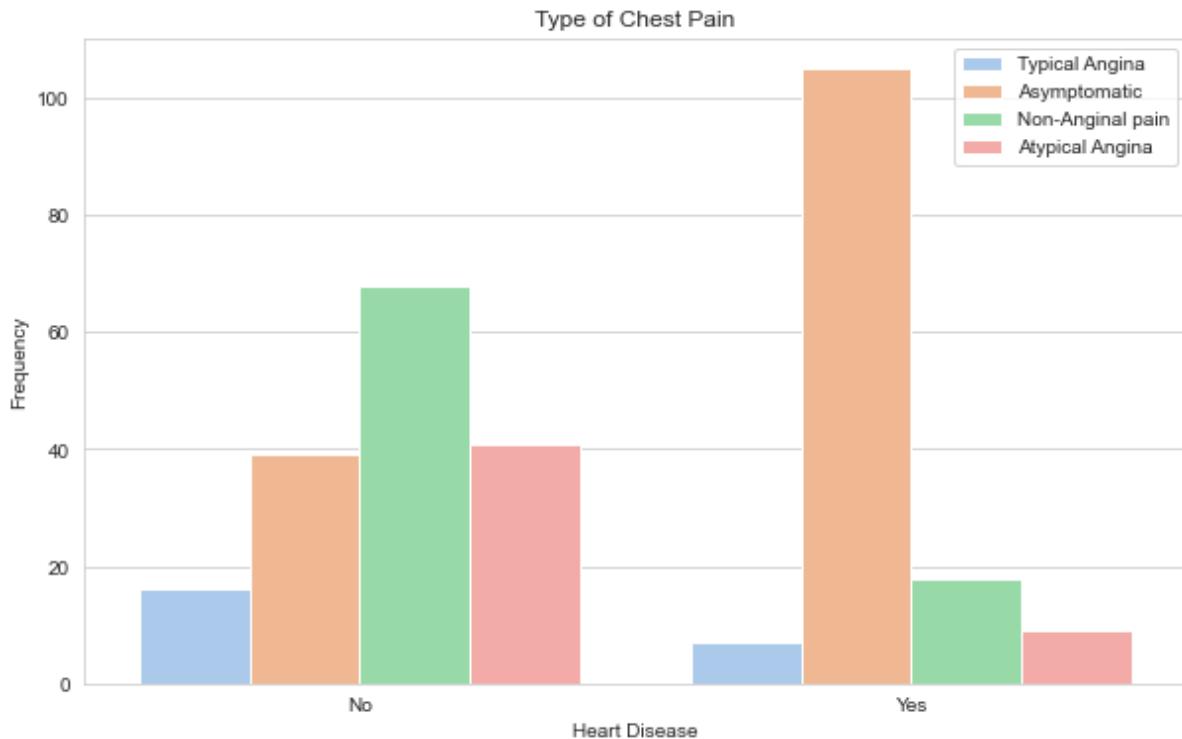
```
fig, ax = plt.subplots()
fig.set_size_inches(10, 6)
sns.histplot(x="Age", data=df, hue="AHD", palette="bright")
sns.set_style("whitegrid")
plt.title("Effect of age on frequency of heart disease")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.legend(["Heart Disease", "No Heart Disease"])
plt.show()
```



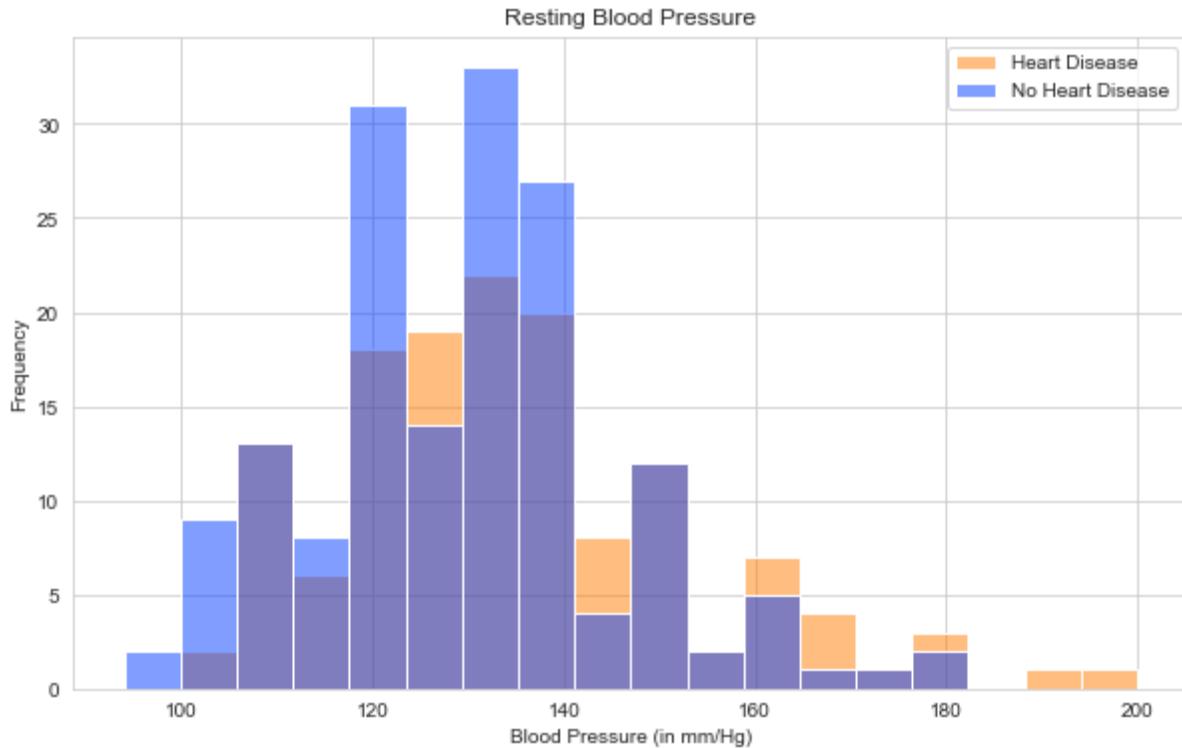
```
df['ChestPain'].unique()
```

```
array(['typical', 'asymptomatic', 'nonanginal', 'nontypical'],
      dtype=object)
```

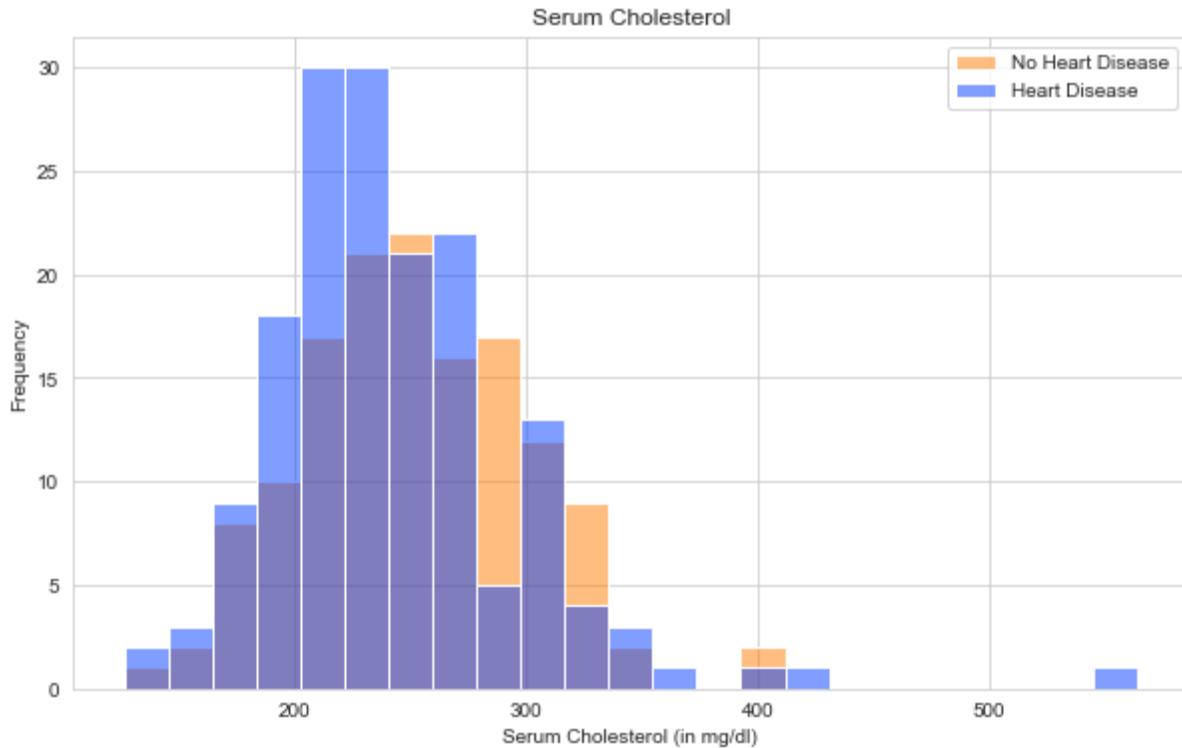
```
fig, ax = plt.subplots()
fig.set_size_inches(10, 6)
sns.countplot(x="AHD", hue="ChestPain", data=df, palette="pastel")
plt.title("Type of Chest Pain")
plt.xlabel("Heart Disease")
plt.ylabel("Frequency")
plt.legend(["Typical Angina", "Asymptomatic", "Non-Anginal pain",
           "Atypical Angina"])
plt.show()
```



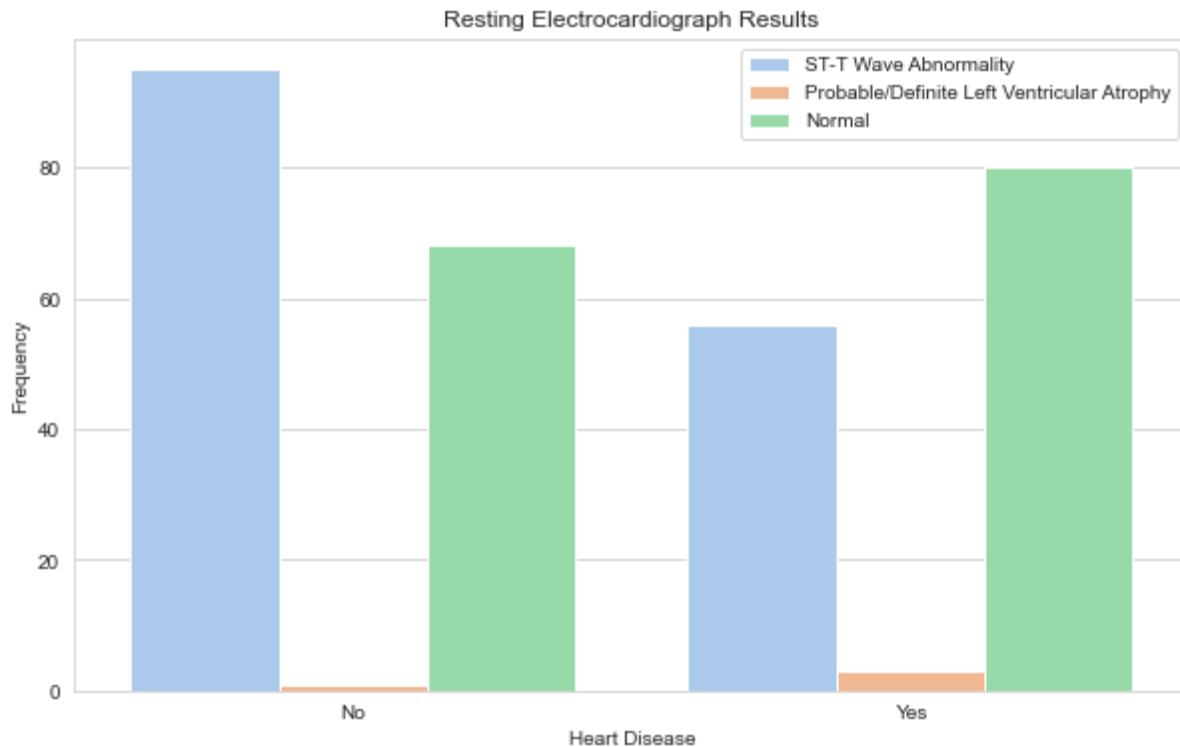
```
fig, ax = plt.subplots()
fig.set_size_inches(10, 6)
sns.histplot(x="RestBP" , data=df, hue="AHD", palette="bright")
sns.set_style("whitegrid")
plt.title("Resting Blood Pressure")
plt.xlabel("Blood Pressure (in mm/Hg) ")
plt.ylabel("Frequency")
plt.legend(["Heart Disease","No Heart Disease"])
plt.show()
```



```
fig, ax = plt.subplots()
fig.set_size_inches(10, 6)
sns.histplot(x="Chol", data=df, hue="AHD", palette="bright")
sns.set_style("whitegrid")
plt.title("Serum Cholesterol")
plt.xlabel("Serum Cholesterol (in mg/dl)")
plt.ylabel("Frequency")
plt.legend(["No Heart Disease", "Heart Disease"])
plt.show()
```



```
fig, ax = plt.subplots()
fig.set_size_inches(10, 6)
sns.countplot(x="AHD", hue="RestECG", data=df, palette="pastel")
plt.title("Resting Electrocardiograph Results")
plt.xlabel("Heart Disease")
plt.ylabel("Frequency")
plt.legend(["ST-T Wave Abnormality", "Probable/Definite Left
Ventricular Atrophy", "Normal"])
plt.show()
```



```
X = df[['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol', 'RestECG', 'MaxHR']]
Y = df['AHD']
```

```
#X= df.values
#Y= df.AHD.values
```

```
from sklearn.model_selection import train_test_split

X_train, Y_train, X_test, Y_test =
train_test_split(X, Y, test_size=0.25)
```

```
X_train.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 227 entries, 245 to 231
Data columns (total 7 columns):
 #   Column      Non-Null Count Dtype  
 --- 
 0   Age         227 non-null   int64  
 1   Sex         227 non-null   int64  
 2   ChestPain  227 non-null   object  
 3   RestBP     227 non-null   int64  
 4   Chol        227 non-null   int64  
 5   RestECG    227 non-null   object  
 6   MaxHR      227 non-null   int64
```

```

2   ChestPain    227 non-null    object
3   RestBP        227 non-null    int64
4   Chol          227 non-null    int64
5   RestECG       227 non-null    int64

6   MaxHR         227 non-null    int64
dtypes: int64(6), object(1)
memory usage: 13.3+ KB

```

```

Y_train.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 76 entries, 121 to 53
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Age         76 non-null    int64  
 1   Sex          76 non-null    int64  
 2   ChestPain    76 non-null    object  
 3   RestBP       76 non-null    int64  
 4   Chol         76 non-null    int64  
 5   RestECG      76 non-null    int64  
 6   MaxHR        76 non-null    int64  

dtypes: int64(6), object(1)
memory usage: 4.5+ KB

```

Conclusion:

Data preparation is recognized for helping businesses and analytics to get ready and prepare the data for operations.

References:

[1]<https://medium.com/@learnbay/6-most-important-steps-for-data-preparation-in-machine-learning-61ae88ab8628>

Assignment No -2

Title: Regression technique

Problem Statement:

This data consists of temperatures of INDIA averaging the temperatures of all place's month wise. Temperatures values are recorded in CELSIUS

- a) Apply Linear Regression using suitable library function and predict the Month-wise temperature.
- b) Assess the performance of regression models using MSE, MAE and R-Square metrics
- c) Visualize simple regression model.

Objective: This assignment will help the students to realize how the Linear Regression can be used and predictions using the same can be performed.

S/W Packages and H/W apparatus used:

Linux OS: Ubuntu/Windows, Jupyter notebook.
PC with the configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD,
15''Color Monitor, Keyboard, Mouse

References:

1. Ethem Alpaydin, Introduction to Machine Learning, PHI 2nd Edition,2013.
2. Peter Flach: Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Cambridge University Press, Edition 2012.

Theory:

Definition of Linear Regression

In layman terms, we can define linear regression as **it is used for learning the linear relationship between the target and one or more forecasters**, and it is probably one of the most popular and well inferential algorithms in statistics. Linear regression endeavours to demonstrate the connection between two variables by fitting a linear equation to observed information. One variable is viewed as an explanatory variable, and the other is viewed as a dependent variable.

Types of Linear Regression

Normally, linear regression is divided into two types: Multiple linear regression and Simple linear regression.

1. Multiple Linear Regression

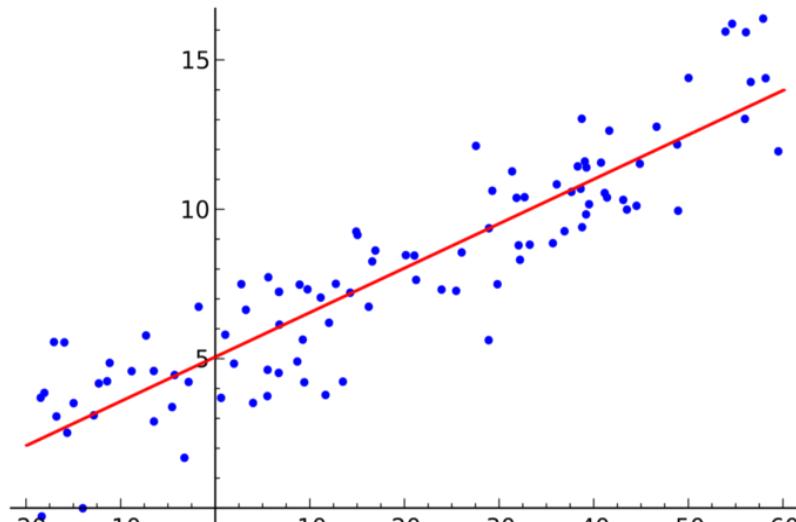
In this type of linear regression, we always attempt to discover the relationship between two or more independent variables or inputs and the corresponding dependent variable or output and the independent variables can be either continuous or categorical.

This linear regression analysis is very helpful in several ways like it helps in foreseeing trends, future values, and moreover predict the impacts of changes.

2. Simple Linear Regression

In simple linear regression, we aim to reveal the relationship between a single independent variable or you can say input, and a corresponding dependent variable or output. We can discuss this in a simple line as $y = \beta_0 + \beta_1 x + \epsilon$

Here, Y speaks to the output or dependent variable, β_0 and β_1 are two obscure constants that speak to the intercept and coefficient that is slope separately, and the error term is ϵ Epsilon. We can also discuss this in the form of a graph and here is a sample simple linear regression model graph.



Simple Linear Regression graph

What Actually is Simple Linear Regression?

It can be described as a method of statistical analysis that can be used to study the relationship between two quantitative variables.

Primarily, there are two things which can be found out by using the method of simple linear regression:

1. **Strength of the relationship between the given duo of variables.** (For example, the relationship between global warming and the melting of glaciers)
2. **How much the value of the dependent variable is at a given value of the independent variable.** (For example, the amount of melting of a glacier at a certain level of global warming or temperature)

Regression models are used for the elaborated explanation of the relationship between two given variables. There are certain types of regression models like logistic regression models, nonlinear regression models, and linear regression models. The linear regression model fits a straight line into the summarized data to establish the relationship between two variables.

Assumptions of Linear Regression

To conduct a simple linear regression, one has to make certain assumptions about the data. This is because it is a parametric test. The assumptions used while performing a simple linear regression are as follows:

- **Homogeneity of variance (homoscedasticity)-** One of the main predictions in a simple linear regression method is that the size of the error stays constant. This simply means that in the value of the independent variable, the error size never changes significantly.
- **Independence of observations-** All the relationships between the observations are transparent, which means that nothing is hidden, and only valid sampling methods are used during the collection of data.
- **Normality-** There is a normal rate of flow in the data.

These three are the assumptions of regression methods.

However, there is one additional assumption that has to be taken into consideration while specifically conducting a linear regression.

- **The line is always a straight line-** There is no curve or grouping factor during the conduction of a linear regression. There is a linear relationship between the variables (dependent variable and independent variable). If the data fails the assumptions of

homoscedasticity or normality, a nonparametric test might be used. (For example, the Spearman rank test)

Example of data that fails to meet the assumptions: One may think that cured meat consumption and the incidence of colorectal cancer in the U.S have a linear relationship. But later on, it comes to the knowledge that there is a very high range difference between the collection of data of both the variables. Since the homoscedasticity assumption is being violated here, there can be no linear regression test. However, a Spearman rank test can be performed to know about the relationship between the given variables.

