

# Data Mining and Data Warehousing Laboratory (CSPC-328)

B.Tech VIth Semester  
(January–June 2024)

Submitted by

Kirtan Gohil(21103073) Group-G3

Submitted to

Dr.Samayveer Singh



Department of Computer Science & Engineering  
Dr. B.R. Ambedkar National Institute of Technology Jalandhar  
-144008, Punjab, India [Table of Content](#)

Sr.No.	PracticalName	Date	Page No.	Remarks
1.	DesigningDatabaseUsingERModelling a)HospitalManagementSystem b)LibraryManagementSystem	29-01-2024	3-6	
2.	NormalizingaDatabase	5-02-2024	7-19	
3.	ProgramstoimplementProcedures, CursorsandTriggersinadatabase	12-02-2024	20-23	
4.	Writeprogramstoimplementand understandusageofDatamarts.	19-02-2024	24-26	
5.	Feature Selection and Variable Filtering.			
6.	Perform Associative Mining In Weka and Orange on large datasets			
7.	Perform K-Nearest Neighbour Classification in Weka and Orange			
8.				
9.				
10.				

# Practical1

## Aim:-DesigningDatabaseUsingERModelling

### Que1CreatedatabasedesignforHospitalManagementSystemusingER Modelling

The patient, physician, department, room, and appointment are the entities that make up the hospital administration system.

The following is a relationship between these entities areas:

An appointment is for one patient and one doctor. A patient may have one or more appointments. A doctor may schedule many appointments with various patients.

One department is assigned to a doctor.

A department may employ several physicians.

One patient can be assigned to one room, and one or more patients can be housed in a room.

A doctor is in charge of each room, however they can oversee more than one. These relationships allow us to develop the subsequent ER model:

#### 1.Entities:

- Patient with attributes (Name, Age, Room Number, and Patient ID).
  - Physician with the following attributes: DepartmentID, Name, Specialty, DoctorID.
  - Department including features like DepartmentName, DepartmentID.
- Room has the following attributes: bed count, supervising doctor ID, room number.
- Appointment with the following attributes: PatientID, DoctorID, Date, Time, Appointment ID.

#### 2. Relationships:

A patient's relationship with an appointment is symbolized by a "has" relationship.

A doctor-patient connection is based on a "conducts" relationship.

A department and a doctor are associated, represented by a "assignedto" relationship.

Multiple doctors are associated with a department through the "employs" relationship.

A patient and a room are connected through a "assignedto" relationship.

A room can have a relationship with numerous patients, represented by a "houses" relationship.

A room has a relationship with a doctor, which is represented by a "supervisedby" relationship. An diagram representing things as boxes and relationships as lines linking these boxes—often with additional symbols to signify the kind and cardinality of the interactions—would be the visual representation of the ER model.

The relationships and entities within the hospital management system are shown in Fig. 1.1.

The patient, doctor, department, room, and appointment are the five main entities that are included. Patients may schedule many appointments, with a doctor and a single patient at each visit. Physicians are assigned to departments, and each department may have more than one physician on staff. Patients are assigned to rooms, and each room can accommodate several patients under a single doctor's care. The ER graphic also shows how a doctor is able to oversee many rooms. The entities are linked together by a number of links, including "has," "conducts," "assigned to," "employees," "houses," and

"supervisedby," which illustrate the many relationships and interactions that exist in a medical setting. The diagram shows the relationships between the various components of the system and acts as a visual representation of the data model.

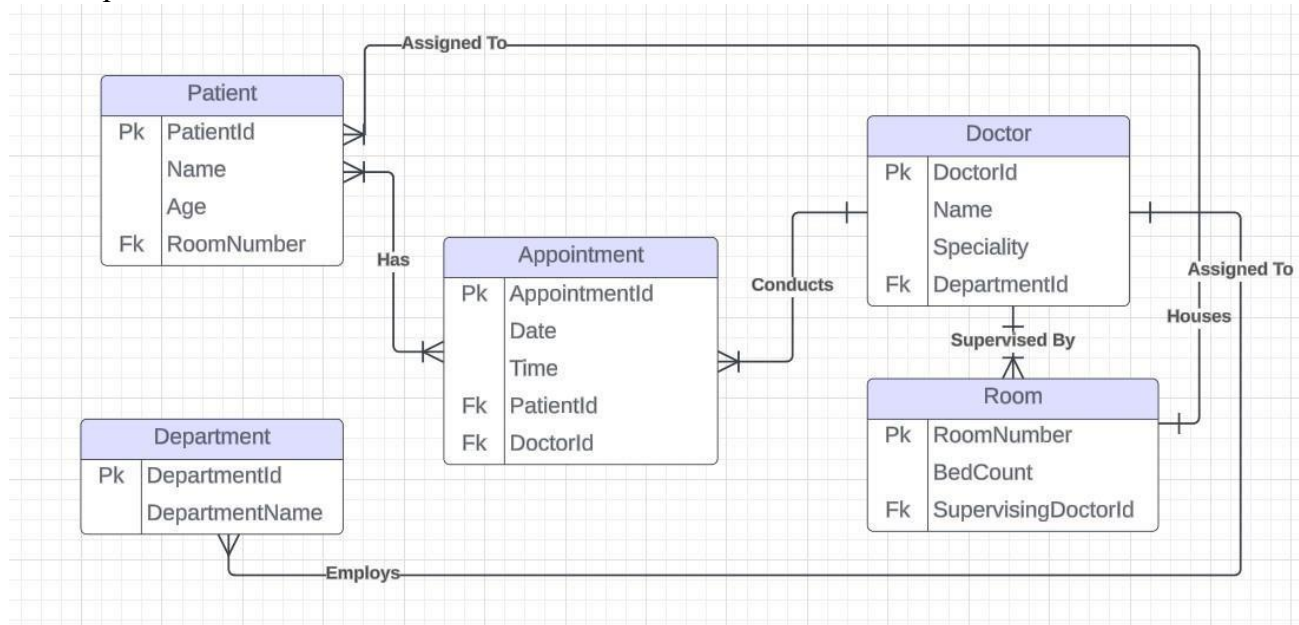


Fig.1.1:ERdiagramforHospitalManagementSystem

## Que2CreatedatabasedesignforLibraryManagementSystemusingER Modelling

The following entities are included in the library management system: book, author, borrower, genre, and loan. The following is a relationship between these entities areas: A book is authored by one or more writers. • A writer can pen one or more books. • A borrower may check out many books, but a book may be checked out by just one borrower. • in real time. • A book falls into a specific genre. • A genre can be connected to more than one book. • The loan specifies when a book was checked out and when it must be returned.

These relationships led us to derive the subsequent ER model:

### 1. Entities

- Book with attributes: Title, ISBN, BookID, GenreID.
- Author with attributes: Name, BirthDate, and AuthorID.
- Borrower with properties: Name, Address, Phone, and Borrower ID.
- Genre with attributes (GenreName, GenreID).
- Loan with attributes: BookID, BorrowerID, Borrow Date, Due Date, Loan ID.

### 2. Relationships:

- A book is linked to its author(s) by means of a "writtenby" relationship.
- One or more books are associated with an author via a "writes" relationship.

- A "borrows" relationship connects a borrower with books.
- A book and borrower have a relationship thanks to the "isborrowedby" connection.
- A book and a genre are connected by a "belongsto" relationship.
- A loan is related to a borrower and a book through a "issued for" relationship. • A genre is connected to many books through a "encompasses" relationship.

To visualize the ER model, entities would be shown as boxes with relationships between them shown as lines or arrows. The types and cardinality of each link would be represented by annotations or symbols.

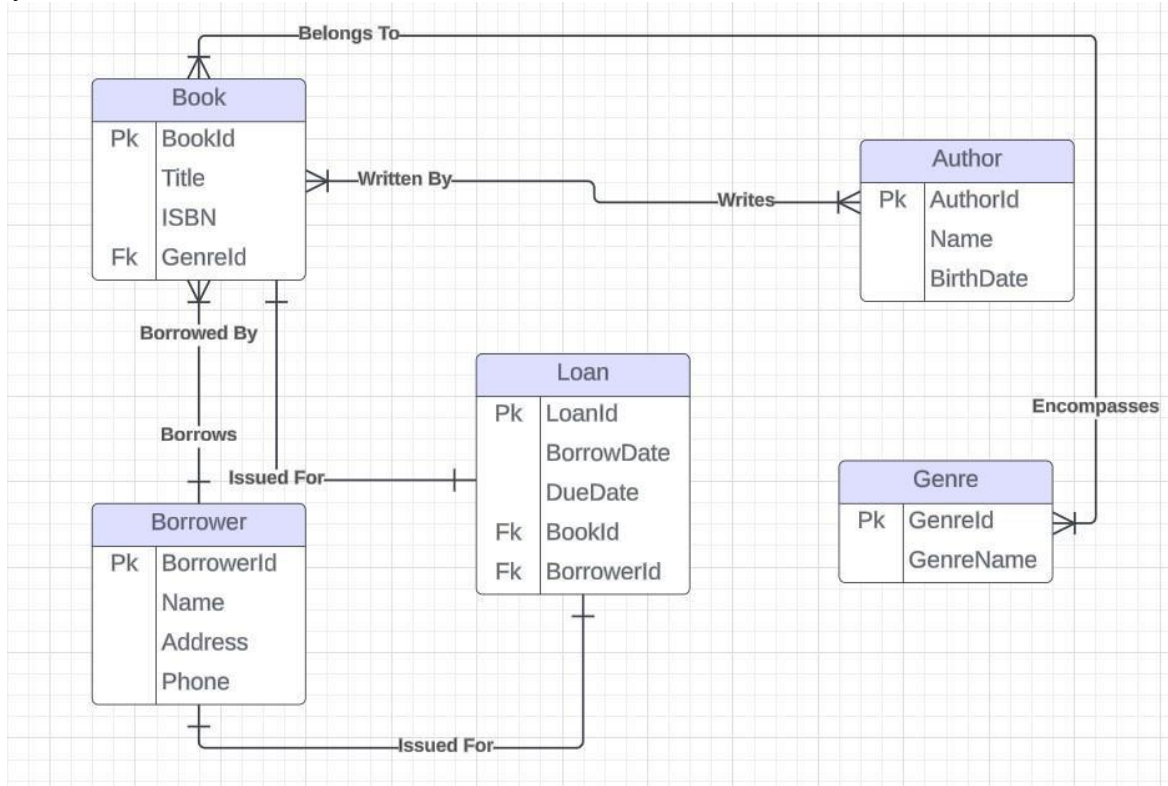


Fig.1.2:ERdiagramforLibraryManagementSystem

Figure 1.2 illustrates the connections and entities in the Library Management System. There are five main components to it: Book, Author, Borrower, Genre, and Loan. The graphic shows how a book is linked to one or more writers by a "written by" relationship, enabling numerous authors to contribute to a single work. Books are linked to authors by a "writes" relationship, meaning that an author is able to write more than one book. The relationship "borrows" links borrowers to books; this means that one borrower may check out numerous books at once, but only one borrower may check out a book at a time. Books are grouped by genres using a "belongsto" relationship, which indicates that a given book is part of a particular genre. Genres might include more than one book. The "issuedfor" relationships bind loans to both borrowers and books, indicating the date a book was borrowed and the return deadline.

