Practical – 1

Knowledge Representation (Prolog programming)

1.1 Write a program in Prolog to implement simple facts and Queries

Code:

```
male(yatharth).
male(karan).
male(sanket).

female(surali).
female(vaishnavi).
female(kavya).
```

Output:

```
yes
| ?- male(yatharth).

yes
| ?- female(surali).

yes
| ?- male(X).

X = yatharth ? ;

X = karan ? ;

X = sanket

yes
| ?-
```

Conclusion: We learned to implement simple program in Prolog.

1.2 Design a Family Tree for Your Family: Write a program which contains three predicates: male, female, parent. Make rules for following family relations: father, mother, grandfather, grandmother, brother, sister, uncle, aunt, nephew and niece, cousin.

Code

```
male(ratilal).
male(tejas).
male(jayesh).
male(dhiren).
male(yatharth).
male(divy).
male(mumux).

female(tara).
female(niru).
female(bhavna).
female(rajeshree).
female(vishwa).
female(drashti).

parent(tejas,yatharth).
parent(niru,yatharth).

parent(jayesh,divy).
parent(bhavna,divy).

parent(jayesh,vishwa).
parent(bhavna,vishwa).

parent(jayesh,drashti).
parent(bhavna,drashti).

parent(dhiren,mumux).
parent(rajeshree,mumux).

father(X,Y) :- male(X),parent(X,Y).
mother(X,Y) :- female(X),parent(X,Y).
son(X,Y) :- male(X),parent(Y,X).
daughter(X,Y) :- female(X),parent(X,Y).
```

grandfather(X,Y) :- male(X),parent(X,Somebody),parent(Somebody,Y).
grandmother(X,Y) :- female(X),parent(X,Somebody),parent(Somebody,Y).

brother(X,Y) :- male(X),parent(Somebody,X),parent(Somebody,Y).
sister(X,Y) :- female(X),parent(Somebody,X),parent(Somebody,Y).

uncle(X,Y) :- male(X),brother(X,Somebody),parent(Somebody,Y).
aunty(X,Y) :- female(Y),uncle(Somebody,X),spouse(Somebody,Y).

Output

```
yes
| ?- father(X,Y).

X = tejas
Y = yatharth ? ;

X = jayesh
Y = divy ? ;

X = jayesh
Y = vishwa ? ;

X = jayesh
Y = drashti ? ;

X = dhiren
Y = mumux ? ;
```

```
| ?- mother(X,Y).

X = niru
Y = yatharth ? ;

X = bhavna
Y = divy ? ;

X = bhavna
Y = vishwa ? ;

X = bhavna
Y = drashti ? ;

X = rajeshree
Y = mumux ? ;

(31 ms) no
| ?-
```

Conclusion: We learned to form functions for relations and ‘and’ & ‘or’ operations.

1.3 Design a Medical Diagnosis Expert System

Code:

```
answer(Inp,X):-
((Inp='Yes',X='y');(Inp='yes',X='y');(Inp='y',X='y');(Inp='Y',X='y');(Inp='YES',X='y'));((Inp='
NO',X='n');(Inp='no',X='n');(Inp='N',X='n');(Inp='n',X='n')).


mainask:-write('Write your name
:'),read(Name),cold(C),cough(F),fever(V),fun(C,F,V,Ans),write(Name),write(Ans).


fever(V):-write('Fever ? '),read(Inp),answer(Inp,V).

cough(F):-write('Cough ? '),read(Inp),answer(Inp,F).

cold(C):-write('Cold ? '),read(Inp),answer(Inp,C).


fun(C,F,V,Ans):-(C='y',F='y',V='y',Ans=' has viral few.');(C='n',F='n',V='y',Ans=' has viral
fever.');

(C='y',F='n',V='n',Ans=' has normal cold.');(C='n',F='n',V='n',Ans=' is alright!');

(C='n',F='y',V='n',Ans=' has normal cough.');(C='y',F='y',V='n',Ans=' has normal cold and
cough.').
```

Output:

```
uncaught exception: error(existence_error(proce
| ?- mainask.
Write your name :dhruv
.
Cold ? Y
.
Cough ? N.
Fever ? N.
dhruv has viral few.

true ?

yes
| ?-
```

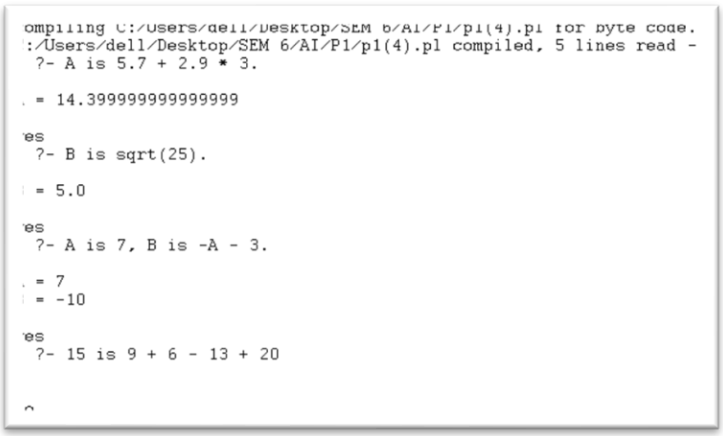
Conclusion: We learned to perform arithmetic operations on prolog, found sum of 1 to n integers and applied recursion.