Applications of Simple Linear Regression

1. **Marks scored by students based on number of hours studied (ideally)-** Here marks scored in exams are dependent and the number of hours studied is independent.
2. **Predicting crop yields based on the amount of rainfall-** Yield is a dependent variable while the measure of precipitation is an independent variable.
3. **Predicting the Salary of a person based on years of experience-** Therefore, Experience becomes the independent while Salary turns into the dependent variable.

Limitations of Simple Linear Regression

Indeed, even the best information doesn't recount a total story. Regression investigation is ordinarily utilized in examination to set up that a relationship exists between variables. However, correlation isn't equivalent to causation: a connection between two variables doesn't mean one causes the other to occur. Indeed, even a line in a simple linear regression that fits the information focuses well may not ensure a circumstances and logical results relationship.

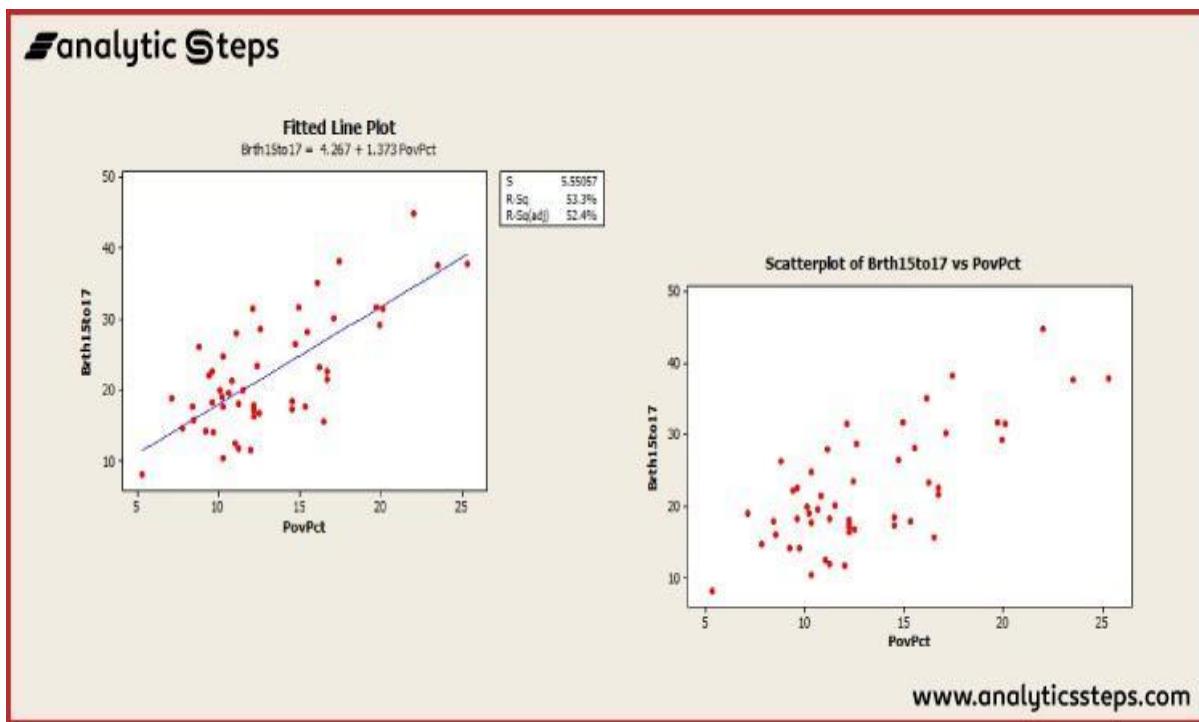
Utilizing a linear regression model will permit you to find whether a connection between variables exists by any means. To see precisely what that relationship is and whether one variable cause another, you will require extra examination and statistical analysis.

Examples of Simple Linear Regression

Now, let's move towards understanding simple linear regression with the help of an example. We will take an example of teen birth rate and poverty level data.

This dataset of size $n = 51$ is for the 50 states and the District of Columbia in the United States. The variables are y = year 2002 birth rate for each 1000 females 15 to 17 years of age and x = destitution rate, which is the percent of the state's populace living in families with wages underneath the governmentally characterized neediness level. (Information source: Mind On Statistics, 3rd version, Utts and Heckard).

Below is the graph (right image) in which you can see the (birth rate on the vertical) is indicating a normally linear relationship, on average, with a positive slope. As the poverty level builds, the birth rate for 15 to 17-year-old females will in general increment too.



Example graph of simple linear regression

Here is another graph (left graph) which is showing a regression line superimposed on the data.

The condition of the fitted regression line is given close to the highest point of the plot. The condition should express that it is for the "average" birth rate (or "anticipated" birth rate would be alright as well) as a regression condition portrays the normal estimation of y as a component of at least one x -variables. In statistical documentation, the condition could be composed $y^=4.267+1.373x$.

- The interpretation of the slope (value = 1.373) is that the 15 to 17-year-old birth rate increases 1.373 units, on average, for each one unit (one per cent) increase in the poverty rate.

- The translation of the intercept (value=4.267) is that if there were states with a population rate = 0, the anticipated normal for the 15 to 17-year-old birth rate would be 4.267 for those states. Since there are no states with a poverty rate = 0 this understanding of the catch isn't basically significant for this model.
- In the chart with a regression line present, we additionally observe the data that $s = 5.55057$ and $r^2 = 53.3\%$.
- The estimation of s discloses to us generally the standard deviation of the contrasts between the y -estimations of individual perceptions and expectations of y dependent on the regression line. The estimation of r^2 can be deciphered to imply that destitution rates "clarify" 53.3% of the noticed variety in the 15 to 17-year-old normal birth paces of the states.

The R₂ (adj) value (52.4%) is a change in accordance with R₂ dependent on the number of x-variables in the model (just one here) and the example size. With just a single x-variable, the charged R₂ isn't significant.

Implementation:

```
#Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the csv file

```
from google.colab import files
uploaded = files.upload()

#Read Student Grades .csv file and divide the data into dependent and independent variables.
data = pd.read_csv('Student_Grades_Data.csv')
data.head()

data.shape

(50, 2)

X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values

X
array([[ 1],
       [ 5],
       [ 7],
       [ 3],
```

```
[ 2],  
[ 9],  
[ 6],  
[12],  
[11],  
[ 2],  
[ 4],  
[ 8],  
[13],  
[ 9],  
[14],  
[10],  
[ 6],  
[12],  
[ 1],  
[ 4],  
[14],  
[10],  
[11],  
[ 4],  
[ 5],  
[ 8],  
[ 1],  
[ 2],  
[ 3],  
[ 7],  
[ 8],  
[14],  
[ 7],  
[ 8],  
[ 1],  
[ 2],  
[ 3],  
[ 4],  
[ 5],  
[ 6],  
[ 7],  
[ 8],  
[ 9],  
[10],  
[11],  
[12],  
[13],  
[14],  
[ 8],  
[ 2]]))
```

Y

```
array([1.5, 2.7, 3.1, 2.1, 1.8, 3.9, 2.9, 4.5, 4.3, 1.8, 2.4, 3.5, 4.8,  
3.9, 5. , 4.1, 2.9, 4.5, 1.5, 2.4, 5. , 4.1, 4.3, 2.4, 2.7, 3.5,  
1.5, 1.8, 2.1, 3.1, 3.5, 5. , 3.1, 3.5, 1.5, 1.8, 2.1, 2.4, 2.7,
```

```
2.9, 3.1, 3.5, 3.9, 4.1, 4.3, 4.5, 4.8, 5. , 3.5, 1.8])

#Split the data into training and test datasets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
random_state = 0)

y_test

array([2.1, 3.5, 2.4, 3.5, 3.1, 1.8, 2.7, 5. , 4.3, 1.8, 3.5, 1.8, 1.5,
1.5, 1.5])

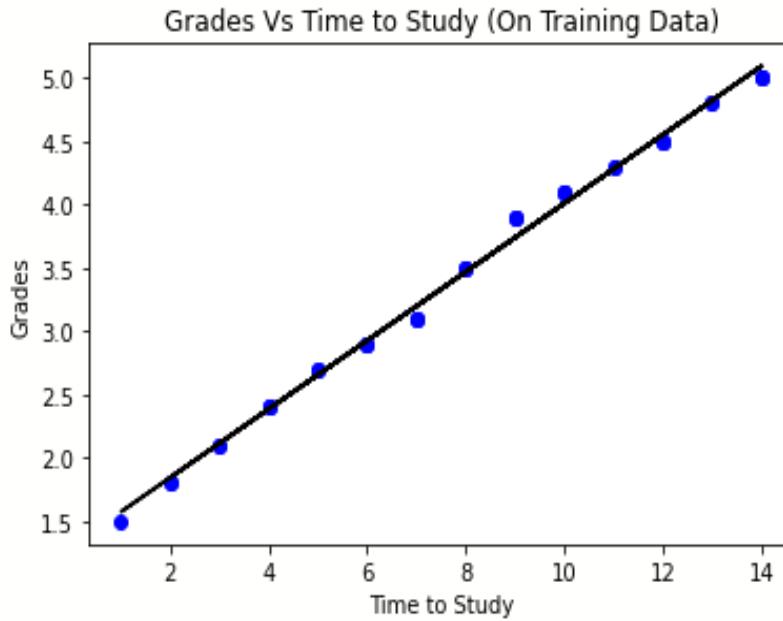
#Fit the Simple Linear Regression Model
from sklearn.linear_model import LinearRegression
LinReg = LinearRegression()
LinReg.fit(X_train, y_train)

#print the
print(f'a0 = {LinReg.intercept_}')
print(f'a1 = {LinReg.coef_}')

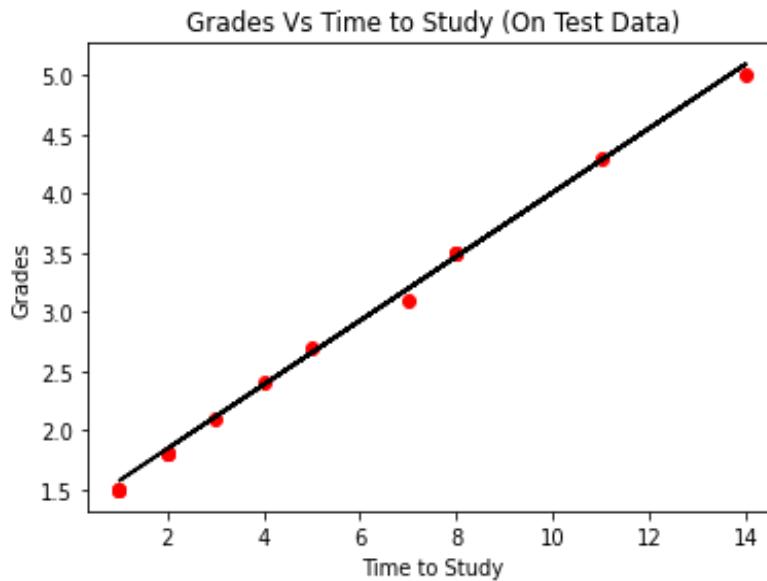
#Predicted grade scores from test dataset
y_predict = LinReg.predict(X_test)
y_predict

#Actual grade scores from test dataset
y_test

#Grades Vs Time to Study visualization on Training Data
plt.scatter(X_train, y_train, color='Blue')
plt.plot(X_train, LinReg.predict(X_train), color='Black')
plt.title('Grades Vs Time to Study (On Training Data)')
plt.xlabel('Time to Study')
plt.ylabel('Grades')
plt.show()
```



```
#Grades Vs Time to Study visualization on Test Data
plt.scatter(X_test, y_test, color='Red')
plt.plot(X_train, LinReg.predict(X_train), color='Black')
plt.title('Grades Vs Time to Study (On Test Data)')
plt.xlabel('Time to Study')
plt.ylabel('Grades')
plt.show()
```



```
#Predicting Grade of a student when he studied for 10 Hrs. Example of how
to pass an external value,
#Independent of Test or Training Dataset
```

```
Predict_Grade = LinReg.predict([[10]])
Predict_Grade
```

```
#Model Evaluation using R-Square
from sklearn import metrics
r_square = metrics.r2_score(y_test, y_predict)
print('R-Square Error:', r_square)

#Model Evaluation using Mean Square Error (MSE)
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_predict))

#Model Evaluation using Root Mean Square Error (RMSE)
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_predict)))

#Model Evaluation using Mean Absolute Error (MAE)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_predict))
```

Conclusion

Simple linear regression is a regression model that figures out the relationship between one independent variable and one dependent variable using a straight line.

References:

[1]RiyaKumari,<https://www.analyticssteps.com/blogs/simple-linear-regression-applications-limitations-examples>

Assignment No -3

Title: Classification using Machine Learning

Problem Statement:

Perform following operations on given dataset:

- a) Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if necessary.
- b) Perform data-preparation (Train-Test Split)
- c) Apply Decision tree classification Algorithm
- d) Evaluate Model.

Objective:

This assignment will help the students to realize how the decision tree classifier can be used and predictions using the same can be performed.

S/W Packages and H/W apparatus used:

Linux OS: Ubuntu/Windows , Jupyter notebook.

PC with the configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15''Color Monitor, Keyboard, Mouse

References:

1. *Ethem Alpaydin, Introduction to Machine Learning, PHI 2nd Edition,2013.*
2. *Peter Flach: Machine Learning: The Art and Science of Algorithms that Make Sense of Data,Cambridge University Press, Edition 2012..*

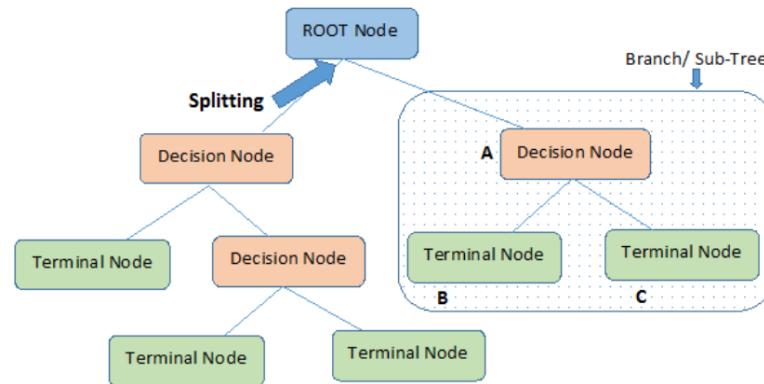
Theory:

Classification:

Classification is a process of categorizing a given set of data into classes. It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

What is a Decision Tree?

It uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.[1]



Root Nodes – It is the node present at the beginning of a decision tree. From this node the population starts dividing according to various features.

Decision Nodes – the nodes we get after splitting the root nodes are called Decision Node

Leaf Nodes – the nodes where further splitting is not possible are called leaf nodes or terminal nodes

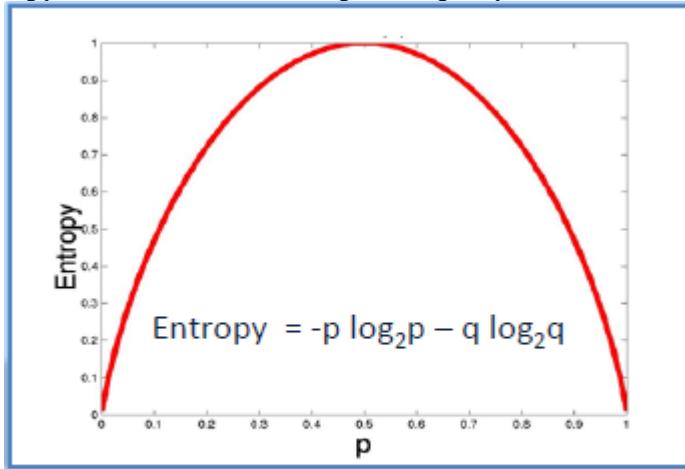
Sub-tree – just like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.

Pruning – It is cutting down some nodes to stop overfitting.



Entropy:

Entropy is used to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is equally divided it has entropy of one.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
 $= \text{Entropy}(0.36, 0.64)$
 $= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$
 $= 0.94$

b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



$$\begin{aligned}
 E(\text{PlayGolf, Outlook}) &= P(\text{Sunny}) * E(3,2) + P(\text{Overcast}) * E(4,0) + P(\text{Rainy}) * E(2,3) \\
 &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\
 &= 0.693
 \end{aligned}$$

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attributes that return the highest information gain (i.e., the most homogeneous branches). [2]

Step 1: Calculate entropy of the target.

$$\begin{aligned}
 \text{Entropy(PlayGolf)} &= \text{Entropy}(5,9) \\
 &= \text{Entropy}(0.36, 0.64) \\
 &= - (0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\
 &= 0.94
 \end{aligned}$$

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated.

Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

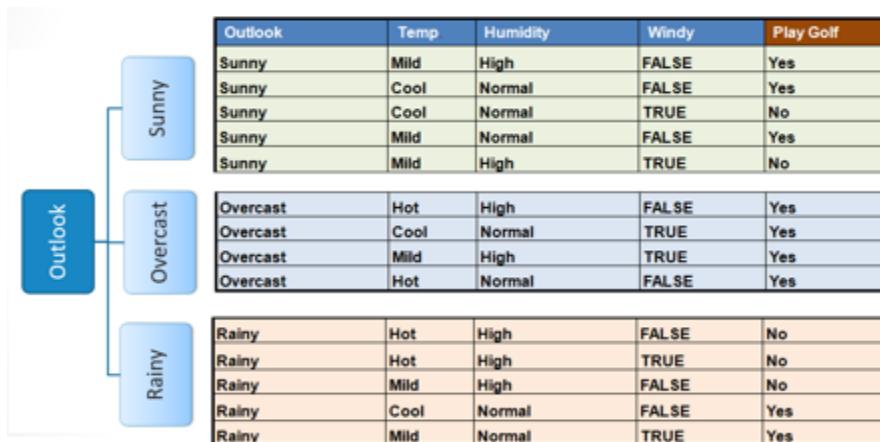
		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

$$\begin{aligned} G(\text{PlayGolf}, \text{Outlook}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Outlook}) \\ &= 0.940 - 0.693 = 0.247 \end{aligned}$$

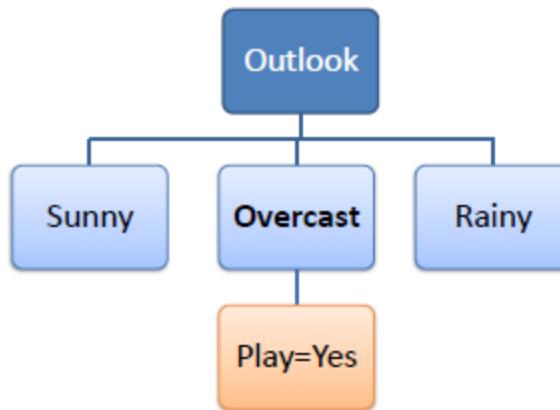
Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



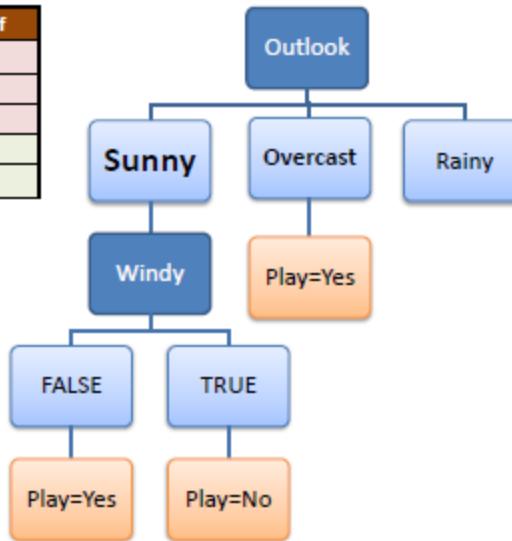
Step 4a: A branch with entropy of 0 is a leaf node.

Temp.	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting.

Temp.	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No

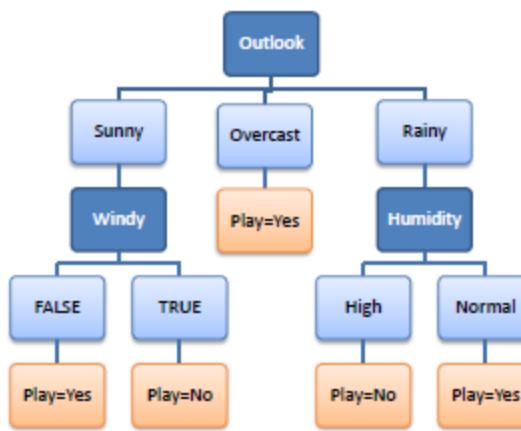


Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision Tree to Decision Rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

$R_1: \text{IF } (\text{Outlook}=\text{Sunny}) \text{ AND } (\text{Windy}=\text{FALSE}) \text{ THEN Play=Yes}$
 $R_2: \text{IF } (\text{Outlook}=\text{Sunny}) \text{ AND } (\text{Windy}=\text{TRUE}) \text{ THEN Play=No}$
 $R_3: \text{IF } (\text{Outlook}=\text{Overcast}) \text{ THEN Play=Yes}$
 $R_4: \text{IF } (\text{Outlook}=\text{Rainy}) \text{ AND } (\text{Humidity}=\text{High}) \text{ THEN Play=No}$
 $R_5: \text{IF } (\text{Outlook}=\text{Rain}) \text{ AND } (\text{Humidity}=\text{Normal}) \text{ THEN Play=Yes}$



Pruning:

It is another method that can help us avoid overfitting. It helps in improving the performance of the tree by cutting the nodes or sub-nodes which are not significant. It removes the branches which have very low importance.

There are mainly 2 ways for pruning:

- (i) **Pre-pruning** – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance **while growing** the tree.
- (ii) **Post-pruning** – once our **tree is built to its depth**, we can start pruning the nodes based on their significance.

Implementation:

Importing all the necessary libraries