## Practical2

### Aim:-NormalisingaDatabaseUsingGriffithNormalisation Tool

Que1Understandthefunctionaldependenciesandnormalizeeach functional dependencyupto2NF,3NF,andBCNFusingnormalizationtoolfrom GriffithUniversity.

Foreachquestion:

- Findtheminimalcover.
- Identifythecandidatekey(s)orprimarykey.
- Checkforpartialdependencies todetermineiftherelationisin2NF.
- Checkfortransitivedependencies toassessiftherelationisin3NF.
- Checkfortransitivedependencies toassessiftherelationisinBCNF.

A.StudentDatabase:

Given therelation:

StudentCourses(StudentID,CourseName,Instructor,CourseCredits)

andthefunctionaldependencies: StudentID,CourseName→Instructor

CourseName→CourseCredits

PreviousFunctionalDependencies

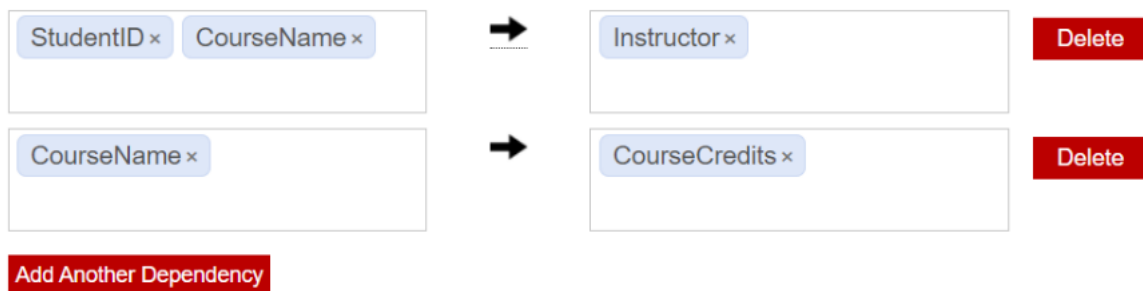


Fig1.A.1 ModifiedFunctionalDependencies

#### Attributes in Table

! Separate attributes using a comma ( , )

StudentID, CourseName, Instructor, CourseCredits

#### Functional Dependencies

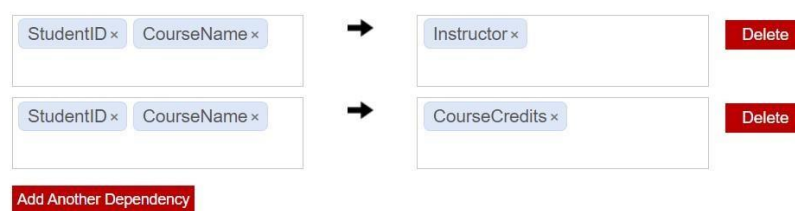


Fig1.A.2

Result

## Check Normal Form



### 2NF

The table is in 2NF



### 3NF

The table is in 3NF



### BCNF

The table is in BCNF

Show Steps



Fig1.A.3

Fig1.A.1 shows previous Functional Dependencies which are not in BCNF. Fig1.A.2 shows new Functional Dependencies which show if you know a student's ID and the name of the course they're retaking, you can determine the instructor who teaches that course and how many credits that course carries. Fig1.A.3 shows the result that new FDs are in BCNF.

## B. Employee Management:

Given the relation:

EmployeeProjects(EmployeeID, ProjectName, Manager, Department)

with the functional dependencies:

EmployeeID  $\rightarrow$  Department

ProjectName  $\rightarrow$  Manager Department  $\rightarrow$  Manager

## Previous Functional Dependencies



Fig1.B.1 Modified Dependencies

## Attributes in Table

! Separate attributes using a comma ( , )

EmployeeID, ProjectName, Manager, Department

## Functional Dependencies

EmployeeID ×	→	Department × ProjectName ×	Delete
Department ×	→	Manager × EmployeeID ×	Delete
<a href="#">Add Another Dependency</a>			

Fig1.B.2 Result

### Check Normal Form



#### 2NF

The table is in 2NF



#### 3NF

The table is in 3NF



#### BCNF

The table is in BCNF

Show Steps



Fig1.B.3

Fig1.B.1 shows previous Functional Dependencies which are not in BCNF. Fig1.B.2 shows new Functional Dependencies which shows Given an EmployeeID, we can determine the ProjectName and Department associated with that employee. Given a Department, we can determine the Manager and EmployeeID associated with that department. Fig1.B.3 shows the result that new FDs are in BCNF.

C.LibrarySystem:

Consider the relation:

BookLending(BookID, MemberID, BorrowDate, DueDate, MemberAddress)

and the functional dependencies: BookID → DueDate

MemberID → MemberAddress

Previous Functional Dependencies





Fig1.C.1

Modified Dependencies

## Attributes in Table

! Separate attributes using a comma ( , )

BookID, MemberID, BorrowDate, DueDate, MemberAddress

## Functional Dependencies



Fig1.C.2 Result

## Check Normal Form



### 2NF

The table is in 2NF



### 3NF

The table is in 3NF



### BCNF

The table is in BCNF

Show Steps



Fig1.C.3

Fig1.C.1 shows previous Functional Dependencies which are not in BCNF. Fig1.C.2 shows new Functional Dependencies which show that if you know which book is borrowed by which member, you can determine the member's address, the due date of the book, and the date it was borrowed. Fig1.C.3 shows the result that the new FDs are in BCNF.

D. Hospital Management:

- For the relation:

PatientTreatment(PatientID,Treatment,Doctor,DoctorSpecialization)  
with the functional dependencies: Doctor→DoctorSpecialization PatientID,Treatment→Doctor

Previous Functional Dependencies



Fig1.D.1 Modified Dependencies

## Attributes in Table

! Separate attributes using a comma ( , )

PatientID, Treatment, Doctor, DoctorSpecialization

## Functional Dependencies



Fig1.D.2 Result

## Check Normal Form



**2NF**  
The table is in 2NF



**3NF**  
The table is in 3NF



**BCNF**  
The table is in BCNF

Show Steps



Fig1.D.3

Fig1.D.1 shows previous Functional Dependencies which are not in BCNF. Fig1.D.2 shows new Functional Dependencies which show that if you know the PatientID and the treatment they are undergoing, you can determine which doctor is responsible for providing that treatment, along with the doctor's specialization. Fig1.D.3 shows the result that new FDs are in BCNF.

E. Airline Reservation System:

-Given the relation:

FlightReservations(FlightNumber,Date,PassengerID,SeatNumber,ClassType,Price,DepartureTime,ArrivalTime,DepartureCity,ArrivalCity) -Functional dependencies are:

FlightNumber,Date→DepartureTime,ArrivalTime,DepartureCity,ArrivalCity

SeatNumber,Date,FlightNumber→PassengerID,ClassType,Price

ClassType→Price

PassengerID→DepartureCity

Previous Functional Dependencies

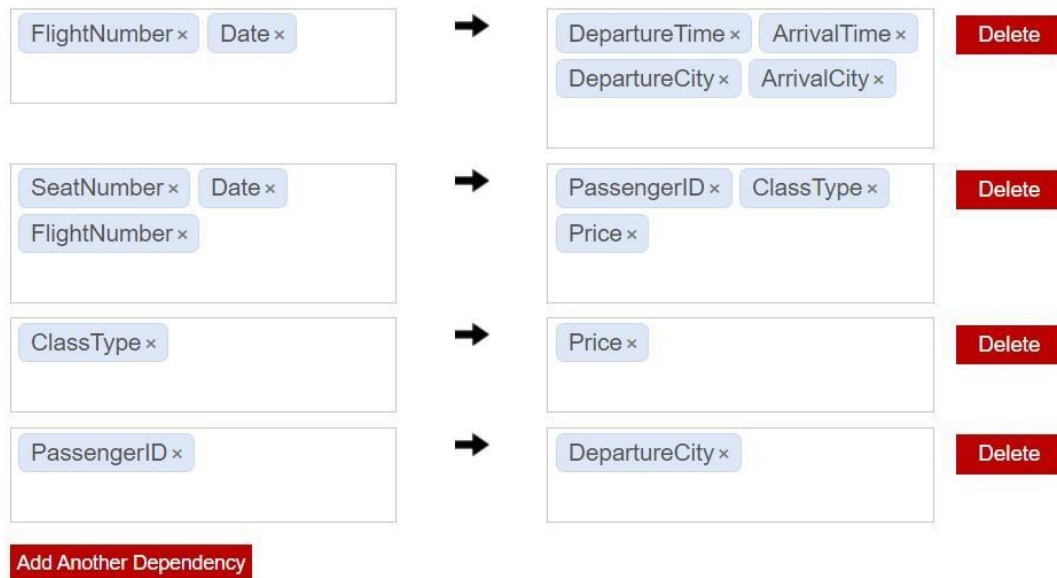


Fig1.E.1 Modified Dependencies

## Attributes in Table

! Separate attributes using a comma ( , )

FlightNumber, Date, PassengerID, SeatNumber, ClassType, Price, DepartureTime, ArrivalTime, DepartureCity, ArrivalCity

## Functional Dependencies



Fig1.E.2 Result

Check Normal Form

✓

**2NF**  
 The table is in 2NF

✓

**3NF**  
 The table is in 3NF

✓

**BCNF**  
 The table is in BCNF

Show Steps ☐

Fig1.E.3

Fig1.E.1 shows previous Functional Dependencies which are not in BCNF. Fig1.E.2 shows new Functional Dependencies which show that if you have information about the flight number, date, and seat number, you can determine the details related to that specific booking, including the departure and arrival times, cities, passenger ID, class type, and price associated with that booking. Fig1.E.3 shows the result that new FDs are in BCNF.