```

import numpy as np # Linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
  
```

Importing the csv file

```
df = pd.read_csv('../input/Admission_Predict.csv')
```

```
# Check null values in the dataset
```

```
df.isnull().sum()
```

Out[]:

Serial No.	0
GRE Score	0
TOEFL Score	0
University Rating	0
SOP	0

```
LOR      0
CGPA     0
Research  0
Chance of Admit    0
dtype: int64
```

```
df.columns
```

```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

Updating chance of admission column

```
# if chance >= 80% CHANCE = 1
# if chance < 80%  CHANCE = 0
```

```
dataset.loc[dataset['Chance of Admit '] < 0.8, 'Chance of Admit '] = 0
dataset.loc[dataset['Chance of Admit '] >= 0.8, 'Chance of Admit '] = 1
```

Initializing the variables

```
X = df.drop(['Chance of Admit ','Serial No.'],axis=1)
y = df['Chance of Admit ']
```

Split the data into training and testing set

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test =
train_test_split(X,y,test_size=0.25,random_state=123)
```

```
# importing required libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

```
# Creating Decision Tree classifier object
clf = DecisionTreeClassifier()
```

```
# Training Decision Tree Classifier
clf = clf.fit(X_train, Y_train)
```

```
#Predicting for the test data
y_pred = clf.predict(X_test)
```

Confusion matrix:

```
print("confusion matrix:\n")
print(metrics.confusion_matrix(Y_test, y_pred))
```

confusion matrix:

```
[[79  3]
 [ 4 39]]
```

```
print("1. Accuracy Score:", metrics.accuracy_score(Y_test, y_pred))
print("2. Precision Score:", metrics.precision_score(Y_test, y_pred))
print("3. Recall Score:", metrics.recall_score(Y_test, y_pred))
print("4. f1 Score:", metrics.f1_score(Y_test, y_pred))

1. Accuracy Score: 0.944
2. Precision Score: 0.9285714285714286
3. Recall Score: 0.9069767441860465
4. f1 Score: 0.9176470588235294
```

Application:

Helpful in solving classification problems.

References:

- [1]Anshul Saini ,Analytics Vidhya,Decision Tree Algorithm – A Complete Guide
- [2]Dr. Saed Sayad,https://www.saedsayad.com/decision_tree.htm

Assignment No -4

Assignment No -4

Title: E-mail Classification using Naïve-Bayes Algorithm.

Problem Statement:

A SMS unsolicited mail (every now and then known as cell smartphone junk mail) is any junk message brought to a cellular phone as textual content messaging via the Short Message Service (SMS). Use probabilistic approach (Naive Bayes Classifier / Bayesian Network) to implement SMS Spam Filtering system. SMS messages are categorized as SPAM or HAM using features like length of message, word depend, unique keywords etc.

Download Data -Set from:

<https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>

This dataset is composed by just one text file, where each line has the correct class followed by the raw message.

- I Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if necessary
- II Perform data-preparation (Train-Test Split)
- III Apply at least two Machine Learning Algorithms and Evaluate Models
- IV Apply Cross-Validation and Evaluate Models and compare performance.
- V Apply Hyper parameter tuning and evaluate models and compare performance.

Objective: This assignment will help the students to realize how the Naïve Bayes algorithm works in classification of text.

S/W Packages and H/W apparatus used:

Linux OS: Ubuntu/Windows , Jupyter notebook.

PC with the configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15''Color Monitor, Keyboard, Mouse

References:

1. Ethem Alpaydin, Introduction to Machine Learning, PHI 2nd Edition,2013.
2. Peter Flach: Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Cambridge University Press, Edition 2012.

Theory:

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for playing golf.

Here is a tabular representation of our dataset.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No

	Outlook	Temperature	Humidity	Windy	Play Golf
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
- Response vector contains the value of **class variable** (prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

Assumption:

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be **independent**.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes' theorem.

Bayes' Theorem

Bayes' theorem is a way to figure out conditional probability. Conditional probability is the probability of an event happening, given that it has some relationship to one or more other events. For example, your probability of getting a parking space is connected to the time of day you park, where you park, and what conventions are going on at any time. Bayes' theorem is slightly more nuanced. In a nutshell, it gives you the actual probability of an **event** given information about **tests**.

- “Events” Are different from “tests.” For example, there is a **test** for liver disease, but that’s separate from the **event** of actually having liver disease.
- **Tests are flawed:** just because you have a positive test does not mean you actually have the disease. Many tests have a high false positive rate. **Rare events tend to have higher false positive rates** than more common events. We’re not just talking about medical tests here. For example, spam filtering can have high false positive rates. Bayes’ theorem takes the test results and calculates your *real probability* that the test has identified the event.

The Formula

Bayes' Theorem (also known as Bayes' rule) is a deceptively simple formula used to calculate conditional probability. The Theorem was named after English mathematician Thomas Bayes (1701-1761). The formal definition for the rule is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

In most cases, you can't just plug numbers into an equation; You have to figure out what your “tests” and “events” are first. For two events, A and B, Bayes' theorem allows you to figure out $p(A|B)$ (the probability that event A happened, given that test B was positive) from $p(B|A)$ (the probability that test B happened, given that event A happened). It can be a little tricky to wrap your head around as technically you're working backwards; you may have to switch your tests and events around, which can get confusing. An example should clarify what I mean by “switch the tests and events around.”

Bayes' Theorem Example #1

You might be interested in finding out a patient's probability of having liver disease if they are an alcoholic. “Being an alcoholic” is the **test** (kind of like a litmus test) for liver disease.

- **A** could mean the event “Patient has liver disease.” Past data tells you that 10% of patients entering your clinic have liver disease. $P(A) = 0.10$.
- **B** could mean the litmus test that “Patient is an alcoholic.” Five percent of the clinic’s patients are alcoholics. $P(B) = 0.05$.
- You might also know that among those patients diagnosed with liver disease, 7% are alcoholics. This is your **B|A**: the probability that a patient is alcoholic, given that they have liver disease, is 7%.

Bayes’ theorem tells you:

$$P(A|B) = (0.07 * 0.1) / 0.05 = 0.14$$

In other words, if the patient is an alcoholic, their chances of having liver disease is 0.14 (14%). This is a large increase from the 10% suggested by past data. But it’s still unlikely that any particular patient has liver disease.

More Bayes’ Theorem Examples

Bayes’ Theorem Problems Example #2

Another way to look at the theorem is to say that one event follows another. Above I said “tests” and “events”, but it’s also legitimate to think of it as the “first event” that leads to the “second event.” There’s no one right way to do this: use the terminology that makes most sense to you.

In a particular pain clinic, 10% of patients are prescribed narcotic pain killers. Overall, five percent of the clinic’s patients are addicted to narcotics (including pain killers and illegal substances). Out of all the people prescribed pain pills, 8% are addicts. *If a patient is an addict, what is the probability that they will be prescribed pain pills?*

Step 1: Figure out what your event “A” is from the question. That information is in the italicized part of this particular question. The event that happens first (A) is being prescribed pain pills. That’s given as 10%.

Step 2: Figure out what your event “B” is from the question. That information is also in the italicized part of this particular question. Event B is being an addict. That’s given as 5%.

Step 3: Figure out what the probability of event B (Step 2) given event A (Step 1). In other words, find what $(B|A)$ is. We want to know “Given that people are prescribed pain pills, what’s the probability they are an addict?” That is given in the question as 8%, or .8.

Step 4: Insert your answers from Steps 1, 2 and 3 into the formula and solve.

$$P(A|B) = P(B|A) * P(A) / P(B) = (0.08 * 0.1) / 0.05 = 0.16$$

The probability of an addict being prescribed pain pills is 0.16 (16%).

Example #3: the Medical Test

A slightly more complicated example involves a medical test (in this case, a genetic test):

There are **several forms of Bayes' Theorem** out there, and they are all equivalent (they are just written in slightly different ways). In this next equation, “X” is used in place of “B.” In addition, you’ll see some changes in the denominator. The proof of why we can rearrange the equation like this is beyond the scope of this article (otherwise it would be 5,000 words instead of 2,000!). However, if you come across a question involving medical tests, you’ll likely be using this alternative formula to find the answer:

$$\Pr(A|X) = \frac{\Pr(X|A) \Pr(A)}{\Pr(X|A) \Pr(A) + \Pr(X|\sim A) \Pr(\sim A)}$$

Bayes Theorem Examples

1% of people have a certain [genetic defect](#).

90% of tests for the gene detect the defect (true positives).

9.6% of the tests are [false positives](#).

If a person gets a positive test result, **what are the odds they actually have the genetic defect?**

The first step into solving Bayes' theorem problems is to assign letters to events:

- A = chance of having the faulty gene. That was given in the question as 1%. That also means the probability of *not* having the gene ($\sim A$) is 99%.
- X = A positive test result.

So:

1. $P(A|X)$ = Probability of having the gene given a positive test result.
2. $P(X|A)$ = Chance of a positive test result given that the person actually has the gene. That was given in the question as 90%.
3. $p(X|\sim A)$ = Chance of a positive test if the person *doesn't* have the gene. That was given in the question as 9.6%

Now we have all of the information we need to put into the equation:
 $P(A|X) = (.9 * .01) / (.9 * .01 + .096 * .99) = 0.0865 (8.65\%)$.

The probability of having the faulty gene on the test is 8.65%.

Bayes' Theorem Problems #4: A Test for Cancer

I wrote about how challenging physicians find [probability and statistics](#) in my post on [reading mammogram results wrong](#). It's not surprising that physicians are way off with their interpretation of results, given that some tricky probabilities are at play. Here's a second example of how Bayes' Theorem works. I've used similar numbers, but the question is worded differently to give you

another opportunity to wrap your mind around how you decide which is event A and which is event X.

Q. Given the following statistics, what is the probability that a woman has cancer if she has a positive mammogram result?

- One percent of women over 50 have breast cancer.
- Ninety percent of women who have breast cancer test positive on mammograms.
- Eight percent of women will have false positives.

Step 1: Assign events to A or X. You want to know what a woman's probability of having cancer is, given a positive mammogram. For this problem, actually having cancer is A and a positive test result is X.

Step 2: List out the parts of the equation (this makes it easier to work the actual equation):

$$P(A) = 0.01$$

$$P(\sim A) = 0.99$$

$$P(X|A) = 0.9$$

$$P(X|\sim A) = 0.08$$

Step 3: Insert the parts into the equation and solve. Note that as this is a medical test, we're using the form of the equation from example #2:

$$(0.9 * 0.01) / ((0.9 * 0.01) + (0.08 * 0.99)) = 0.10.$$

The probability of a woman having cancer, given a positive test result, is 10%.

Remember when (up there ^^) I said that there are many equivalent ways to write Bayes Theorem? Here is another equation, that you can use to figure out the above problem. You'll get exactly the same result:

$$P(B|A) = \frac{P(B \cap A)}{P(A)} = \frac{P(B \cap A)}{P(B \cap A) + P(B^c \cap A)}$$

The main difference with this form of the equation is that it uses the probability terms intersection(\cap) and complement (c). Think of it as shorthand: it's the same equation, written in a different way.

In order to find the probabilities on the right side of this equation, use the multiplication rule:

$$P(B \cap A) = P(B) * P(A|B)$$

The two sides of the equation are equivalent, and $P(B) * P(A|B)$ is what we were using when we solved the numerator in the problem above.

$$P(B) * P(A|B) = 0.01 * 0.9 = 0.009$$

For the denominator, we have $P(B^c \cap A)$ as part of the equation. This can be (equivalently) rewritten as $P(B^c * P(A|B^c))$. This gives us:

$$P(B^c * P(A|B^c)) = 0.99 * 0.08 = 0.0792.$$

Inserting those two solutions into the formula, we get:

$$0.009 / (0.009 + 0.0792) = 10\%.$$

Bayes' Theorem Problems: Another Way to Look at It.

Bayes' theorem problems can be figured out *without* using the equation (although using the equation is probably simpler). But if you can't wrap your head around why the equation works (or what it's doing), here's the non-equation solution for the same problem in #1 (the genetic test problem) above.

Step 1: Find the probability of a true positive on the test. That equals people who actually have the defect (1%) * true positive results (90%) = .009.

Step 2: Find the probability of a false positive on the test. That equals people who don't have the defect (99%) * false positive results (9.6%) = .09504.

Step 3: Figure out the probability of getting a positive result on the test. That equals the chance of a true positive (Step 1) plus a false positive (Step 2) = .009 + .09504 = .010404.

Step 4: Find the probability of actually having the gene, given a positive result. Divide the chance of having a real, positive result (Step 1) by the chance of getting any kind of positive result (Step 3) = .009/.010404 = 0.0865 (8.65%).

Other forms of Bayes' Theorem

Bayes' Theorem has several forms. You probably won't encounter any of these other forms in an elementary stats class. The different forms can be used for different purposes. For example, one version uses what Rudolf Carnap called the "**probability ratio**". The probability ratio rule states that any event (like a patient having liver disease) must be multiplied by this factor $PR(H,E)=P_E(H)/P(H)$. That gives the event's probability conditional on E. The **Odds Ratio Rule** is very similar to the probability ratio, but the [likelihood ratio](#) divides a test's true positive rate divided by its false positive rate. The formal definition of the Odds Ratio rule is $OR(H,E)=P_{H,E}/P_{\neg H,E}$.

Bayesian Spam Filtering

Although Bayes' Theorem is used extensively in the medical sciences, there are other applications. For example, it's used to [filter spam](#). The **event** in this case is that the message is spam. The **test** for spam is that the message contains some flagged words (like "viagra" or "you have won"). Here's the equation set up (from Wikipedia), read as "The probability a message is spam given that it contains certain flagged words":

$$\Pr(\text{spam}|\text{words}) = \frac{\Pr(\text{words}|\text{spam}) \Pr(\text{spam})}{\Pr(\text{words})}$$

The actual equations used for spam filtering are a little more complex; they contain more flags than just content. For example, the timing of the message, or how often the filter has seen the same content before, are two other spam tests.

Implementation

```
In [70]: import pandas as pd
In [71]: df = pd.read_csv('Email_Spam.csv')
df.head()
Out[71]:
   Category          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...
In [72]: df.groupby('Category').describe()
Out[72]:
           Message
           count  unique   top
Category
  ham      4825     4516  Sorry, I'll call later    30
  spam      747      641 Please call our customer service representativ...
In [73]: df['spam']=df['Category'].apply(lambda x: 1 if x=='spam' else 0)
df.head()
Out[73]:
   Category          Message  spam
0    ham  Go until jurong point, crazy.. Available only ...    0
1    ham           Ok lar... Joking wif u oni...    0
2   spam  Free entry in 2 a wkly comp to win FA Cup fina...    1
3    ham  U dun say so early hor... U c already then say...    0
4    ham  Nah I don't think he goes to usf, he lives aro...
```

```
In [74]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.Message,df.spam)

In [75]: from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer()
X_train_count = v.fit_transform(X_train.values)
X_train_count.toarray()[:2]
#print(v.get_feature_names())

Out[75]: array([[0, 0, 0, ..., 0, 0, 0],
   [0, 0, 0, ..., 0, 0, 0]], dtype=int64)

In [76]: from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_count,y_train)

Out[76]: MultinomialNB()

In [77]: emails = [
    'How are you brother?',
    'Get Flat 50% on Your body outfits'
]
emails_count = v.transform(emails)
model.predict(emails_count)

Out[77]: array([0, 0], dtype=int64)

In [78]: X_test_count = v.transform(X_test)
model.score(X_test_count, y_test)

Out[78]: 0.9877961234745154
```

Assignment No -5

Title: K Means Clustering

Problem Statement:

Perform following operations on given dataset:

- a) Apply Data pre-processing (Label Encoding, Data Transformation....) techniques if necessary.
- b) Perform data-preparation (Train-Test Split)
- c) Apply Machine Learning Algorithm
- d) Evaluate Model.
- e) Apply Cross-Validation and Evaluate Model

Objective:

This assignment will help the students to realize how to do Clustering using K Means Clustering algorithm.

S/W Packages and H/W apparatus used:

Linux OS: Ubuntu/Windows , Jupyter notebook.

PC with the configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15''Color Monitor, Keyboard, Mouse

References:

3. Ethem Alpaydin, *Introduction to Machine Learning*, PHI 2nd Edition, 2013.
4. Peter Flach: *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*, Cambridge University Press, Edition 2012..

Theory:

Introduction to K-means Clustering

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. [1]

The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity.

The results of the K-means clustering algorithm are:

1. The centroids of the K clusters, which can be used to label new data
2. Labels for the training data (each data point is assigned to a single cluster) Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have formed organically.

The "Choosing K" section below describes how the number of groups can be determined. Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

This introduction to the K-means clustering algorithm covers:

- Common business cases where K-means is used
- The steps involved in running the algorithm

Some examples of use cases are:

Behavioral segmentation:

- o Segment by purchase history
- o Segment by activities on application, website, or platform.
- o Define personas based on interests
- o Create profiles based on activity monitoring

Inventory categorization:

- o Group inventory by sales activity
- o Group inventory by manufacturing metrics

Sorting sensor measurements:

- o Detect activity types in motion sensors
- o Group images
- o Separate audio
- o Identify groups in health monitoring

Detecting bots or anomalies:

- o Separate valid activity groups from bots
- o Group valid activity to clean up outlier detection In addition, monitoring if a tracked data point switches between groups over time can be used to detect meaningful changes in the data.

Algorithm:

The K-means clustering algorithm uses iterative refinement to produce a final result. The

algorithm inputs are the number of clusters K and the data set. The data set is a collection of features for each data point. The algorithms start with initial estimates for the K centroids, which can either be randomly generated or randomly selected from the data set. The algorithm then iterates between two steps:

1. Data assignment step:

Each centroid defines one of the clusters. In this step, each data point is assigned to its nearest centroid, based on the squared Euclidean distance. More formally, if c_i is the collection of centroids in set C, then each data point x is assigned to a cluster based on

$$\operatorname{argmin}_{c_i \in C} \text{dist}(c_i, x)^2$$

where $\text{dist}(\cdot)$ is the standard (L2) Euclidean distance. Let the set of data point assignments for each i th cluster centroid be S_i .

2. Centroid update step:

In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid's cluster.

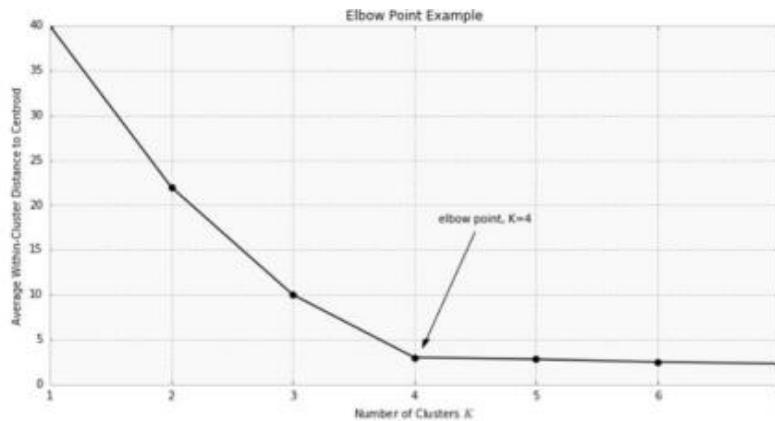
$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$$

The algorithm iterates between steps one and two until a stopping criteria is met (i.e., no data points change clusters, the sum of the distances is minimized, or some maximum number of iterations is reached).

This algorithm is guaranteed to converge to a result. The result may be a local optimum (i.e. not necessarily the best possible outcome), meaning that assessing more than one run of the algorithm with randomized starting centroids may give a better outcome.

Choosing K

The algorithm described above finds the clusters and data set labels for a particular pre-chosen K. To find the number of clusters in the data, the user needs to run the K-means clustering algorithm for a range of K values and compare the results. In general, there is no method for determining exact value of K, but an accurate estimate can be obtained using the following techniques. One of the metrics that is commonly used to compare results across different values of K is the mean distance between data points and their cluster centroid. Since increasing the number of clusters will always reduce the distance to data points, increasing K will always decrease this metric, to the extreme of reaching zero when K is the same as the number of data points. Thus, this metric cannot be used as the sole target. Instead, mean distance to the centroid as a function of K is plotted and the "elbow point," where the rate of decrease sharply shifts, can be used to roughly determine K. A number of other techniques exist for validating K, including cross-validation, information criteria, the information theoretic jump method, the silhouette method, and the G-means algorithm. In addition, monitoring the distribution of data points across groups provides insight into how the algorithm is splitting the data for each K.



Implementation:

Importing Dataset

```
from pandas import read_csv
A=read_csv("E:/DS1/Mall_Customers.csv")
```

Dropping the irrelevant columns

```
B=A.drop(["Customer_ID"],axis=1)
```

Label encoding

```
# Import label encoder
from sklearn import preprocessing
```

```
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
```

```
# Encode labels in column 'species'.
B['Genre']=label_encoder.fit_transform(B['Genre'])
```

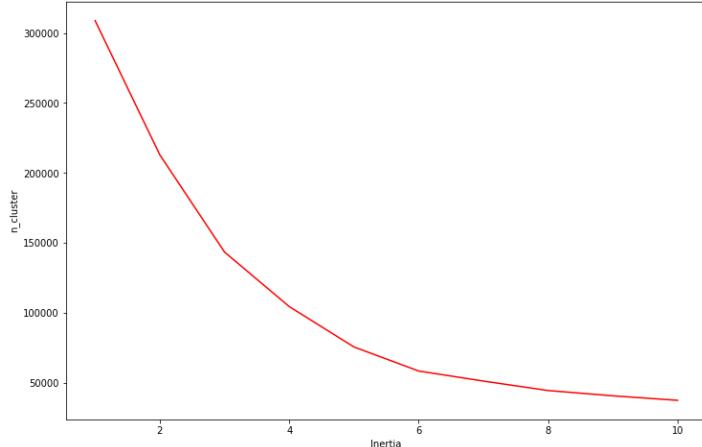
```
B['Genre'].unique()
array([1, 0], dtype=int64)
```

Finding K

```
from sklearn.cluster import KMeans
cluster = []
for k in range (1, 11):
    kmean = KMeans(n_clusters=k).fit(B)
    cluster.append(kmean.inertia_)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 8))
```

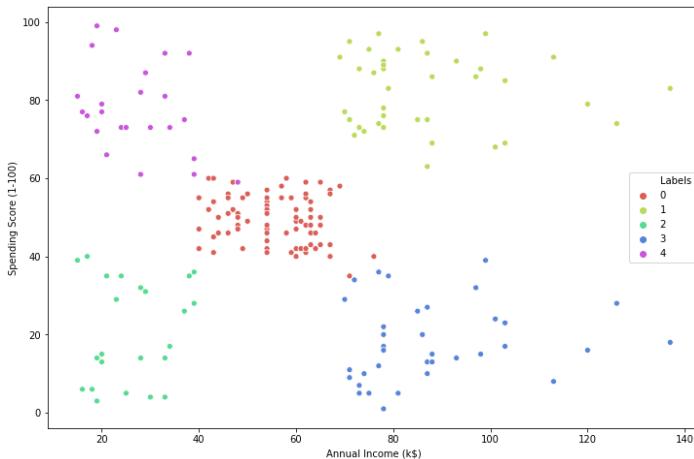
```
plt.plot(range(1, 11), cluster, 'r-')
plt.xlabel('Inertia')
plt.ylabel('n_cluster')
plt.show()
```



With above value of K, Create K means clustering model

```
km = KMeans(n_clusters=5).fit(B)
B['Labels'] = km.labels_
```

```
import seaborn as sns
plt.figure(figsize=(12, 8))
sns.scatterplot(B['Annual Income (k$)'], B['Spending Score (1-100)'], hue=B['Labels'],
palette=sns.color_palette('hls', 5))
plt.show()
```



References:

- [1] Andrea Trevino, Introduction to K-means Clustering , Oracle AI & Data Science Blog

Assignment No -6

Title: Association Rule Learning

Problem Statement:

Download Market Basket Optimization dataset from below link.

Data Set: <https://www.kaggle.com/hemanthkumar05/market-basket-optimization>

This dataset comprises the list of transactions of a retail company over the period of one week. It contains a total of 7501 transaction records where each record consists of the list of items sold in one transaction. Using this record of transactions and items in each transaction, find the association rules between items.

There is no header in the dataset and the first row contains the first transaction, so mentioned header = None here while loading dataset.

- a. Follow following steps :
- b. Data Preprocessing
- c. Generate the list of transactions from the dataset
- d. Train Apriori algorithm on the dataset
- e. Visualize the list of rules

Generated rules depend on the values of hyper parameters. By increasing the minimum confidence value and find the rules accordingly

Objective:

This assignment will help the students to understand and implement Apriori Algorithm.

S/W Packages and H/W apparatus used:

Linux OS: Ubuntu/Windows , Jupyter notebook.

PC with the configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15''Color Monitor, Keyboard, Mouse

References:

- 1.Ethem Alpaydin, Introduction to Machine Learning, PHI 2nd Edition,2013.
2. Peter Flach: Machine Learning: The Art and Science of Algorithms that Make Sense of Data,Cambridge University Press, Edition 2012.

Theory:

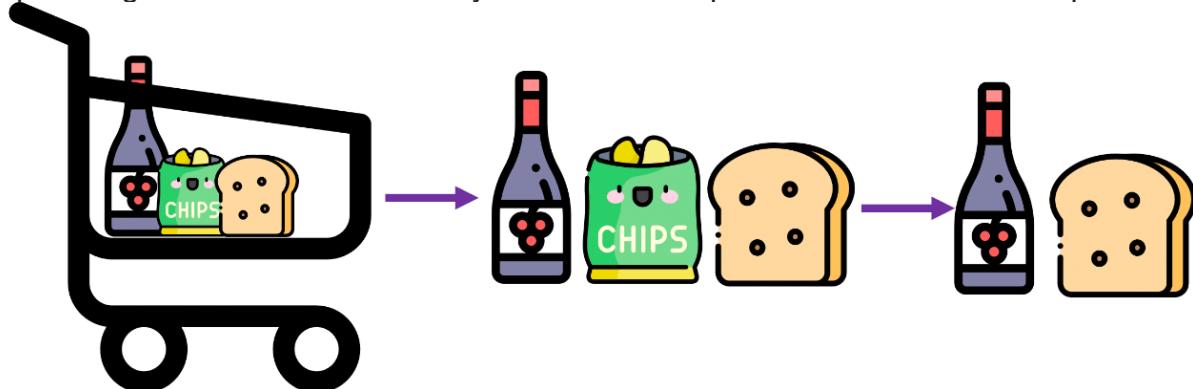
the Apriori algorithm is used for the purpose of [association rule mining](#).Association rule mining is a technique to identify frequent patterns and associations among a set of items.

For example, understanding customer buying habits. By finding correlations and associations between different items that customers place in their 'shopping basket,' recurring patterns can be derived.

This process of identifying an association between products/items is called association rule mining. To implement association rule mining, many algorithms have been developed. Apriori algorithm is one of the most popular and arguably the most efficient algorithms among them.

Apriori Algorithm

Apriori algorithm assumes that any subset of a frequent itemset must be frequent.



Say, a transaction containing {wine, chips, bread} also contains {wine, bread}. So, according to the principle of Apriori, if {wine, chips, bread} is frequent, then {wine, bread} must also be frequent.

The key concept in the Apriori algorithm is that it assumes all subsets of a frequent itemset to be frequent. Similarly, for any infrequent itemset, all its supersets must also be infrequent.

Here is a dataset consisting of six transactions in an hour. Each transaction is a combination of 0s and 1s, where 0 represents the absence of an item and 1 represents the presence of it.

Here is a dataset consisting of six transactions in an hour. Each transaction is a combination of 0s and 1s, where 0 represents the absence of an item and 1 represents the presence of it.

Transaction ID	Wine	Chips	Bread	Milk
T1	1	0	1	0
T2	1	1	1	0
T3	0	1	1	0
T4	1	0	0	0
T5	1	1	0	0
T6	0	0	1	0

1	1	1	1	1
2	1	0	1	1
3	0	0	1	1
4	0	1	0	0
5	1	1	1	1
6	1	1	0	1

We can find multiple rules from this scenario. For example, in a transaction of wine, chips, and bread, if wine and chips are bought, then customers also buy bread.

{wine, chips} => {bread}

In order to select the interesting rules out of multiple possible rules from this small business scenario, we will be using the following measures:

- Support
- Confidence
- List
- Conviction

Support

Support of item x is nothing but the ratio of the number of transactions in which item x appears to the total number of transactions.

i.e.,

$$\text{Support(wine)} = \frac{\text{Number of transactions in which the item wine appears}}{\text{Total number of transactions}}$$

$$\text{Support(wine)} = 4/6 = 0.66667$$

Confidence

Confidence ($x \Rightarrow y$) signifies the likelihood of the item y being purchased when item x is purchased. This method takes into account the popularity of item x .

i.e.,

$$\text{Conf}(\{\text{wine, chips}\} \Rightarrow \{\text{bread}\}) = \frac{\text{support}(\text{wine, chips, bread})}{\text{support}(\text{wine, chips})}$$

$$\text{Conf}(\{\text{wine, chips}\} \Rightarrow \{\text{bread}\}) = \frac{\frac{2}{6}}{\frac{3}{6}} : = 0.667$$

Lift

Lift ($x \Rightarrow y$) is nothing but the ‘interestingness’ or the likelihood of the item y being purchased when item x is sold. Unlike confidence ($x \Rightarrow y$), this method takes into account the popularity of the item y .

i.e.,

$$\text{lift } (\{\text{wine, chips}\} \Rightarrow \{\text{bread}\}) = \frac{\text{support}(\text{wine, chips, bread})}{\text{support}(\text{wine, chips})}$$

$$\text{lift } (\{\text{wine, chips}\} \Rightarrow \{\text{bread}\}) = \frac{\frac{2}{6}}{\frac{3 * 4}{6 * 6}} = 1$$

- Lift ($x \Rightarrow y$) = 1 means that there is no correlation within the itemset.
- Lift ($x \Rightarrow y$) > 1 means that there is a positive correlation within the itemset, i.e., products in the itemset, x and y , are more likely to be bought together.
- Lift ($x \Rightarrow y$) < 1 means that there is a negative correlation within the itemset, i.e., products in itemset, x and y , are unlikely to be bought together.

Dataset

Below is the transaction data from Day 1. This dataset contains 6 items and 22 transaction records.

	A	B	C	D	E	F
1	Wine	Chips	Bread	Butter	Milk	Apple
2	Wine		Bread	Butter	Milk	
3			Bread	Butter	Milk	
4		Chips				Apple
5	Wine	Chips	Bread	Butter	Milk	Apple
6	Wine	Chips			Milk	
7	Wine	Chips	Bread	Butter		Apple
8	Wine	Chips			Milk	
9	Wine		Bread			Apple
10	Wine		Bread	Butter	Milk	
11		Chips	Bread	Butter		Apple
12	Wine			Butter	Milk	Apple
13	Wine	Chips	Bread	Butter	Milk	
14	Wine		Bread		Milk	Apple
15	Wine		Bread	Butter	Milk	Apple
16	Wine	Chips	Bread	Butter	Milk	Apple
17		Chips	Bread	Butter	Milk	Apple
18		Chips		Butter	Milk	Apple
19	Wine	Chips	Bread	Butter	Milk	Apple
20	Wine		Bread	Butter	Milk	Apple
21	Wine	Chips	Bread		Milk	Apple
22		Chips				

Environment Setup:

Before we move forward, we need to install the ‘apyori’ package first.

```
pip install apyori
```

Market Basket Analysis Implementation within Python

With the help of the apyori package, we will be implementing the Apriori algorithm in order to help the manager in market basket analysis.

Step 1: Import the libraries

```
In [1]: #Importing the required datasets
import numpy as np
import pandas as pd
from apyori import apriori
```

Step 2: Load the dataset

```
In [2]: #Loading the dataset
store_data = pd.read_csv('Day1.csv', header=None)
```

Step 3: Have a glance at the records

```
In [3]: #Having a glance at the records
store_data
```

```
out[3]:
      0   1   2   3   4   5
  0  Wine  Chips  Bread  Butter  Milk  Apple
  1  Wine    NaN  Bread  Butter  Milk    NaN
  2    NaN    NaN  Bread  Butter  Milk    NaN
  3    NaN  Chips    NaN  NaN    NaN  Apple
  4  Wine  Chips  Bread  Butter  Milk  Apple
  5  Wine  Chips    NaN  NaN    NaN  Milk
  6  Wine  Chips  Bread  Butter  NaN  Apple
  7  Wine  Chips    NaN  NaN  Milk    NaN
  8  Wine    NaN  Bread  NaN    NaN  Apple
  9  Wine    NaN  Bread  Butter  Milk    NaN
 10   NaN  Chips  Bread  Butter  NaN  Apple
 11  Wine    NaN    NaN  Butter  Milk  Apple
 12  Wine  Chips  Bread  Butter  Milk    NaN
 13  Wine    NaN  Bread  NaN  Milk  Apple
 14  Wine    NaN  Bread  Butter  Milk  Apple
 15  Wine  Chips  Bread  Butter  Milk  Apple
 16   NaN  Chips  Bread  Butter  Milk  Apple
 17   NaN  Chips    NaN  Butter  Milk  Apple
 18  Wine  Chips  Bread  Butter  Milk  Apple
 19  Wine    NaN  Bread  Butter  Milk  Apple
 20  Wine  Chips  Bread  NaN  Milk  Apple
 21   NaN  Chips    NaN  NaN    NaN  NaN
```

Step 4 :Convert Pandas DataFrame into a list of lists

```
In [5]: #Converting the pandas dataframe into a list of lists
records = []
for i in range(0, 22):
    records.append([str(store_data.values[i,j]) for j in range(0, 6)])
```

Step 5: Build the Apriori model

```
In [7]: #Building the first apriori model
association_rules = apriori(records, min_support=0.50, min_confidence=0.7, min_lift=1.2, min_length=2)
association_results = list(association_rules)
```

Step 6: Print out the number of rules

```
In [8]: #Getting the number of rules
print(len(association_results))
```

Step 7: Have a glance at the rule

```
In [10]: #Glancing at the first rule
print(association_results)
```

```
[RelationRecord(items=frozenset({'Milk', 'Butter', 'Bread'}), support=0.5, ordered_statistics=[OrderedStatistic(items_base=frozenset({'Milk', 'Bread'}), items_add=frozenset({'Butter'}), confidence=0.8461538461538461, lift=1.241025641025641)])]
```

The support value for the first rule is 0.5. This number is calculated by dividing the number of transactions containing ‘Milk,’ ‘Bread,’ and ‘Butter’ by the total number of transactions.

The confidence level for the rule is 0.846, which shows that out of all the transactions that contain both “Milk” and “Bread”, 84.6 % contain ‘Butter’ too.

The lift of 1.241 tells us that 'Butter' is 1.241 times more likely to be bought by the customers who buy both 'Milk' and 'Butter' compared to the default likelihood sale of 'Butter.'

Conclusion:

Apriori algorithm is implemented

PART B:

I. Design & Analysis of Algorithms

ASSIGNMENT 1

TITLE: Fractional knapsack using Greedy algorithm and 0/1 knapsack using dynamic programming

PROBLEM STATEMENT:

Write a program to implement Fractional knapsack using Greedy algorithm and 0/1 knapsack using dynamic programming. Show that Greedy strategy does not necessarily yield an optimal solution over a dynamic programming approach.

OBJECTIVE:

To understand and implement the program for Fractional knapsack using Greedy algorithm and 0/1 knapsack using dynamic programming

THEORY:

Among all the algorithmic approaches, the simplest and straightforward approach is the Greedy method. In this approach, the decision is taken on the basis of current available information without worrying about the effect of the current decision in future.

Greedy algorithms build a solution part by part, choosing the next part in such a way, that it gives an immediate benefit. This approach never reconsiders the choices taken previously. This approach is mainly used to solve optimization problems. Greedy method is easy to implement and quite efficient in most of the cases. Hence, we can say that Greedy algorithm is an algorithmic paradigm based on heuristic that follows local optimal choice at each step with the hope of finding global optimal solution.

In many problems, it does not produce an optimal solution though it gives an approximate (near optimal) solution in a reasonable time.

Components of Greedy Algorithm

Greedy algorithms have the following five components:

- **A candidate set:** A solution is created from this set.
- **A selection function:** Used to choose the best candidate to be added to the solution.
- **A feasibility function:** Used to determine whether a candidate can be used to contribute to the solution.
- **An objective function:** Used to assign a value to a solution or a partial solution.
- **A solution function:** Used to indicate whether a complete solution has been reached.

Areas of Application

Greedy approach is used to solve many problems, such as

- Finding the shortest path between two vertices using Dijkstra's algorithm.
- Finding the minimal spanning tree in a graph using Prim's /Kruskal's algorithm, etc.

Where Greedy Approach Fails

In many problems, Greedy algorithm fails to find an optimal solution, moreover it may produce a worst solution. Problems like Travelling Salesman and Knapsack cannot be solved using this approach.

Knapsack Problem

Given a set of items, each with a weight and a value, determine a subset of items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

The knapsack problem is a combinatorial optimization problem. It appears as a subproblem in many, more complex mathematical models of real-world problems. One general approach to difficult problems is to identify the most restrictive constraint, ignore the others, solve a knapsack problem, and somehow adjust the solution to satisfy the ignored constraints.

Applications

In many cases of resource allocation along with some constraint, the problem can be derived in a similar way of Knapsack problem. Following is a set of example.

- Finding the least wasteful way to cut raw materials
- portfolio optimization
- Cutting stock problems

Problem Scenario

A thief is robbing a store and can carry a maximal weight of W into his knapsack. There are n items available in the store and weight of i^{th} item is w_i and its profit is p_i . What items should the thief take?

In this context, the items should be selected in such a way that the thief will carry those items for which he will gain maximum profit. Hence, the objective of the thief is to maximize the profit.

Based on the nature of the items, Knapsack problems are categorized as

- Fractional Knapsack
- Knapsack

Fractional Knapsack problem using Greedy approach

In this case, items can be broken into smaller pieces, hence the thief can select fractions of items.