#### F.6. University Enrolment System:

- Given the relation:

Enrollments(StudentID, CourseCode, Semester, Grade, InstructorID, CourseName, CourseCredits, Department)

- Functional dependencies are:

StudentID, CourseCode, Semester → Grade, InstructorID

CourseCode → CourseName, CourseCredits, Department

InstructorID, CourseCode → Department

InstructorID → Department

Previous Functional Dependencies

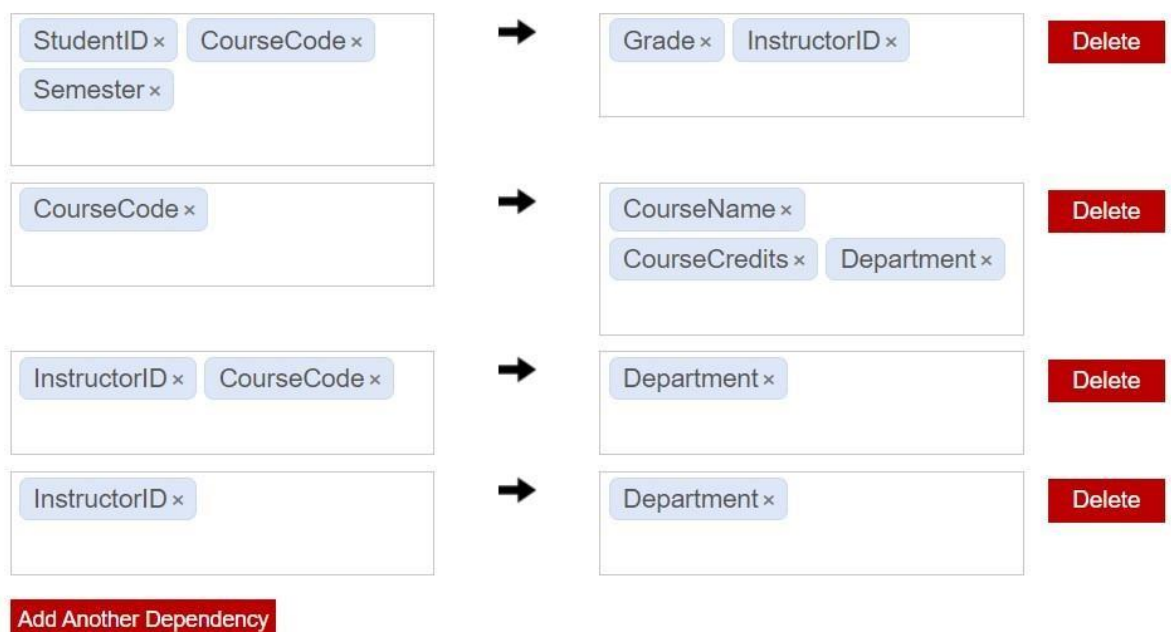


Fig1.F.1

Modified Dependencies

## Attributes in Table

! Separate attributes using a comma ( , )

StudentID, CourseCode, Semester, Grade, InstructorID, CourseName,  
CourseCredits, Department

## Functional Dependencies

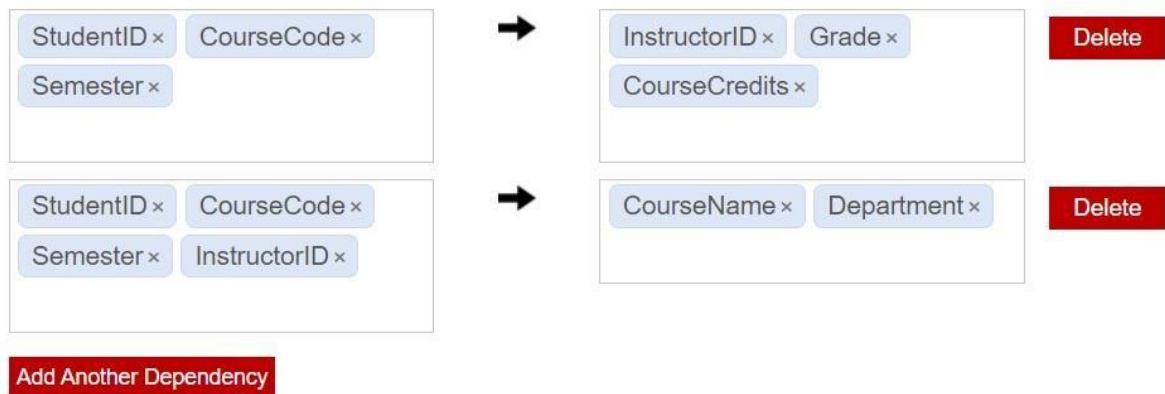


Fig1.F.2 Result

Check Normal Form



**2NF**

The table is in 2NF



**3NF**

The table is in 3NF



**BCNF**

The table is in BCNF

Show Steps



Fig1.F.3

Fig1.F.1 shows previous Functional Dependencies which are not in BCNF. Fig1.F.2 shows new Functional Dependencies which show that for a given student, a specific course in a particular semester uniquely determines the grade received by the student, the instructor teaching the course, and the number of credits associated with the course. It means that for a given student taking a specific course in a particular semester with a particular instructor, there is only one department to which the course belongs and one specific name for the course. Fig1.F.3 shows the result that new FDs are in BCNF.

G.MusicStreamingPlatform:

-For the relation:

UserPlays(UserID, SongID, Date, ArtistName, Album, Genre, PlayCount, SubscriptionType)

-Functional dependencies are:

UserID, SongID, Date  $\rightarrow$  PlayCount

SongID  $\rightarrow$  ArtistName, Album, Genre

UserID  $\rightarrow$  SubscriptionType

ArtistName, Album  $\rightarrow$  Genre

## PreviousFunctionalDependencies



Fig1.G.1 ModifiedDependencies

## Attributes in Table

ⓘ Separate attributes using a comma ( , )

UserID, SongID, Date, ArtistName, Album, Genre, PlayCount, SubscriptionType

## Functional Dependencies



Fig1.G.2 Result

Check Normal Form

- ☒ **2NF**  
The table is in 2NF
- ☒ **3NF**  
The table is in 3NF
- ☒ **BCNF**  
The table is in BCNF

Show Steps ☐

Fig1.G.3

Fig1.G.1 shows previous Functional Dependencies which are not in BCNF. Fig1.G.2 shows new Functional Dependencies which show that for a given user, listening to a specific song on a particular date uniquely determines various attributes related to

that listening event, such as how many times the song was played (PlayCount), the type of subscription the user has (SubscriptionType), the name of the artist, the album, and the genre of the song. Fig1.G.3 shows the result that new FDs are in BCNF.

H. Real Estate System:

- For the relation:

PropertyListings(PropertyID, OwnerID, AgentID, Price, Location, HouseType, NumberOfRooms, AgentName, CommissionRate) - Functional dependencies are:

PropertyID → Price, Location, HouseType, NumberOfRooms, OwnerID, AgentID

AgentID → AgentName, CommissionRate HouseType → NumberOfRooms

Previous Functional Dependencies



Fig1.H.1 Modified Dependencies



## Attributes in Table

① Separate attributes using a comma ( , )

PropertyID, OwnerID, AgentID, Price, Location, HouseType,  
NumberOfRooms, AgentName, CommissionRate

## Functional Dependencies



Fig1.H.2 Result

Check Normal Form



**2NF**

The table is in 2NF



**3NF**

The table is in 3NF



**BCNF**

The table is in BCNF

Show Steps



Fig1.H.3

Fig1.H.1 shows previous Functional Dependencies which are not in BCNF. Fig1.H.2 shows new Functional Dependencies which show that each property in the table is uniquely identified by its PropertyID, and for each PropertyID, there is a fixed price, location, house type, ownerID, and agentID associated with it.

It means that each agent assigned to a specific property is uniquely identified by their AgentID, and for each combination of AgentID and PropertyID, there is a fixed name for the agent and a fixed commission rate associated with that agent's involvement in that property transaction.

It means that the number of rooms in a property is uniquely determined by the combination of its PropertyID and HouseType. Fig1.H.3 shows the result that new FDs are in BCNF.

Que2 Design a BCNF Normalized Database and verify using Griffith Tool.

Ans Database is Flight Reservation System.