According to the problem statement,

- There are n items in the store
- Weight of i^{th} item $w_i > 0$
- Profit of i^{th} item $p_i > 0$

- Profit for i^{th} item $p_i > 0$
- Capacity of the Knapsack is W

In this version of Knapsack problem, items can be broken into smaller pieces. So, the thief may take only a fraction x_i of i^{th} item.

$$0 \leq X_i \leq 1$$

The i^{th} item contributes the weight $x_i \cdot w_i$ to the total weight in the knapsack and profit $x_i \cdot p_i$ to the total profit.

Hence, the objective of this algorithm is to

$$\text{maximize } \sum_{n=1}^n (x_i \cdot p_i)$$

subject to constraint,

$$\sum_{n=1}^n (x_i \cdot w_i) \leq W$$

```
Algorithm: Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)
for i = 1 to n
    do x[i] = 0
weight = 0
for i = 1 to n
    if weight + w[i] ≤ W then
        x[i] = 1
        weight = weight + w[i]
    else
        x[i] = (W - weight) / w[i]
        weight = W
        break
return x
```

Analysis

If the provided items are already sorted into a decreasing order of p_i/w_i , then the whileloop takes a time in $O(n)$; Therefore, the total time including the sort is in $O(n \log n)$.

0/1 Knapsack problem using Dynamic Programming

Let i be the highest-numbered item in an optimal solution S for W dollars. Then $S' = S - \{i\}$ is an optimal solution for $W - w_i$ dollars and the value to the solution S is V_i plus the value of the subproblem.

We can express this fact in the following formula: define $c[i, w]$ to be the solution for items **1,2, ..., i** and the maximum weight w .

The algorithm takes the following inputs

- The maximum weight W
- The number of items n
- The two sequences $v = \langle v_1, v_2, \dots, v_n \rangle$ and $w = \langle w_1, w_2, \dots, w_n \rangle$

```
Dynamic-0-1-knapsack (v, w, n, W)
for w = 0 to W do
    c[0, w] = 0
for i = 1 to n do
    c[i, 0] = 0
    for w = 1 to W do
        if  $w_i \leq w$  then
            if  $v_i + c[i-1, w-w_i]$  then
                c[i, w] =  $v_i + c[i-1, w-w_i]$ 
            else c[i, w] = c[i-1, w]
        else
            c[i, w] = c[i-1, w]
```

Analysis

This algorithm takes $\theta(n, w)$ times as table c has $(n + 1).(w + 1)$ entries, where each entry requires $\theta(1)$ time to compute.

Example

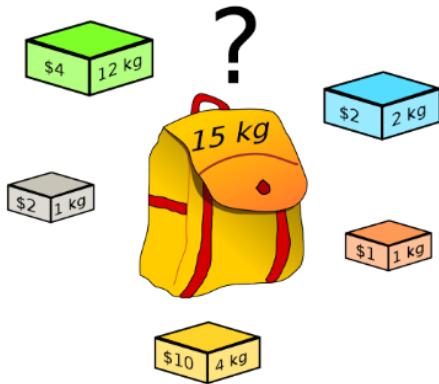
Knapsack Problem-

You are given the following-

- A knapsack (kind of shoulder bag) with limited weight capacity.
- Few items each having some weight and value.

The problem states- Which items should be placed into the knapsack such that-

- The value or profit obtained by putting the items into the knapsack is maximum.
- And the weight limit of the knapsack does not exceed.

**Knapsack Problem**

[2]

Knapsack Problem Variants-

Knapsack problem has the following two variants-

1. Fractional Knapsack Problem
2. 0/1 Knapsack Problem

Fractional Knapsack Problem-

In Fractional Knapsack Problem,

- As the name suggests, items are divisible here.
- We can even put the fraction of any item into the knapsack if taking the complete item is not possible.
- It is solved using Greedy Method.

Fractional Knapsack Problem Using Greedy Method-

Fractional knapsack problem is solved using greedy method in the following steps-

Step-01:

For each item, compute its value / weight ratio.

Step-02:

Arrange all the items in decreasing order of their value / weight ratio.

Step-03:

Start putting the items into the knapsack beginning from the item with the highest ratio.

Put as many items as you can into the knapsack.

Time Complexity-

The main time taking step is the sorting of all items in decreasing order of their value / weight ratio.

- If the items are already arranged in the required order, then while loop takes $O(n)$ time.
- The average time complexity of [Quick Sort](#) is $O(n\log n)$.
- Therefore, total time taken including the sort is $O(n\log n)$.

Problem-

For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Item	Weight	Value
1	5	30
2	10	40
3	15	45
4	22	77
5	25	90

OR

Find the optimal solution for the fractional knapsack problem making use of greedy approach.

Consider-

$$n = 5$$

$$w = 60 \text{ kg}$$

$$(w_1, w_2, w_3, w_4, w_5) = (5, 10, 15, 22, 25)$$

$$(b_1, b_2, b_3, b_4, b_5) = (30, 40, 45, 77, 90)$$

OR

A thief enters a house for robbing it. He can carry a maximal weight of 60 kg into his bag. There are 5 items in the house with the following weights and values. What items should thief take if he can even take the fraction of any item with him?

Item	Weight	Value
1	5	30
2	10	40
3	15	45
4	22	77
5	25	90

Solution-

Step-01:

Compute the value / weight ratio for each item-

Items	Weight	Value	Ratio
1	5	30	6
2	10	40	4
3	15	45	3
4	22	77	3.5
5	25	90	3.6

Step-02:

Sort all the items in decreasing order of their value / weight ratio-

I1 I2 I5 I4 I3

(6) (4) (3.6) (3.5) (3)

Step-03:

Start filling the knapsack by putting the items into it one by one.

Knapsack Weight	Items in Knapsack	Cost
60	Ø	0
55	I1	30
45	I1, I2	70
20	I1, I2, I5	160

Now,

- Knapsack weight left to be filled is 20 kg but item-4 has a weight of 22 kg.
- Since in fractional knapsack problem, even the fraction of any item can be taken.
- So, knapsack will contain the following items-

< I1 , I2 , I5 , (20/22) I4 >

Total cost of the knapsack

$$= 160 + (20/27) \times 77$$

$$= 160 + 70$$

$$= 230 \text{ units}$$

Important Note-

Had the problem been a 0/1 knapsack problem, knapsack would contain the following items- < I1 , I2 , I5 >

The knapsack's total cost would be 160 units.

0/1 Knapsack Problem-

In 0/1 Knapsack Problem,

- As the name suggests, items are indivisible here.
- We can not take the fraction of any item.
- We have to either take an item completely or leave it completely.
- It is solved using dynamic programming approach.

0/1 Knapsack Problem Using Dynamic Programming-

Consider-

- Knapsack weight capacity = w
- Number of items each having some weight and value = n

0/1 knapsack problem is solved using dynamic programming in the following steps-

Step-01:

- Draw a table say 'T' with (n+1) number of rows and (w+1) number of columns.
- Fill all the boxes of 0th row and 0th column with zeroes as shown-

	0	1	2	3	w
0	0	0	0	0	0
1	0					
2	0					
.....					
n	0					

T-Table

[2]

Step-02:

Start filling the table row wise top to bottom from left to right.

Use the following formula-

$$T(i, j) = \max \{ T(i-1, j), \text{value}_i + T(i-1, j - \text{weight}_i) \}$$

Here, $T(i, j)$ = maximum value of the selected items if we can take items 1 to i and have weight restrictions of j.

This step leads to completely filling the table.

- Then, value of the last box represents the maximum possible value that can be put into the knapsack.

Step-03:

To identify the items that must be put into the knapsack to obtain that maximum profit,

- Consider the last column of the table.
- Start scanning the entries from bottom to top.

- On encountering an entry whose value is not same as the value stored in the entry immediately above it, mark the row label of that entry.
- After all the entries are scanned, the marked labels represent the items that must be put into the knapsack.

Time Complexity-

Each entry of the table requires constant time $\theta(1)$ for its computation.

- It takes $\theta(nw)$ time to fill $(n+1)(w+1)$ table entries.
- It takes $\theta(n)$ time for tracing the solution since tracing process traces the n rows.
- Thus, overall $\theta(nw)$ time is taken to solve 0/1 knapsack problem using dynamic programming.

PRACTICE PROBLEM BASED ON 0/1 KNAPSACK PROBLEM-**Problem-**

For the given set of items and knapsack capacity = 5 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

Item	Weight	Value
1	2	3
2	3	4
3	4	5
4	5	6

OR

Find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach. Consider-

$$n = 4$$

$$w = 5 \text{ kg}$$

$$(w_1, w_2, w_3, w_4) = (2, 3, 4, 5)$$

$$(b_1, b_2, b_3, b_4) = (3, 4, 5, 6)$$

OR

A thief enters a house for robbing it. He can carry a maximal weight of 5 kg into his bag. There are 4 items in the house with the following weights and values. What items should thief take if he either takes the item completely or leaves it completely?

Item	Weight (kg)	Value (\$)
Mirror	2	3

Silver nugget	3	4
Painting	4	5
Vase	5	6

Solution-**Given-**

- Knapsack capacity (w) = 5 kg
- Number of items (n) = 4

Step-01:

- Draw a table say 'T' with $(n+1) = 4 + 1 = 5$ number of rows and $(w+1) = 5 + 1 = 6$ number of columns.
- Fill all the boxes of 0th row and 0th column with 0.

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

T-Table

[2]

Step-02:

Start filling the table row wise top to bottom from left to right using the formula-

$$T(i, j) = \max \{ T(i-1, j), \text{value}_i + T(i-1, j - \text{weight}_i) \}$$

Finding T(1,1)-

We have,

- $i = 1$
- $j = 1$
- $(\text{value})_i = (\text{value})_1 = 3$
- $(\text{weight})_i = (\text{weight})_1 = 2$

Substituting the values, we get-

$$T(1,1) = \max \{ T(1-1, 1), 3 + T(1-1, 1-2) \}$$

$$T(1,1) = \max \{ T(0,1), 3 + T(0,-1) \}$$

$$T(1,1) = T(0,1) \{ \text{Ignore } T(0,-1) \}$$

$$T(1,1) = 0$$

Finding T(1,2)-

We have,

- $i = 1$
- $j = 2$
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

$$T(1,2) = \max \{ T(1-1, 2), 3 + T(1-1, 2-2) \}$$

$$T(1,2) = \max \{ T(0,2), 3 + T(0,0) \}$$

$$T(1,2) = \max \{ 0, 3+0 \}$$

$$T(1,2) = 3$$

Finding T(1,3)-

We have,

- $i = 1$
- $j = 3$
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

$$T(1,3) = \max \{ T(1-1, 3), 3 + T(1-1, 3-2) \}$$

$$T(1,3) = \max \{ T(0,3), 3 + T(0,1) \}$$

$$T(1,3) = \max \{ 0, 3+0 \}$$

$$T(1,3) = 3$$

Finding T(1,4)-

We have,

- $i = 1$
- $j = 4$
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

$$T(1,4) = \max \{ T(1-1, 4), 3 + T(1-1, 4-2) \}$$

$$T(1,4) = \max \{ T(0,4), 3 + T(0,2) \}$$

$$T(1,4) = \max \{ 0, 3+0 \}$$

$$T(1,4) = 3$$

Finding T(1,5)-

We have,

- $i = 1$
- $j = 5$
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

$$T(1,5) = \max \{ T(1-1, 5), 3 + T(1-1, 5-2) \}$$

$$T(1,5) = \max \{ T(0,5), 3 + T(0,3) \}$$

$$T(1,5) = \max \{ 0, 3+0 \}$$

$$T(1,5) = 3$$

Finding T(2,1)-

We have,

- $i = 2$
- $j = 1$
- $(value)_i = (value)_2 = 4$
- $(weight)_i = (weight)_2 = 3$

Substituting the values, we get-

$$T(2,1) = \max \{ T(2-1, 1), 4 + T(2-1, 1-3) \}$$

$$T(2,1) = \max \{ T(1,1), 4 + T(1,-2) \}$$

$$T(2,1) = T(1,1) \{ \text{Ignore } T(1,-2) \}$$

$$T(2,1) = 0$$

Finding T(2,2)-

We have,

- $i = 2$
- $j = 2$
- $(value)_i = (value)_2 = 4$
- $(weight)_i = (weight)_2 = 3$

Substituting the values, we get-

$$T(2,2) = \max \{ T(2-1, 2), 4 + T(2-1, 2-3) \}$$

$$T(2,2) = \max \{ T(1,2), 4 + T(1,-1) \}$$

$$T(2,2) = T(1,2) \{ \text{Ignore } T(1,-1) \}$$

$$T(2,2) = 3$$

Finding T(2,3)-

We have,

- $i = 2$
- $j = 3$
- $(value)_i = (value)_2 = 4$
- $(weight)_i = (weight)_2 = 3$

Substituting the values, we get-

$$T(2,3) = \max \{ T(2-1, 3), 4 + T(2-1, 3-3) \}$$

$$T(2,3) = \max \{ T(1,3), 4 + T(1,0) \}$$

$$T(2,3) = \max \{ 3, 4+0 \}$$

$$T(2,3) = 4$$

Finding T(2,4)-

We have,

- $i = 2$

- $j = 4$
- $(\text{value})_i = (\text{value})_2 = 4$
- $(\text{weight})_i = (\text{weight})_2 = 3$

Substituting the values, we get-

$$T(2,4) = \max \{ T(2-1, 4), 4 + T(2-1, 4-3) \}$$

$$T(2,4) = \max \{ T(1,4), 4 + T(1,1) \}$$

$$T(2,4) = \max \{ 3, 4+0 \}$$

$$T(2,4) = 4$$

Finding $T(2,5)$ -

We have,

- $i = 2$
- $j = 5$
- $(\text{value})_i = (\text{value})_2 = 4$
- $(\text{weight})_i = (\text{weight})_2 = 3$

Substituting the values, we get-

$$T(2,5) = \max \{ T(2-1, 5), 4 + T(2-1, 5-3) \}$$

$$T(2,5) = \max \{ T(1,5), 4 + T(1,2) \}$$

$$T(2,5) = \max \{ 3, 4+3 \}$$

$$T(2,5) = 7$$

Similarly, compute all the entries.

After all the entries are computed and filled in the table, we get the following table-

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

T-Table

[2]

The last entry represents the maximum possible value that can be put into the knapsack.

- So, maximum possible value that can be put into the knapsack = 7.

CONCLUSION:

From this assignment we have understood that Greedy strategy does not necessarily yield an optimal solution over a dynamic programming approach.

References:

1. AnanyLevitin, Introduction to the Design & Analysis of Algorithm, Pearson, ISBN 81- 7758-835-4.
2. <https://www.geeksforgeeks.org/>

ASSIGNMENT 2

TITLE: Bellman-Ford Algorithm using Dynamic Programming and verify the time complexity

PROBLEM STATEMENT:

Write a program to implement Bellman-Ford Algorithm using Dynamic Programming and verify the time complexity.

OBJECTIVE:

To understand and implement the program for Bellman-Ford algorithm using dynamic programming.

THEORY:

Dynamic Programming is also used in optimization problems. Like divide-and-conquer method, Dynamic Programming solves problems by combining the solutions of sub-problems. Moreover, Dynamic Programming algorithm solves each sub-problem just once and then saves its answer in a table, thereby avoiding the work of re-computing the answer every time.

Two main properties of a problem suggest that the given problem can be solved using Dynamic Programming. These properties are overlapping sub-problems and optimal substructure.

Overlapping Sub-Problems

Similar to Divide-and-Conquer approach, Dynamic Programming also combines solutions to sub-problems. It is mainly used where the solution of one sub-problem is needed repeatedly. The computed solutions are stored in a table, so that these don't have to be re-computed. Hence, this technique is needed where overlapping sub-problem exists.

For example, Binary Search does not have overlapping sub-problem. Whereas recursive program of Fibonacci numbers have many overlapping sub-problems.

Optimal Sub-Structure

A given problem has Optimal Substructure Property, if the optimal solution of the given problem can be obtained using optimal solutions of its sub-problems.

For example, the Shortest Path problem has the following optimal substructure property: If a node x lies in the shortest path from a source node u to destination node v , then the shortest path from u to v is the combination of the shortest path from u to x , and the shortest path from x to v .

The standard All Pair Shortest Path algorithms like Floyd-Warshall and Bellman-Ford are typical examples of Dynamic Programming.

Steps of Dynamic Programming Approach

Dynamic Programming algorithm is designed using the following four steps:

- Characterize the structure of an optimal solution.
- Recursively define the value of an optimal solution.
- Compute the value of an optimal solution, typically in a bottom-up fashion.
- Construct an optimal solution from the computed information.

Applications of Dynamic Programming Approach

- Matrix Chain Multiplication
- Longest Common Subsequence
- Travelling Salesman Problem

Bellman Ford Algorithm

This algorithm solves the single source shortest path problem of a directed graph $G = (V, E)$ in which the edge weights may be negative. Moreover, this algorithm can be applied to find the shortest path, if there does not exist any negative weighted cycle.

Algorithm: Bellman-Ford-Algorithm (G, w, s)

```

for each vertex  $v \in G.V$ 
     $v.d := \infty$ 
     $v.\Pi := NIL$ 
 $s.d := 0$ 
for  $i = 1$  to  $|G.V| - 1$ 
    for each edge  $(u, v) \in G.E$ 
        if  $v.d > u.d + w(u, v)$ 
             $v.d := u.d + w(u, v)$ 
             $v.\Pi := u$ 
for each edge  $(u, v) \in G.E$ 
    if  $v.d > u.d + w(u, v)$ 
        return FALSE
return TRUE

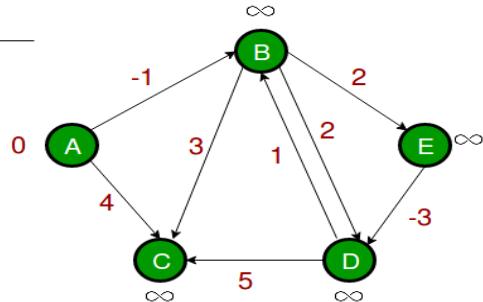
```

Analysis

The first for loop is used for initialization, which runs in (V) times. The next for loop runs $|V - 1|$ passes over the edges, which takes (E) times. Hence, Bellman-Ford algorithm runs in ($V \cdot E$) time.

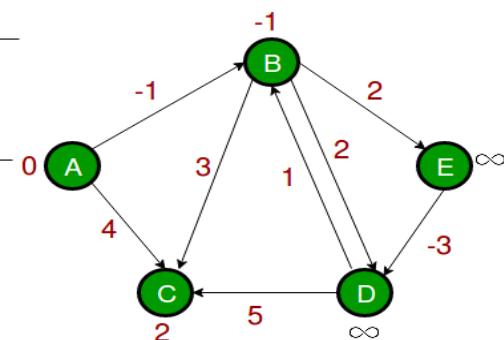
Example:

A	B	C	D	E
0	∞	∞	∞	∞



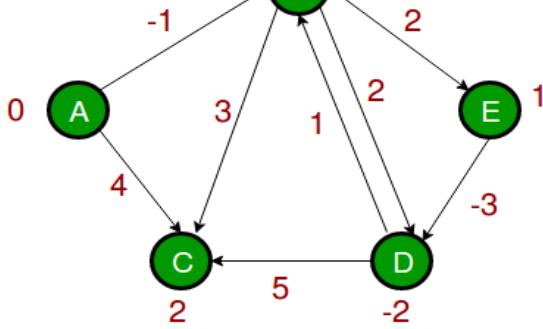
Let all edges are processed in the following order: (B, E), (D, B), (B, D), (A, B), (A, C), (D, C), (B, C), (E, D). We get the following distances when all edges are processed the first time. The first row shows initial distances. The second row shows distances when edges (B, E), (D, B), (B, D) and (A, B) are processed. The third row shows distances when (A, C) is processed. The fourth row shows when (D, C), (B, C) and (E, D) are processed.

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



The first iteration guarantees to give all shortest paths which are at most 1 edge long. We get the following distances when all edges are processed a second time (The last row shows final values).

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1



[2]

The second iteration guarantees to give all shortest paths which are at most 2 edges long. The algorithm processes all edges 2 more times. The distances are minimized after the second iteration, so third and fourth iterations don't update the distances.

Conclusion:

From this assignment we have understood the dynamic programming strategy and also successfully implemented Bellman-Ford algorithm.