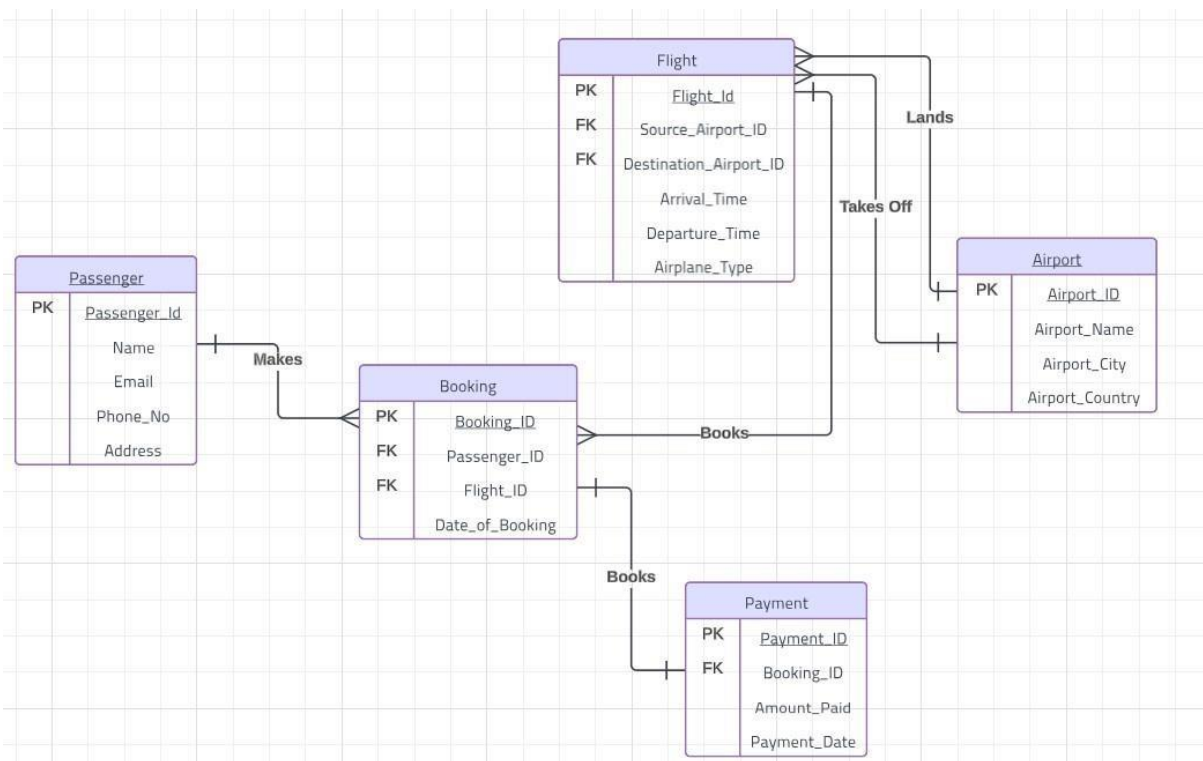


Fig2.1

Fig2.1showsthedesignofairlinerreservationsystemdatabase.

FunctionalDependenciesare:

FlightsTable:

- Flight\_ID->Source\_Airport\_ID
- Flight\_ID->Destination\_Airport\_ID
- Flight\_ID->Departure\_Time
- Flight\_ID->Arrival\_Time
- Flight\_ID->Airplane\_Type

AirportsTable: Airport\_Code->Airport\_Name  
 Airport\_Code->Airport\_City

Airport\_Code->Airport\_Country PassengerTable:

- Customer\_ID->Name
- Customer\_ID->Email
- Customer\_ID->Phone\_No
- Customer\_ID->Address

BookingsTable:

Booking\_ID->Flight\_ID

Booking\_ID->Passenger\_ID

Booking\_ID->Date\_of\_Booking PaymentsTable:

- Payment\_ID->Booking\_ID
- Payment\_ID->Amount\_Paid

- Payment\_ID->Payment\_Date

VerificationUsingGriffithTool

Check Normal Form

---



**2NF**

The table is in 2NF



**3NF**

The table is in 3NF



**BCNF**

The table is in BCNF

Show Steps



Fig2.2

Result

Fig2.2showsthatEachTableisinBCNF.

## Practical-3

Aim:-CreateProcedures,TriggersandCursors

Que1WriteastoredprocedurenamedUpdateCountryPopulationthat updatesthepopulationofagivencountrybasedonaprovidedcountry codeandnewpopulationvalue.Additionally,theprocedureshouldlog theoldandnewpopulationvaluestoapopulation\_change\_logtable. Ans

DELIMITER//

```
CREATEPROCEDUREUpdateCountryPopulation(INCountryCodeCHAR(3),IN  
NewPopulationINT)
```

```
BEGIN
```

```
    DECLAREOldPopulationINT;
```

```
    --Gettheoldpopulation
```

```
    SELECTPopulationINTOOldPopulation
```

```
    FROMcountry
```

```
    WHERECode=CountryCode;
```

```
    --Updatethepopulation
```

```
    UPDATEcountry
```

```
    SETPopulation=NewPopulation WHERECode=CountryCode;
```

```
    --Logthepopulationchange
```

```
    INSERTINTOpopulation_change_log(CountryCode,OldPopulation,  
NewPopulation,ChangeDate)
```

```
    VALUES(CountryCode,OldPopulation,NewPopulation,NOW());--NOW()isused  
fortheurrenttimestampinMySQL
```

```
END//
```

```
DELIMITER;
```

```
CALLUpdateCountryPopulation('USA',350000000);
```

	LogID	CountryCode	OldPopulation	NewPopulation	ChangeDate
▶	1	USA	NULL	2000000	NULL
	2	USA	2000000	350000000	2024-02-18 15:21:44
*	NULL	NULL	NULL	NULL	NULL

Fig3.1

Fig3.1showspopulation\_change\_logtablewhichhasoldpopulation,new populationanddateofchange.

Que2Developatriggernamedafter\_country\_insertthatchecksifthe insertedcountry'spopulationexceeds1million.Ifitdoes,inserta recordintoahigh\_population\_countriestable.

Ans

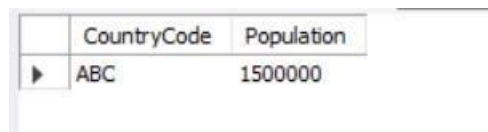
```
CREATETRIGGERafter_country_insert
AFTERINSERTONcountry
FOREACHROW
BEGIN
    DECLARECountryPopulationINT;

    --Getthepopulationoftheinsertedcountry
    SELECTPopulationINTOCountryPopulation
    FROMcountry
    WHERECode=NEW.Code;

    --Checkifpopulationexceeds1million
    IFCountryPopulation>1000000THEN
        --Insertinto high_population_countries table
        INSERTINTO high_population_countries(CountryCode,Population)
        VALUES(NEW.Code,CountryPopulation);
    ENDIF; END//
DELIMITER;

INSERTINTOcountry(Code,Population)VALUES('ABC',1500000);

select*fromhigh_population_countries;
```



	CountryCode	Population
▶	ABC	1500000

Fig3.2

Fig3.2 showshigh\_population\_countriestablewithcountrycodeandpopulation. Que3 DevelopaprocedureAdjustCityPopulationsusingacursorthat decreasesthepopulationby10%forallcitiesinagivencountrycode, providedthecurrentpopulationisbetween500,000and1million. Additionally,logthesechangestoacity\_population\_adjustmentstable withcityID,oldpopulation,andnewpopulation.

Ans

```
DELIMITER//
CREATEPROCEDUREAdjustCityPopulations(INCountryCodeCHAR(3))
BEGIN
    DECLAREdoneINTDEFAULTFALSE;
    DECLARECityIDINT;
    DECLAREOldPopulationINT;
    DECLARENewPopulationINT;
    --Declarecursor
```

```

DECLAREcity_cursorCURSORFOR
SELECTCityID,Population
FROMcity
WHERECountryCode=CountryCode
ANDPopulationBETWEEN500000AND1000000;

--Declarehandlerfornomorerows
DECLARECONTINUEHANDLERFORNOTFOUNDSETdone=TRUE;

--Openthecursor
OPENcity_cursor;

--Startloopingthroughthecursor
adjust_loop:LOOP --Fetchtherow
    FETCHcity_cursorINTOCityID,OldPopulation;

    --Checkifnomorerows IFdoneTHEN
        LEAVEadjust_loop;
    ENDIF;

    --Calclatenewpopulation(decreaseby10%)
    SETNewPopulation=ROUND(OldPopulation*0.9,0);

    --Updatecitypopulation
    UPDATEcity
    SETPopulation=NewPopulation WHERECityID=CityID;

    --Logpopulationadjustment
    INSERTINTOcity_population_adjustment(CityID,OldPopulation,
NewPopulation,AdjustmentDate)
    VALUES(CityID,OldPopulation,NewPopulation,NOW()); ENDLOOPadjust_loop;

--Closethecursor
CLOSEcity_cursor;
END//
DELIMITER;
CALLAdjustCityPopulations('USA'); select*fromcity_population_adjustment;

```

	CityID	OldPopulation	NewPopulation	AdjustmentDate
►	NULL	731200	658080	2024-02-18 16:17:55
	NULL	593321	533989	2024-02-18 16:17:55
	NULL	609823	548841	2024-02-18 16:17:55
	NULL	669181	602263	2024-02-18 16:17:55
	NULL	907718	816946	2024-02-18 16:17:55
	NULL	622013	559812	2024-02-18 16:17:55
	NULL	559249	503324	2024-02-18 16:17:55
	NULL	538918	485026	2024-02-18 16:17:55
	NULL	521936	469742	2024-02-18 16:17:55
	NULL	512880	461592	2024-02-18 16:17:55
	NULL	978100	880290	2024-02-18 16:17:55
	NULL	663340	597006	2024-02-18 16:17:55
	NULL	536827	483144	2024-02-18 16:17:55
	NULL	935361	841825	2024-02-18 16:17:55
	NULL	758141	682327	2024-02-18 16:17:55

Fig3.3

Fig3.3 shows city\_population\_adjustment table which records the population statistics and date of change.

## Practical-4

Aim:-Writeprogramstoimplementandunderstandusageof Datamarts.