References

1. AnanyLevitin, Introduction to the Design & Analysis of Algorithm, Pearson, ISBN 81- 7758-835-4.
2. <https://www.geeksforgeeks.org/>

ASSIGNMENT 3

TITLE: n-queens on chessboard so that no two queens attack each other using Backtracking.

PROBLEM STATEMENT:

Write a recursive program to find the solution of placing n queens on chessboard so that no two queens attack each other using Backtracking.

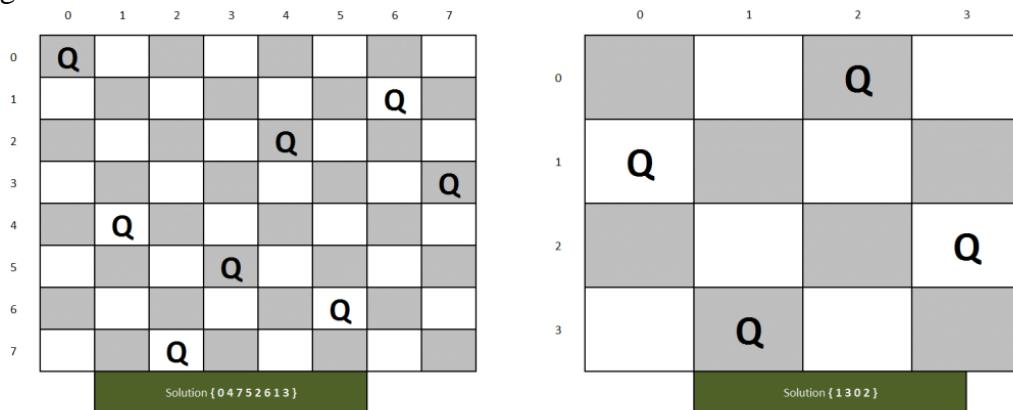
OBJECTIVE:

To understand and implement the program for n-queens problem using backtracking strategy.

THEORY:

In chess, a queen can move as far as she pleases, horizontally, vertically, or diagonally. A chess board has 8 rows and 8 columns. The standard 8 by 8 Queens problem asks how to place 8 queens on an ordinary chess board so that none of them can hit any other in one move. In chess, a queen can move as far as she pleases, horizontally, vertically, or diagonally. A chess board has 8 rows and 8 columns. The standard 8 by 8 Queens problem asks how to place 8 queens on an ordinary chess board so that none of them can hit any other in one move.

Naive Algorithm



Generate all possible configurations of queens on board and print a configuration that satisfies the given constraints.

Algorithm

while there are untried configurations

{

generate the next configuration

```

if queens don't attack in this configuration, then
{
    print this configuration;
}
}

```

Backtracking Algorithm

The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes then we backtrack and return false.

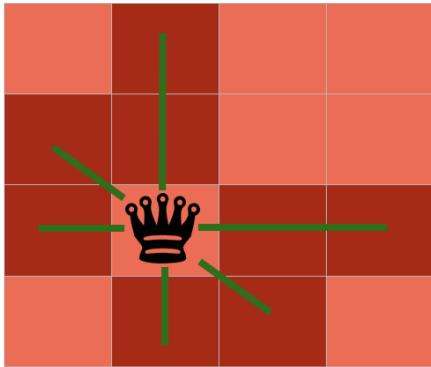
Algorithm

1. Start in the leftmost column
2. If all queens are placed return true
3. Try all rows in the current column. Do following for every tried row.
 - a. If the queen can be placed safely in this row, then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.
 - b. If placing queen in [row, column] leads to a solution then return true.
 - c. If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.
4. If all rows have been tried and nothing worked, return false to trigger backtracking.

Example

N Queens Problem

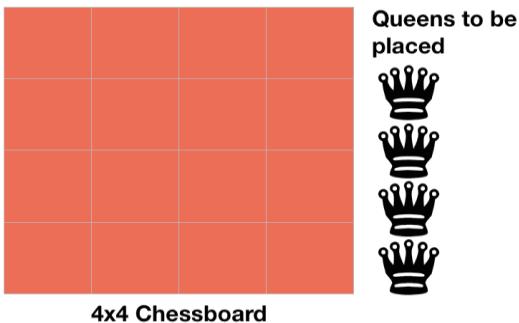
N Queens problem is one of the most common examples of backtracking. Our goal is to arrange N queens on an NxN chessboard such that no queen can strike down any other queen. A queen can attack horizontally, vertically, or diagonally.



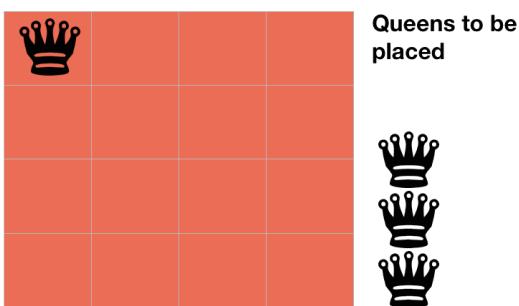
Cells attacked by the queen [3]

So, we start by placing the first queen anywhere arbitrarily and then place the next queen in any of the safe places. We continue this process until the number of unplaced queens becomes zero (a solution is found) or no safe place is left. If no safe place is left, then we change the position of the previously placed queen. Let's test this algorithm on a 4x4 chessboard.

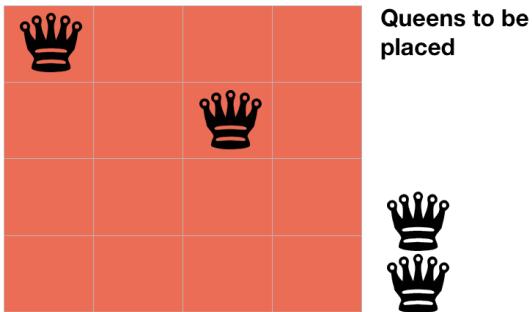
Using Backtracking to Solve N Queens



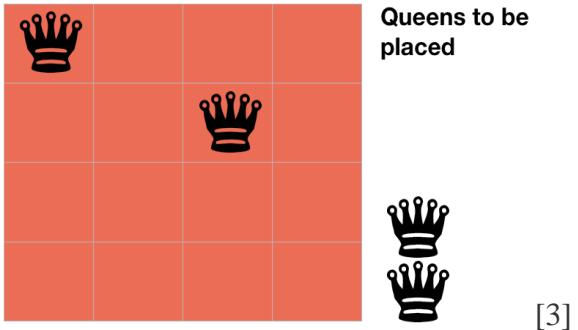
The above picture shows a 4x4 chessboard and we have to place 4 queens on it. So, we will start by placing the first queen in the first row.



Now, the second step is to place the second queen in a safe position. Also, we can't place the queen in the first row, so we will try putting the queen in the second row this time.

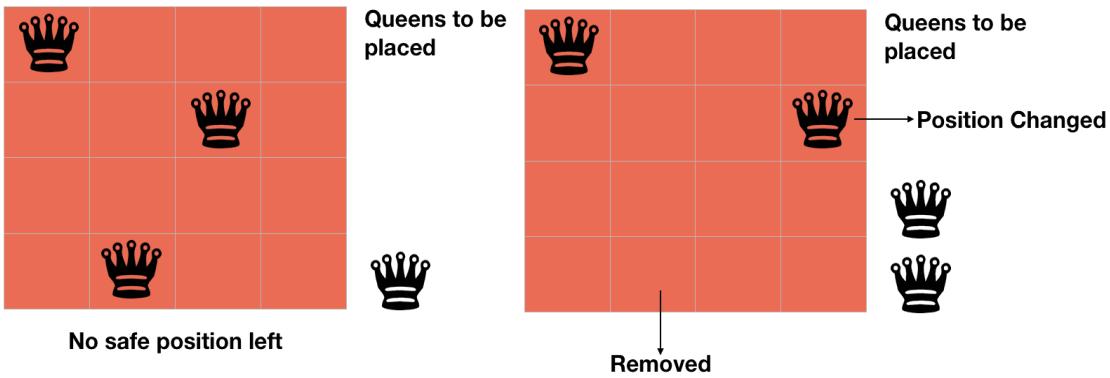


Let's place the third queen in a safe position, somewhere in the third row.

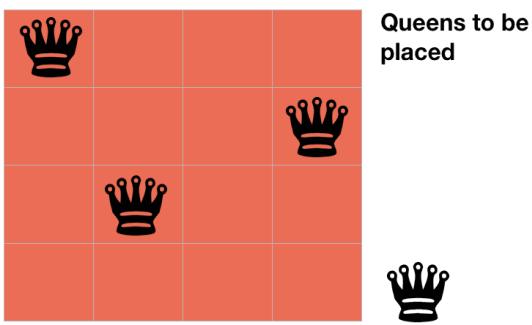


Now, we can see that there is no safe place where we can put the last queen. So, we will just change the position of the previous queen i.e., backtrack and change the previous decision.

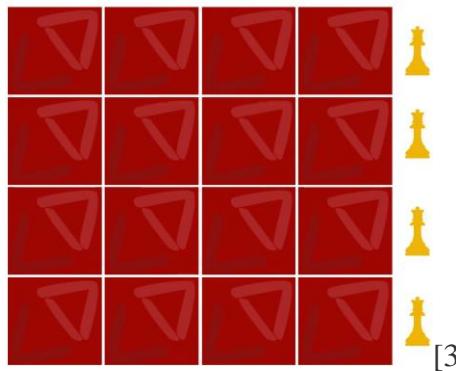
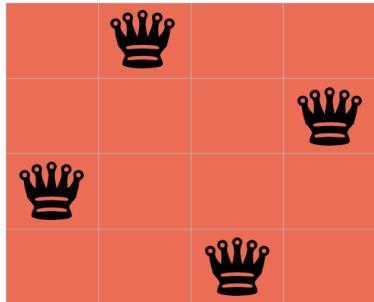
Also, there is no other position where we can place the third queen, so we will go back one more step and change the position of the second queen.



And now we will place the third queen again in a safe position other than the previously placed position in the third row.



We will continue this process and finally, we will get the solution as shown below.



CONCLUSION:

From this assignment we have understood the backtracking strategy and also successfully implemented n-queens problem.

References:

3. S. Sridhar, Design and Analysis of Algorithms, Oxford, ISBN 10: 0-19-809369-1.
4. AnanyLevitin, Introduction to the Design & Analysis of Algorithm, Pearson, ISBN 81- 7758-835-4.
5. <https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/>

ASSIGNMENT 4

TITLE:

The travelling salesman problem and to print the path and the cost using Branch and Bound.

PROBLEM STATEMENT:

Write a program to solve the travelling salesman problem and to print the path and the cost using Branch and Bound.

OBJECTIVE:

To understand and implement the program for travelling salesman using branch and bound.

THEORY:

A branch and bound algorithm is an optimization technique to get an optimal solution to the problem. It looks for the best solution for a given problem in the entire space of the solution. The bounds in the function to be optimized are merged with the value of the latest best solution. It allows the algorithm to find parts of the solution space completely.

The purpose of a branch and bound search is to maintain the lowest-cost path to a target. Once a solution is found, it can keep improving the solution. Branch and bound search is implemented in depth-bounded search and depth-first search.

Let us see how to apply it state space search tree. We start enumerating all possible nodes (preferably in lexicographical order)

The Root Node: Without loss of generality, we assume we start at vertex “0” for which the lower bound has been calculated above.

Dealing with Level 2: The next level enumerates all possible vertices we can go to (keeping in mind that in any path a vertex has to occur only once) which are, 1, 2, 3... n (Note that the graph is complete). Consider we are calculating for vertex 1, Since we moved from 0 to 1, our tour has now included the edge 0-1. This allows us to make necessary changes in the lower bound of the root.

Lower Bound for vertex 1 = Old lower bound - ((minimum edge cost of 0 + minimum edge cost of 1) / 2) + (edge cost 0-1)

To include edge 0-1, we add the edge cost of 0-1, and subtract an edge weight such that the lower bound remains as tight as possible which would be the sum of the minimum edges of 0 and 1 divided by 2. Clearly, the edge subtracted can't be smaller than this.

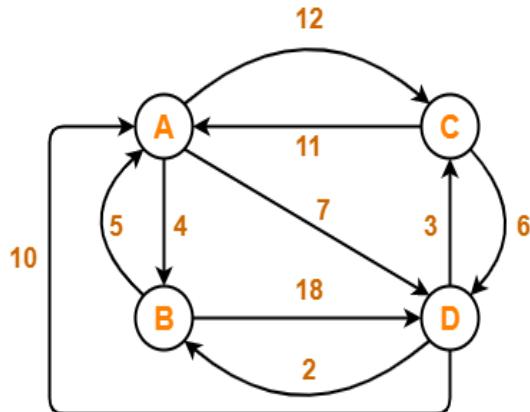
Dealing with other levels: As we move on to the next level, we again enumerate all possible vertices. For the above case going further after 1, we check out for 2, 3, 4, ...n. Consider lower bound for 2 as we moved from 1 to 1, we include the edge 1-2 to the tour and alter the new lower bound for this node. Lower bound(2) = Old lower bound - ((second minimum edge cost of 1 + minimum edge cost of 2)/2) + edge cost 1-2

Time Complexity

The worst-case complexity of Branch and Bound remains the same as that of the Brute Force clearly because in the worst case, we may never get a chance to prune a node. Whereas, in practice it performs very well depending on the different instance of the TSP. The complexity also depends on the choice of the bounding function as they are the ones deciding how many nodes to be pruned.

Problem-

Solve Travelling Salesman Problem using Branch and Bound Algorithm in the following graph-



Solution-

Step-01:

Write the initial cost matrix and reduce it-

	A	B	C	D
A	∞	4	12	7
B	5	∞	∞	18
C	11	∞	∞	6
D	10	2	3	∞

Rules

- To reduce a matrix, perform the row reduction and column reduction of the matrix separately.

- A row or a column is said to be reduced if it contains at least one entry ‘0’ in it.

Row Reduction-

Consider the rows of above matrix one by one.

If the row already contains an entry ‘0’, then-

- There is no need to reduce that row.

If the row does not contain an entry ‘0’, then-

- Reduce that particular row.
- Select the least value element from that row.
- Subtract that element from each element of that row.
- This will create an entry ‘0’ in that row, thus reducing that row.

Following this, we have-

- Reduce the elements of row-1 by 4.
- Reduce the elements of row-2 by 5.
- Reduce the elements of row-3 by 6.
- Reduce the elements of row-4 by 2.

Performing this, we obtain the following row-reduced matrix-

$$\begin{array}{cccc}
 & \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} \\
 \mathbf{A} & \left[\begin{array}{cccc}
 \infty & 0 & 8 & 3
 \end{array} \right] \\
 \mathbf{B} & \left[\begin{array}{cccc}
 0 & \infty & \infty & 13
 \end{array} \right] \\
 \mathbf{C} & \left[\begin{array}{cccc}
 5 & \infty & \infty & 0
 \end{array} \right] \\
 \mathbf{D} & \left[\begin{array}{cccc}
 8 & 0 & 1 & \infty
 \end{array} \right]
 \end{array}$$

Column Reduction-

Consider the columns of above row-reduced matrix one by one.

If the column already contains an entry ‘0’, then-

- There is no need to reduce that column.

If the column does not contain an entry ‘0’, then-

- Reduce that particular column.

- Select the least value element from that column.
- Subtract that element from each element of that column.
- This will create an entry ‘0’ in that column, thus reducing that column.

Following this, we have-

- There is no need to reduce column-1.
- There is no need to reduce column-2.
- Reduce the elements of column-3 by 1.
- There is no need to reduce column-4.

Performing this, we obtain the following column-reduced matrix-

	A	B	C	D
A	∞	0	7	3
B	0	∞	∞	13
C	5	∞	∞	0
D	8	0	0	∞

Finally, the initial distance matrix is completely reduced.

Now, we calculate the cost of node-1 by adding all the reduction elements.

Cost(1)

$$\begin{aligned}
 &= \text{Sum of all reduction elements} \\
 &= 4 + 5 + 6 + 2 + 1 \\
 &= 18
 \end{aligned}$$

Step-02:

- We consider all other vertices one by one.
- We select the best vertex where we can land upon to minimize the tour cost.

Choosing To Go To Vertex-B: Node-2 (Path A → B)

From the reduced matrix of step-01, $M[A,B] = 0$

- Set row-A and column-B to ∞
- Set $M[B,A] = \infty$

Now, resulting cost matrix is-

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	13
C	5	∞	∞	0
D	8	∞	0	∞

Now,

- We reduce this matrix.
- Then, we find out the cost of node-02.

Row Reduction-

- We can not reduce row-1 as all its elements are ∞ .
- Reduce all the elements of row-2 by 13.
- There is no need to reduce row-3.
- There is no need to reduce row-4.

Performing this, we obtain the following row-reduced matrix-

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	0
C	5	∞	∞	0
D	8	∞	0	∞

Column Reduction-

- Reduce the elements of column-1 by 5.
- We can not reduce column-2 as all its elements are ∞ .
- There is no need to reduce column-3.
- There is no need to reduce column-4.

Performing this, we obtain the following column-reduced matrix-

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	0
C	0	∞	∞	0
D	3	∞	0	∞

Finally, the matrix is completely reduced.

Now, we calculate the cost of node-2.

$$\begin{aligned}
 \text{Cost}(2) &= \text{Cost}(1) + \text{Sum of reduction elements} + M[A,B] \\
 &= 18 + (13 + 5) + 0 \\
 &= 36
 \end{aligned}$$

Choosing To Go To Vertex-C: Node-3 (Path A → C)

- From the reduced matrix of step-01, $M[A,C] = 7$
- Set row-A and column-C to ∞
- Set $M[C,A] = \infty$

Now, resulting cost matrix is-

	A	B	C	D
A	∞	∞	∞	∞
B	0	∞	∞	13
C	∞	∞	∞	0
D	8	0	∞	∞

Now,

- We reduce this matrix.
- Then, we find out the cost of node-03.

Row Reduction-

- We can not reduce row-1 as all its elements are ∞ .
- There is no need to reduce row-2.
- There is no need to reduce row-3.
- There is no need to reduce row-4.

Thus, the matrix is already row-reduced.

Column Reduction-

- There is no need to reduce column-1.
- There is no need to reduce column-2.
- We can not reduce column-3 as all its elements are ∞ .
- There is no need to reduce column-4.

Thus, the matrix is already column reduced.

Finally, the matrix is completely reduced.

Now, we calculate the cost of node-3.

Cost(3)

$$\begin{aligned} &= \text{Cost}(1) + \text{Sum of reduction elements} + M[A,C] \\ &= 18 + 0 + 7 \\ &= 25 \end{aligned}$$

Choosing To Go To Vertex-D: Node-4 (Path A → D)

- From the reduced matrix of step-01, $M[A,D] = 3$
- Set row-A and column-D to ∞
- Set $M[D,A] = \infty$

Now, resulting cost matrix is-

	A	B	C	D
A	∞	∞	∞	∞
B	0	∞	∞	∞
C	5	∞	∞	∞
D	∞	0	0	∞

Now,

- We reduce this matrix.
- Then, we find out the cost of node-04.

Row Reduction-

- We can not reduce row-1 as all its elements are ∞ .
- There is no need to reduce row-2.
- Reduce all the elements of row-3 by 5.
- There is no need to reduce row-4.

Performing this, we obtain the following row-reduced matrix-

	A	B	C	D
A	∞	∞	∞	∞
B	0	∞	∞	∞
C	0	∞	∞	∞
D	∞	0	0	∞

Column Reduction-

- There is no need to reduce column-1.
- There is no need to reduce column-2.
- There is no need to reduce column-3.
- We can not reduce column-4 as all its elements are ∞ .

Thus, the matrix is already column-reduced.

Finally, the matrix is completely reduced.

Now, we calculate the cost of node-4.

$$\begin{aligned}
 \text{Cost}(4) &= \text{Cost}(1) + \text{Sum of reduction elements} + M[A,D] \\
 &= 18 + 5 + 3 \\
 &= 26
 \end{aligned}$$

Thus, we have-

- Cost(2) = 36 (for Path A → B)
- Cost(3) = 25 (for Path A → C)
- Cost(4) = 26 (for Path A → D)

We choose the node with the lowest cost.

Since cost for node-3 is lowest, so we prefer to visit node-3.

Thus, we choose node-3 i.e. path A → C.

Step-03:

We explore the vertices B and D from node-3.

We now start from the cost matrix at node-3 which is-

	A	B	C	D
A	∞	∞	∞	∞
B	0	∞	∞	13
C	∞	∞	∞	0
D	8	0	∞	∞

Cost(3) = 25

Choosing To Go To Vertex-B: Node-5 (Path A → C → B)

- From the reduced matrix of step-02, $M[C,B] = \infty$
- Set row-C and column-B to ∞
- Set $M[B,A] = \infty$

Now, resulting cost matrix is-

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	13
C	∞	∞	∞	∞
D	8	∞	∞	∞

Now,

- We reduce this matrix.
- Then, we find out the cost of node-5.

Row Reduction-

- We can not reduce row-1 as all its elements are ∞ .
- Reduce all the elements of row-2 by 13.
- We can not reduce row-3 as all its elements are ∞ .
- Reduce all the elements of row-4 by 8.

Performing this, we obtain the following row-reduced matrix-

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	0
C	∞	∞	∞	∞
D	0	∞	∞	∞

Column Reduction-

- There is no need to reduce column-1.
- We can not reduce column-2 as all its elements are ∞ .
- We can not reduce column-3 as all its elements are ∞ .

- There is no need to reduce column-4.

Thus, the matrix is already column reduced.

Finally, the matrix is completely reduced.

Now, we calculate the cost of node-5.

Cost(5)

$$= \text{cost}(3) + \text{Sum of reduction elements} + M[C,B]$$

$$= 25 + (13 + 8) + \infty$$

$$= \infty$$

Choosing To Go To Vertex-D: Node-6 (Path A → C → D)

- From the reduced matrix of step-02, $M[C,D] = \infty$
- Set row-C and column-D to ∞
- Set $M[D,A] = \infty$

Now, resulting cost matrix is-

	A	B	C	D
A	∞	∞	∞	∞
B	0	∞	∞	∞
C	∞	∞	∞	∞
D	∞	0	∞	∞

Now,

- We reduce this matrix.
- Then, we find out the cost of node-6.

Row Reduction-

- We can not reduce row-1 as all its elements are ∞ .
- There is no need to reduce row-2.
- We can not reduce row-3 as all its elements are ∞ .
- We can not reduce row-4 as all its elements are ∞ .

Thus, the matrix is already row reduced.

Column Reduction-

- There is no need to reduce column-1.
- We can not reduce column-2 as all its elements are ∞ .
- We can not reduce column-3 as all its elements are ∞ .
- We can not reduce column-4 as all its elements are ∞ .

Thus, the matrix is already column reduced.

Finally, the matrix is completely reduced.

Now, we calculate the cost of node-6.

Cost(6)

$$\begin{aligned} &= \text{cost}(3) + \text{Sum of reduction elements} + M[C,D] \\ &= 25 + 0 + 0 \\ &= 25 \end{aligned}$$

Thus, we have-

- Cost(5) = ∞ (for Path A → C → B)
- Cost(6) = 25 (for Path A → C → D)

We choose the node with the lowest cost.

Since cost for node-6 is lowest, so we prefer to visit node-6.

Thus, we choose node-6 i.e. path **C → D**.

Step-04:

We explore vertex B from node-6.

We start with the cost matrix at node-6 which is-

	A	B	C	D
A	∞	∞	∞	∞
B	0	∞	∞	∞
C	∞	∞	∞	∞
D	∞	0	∞	∞

Cost(6) = 25

Choosing To Go To Vertex-B: Node-7 (Path A → C → D → B)

- From the reduced matrix of step-03, $M[D,B] = 0$
- Set row-D and column-B to ∞
- Set $M[B,A] = \infty$

Now, resulting cost matrix is-

	A	B	C	D
A	∞	∞	∞	∞
B	∞	∞	∞	∞
C	∞	∞	∞	∞
D	∞	∞	∞	∞

Now,

- We reduce this matrix.
- Then, we find out the cost of node-7.

Row Reduction-

- We can not reduce row-1 as all its elements are ∞ .
- We can not reduce row-2 as all its elements are ∞ .
- We can not reduce row-3 as all its elements are ∞ .
- We can not reduce row-4 as all its elements are ∞ .

Column Reduction-

- We can not reduce column-1 as all its elements are ∞ .
- We can not reduce column-2 as all its elements are ∞ .
- We can not reduce column-3 as all its elements are ∞ .
- We can not reduce column-4 as all its elements are ∞ .

Thus, the matrix is already column reduced.

Finally, the matrix is completely reduced.

All the entries have become ∞ .

Now, we calculate the cost of node-7.

$$\begin{aligned}
 \text{Cost}(7) &= \text{cost}(6) + \text{Sum of reduction elements} + M[D,B] \\
 &= 25 + 0 + 0 \\
 &= 25
 \end{aligned}$$

Thus,

- Optimal path is: A → C → D → B → A
- Cost of Optimal path = **25 units**

CONCLUSION:

From this assignment we have understood the branch and bound method and also successfully implemented the travelling salesman problem.

References:

6. Horowitz and Sahani, Fundamentals of computer Algorithms, Galgotia, ISBN 81-7371-612-9.
7. <https://www.gatevidyalay.com/travelling-salesman-problem-using-branch-and-bound-approach/>

PART B:

II. Advanced Database Management Systems

Assignment: 1

AIM:

Create a database with suitable example using MongoDB and implement

1. Inserting and saving document (batch insert, insert validation)
2. Removing document
3. Updating document (document replacement, using modifiers, up inserts, updating multiple documents, returning updated documents)
4. Execute at least 10 queries on any suitable MongoDB database that demonstrates following:
 - Find and find One (specific values)
 - Query criteria (Query conditionals, OR queries, \$not, Conditional semantics)
 - Type-specific queries (Null, Regular expression, Querying arrays) where queries
 - Cursors (Limit, skip, sort, advanced query options)

PROBLEM STATEMENT /DEFINITION

Create a database with suitable example using MongoDB and implement

- Inserting and saving document (batch insert, insert validation)
- Removing document
- Updating document (document replacement, using modifiers, upserts, updating multiple documents, returning updated documents)

OBJECTIVE:

To understand Mongodbd basic commands

To implement the concept of document oriented databases.

To understand Mongodbd retrieval commands

THEORY:

SQL VsMongoDB

SQL Concepts	MongoDB Concepts
Database	Database
Table	Collection

Row	Document Or BSON Document
Column	Field
Index	Index
Table Join	Embedded Documents & Linking
Primary Key	Primary Key
Specify Any Unique Column Or Column Combination As Primary Key.	In Mongodb, The Primary Key Is Automatically Set To The <u>Id</u> Field.
Aggregation (E.G. Group By)	Aggregation Pipeline

1.Create a collection in mongodb

```
db.createCollection("Teacher_info")
```

2.Create a capped collection in mongodb

```
>db.createCollection("audit", {capped:true, size:20480})
```

```
{ "ok" : 1 }
```

3.Insert a document into collection

```
db.Teacher_info.insert( { Teacher_id: "Pic001", Teacher_Name: "Ravi",Dept_Name: "IT", Sal:30000, status: "A" } )
```

```
db.Teacher_info.insert( { Teacher_id: "Pic002", Teacher_Name: "Ravi",Dept_Name: "IT", Sal:20000, status: "A" } )
```

```
db.Teacher_info.insert( { Teacher_id: "Pic003", Teacher_Name: "Akshay",Dept_Name: "Comp", Sal:25000, status: "N" } )
```

4.Update a document into collection

```
db.Teacher_info.update( { sal: { $gt: 25000 } }, { $set: { Dept_name: "ETC" } }, { multi: true } )
```

```
db.Teacher_info.update( { status: "A" } , { $inc: { sal: 10000 } }, { multi: true } )
```

5.Remove a document from collection

```
db.Teacher_info.remove({Teacher_id: "pic001"});  
db.Teacher_info.remove({})
```

6.Alter a field into a mongodb document

```
db.Teacher_info.update( { }, { $set: { join_date: new Date() } }, { multi: true } )
```

7.To drop a particular collection

```
db.Teacher_info.drop()
```

Retrieval From Database:-

When we retrieve a document from mongodb collection it always add a _id field in the every document which conatin unique _id field.

ObjectId(<hexadecimal>)

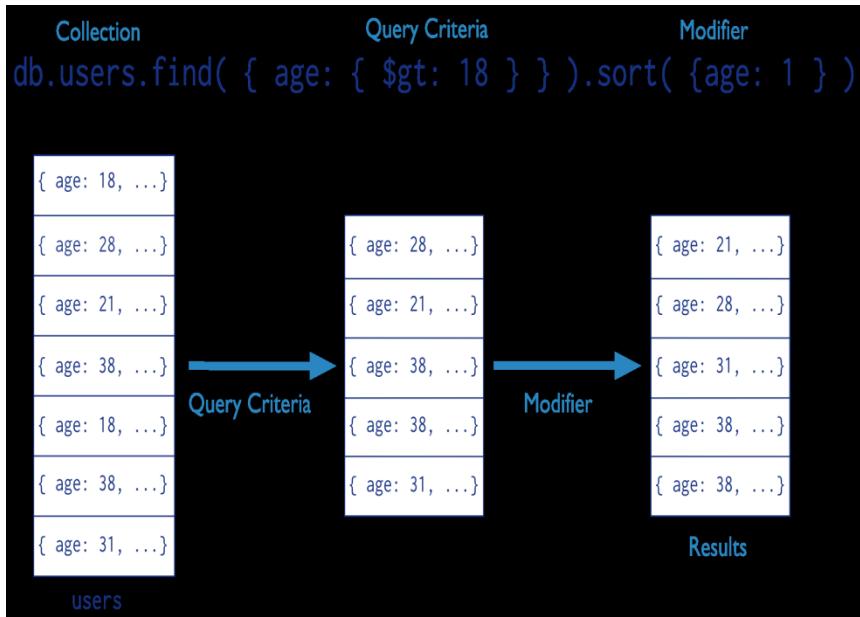
Returns a new ObjectId value. The 12-byte ObjectId value consists of:

4-byte value representing the seconds since the Unix epoch,

3-byte machine identifier,

2-byte process id, and

3-byte counter,starting with a random value.



1. Retrieve a collection in mongodb using Find command

```
db.Teacher.find()
```

```
{
  "_id" : 101, "Name" : "Dev",
  "Address" : [ { "City" : "Pune", "Pin" : 444043 } ],
  "Department" : [ { "Dept_id" : 111, "Dept_name" : "IT" } ],
  "Salary" : 78000 }

{
  "_id" : 135, "Name" : "Jennifer", "Address" : [ { "City" :
  "Mumbai", "Pin" : 444111 } ], "Department" : [ { "Dept_id" : 112,
  "Dept_name" : "COMP" } ], "Salary" : 65000 }
{ "_id" : 126, "Name" : "Gaurav", "Address" : [ { "City" : "Nashik",
  "Pin" : 444198 } ], "Department" : [ { "Dept_id" : 112, "Dept_name"
  : "COMP" } ], "Salary" : 90000 }
{ "_id" : 175, "Name" : "Shree", "Address" : [ { "City" : "Nagpur",
  "Pin" : 444158 } ], "Department" : [ { "Dept_id" : 113, "Dept_name"
  : "ENTC" } ], "Salary" : 42000 }
{ "_id" : 587, "Name" : "Raman", "Address" : [ { "City" : "Banglore",
  "Pin" : 445754 } ], "Department" : [ { "Dept_id" : 113, "Dept_name"
  : "ENTC" } ], "Salary" : 79000 }
{ "_id" : 674, "Name" : "Mandar", "Address" : [ { "City" : "Jalgaon",
  "Pin" : 465487 } ], "Department" : [ { "Dept_id" : 111, "Dept_name"
  : "IT" } ], "Salary" : 88000 }
{ "_id" : 573, "Name" : "Manish", "Address" : [ { "City" : "Washim",
  "Pin" : 547353 } ], "Department" : [ { "Dept_id" : 112, "Dept_name"
  : "COMP" } ], "Salary" : 65000 }
```

2. Retrieve a document from collection in mongodb using Find command using condition

```
>db.Teacher_info.find({sal: 25000})
```

3. Retrieve a document from collection in mongodb using Find command using or operator

```
>db.Teacher_info.find( { $or: [ { status: "A" } , { sal:50000 } ] } )
```

4. Retrieve a document from collection in mongodb using Find command using greater than , less than, greater than and equal to ,less than and equal to operator

```
>db.Teacher_info.find( { sal: { $gt: 40000 } } )
```

```
>db.media.find( { Released : {$gt : 2000} }, { "Cast" : 0 } )
```

```
{ "_id" : ObjectId("4c4369a3c603000000007ed3"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 }
```

```
>db.media.find ( { Released : {$gte : 1999} }, { "Cast" : 0 } )
```

```
{ "_id" : ObjectId("4c43694bc603000000007ed1"), "Type" : "DVD", "Title" : "Matrix, The", "Released" : 1999 }
```

```
{ "_id" : ObjectId("4c4369a3c603000000007ed3"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 }
```

```
>db.media.find ( { Released : {$lt : 1999} }, { "Cast" : 0 } )
```

```
{ "_id" : ObjectId("4c436969c603000000007ed2"), "Type" : "DVD", "Title" : "Blade Runner", "Released" : 1982 }
```

```
>db.media.find( {Released : {$lte: 1999}}, { "Cast" : 0 })
```

```
{ "_id" : ObjectId("4c43694bc603000000007ed1"), "Type" : "DVD", "Title" : "Matrix, The", "Released" : 1999 }
```

```
{ "_id" : ObjectId("4c436969c603000000007ed2"), "Type" : "DVD", "Title" : "Blade Runner", "Released" : 1982 }
```

```
>db.media.find( {Released : {$gte: 1990, $lt : 2010}}, { "Cast" : 0 })
```

```
{ "_id" : ObjectId("4c43694bc603000000007ed1"), "Type" : "DVD", "Title" : "Matrix, The", "Released" : 1999 }
```

Retrieval a value from document which contain array field

Exact Match on an Array

```
db.inventory.find( { tags: [ 'fruit', 'food', 'citrus' ] } )
```

Match an Array Element

```
db.inventory.find( { tags: 'fruit' } )
```

Match a Specific Element of an Array

```
db.inventory.find( { 'tags.0' : 'fruit' } )
```

6.MongoDB provides a db.collection.findOne() method as a special case of find() that returns a single document.

7.Exclude One Field from a Result Set

```
>db.records.Find( { "user_id": { $lt: 42 } }, { history: 0 } )
```

8.Return Two fields and the _id Field

```
>db.records.find( { "user_id": { $lt: 42 } }, { "name": 1, "email": 1 } )
```

9.Return Two Fields and Exclude _id

```
>db.records.find( { "user_id": { $lt: 42 } }, { "_id": 0, "name": 1, "email": 1 } )
```

10. Retrieve a collection in mongodb using Find command and pretty appearance

```
>db.<collection>.find().pretty()
```

db.users.find(collection
{age:{\$gt:18}},	query
criteria	
{name :1,address:1}	projection

Retrieve a document in ascending or descending order using 1 for ascending and -1 for descending from collection in mongodb

```
>db.Teacher_info.find( { status: "A" } ).sort( { sal: -1 } )
```

```
>db.audit.find().sort( { $natural: -1 } ).limit( 10 )
```

```
>db.Employee.find().sort({_id:-1})
```

```
{
  "_id" : 106, "Name" : "RAJ", "Address" : [ { "City" : "NASIK", "Pin" : 411002 } ], "Department" : [ { "Dept_id" : 113, "Dept_name" : "ACCOUNTING" } ], "Salary" : 50000
{
  "_id" : 105, "Name" : "ASHOK", "Address" : [ { "City" : "NASIK", "Pin" : 411002 } ], "Department" : [ { "Dept_id" : 113, "Dept_name" : "ACCOUNTING" } ], "Salary" : 40000
{
  "_id" : 104, "Name" : "JOY", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 112, "Dept_name" : "SALES" } ], "Salary" : 20000
{
  "_id" : 103, "Name" : "RAM", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 112, "Dept_name" : "SALES" } ], "Salary" : 10000
{
  "_id" : 102, "Name" : "AKASH", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 111, "Dept_name" : "HR" } ], "Salary" : 80000
{
  "_id" : 101, "Name" : "Dev", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 111, "Dept_name" : "HR" } ], "Salary" : 78000 }
```

>db.Employee.find().sort({_id:1})

```
{
  "_id" : 101, "Name" : "Dev", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 111, "Dept_name" : "HR" } ], "Salary" : 78000
{
  "_id" : 102, "Name" : "AKASH", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 111, "Dept_name" : "HR" } ], "Salary" : 80000 }
```

```

{ "_id" : 103, "Name" : "RAM", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 112, "Dept_name" : "SALES" } ], "Salary" : 10000 }
{ "_id" : 104, "Name" : "JOY", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 112, "Dept_name" : "SALES" } ], "Salary" : 20000 }
{ "_id" : 105, "Name" : "ASHOK", "Address" : [ { "City" : "NASIK", "Pin" : 411002 } ], "Department" : [ { "Dept_id" : 113, "Dept_name" : "ACCOUNTING" } ], "Salary" : 40000 }
{ "_id" : 106, "Name" : "RAJ", "Address" : [ { "City" : "NASIK", "Pin" : 411002 } ], "Department" : [ { "Dept_id" : 113, "Dept_name" : "ACCOUNTING" } ], "Salary" : 50000 }
>db.Employee.find().sort({$natural:-1}).limit(2)
{ "_id" : 106, "Name" : "RAJ", "Address" : [ { "City" : "NASIK", "Pin" : 411002 } ], "Department" : [ { "Dept_id" : 113, "Dept_name" : "ACCOUNTING" } ], "Salary" : 50000 }
{ "_id" : 105, "Name" : "ASHOK", "Address" : [ { "City" : "NASIK", "Pin" : 411002 } ], "Department" : [ { "Dept_id" : 113, "Dept_name" : "ACCOUNTING" } ], "Salary" : 40000 }

```

>db.Employee.find().sort({\$natural:1}).limit(2)

```

{ "_id" : 101, "Name" : "Dev", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 111, "Dept_name" : "HR" } ], "Salary" : 78000 }
{ "_id" : 102, "Name" : "AKASH", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 111, "Dept_name" : "HR" } ], "Salary" : 80000 }
>db.Employee.find({Salary:{$in:[10000,30000]}})
{ "_id" : 103, "Name" : "RAM", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ], "Department" : [ { "Dept_id" : 112, "Dept_name" : "SALES" } ], "Salary" : 10000 }
>db.Employee.update({"Name":"RAM"}, { $set :{Address:{City: "Nasik"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>db.Employee.find({"Name":"RAM"})
{ "_id" : 103, "Name" : "RAM", "Address" : { "City" : "Nasik" }, "Department" : [ { "Dept_id" : 112, "Dept_name" : "SALES" } ], "Salary" : 10000 }
>db.Employee.update({"Name":"RAM"}, {$inc :{"Salary": 10000 } })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>db.Employee.find({"Name":"RAM"})
{ "_id" : 103, "Name" : "RAM", "Address" : { "City" : "Nasik" }, "Department" : [ { "Dept_id" : 112, "Dept_name" : "SALES" } ], "Salary" : 20000 }

```

Retrieve documentwith a particular from collection in mongodb

>db.Employee.find().limit(2).pretty()

```
{
  "_id" : 101,
```

```

    "Name" : "Dev",
    "Address" : [
        {
            "City" : "Pune",
            "Pin" : 444043
        }
    ],
    "Department" : [
        {
            "Dept_id" : 111,
            "Dept_name" : "HR"
        }
    ],
    "Salary" : 78000
}
{
    "_id" : 102,
    "Name" : "AKASH",
    "Address" : [
        {
            "City" : "Pune",
            "Pin" : 444043
        }
    ],
    "Department" : [
        {
            "Dept_id" : 111,
            "Dept_name" : "HR"
        }
    ],
    "Salary" : 80000
}

```

Retrieve document skipping some documents from collection in mongodb

```

>db.Employee.find().skip(3).pretty()
{
    "_id" : 104,
    "Name" : "JOY",
    "Address" : [
        {
            "City" : "Pune",
            "Pin" : 444043
        }
    ],
    "Department" : [
        {
            "Dept_id" : 112,
            "Dept_name" : "SALES"
        }
    ],
    "Salary" : 20000
}

```

```

}
{
    "_id" : 105,
    "Name" : "ASHOK",
    "Address" : [
        {
            "City" : "NASIK",
            "Pin" : 411002
        }
    ],
    "Department" : [
        {
            "Dept_id" : 113,
            "Dept_name" : "ACCOUNTING"
        }
    ],
    "Salary" : 40000
}
{
    "_id" : 106,
    "Name" : "RAJ",
    "Address" : [
        {
            "City" : "NASIK",
            "Pin" : 411002
        }
    ],
    "Department" : [
        {
            "Dept_id" : 113,
            "Dept_name" : "ACCOUNTING"
        }
    ],
    "Salary" : 50000
}

```

REFERENCE BOOK:

Kristina Chodorow, MongoDB The definitive guide, O'Reilly Publications, ISBN:978-93-5110-269-4,2nd Edition.