Question1:Designadatamartforabanktostorethecredithistoryof customersinabank.Usethiscreditprofilingtoprocessfutureloan applications.(Suggestivetables:CustomerProfile,accounts,loans, creditcards,paymenthistorytable,inquiries,Collections,CreditScore History). Ans createdatabasebank;

```
createtablecustomer_profile(customer_idintprimarykey,first_name
varchar(25),last_namevarchar(25),d_o_bdate,addressvarchar(50),phone_no
int,emailvarchar(25),incomeint);
```

```
createtableaccounts(account_idintprimarykey,customer_idint,accounttype
varchar(25),dateofopendate,accountstatusvarchar(25),foreignkey(customer_id)
referencescustomer_profile(customer_id),balanceint);
```

```
createtableloans(loan_idintprimarykey,customer_idint,loantype
varchar(25),loanamountint,termint,interest_ratedecimal(4,2),loanstatus
varchar(25),foreignkey(customer_id)referencescustomer_profile(customer_id));
```

```
createtablecreditcards(card_idintprimarykey,customer_idint,cardtype
varchar(25),creditlimitdecimal(10,2),cardissuedatedate,foreignkey(customer_id)
referencescustomer_profile(customer_id),currentbalancedecimal(10,2));
```

```
createtablepaymenthistory(payment_idintprimarykey,customer_idint,account_id
int,paymentamountdecimal(10,2),paymentdatedate,foreignkey(customer_id)
referencescustomer_profile(customer_id),foreignkey(account_id)references
accounts(account_id));
```

```
createtableinquiries(inquiry_idintprimarykey,customer_idint,inquirydate
date,inquirytypevarchar(25),foreignkey(customer_id)references
customer_profile(customer_id));
```

```
createtablecollections(collection_idintprimarykey,customer_idint,collectiondate
date,collectiontypevarchar(25),amountint,foreignkey(customer_id)references
customer_profile(customer_id));
```

```
createtablecredit_score_history(creditscore_idintprimarykey,customer_id
int,creditscoreint,scoredatedate,foreignkey(customer_id)references
customer_profile(customer_id)); --DATAMART:
```

```

createtablecustomerrisk(customer_idintprimarykey,riskcategoryvar
char(25));
insertintocustomerrisk(customer_id,riskcategory)selectc.customer_id,case
whenc.income>75000andsum(a.balance)>100000then'lowrisk'
whenc.income>50000andsum(a.balance)>60000then'moderaterisk' else'highrisk'
endasriskcategory
fromcustomer_profilecjoinaccountsaonc.customer_id=a.customer_idgroupby c.customer_id;

```

customer_id	riskcategory
1	low risk
2	high risk
3	moderate risk
4	moderate risk
5	high risk
NULL	NULL

Fig4.1

InFig4.1,itshowsthatitdividesthecustomersintodifferentriskcategorybaseon incomeandbalanceofcustomers.

```

createtableloanassessmentasselectc.customer_idas
customer_id,c.collectionstatusascollectionstatus,l.loanstatusasloanstatusfrom
collectionscjoinloanslonc.customer_id=l.customer_idwherecollectionstatus='ontime'andloanst
atus='paid_off';

```

customer_id	collectionstatus	loanstatus
1	on-time	paid_off
4	on-time	paid_off

Fig4.2

InFig4.2itshowstheresultofcustomerswhoseloanstatusispaidoffand collectionstatusisontime.

```

createtableloanpassasselectl.customer_idfromloanassessmentljoin
customerriskconl.customer_id=c.customer_idjoincredit_score_historychon
ch.customer_id=c.customer_idwhererec.riskcategory='lowrisk'and ch.creditscore>750;

```

customer_id
1

Fig4.3

InFig4.3itshowsthecustomerswhichhaslowriskcategoryhasloanstatusas paidoffandontimeandcreditscoregreaterthan750.

```

CREATEPROCEDURELOAN_PASS_RESULT(INCUSTOMERIDINT) BEGIN
DECLAREMESSAGE_TEXTVARCHAR(50);
IFEXISTS(
    SELECT1FROMloanpass

```



```

WHERE customer_id=CUSTOMERID
)THEN
    SELECT CUSTOMERID,'PASSED' AS LOAN_ELIGIBILITY;
ELSE
    SELECT CUSTOMERID,'REJECTED' AS LOAN_ELIGIBILITY;
ENDIF;
END//
DELIMITER; call LOAN_PASS_RESULT(1);

```

Output1

	CUSTOMERID	LOAN_ELIGIBILITY
▶	1	PASSED

Fig4.4

call LOAN\_PASS\_RESULT(2); Output2

	CUSTOMERID	LOAN_ELIGIBILITY
▶	2	REJECTED

Fig4.5

RESULT: Successfully implemented and learnt the usage of Datamarts.

## PRACTICAL#5

**Objective:** Feature Selection and Variable Filtering.

**Question#:**

- A) Select a dataset that has a minimum of 150 features.
- B) Apply 3 Feature Selection Techniques
- C) For each feature selection technique apply 3 machine learning models on it.
- D) Compare the results.

**TOOL USED: Weka**

**Feature Selection Technique->Gain Ratio->**The gain ratio is a metric in decision trees that balances the information gain with the intrinsic information of attributes, helping to select the best attribute for splitting nodes.

**No. of selected attribute->** 20

**Algorithm: Naive Bayes->**Probabilistic classification algorithm based on Bayes' theorem with an assumption of independence between features

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      150           86.2069 %
Incorrectly Classified Instances    24           13.7931 %
Kappa statistic                    0.7249
Mean absolute error                 0.1365
Root mean squared error             0.3592
Relative absolute error             27.3077 %
Root relative squared error         71.8425 %
Total Number of Instances          174

=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.798	0.071	0.922	0.798	0.855	0.932	P
	0.929	0.202	0.814	0.929	0.868	0.936	H
Weighted Avg.	0.862	0.135	0.869	0.862	0.862	0.934	

```
=== Confusion Matrix ===

 a  b  <-- classified as
71 18 | a = P
 6 79 | b = H
```

**Fig 5.1 Naive Bayes with 20 attributes**

**Algorithm:Random tree->**It works by building multiple decision trees during training, where each tree is trained on a random subset of the training data and a random subset of the features.The random trees vote on the final classification or regression output, and the most popular outcome is chosen.Random Trees help reduce overfitting and improve accuracy, especially when dealing with noisy or high-dimensional data

```

Correctly Classified Instances      135          77.5862 %
Incorrectly Classified Instances    39          22.4138 %
Kappa statistic                    0.5518
Mean absolute error                 0.2241
Root mean squared error             0.4734
Relative absolute error             44.8438 %
Root relative squared error         94.6953 %
Total Number of Instances          174

=== Detailed Accuracy By Class ===

                TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area   Class
                0.764     0.212     0.791      0.764     0.777       0.776      P
                0.788     0.236     0.761      0.788     0.775       0.776      H
Weighted Avg.   0.776     0.224     0.776      0.776     0.776       0.776

=== Confusion Matrix ===

  a  b  <-- classified as
68 21 |  a = P
18 67 |  b = H

```

**Fig 5.2 Random Tree with 20 attributes**

**Algorithm: AdaBoost**→It works by combining multiple weak learners (typically decision trees) to create a strong learner. It begins by assigning equal weights to all training samples. Then, it iteratively trains weak learners, focusing more on incorrectly classified samples in each iteration. The predictions of weak learners are combined through weighted voting, where more accurate learners have higher weights. This process continues until a predetermined number of iterations is reached or until perfect predictions are achieved.

```

Correctly Classified Instances      147          84.4828 %
Incorrectly Classified Instances    27          15.5172 %
Kappa statistic                    0.6904
Mean absolute error                0.1812
Root mean squared error            0.3165
Relative absolute error            36.2572 %
Root relative squared error        63.3048 %
Total Number of Instances         174

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
      0.787    0.094    0.897    0.787    0.838     0.941    P
      0.906    0.213    0.802    0.906    0.851     0.941    H
Weighted Avg.   0.845    0.152    0.851    0.845    0.844     0.941

=== Confusion Matrix ===

  a  b  <-- classified as
70 19 |  a = P
 8 77 |  b = H

```

**Fig 5.3 AdaBoost with 20 attributes**

**Feature Selection Technique->Gain Ratio**

**No. of selected attribute-> 40**

**Algorithm: Naive Bayes**

```

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
      0.865    0.106    0.895    0.865    0.88     0.955    P
      0.894    0.135    0.864    0.894    0.879    0.959    H
Weighted Avg.   0.879    0.12    0.88     0.879    0.879    0.957

=== Confusion Matrix ===

  a  b  <-- classified as
77 12 |  a = P
 9 76 |  b = H

```

**Fig 5.4 Naive Bayes with 40 attributes**

**Algorithm: Random Tree**

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      141           81.0345 %
Incorrectly Classified Instances    33           18.9655 %
Kappa statistic                    0.6208
Mean absolute error                 0.1897
Root mean squared error             0.4355
Relative absolute error             37.9447 %
Root relative squared error         87.107 %
Total Number of Instances          174

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                0.798    0.176    0.826    0.798    0.811     0.811    P
                0.824    0.202    0.795    0.824    0.809     0.811    H
Weighted Avg.   0.81     0.189    0.811    0.81     0.81      0.811

=== Confusion Matrix ===

  a  b  <-- classified as
71 18 |  a = P
15 70 |  b = H

```

**Fig 5.5 Random Tree with 40 attributes**

**Algorithm:AdaBoost**

**Fig 5.6**

```

Correctly Classified Instances      151           86.7816 %
Incorrectly Classified Instances    23           13.2184 %
Kappa statistic                    0.7356
Mean absolute error                 0.1766
Root mean squared error             0.3322
Relative absolute error             35.3247 %
Root relative squared error         66.441 %
Total Number of Instances          174

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                0.865    0.129    0.875    0.865    0.87      0.929    P
                0.871    0.135    0.86     0.871    0.865     0.929    H
Weighted Avg.   0.868    0.132    0.868    0.868    0.868     0.929

=== Confusion Matrix ===

  a  b  <-- classified as
77 12 |  a = P
11 74 |  b = H

```

**Fig 5.6 AdaBoost with 40 attributes**

**Feature Selection Technique->Gain Ratio**

**No. of selected attribute-> 50**

**Algorithm:** Naive Bayes

```
=== Detailed Accuracy By Class ===  
  
          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class  
          0.876    0.118    0.886    0.876    0.881    0.952    P  
          0.882    0.124    0.872    0.882    0.877    0.96     H  
Weighted Avg.  0.879    0.121    0.879    0.879    0.879    0.956  
  
=== Confusion Matrix ===  
  
  a  b  <-- classified as  
78 11 |  a = P  
10 75 |  b = H
```

**Fig 5.7 Naive Bayes with 50 attributes**

**Algorithm:** Random Tree

```
Correctly Classified Instances      140          80.4598 %  
Incorrectly Classified Instances    34          19.5402 %  
Kappa statistic                    0.6086  
Mean absolute error                 0.1954  
Root mean squared error             0.442  
Relative absolute error             39.0946 %  
Root relative squared error         88.4169 %  
Total Number of Instances          174  
  
=== Detailed Accuracy By Class ===  
  
          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class  
          0.831    0.224    0.796    0.831    0.813    0.804    P  
          0.776    0.169    0.815    0.776    0.795    0.804    H  
Weighted Avg.  0.805    0.197    0.805    0.805    0.804    0.804  
  
=== Confusion Matrix ===  
  
  a  b  <-- classified as  
74 15 |  a = P  
19 66 |  b = H
```

**Fig 5.8 Random Tree with 50 attributes**

**Algorithm:** AdaBoost

```

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                0.809    0.129    0.867    0.809    0.837    0.93     P
                0.871    0.191    0.813    0.871    0.841    0.93     H
Weighted Avg.    0.839    0.16    0.841    0.839    0.839    0.93

=== Confusion Matrix ===
  a  b  <-- classified as
72 17 |  a = P
11 74 |  b = H

```

**Fig 5.9 AdaBoost with 50 attributes**

#### **TOOL USED:- ORANGE**

Orange is an open-source data visualization, analysis, and machine learning toolkit. It provides a user-friendly interface for data preprocessing, exploration, visualization, and predictive modeling. Orange offers a wide range of machine learning algorithms for classification, regression, clustering, and other tasks. Users can easily compare and evaluate different algorithms using built-in evaluation widgets.

**KNN**->K-Nearest Neighbors (KNN) is a simple yet effective supervised machine learning algorithm used for both classification and regression tasks. It's based on the idea that similar data points tend to belong to the same class or have similar values. When making predictions for a new data point, KNN calculates the distance between that point and all other points in the training dataset. Common distance metrics include

Euclidean distance, Manhattan distance, or cosine similarity.

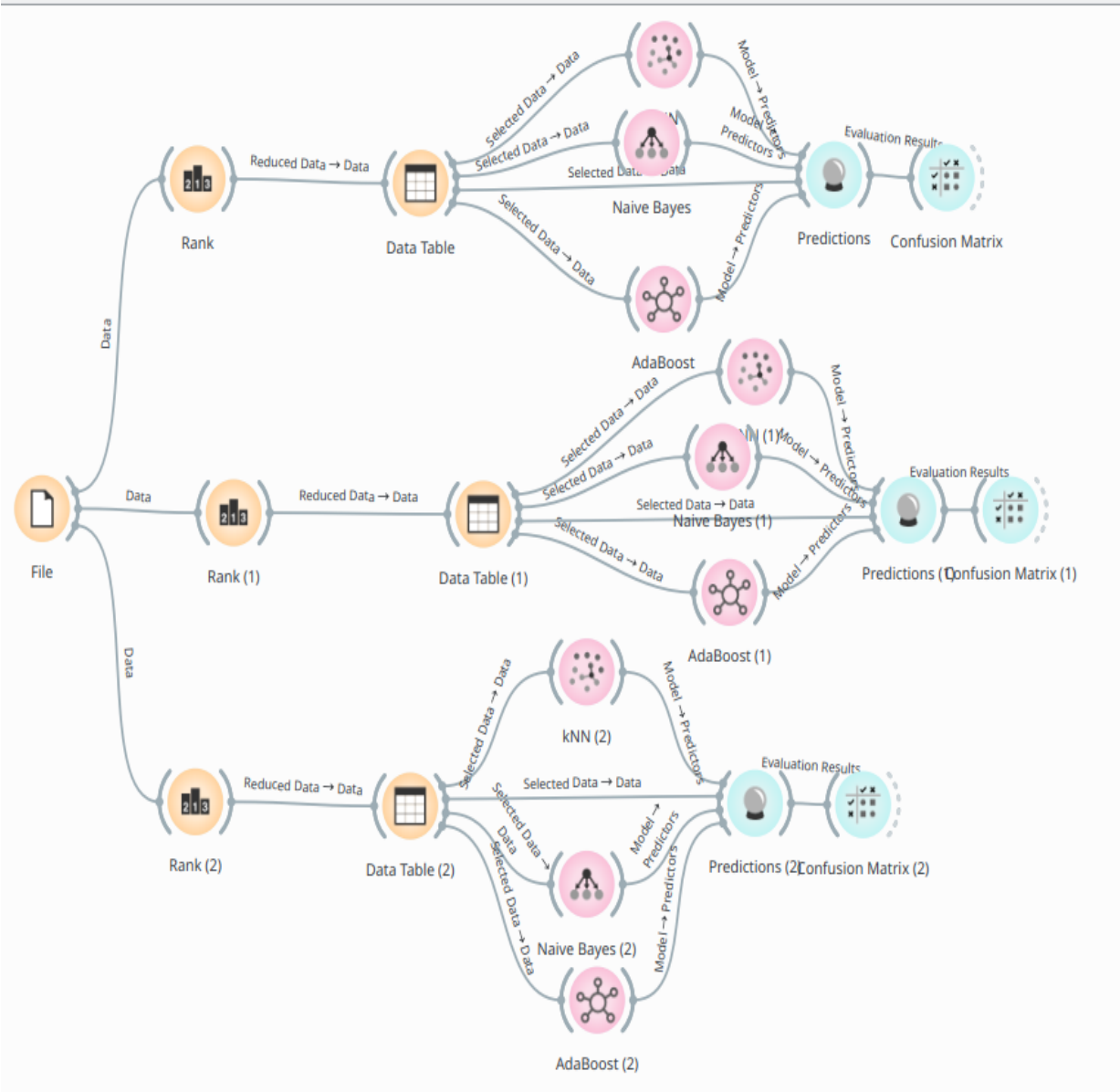


Fig 5.10

Feature Selection Technique->Gain Ratio

No. of selected attribute-> 20

	Model	AUC	CA	F1	Prec	Recall	MCC
selected	kNN	0.957	0.868	0.867	0.874	0.868	0.742
	Naive Bayes	0.966	0.908	0.908	0.908	0.908	0.816
	AdaBoost	1.000	1.000	1.000	1.000	1.000	1.000

No. of



**attribute-> 40**

No. of	Model	AUC	CA	F1	Prec	Recall	MCC
	kNN	0.963	0.885	0.885	0.890	0.885	0.776
	Naive Bayes (1)	0.976	0.885	0.885	0.887	0.885	0.772
	AdaBoost (1)	1.000	1.000	1.000	1.000	1.000	1.000

**selected attribute-> 50**

Model	AUC	CA	F1	Prec	Recall	MCC
kNN	0.951	0.868	0.868	0.870	0.868	0.738
Naive Bayes (2)	0.973	0.891	0.891	0.891	0.891	0.782
AdaBoost (2)	1.000	1.000	1.000	1.000	1.000	1.000

## PRACTICAL#6

**Aim :-** Perform Associative Mining In Weka and Orange on large datasets

**Theory :-** To perform association mining on large datasets, algorithms such as Apriori or FP-growth are employed. These algorithms efficiently extract frequent itemsets by iteratively identifying patterns within transactional data. With the support of these algorithms, associations between items can be discovered, aiding in tasks such as market basket analysis or recommendation systems. Efficient implementation and optimization are crucial for handling the computational complexity posed by large datasets, ensuring scalability and practical applicability in real-world scenarios.

### PROCEDURE:

#### USING WEKA TOOL:

**Scenario#1:WITH VALUE OF SUPPORT = 0.3 AND CONFIDENCE = 0.5**

```
Apriori
=====

Minimum support: 0.45 (2082 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 11

Generated sets of large itemsets:

Size of set of large itemsets L(1): 13

Size of set of large itemsets L(2): 7

Best rules found:

1. biscuits=t 2605 ==> bread and cake=t 2083    conf:(0.8)
2. milk-cream=t 2939 ==> bread and cake=t 2337    conf:(0.8)
3. fruit=t 2962 ==> bread and cake=t 2325    conf:(0.78)
4. baking needs=t 2795 ==> bread and cake=t 2191    conf:(0.78)
5. frozen foods=t 2717 ==> bread and cake=t 2129    conf:(0.78)
6. vegetables=t 2961 ==> bread and cake=t 2298    conf:(0.78)
7. vegetables=t 2961 ==> fruit=t 2207    conf:(0.75)
8. fruit=t 2962 ==> vegetables=t 2207    conf:(0.75)
9. bread and cake=t 3330 ==> milk-cream=t 2337    conf:(0.7)
10. bread and cake=t 3330 ==> fruit=t 2325    conf:(0.7)
```

**Fig 6.1 he rules found based on Support = 0.3 and Confidence = 0.5**

The Apriori method, with a minimum support of 0.3 and a minimum confidence of 0.5 over 11 cycles, produced 2082 instances in Figure 6.1. Large itemset sets were produced by it; L(1) contained 13 sets, while L(2) contained 7. Among the notable rules are those describing associations, such biscuits leading to cake and bread or fruit leading to cake and bread.

#### Scenario#2:WITH VALUE OF SUPPORT = 0.5 AND CONFIDENCE = 0.7