CONCLUSION:

Understand to implement data from mongodb database with the help of statement and operators.

Assignment: 2

AIM: Implement Map reduces operation with suitable example on above MongoDB database

- Aggregation framework
- Create and drop different types of indexes and explain () to show the advantage of the indexes.

PROBLEM STATEMENT /DEFINITION

Implement Map reduces operation with suitable example on above MongoDB database

- Aggregation framework
- Create and drop different types of indexes and explain () to show the advantage of the indexes.

OBJECTIVE:

To understand the concept of Mapreduce in mongodb.

To understand the concept of Aggregation in mongodb.

To implement the concept of document oriented databases.

THEORY:

- Implements the MapReduce model for processing large data sets.
- Can choose from one of several output options (inline, new collection, merge, replace, reduce)
- MapReduce functions are written in JavaScript.
- Supports non-sharded and sharded input collections.
- Can be used for incremental aggregation over large collections.
- MongoDB 2.2 implements much better support for sharded map reduce output.
- New feature in the Mongodb2.2.0 production release (August, 2012).
- Designed with specific goals of **improving performance** and **usability**.
- Returns result set inline.
- Supports **non-sharded and sharded** input collections.

- Uses a "**pipeline**" approach where objects are transformed as they pass through a series of pipeline operators such as matching, projecting, sorting, and grouping.
- **Pipeline operators need not produce one output document for every input document:** operators may also generate new documents or filter out documents.
- Map/Reduce involves two steps:
- first, map the data from the collection specified;
- second, reduce the results.

```
>db.createCollection("Order")
• { "ok" : 1 }

>db.order.insert({cust_id:"A123",amount:500,status:"A"})
• WriteResult({ "nInserted" : 1 })

>db.order.insert({cust_id:"A123",amount:250,status:"A"})
• WriteResult({ "nInserted" : 1 })

>db.order.insert({cust_id:"B212",amount:200,status:"A"})
• WriteResult({ "nInserted" : 1 })

>db.order.insert({cust_id:"A123",amount:300,status:"d"})
• WriteResult({ "nInserted" : 1 })
```

• **Map Function**

- var mapFunction1 = function()
- { emit(this.cust_id, this.amount);};

• **Reduce Function**

- var reduceFunction1 = function(key, values)
- {return Array.sum(values);};

db.order.mapReduce

```
(mapFunction1, reduceFunction1, {query: {status: "A"}},  
out: "order_totals"));
```

```

"result" : "order_totals",
"timeMillis" : 28,
"counts" : {
    "input" : 3,
    "emit" : 3,
    "reduce" : 1,
    "output" : 2
},
"ok" : 1,

```

>db.order.mapReduce(

Map Function -> function() { emit(this.cust_id, this.amount);},

Reduce Function -> function(key, values) { return Array.sum (values)},

Query à {query: { status:"A"},

Output collection à out: "order_ totals"})

```

{
    "result" : "order_totals",
    "timeMillis" : 27,
    "counts" : {
        "input" : 3,
        "emit" : 3,
        "reduce" : 1,
        "output" : 2
    },
    "ok" : 1,
}

```

To display result of mapReduce function use collection created in OUT.

Db.<collection name>.find();

```
db.order_totals.find();
{ "_id" : "A123", "value" : 750 }
{ "_id" : "B212", "value" : 200 }
```

Implementation of Aggregation:-

```
> use Teacher
switched to db Teacher
>db.Teacher.find()
{ "_id" : 101, "Name" : "Dev", "Address" : [ { "City" : "Pune", "Pin" : 444043 } ],
"Department" : [ { "Dept_id" : 111, "Dept_name" : "IT" } ], "Salary" : 78000 }
{ "_id" : 135, "Name" : "Jennifer", "Address" : [ { "City" : "Mumbai", "Pin" : 444111 } ],
"Department" : [ { "Dept_id" : 112, "Dept_name" : "COMP" } ], "Salary" : 65000 }
{ "_id" : 126, "Name" : "Gaurav", "Address" : [ { "City" : "Nashik", "Pin" : 444198 } ],
"Department" : [ { "Dept_id" : 112, "Dept_name" : "COMP" } ], "Salary" : 90000 }
{ "_id" : 175, "Name" : "Shree", "Address" : [ { "City" : "Nagpur", "Pin" : 444158 } ],
"Department" : [ { "Dept_id" : 113, "Dept_name" : "ENTC" } ], "Salary" : 42000 }
{ "_id" : 587, "Name" : "Raman", "Address" : [ { "City" : "Banglore", "Pin" : 445754 } ],
"Department" : [ { "Dept_id" : 113, "Dept_name" : "ENTC" } ], "Salary" : 79000 }
{ "_id" : 674, "Name" : "Mandar", "Address" : [ { "City" : "Jalgaon", "Pin" : 465487 } ],
"Department" : [ { "Dept_id" : 111, "Dept_name" : "IT" } ], "Salary" : 88000 }
{ "_id" : 573, "Name" : "Manish", "Address" : [ { "City" : "Washim", "Pin" : 547353 } ],
"Department" : [ { "Dept_id" : 112, "Dept_name" : "COMP" } ], "Salary" : 65000 }

>db.Teacher.aggregate([
... {$group:{_id:"$Department",totalsalary:{$sum:"$Salary"}},}
... ])
{
  "result" : [
    {
      "_id" : [
        {
          "Dept_id" : 113,
          "Dept_name" : "ENTC"
        }
      ],
      "totalsalary" : 121000
    },
    {
      "_id" : [
        {
          "Dept_id" : 112,
          "Dept_name" : "COMP"
        }
      ]
    }
  ]
}
```

```

        ],
        "totalsalary" : 220000
    },
    {
        "_id" : [
            {
                "Dept_id" : 111,
                "Dept_name" : "IT"
            }
        ],
        "totalsalary" : 166000
    }
],
"ok" : 1
}
>db.Teacher.aggregate([
{$group:{_id:"$Department",totalsalary:{$sum:"$Salary"}},{$group:{_id:"$_id.Department",AvgSal:{$sum:"$totalsalary"}}}}]
{ "result" : [ { "_id" : [ ], "AvgSal" : 507000 } ], "ok" : 1 }
>db.Teacher.aggregate([
{$group:{_id:"$Department",totalsalary:{$sum:"$Salary"}},{$match:{totalsalary:{$gte:200000}}}}]
{
    "result" : [
        {
            "_id" : [
                {
                    "Dept_id" : 112,
                    "Dept_name" : "COMP"
                }
            ],
            "totalsalary" : 220000
        }
    ],
    "ok" : 1
}
>db.Teacher.aggregate([ {$group:{_id:"$Department",totalsalary:{$sum:"$Salary"}}}, {
$sort:{totalsalary:1}}])
{
    "result" : [
        {
            "_id" : [
                {
                    "Dept_id" : 113,
                    "Dept_name" : "ENTC"
                }
            ]
        }
    ]
}

```

```

        ],
        "totalsalary" : 121000
    },
    {
        "_id" : [
            {
                "Dept_id" : 111,
                "Dept_name" : "IT"
            }
        ],
        "totalsalary" : 166000
    },
    {
        "_id" : [
            {
                "Dept_id" : 112,
                "Dept_name" : "COMP"
            }
        ],
        "totalsalary" : 220000
    }
],
"ok" : 1
}
>db.Teacher.aggregate([
    { $group: { _id: "$Department", totalsalary: { $sum: "$Salary" } } },
    { $group: { _id: "$_id.Department", big: { $last: "$_id.Dept_name" }, bigsalary: { $last: "$totalsalary" }, small: { $first: "$_id.Dept_name" }, smallsalary: { $first: "$totalsalary" } } }
])
{
    "result" : [
        {
            "_id" : [ ],
            "big" : [
                "IT"
            ],
            "bigsalary" : 166000,
            "small" : [
                "ENTC"
            ],
            "smallsalary" : 121000
        }
    ],
    "ok" : 1
}

```

REFERENCE BOOK:

Kristina Chodorow, MongoDB The definitive guide, O'Reilly Publications, ISBN:978-93-5110-269-4,2nd Edition.

CONCLUSION:

Understand to mapreduce operation in mongodb

Assignment 3

Aim : Case Study: Design conceptual model using Star and Snowflake schema for any one database

Problem statement: Design conceptual model using Star and Snowflake schema for any one database

OBJECTIVE:

1. To understand concepts of multidimensional data.
2. To understand the relational implementation of the multidimensional data model is typically a star schema, or a snowflake schema.

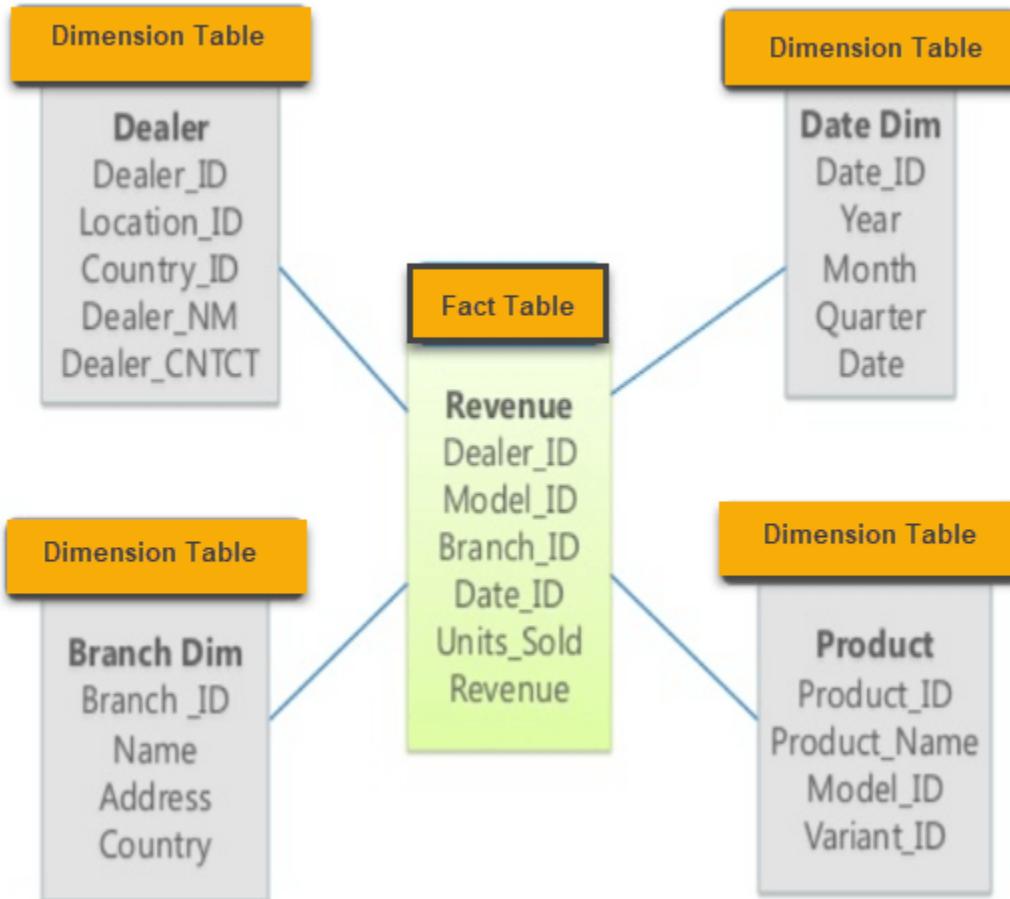
Theory:

The data warehouses are considered modern ancient techniques, since the early days for the relational databases, the idea of keeping a historical data for reference when it needed has been originated, and the idea was primitive to create archives for the historical data to save these data, despite of the usage of a special techniques for the recovery of these data from the different storage modes. This research applied of structured databases for a trading company operating across the continents, has a set of branches each one has its own stores and showrooms, and the company branch's group of sections with specific activities, such as stores management, showrooms management, accounting management, contracts and other departments. It also assumes that the company center exported software to manage databases for all branches to ensure the safety performance, standardization of processors and prevent the possible errors and bottlenecks problems. Also the research provides this methods the best requirements have been used for the applied of the data warehouse (DW), the information that managed by such an applied must be with high accuracy. It must be emphasized to ensure compatibility information and hedge its security, in schemes domain, been applied to a comparison between the two schemes (Star and Snowflake Schemas) with the concepts of multidimensional database. It turns out that Star Schema is better than Snowflake Schema in (Query complexity, Query performance, Foreign Key Joins). And finally it has been concluded that Star Schema center fact and change, while Snowflake Schema center fact and not change.

Example:

1. Star Schema

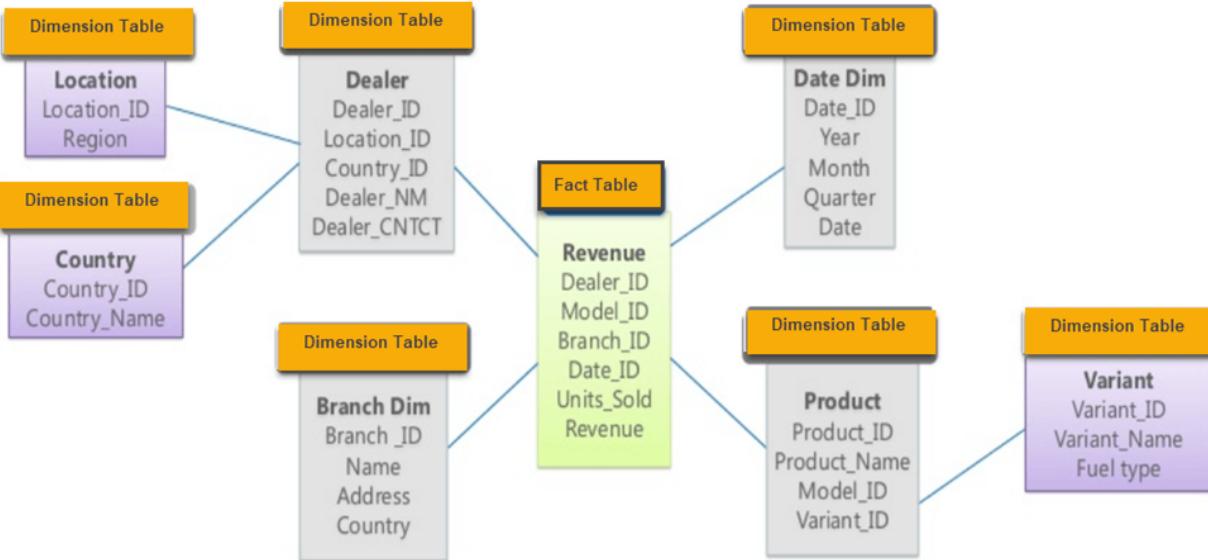
Star Schema in data warehouse, in which the center of the star can have one fact table and a number of associated dimension tables. It is known as star schema as its structure resembles a star. The Star Schema data model is the simplest type of Data Warehouse schema. It is also known as Star Join Schema and is optimized for querying large data sets.



2. Snowflake Schema?

Snowflake Schema in data warehouse is a logical arrangement of tables in a multidimensional database such that the [ER diagram](#) resembles a snowflake shape. A Snowflake Schema is an extension of a Star Schema, and it adds additional dimensions. The dimension tables are normalized which splits data into additional tables.

In the following Snowflake Schema example, Country is further normalized into an individual table.



- Multidimensional schema is especially designed to model data warehouse systems
- The star schema is the simplest type of Data Warehouse schema. It is known as star schema as its structure resembles a star.
- Comparing Snowflake vs Star schema, a Snowflake Schema is an extension of a Star Schema, and it adds additional dimensions. It is called snowflake because its diagram resembles a Snowflake.
- In a star schema, only single join defines the relationship between the fact table and any dimension tables.
- Star schema contains a fact table surrounded by dimension tables.
- Snowflake schema is surrounded by dimension table which are in turn surrounded by dimension table
- A snowflake schema requires many joins to fetch the data.
- Comparing Star vs Snowflake schema, Start schema has simple DB design, while Snowflake schema has very complex DB design.

REFERENCE BOOK:

Jiawei Han, Micheline Kamber, Jian Pei "Data Mining: concepts and techniques", 2nd Edition, Publisher: Elsevier/Morgan Kaufmann.

CONCLUSION:

Understand to concept of multidimensional data.

PART C:
MINI PROJECT

Mini Project

AIM: Build the mini project based on the relevant applicable concepts of Machine Learning / DAA / ADBMS by forming teams of around 3 to 4 students.

A] Sample ML mini-project Format:

PROBLEM STATEMENT /DEFINITION

Design and Implement any Data science Application using Python. Obtain Data, Scrub data (Data cleaning), Explore data, Prepare and validate data model and Interpret data (Data Visualization). Visualize data using any visualization tool like Matplotlib, ggplot, Tableau etc. Prepare Project Report.

OBJECTIVE:

1. To explore the Data science project life cycle.
2. To identify need of project and define problem statement.
3. To extract and process data.
4. To interpret and analyze results using data visualization.

Mini Project Report Format:

Abstract

Acknowledgement

List of Tables & Figures

Contents

1. Introduction

 1.1 Purpose, Problem statement

 1.2 Scope, Objective

 1.3 Definition, Acronym, and Abbreviations

 1.4 References

2. Literature Survey

 2.1 Introduction

 2.2 Detail Literature survey

2.4 Findings of Literature survey

3. System Architecture and Design

3.1 Detail Architecture

3.2 Dataset Description

3.3 Detail Phases

3.4 Algorithms

4. Experimentation and Results

4.1 Phase-wise results

4.2 Explanation with example

4.3 Comparison of result with standard

4.4 Accuracy

4.5 Visualization

4.6 Tools used

5. Conclusion and Future scope

5.1 Conclusion

5.2 Future scope

References

Annexure:

A. GUIs / Screen Snapshot of the System Developed

B. Implementation /code

B] Sample ADBMS mini-project Format:**PROBLEM STATEMENT /DEFINITION**

Build the mini project based on the requirement document and design prepared as a part of Database Management Lab in second year.

Form teams of around 3 to 4 people.

- A. Develop the application: Build a suitable GUI by using forms and placing the controls on it for any application. Proper data entry validations are expected.
- B. Add the database connection with front end. Implement the basic CRUD operations.
- C. Prepare and submit report to include Title of the Project, Abstract, List the hardware and software requirements at the backend and at the front end, Source Code, Graphical User Interface, Conclusion.

OBJECTIVE:

3. To understand applications of document-oriented database by implementing mini project.
4. To learn effective UI designs.
5. To learn to design & implement database system for specific domain.
6. To learn to design system architectural & flow diagram.

Mini Project Report Format:

Abstract

Acknowledgement

List of Tables & Figures

Contents

1. Introduction

 1.1 Purpose

 1.2 Scope

 1.3 Definition, Acronym, and Abbreviations

 1.4 References

 1.5 Developers' Responsibilities: An Overview

2. General Description

 2.1 Product Function Perspective

2.2 User Characteristics.

2.4 General Constraints

2.5 Assumptions and Dependencies

3. Specific Requirements

3.1 Inputs and Outputs

3.2 Functional Requirements

3.3 Functional Interface Requirements

3.3 Performance Constraints

3.4 Design Constraints

3.6 Acceptance criteria

4. System Design

5. System Implementation

5.1 Hardware and Software Platform description

5.2 Tools used

5.3 System Verification and Testing (Test Case Execution)

5.4 Future work / Extension

5.5 Conclusion

References

Annexure:

A. GUIs / Screen Snapshot of the System Developed