```
Apriori
=====

Minimum support: 0.5 (2314 instances)
Minimum metric <confidence>: 0.7
Number of cycles performed: 10

Generated sets of large itemsets:

Size of set of large itemsets L(1): 10

Size of set of large itemsets L(2): 2

Best rules found:

1. milk-cream=t 2939 ==> bread and cake=t 2337    conf:(0.8)
2. fruit=t 2962 ==> bread and cake=t 2325    conf:(0.78)
3. bread and cake=t 3330 ==> milk-cream=t 2337    conf:(0.7)
```

**Fig 6.2 The rules found based on Support = 0.5 and Confidence = 0.7**

The Apriori method, with a minimum support of 0.5 and a minimum confidence of 0.7 over 10 cycles, produced 2314 instances in Figure 6.2. Large itemset sets were produced by it. L(1) contained 10 sets, while L(2) contained 2.

Some Associations are milk-cream to bread and cake or fruit to bread and cake

#### Scenario#3:WITH VALUE OF SUPPORT = 0.3 AND CONFIDENCE = 0.7

```
=== Run information ===

Scheme:      weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.7 -S -1.0 -c -1
Relation:    supermarket
Instances:   4627
Attributes:  217
[list of attributes omitted]
=== Associator model (full training set) ===

No large itemsets and rules found!
```

**Fig 6.3 The rules found based on Support = 0.7 and Confidence = 0.9**

In Figure 6.3, no rules were found in this iteration of the Apriori method, with a minimum support of 0.7 and a minimum confidence of 0.9 applied. This finding might be explained by the strict confidence and support standards that were established, which might have led to too few examples satisfying these requirements to create meaningful correlations. The lack of rules implies that there might not be frequent itemsets in the dataset that meet the designated confidence and support requirements.

## Practical #7

**Aim :** Perform K-Nearest Neighbour Classification in Weka and Orange

### Theory:

K-Nearest Neighbors (KNN) is a simple yet powerful algorithm used for classification and regression tasks in machine learning. It's a non-parametric and instance-based learning algorithm, meaning it doesn't make strong assumptions about the underlying data distribution and it memorizes the entire training dataset. The choice of K (the number of nearest neighbors to consider) is crucial. A smaller K value can lead to more complex decision boundaries, while a larger K value can lead to smoother boundaries.

**Using :** WEKA tool

**Database used :** Darwin

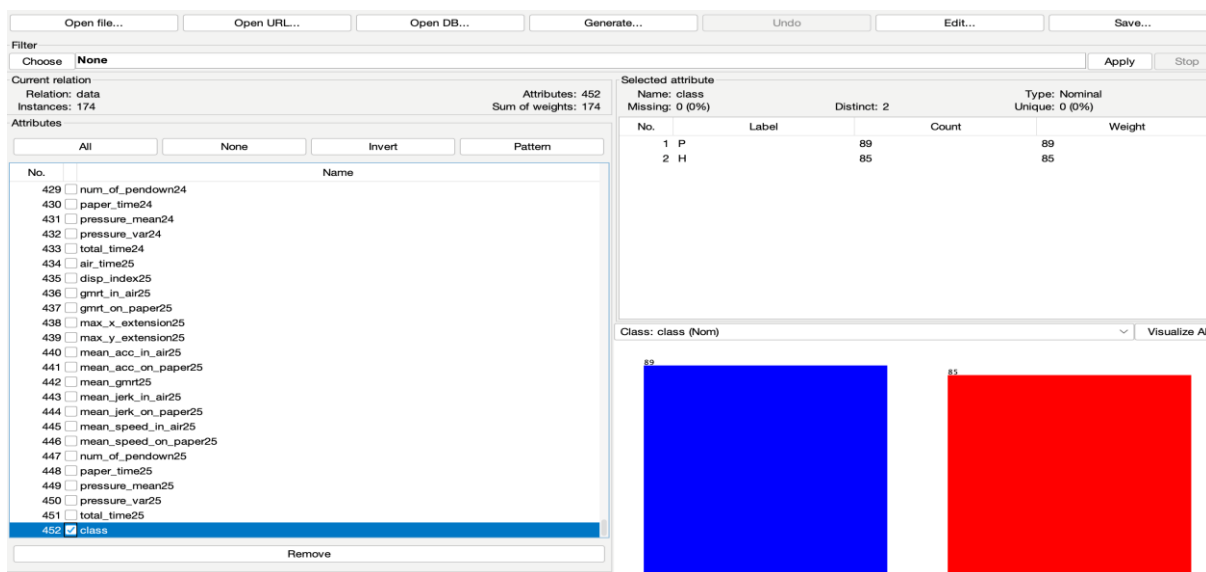


Fig 7.1 uploading labelled data

Linear NN Search

Classifier

Choose **IBk -K 5 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""**

Test options

☐ Use training set

☐ Supplied test set

☒ Cross-validation Folds **10**

☐ Percentage split % **66**

More options...

(Nom) class

Start Stop

Result list (right-click for options)

13:57:37 - lazy.IBk

13:59:20 - lazy.IBk

14:00:42 - lazy.IBk

Classifier output

=== Run information ===

Scheme: weka.classifiers.lazy.IBk -K 5 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Relation: data

Instances: 174

Attributes: 452

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier  
using 5 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	128	73.5632 %
Incorrectly Classified Instances	46	26.4368 %
Kappa statistic	0.4773	
Mean absolute error	0.2661	
Root mean squared error	0.4341	
Relative absolute error	53.2422 %	
Root relative squared error	86.8301 %	
Total Number of Instances	174	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.483	0.000	1.000	0.483	0.652	0.560	0.902	0.908	P
	1.000	0.517	0.649	1.000	0.787	0.560	0.902	0.843	H
Weighted Avg.	0.736	0.252	0.828	0.736	0.718	0.560	0.902	0.877	

=== Confusion Matrix ===

a b <-- classified as

43	46	a = P
0	85	b = H

Fig 7.2 KNN using linear NN search

## Cover Tree

```
=== Classifier model (full training set) ===

IB1 instance-based classifier
using 5 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      128                73.5632 %
Incorrectly Classified Instances    46                26.4368 %
Kappa statistic                    0.4773
Mean absolute error                 0.2661
Root mean squared error             0.4341
Relative absolute error             53.2422 %
Root relative squared error         86.8301 %
Total Number of Instances          174

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.483    0.000    1.000     0.483    0.652     0.560    0.902    0.908     P
                1.000    0.517    0.649     1.000    0.787     0.560    0.902    0.843     H
Weighted Avg.   0.736    0.252    0.828     0.736    0.718     0.560    0.902    0.877

=== Confusion Matrix ===

  a  b  <-- classified as
43 46 |  a = P
 0 85 |  b = H
```

Fig 7.3 KNN using Cover tree

## Ball Tree

```

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 5 nearest neighbour(s) for classification

Time taken to build model: 0.05 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      128           73.5632 %
Incorrectly Classified Instances    46           26.4368 %
Kappa statistic                    0.4773
Mean absolute error                 0.2661
Root mean squared error             0.4341
Relative absolute error             53.2422 %
Root relative squared error         86.8301 %
Total Number of Instances          174

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.483    0.000    1.000     0.483    0.652     0.560    0.902    0.908     P
                1.000    0.517    0.649     1.000    0.787     0.560    0.902    0.843     H
Weighted Avg.   0.736    0.252    0.828     0.736    0.718     0.560    0.902    0.877

=== Confusion Matrix ===
  a  b  <-- classified as
43 46 | a = P
 0 85 | b = H

```

Fig 7.4 KNN using Ball tree

Using : Orange tool

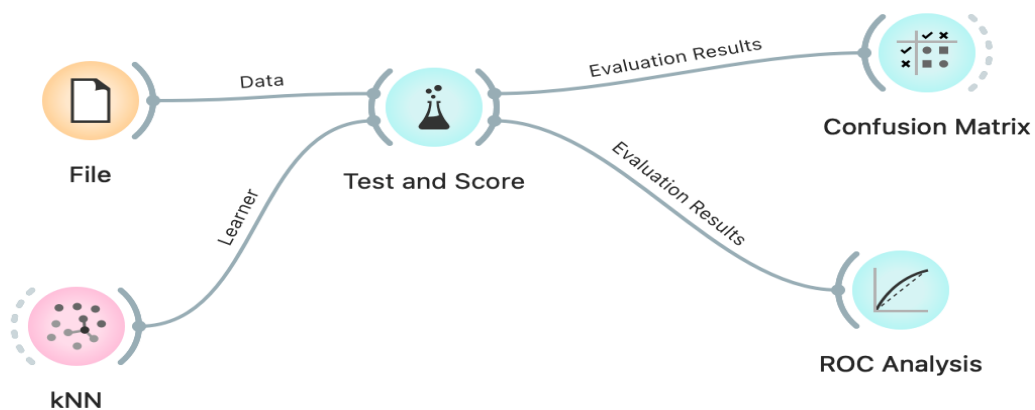


Fig 7.5 KNN using Orange tool

Matric : Euclidean

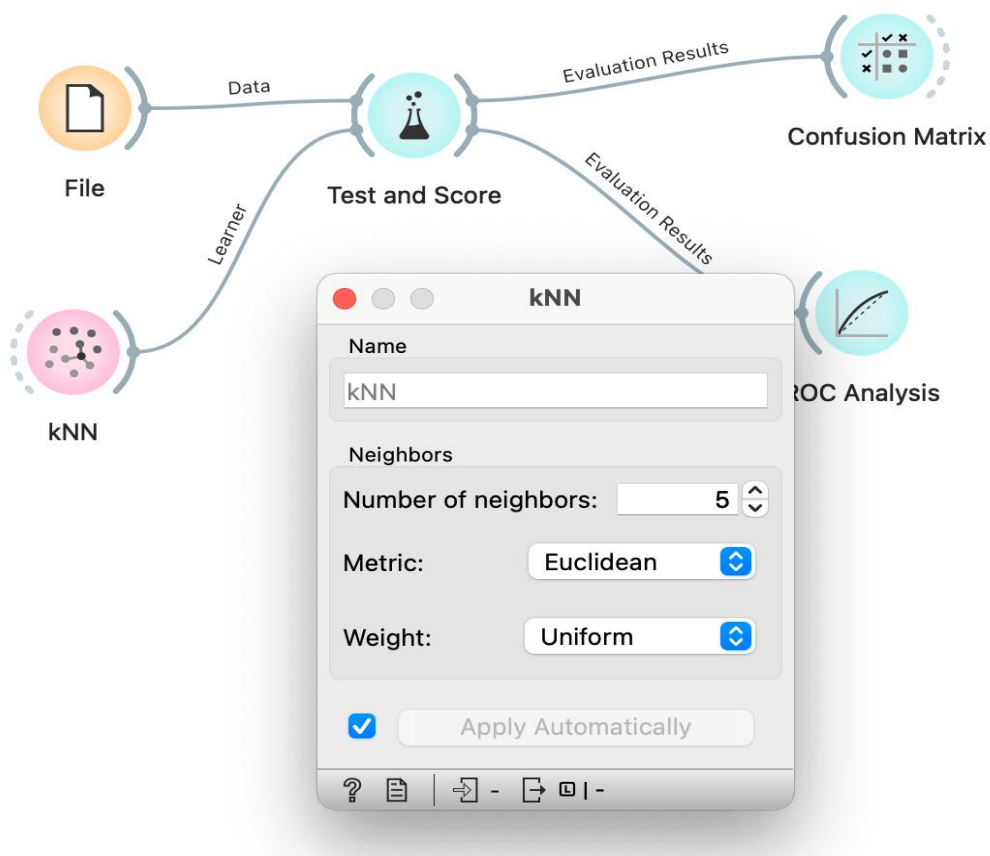


Fig 7.6 using Euclidean metric

☒ Cross validation

Number of folds: 5

☒ Stratified

☐ Cross validation by feature

☐ Random sampling

Repeat train/test: 10

Training set size: 66 %

☒ Stratified

☐ Leave one out

☐ Test on train data

☐ Test on test data

Evaluation results for target (None, show average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
kNN	0.815	0.724	0.713	0.773	0.724	0.497

Fig 7.7 Evaluation result

Matric : Manhattan



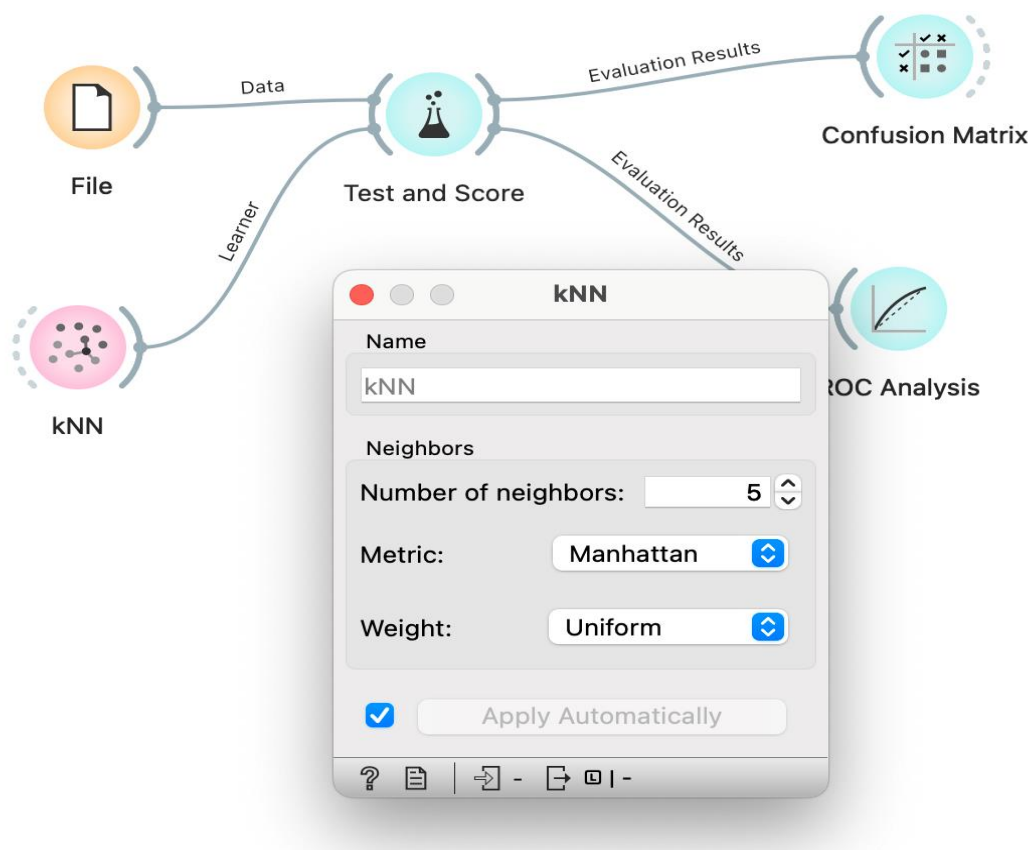


Fig 7.8 KNN using Manhattan metric

☒ Cross validation  
 Number of folds: 5  
☒ Stratified  
☐ Cross validation by feature  
☐ Random sampling  
 Repeat train/test: 10  
 Training set size: 66 %  
☒ Stratified  
☐ Leave one out  
☐ Test on train data  
☐ Test on test data

Evaluation results for target (None, show average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
kNN	0.898	0.718	0.696	0.821	0.718	0.534

Fig 7.9 Evaluation result

Matric: Chebyshev

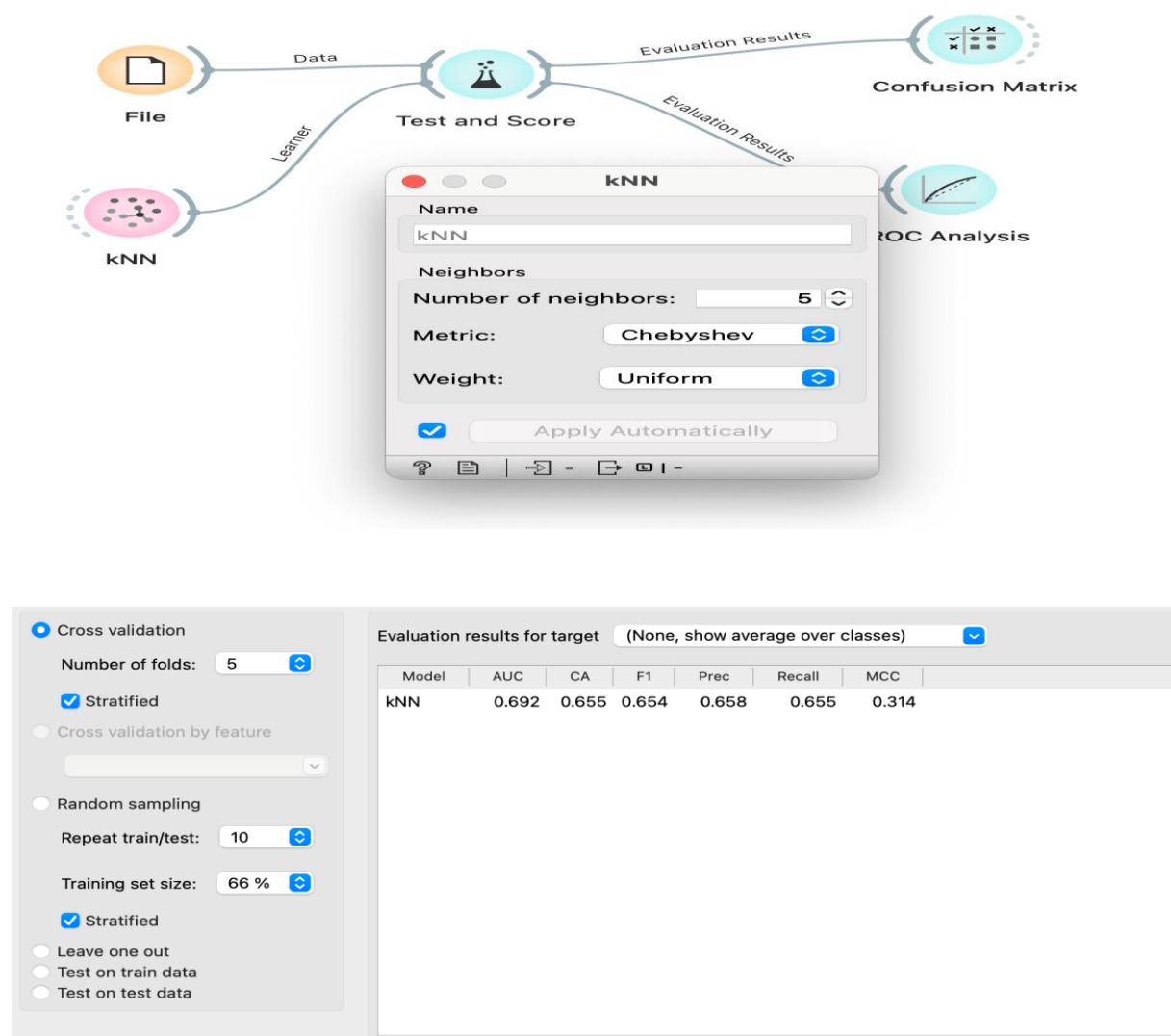


Fig 7.10 using Chebyshev metric