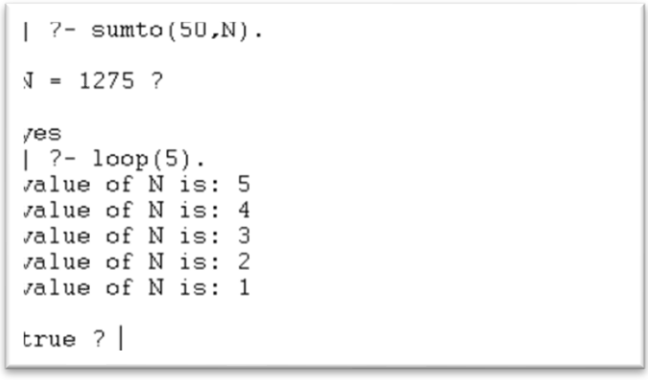
1.4 Write a Prolog program to demonstrate arithmetic operations and find addition of 1 to N numbers. Also demonstrate examples of recursion.

Code:

```
sumto(1, 1).
sumto(N, M) :- N>1, N1 is N-1, sumto(N1, M1), M is M1+N.

loop(0).
loop(N) :- N>0, write('value of N is: '), write(N), nl,
S is N-1, loop(S).
```

Output:





Conclusion: We learned to perform arithmetic operations on prolog, found sum of 1 to n integers and applied recursion.

1.5 Write a program to display Fibonacci series in prolog.

Code:

```
fab1(1,1).
fab1(2,1).
fab1(N,T):-N>2,N1 is N-1,N2 is N-2,fab1(N1,T1),fab1(N2,T2),T is (T1+T2).
```

Output:

```
yes
| ?- fab1(4,T).

T = 3 ?

yes
| ?-
```

Conclusion: We compiled the Fibonacci series.

1.6 Write a program to find factorial of a number in prolog using recursion.

Code:

```
fact(0,1).
fact(N,ANS):- N>0,
N1 is N-1,
fact(N1,A),
ANS is A*N.
```

Output:

```
yes
| ?- fact(4, ANS).

ANS = 24 ?
```

Conclusion: We implemented factorial

1.7 Write predicates one converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing.

- **Software requirements:** GNU Prolog
- **Theory:**

Here are the formulae for converting between degrees Fahrenheit and degrees Celsius:

DegreesF = (9 / 5 * DegreesC) + 32

DegreesC = 5 / 9 * (DegreesF - 32)

convert(C, F) that first checks which temperature (C or F) is given, and then does the appropriate conversion. That is, when C is given, convert(C, F) uses c2f(C, F).

Code

```
convert(F,C) :- C is (F-32)*(5/9).
c_to_f(C,F) :- F is C*(9/5) + 32.
frrezinf(F) :- F=<32.
```

Output

```
(16 ms) no
| ?- [pr3].
compiling C:/GNU-Prolog/bin/pr3.pl for byte code...
C:/GNU-Prolog/bin/pr3.pl compiled, 2 lines read - 1019 byte

yes
| ?- convert(32,C).

C = 0.0

yes
| ?- convert(33,C).

C = 0.55555555555555558

yes
| ?- convert(33,C).

C = 0.55555555555555558

yes
| ?- convert(36,C).

C = 2.2222222222222223

yes
| ?-
```

Conclusion: We converted degree of temperatures into different units using prolog.

1.8 Write a program in prolog to implement phone list which store name, phone number and birthdays of friends and family members. Write a query to get birthday a list of people whose birthdays are in the current month.

- **Software requirements**: GNU Prolog
- **Theory:**
  We first take the data in form of fact.
  Then we create a query for finding the appropriate data with condition.
  A single underscore (_) denotes an anonymous variable and means "any term".
  Unlike other variables, the underscore does not represent the same value everywhere it occurs within a predicate definition.
  We then define needed variable and get the answer.
- **Code**

```
phone_list(person(yatharth,chauhan),'1234567890',bdate(day(23),month(06),year(2002))).
phone_list(person(surali,asodariya),'1234567891',bdate(day(29),month(06),year(2002))).
phone_list(person(karan,bhatt),'1234567892',bdate(day(13),month(11),year(2002))).
phone_list(person(sanket,bhimani),'1234567892',bdate(day(2),month(2),year(2002))).
phone_list(person(vaishnavi,bhalodiya),'1234567893',bdate(day(3),month(3),year(2002))).
phone_list(person(kavya,bhalodiya),'123457789',bdate(day(4),month(4),year(2002))).
```

- **Output:**

```
| ?- phone_list(person(shubh,pate.

B = '123456798'
X = 2001
Y = 4 ?

yes
| ?-
```

- **Conclusion:** We implemented phone list query.

1.9 Implement Water Jug Problem in Prolog.

Variation1: A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?
Variation2: You have to defuse a bomb by placing exactly 4 gallons (15 L) of water on a sensor. The problem is, you only have a 5-gallon (18.9 L) jug and a 3 gallons (11 L) jug on hand! Neither jug has any measuring markings on it. Variation3: A milkman carries a full 12-liter container of milk. He needs to deliver exactly liters. However, the customer only has 8 and 5-liter jugs.
Neither jug has any measuring markings on it.
Write a program in Prolog to implement simple facts and Queries

- **Theory:**
  A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

- **Code:**

```
member(X,[X|_]). member(X,[Y|Z]):-
member(X,Z).

move(X,Y,_):-X=:=2,Y=:=0,write('done'),!. move(X,Y,Z):-
X<4,\+member((4,Y),Z),write("fill 4 jug"),nl,move(4,Y,[(4,Y)|Z]). move(X,Y,Z):-
Y<3,\+member((X,3),Z),write("fill 3 jug"),nl,move(X,3,[(X,3)|z]). move(X,Y,Z):-
X>0,\+member((0,Y),Z),write("pour 4 jug"),nl,move(0,Y,[(0,Y)|Z]). move(X,Y,Z):-
Y>0,\+member((X,0),Z),write("pour 3 jug"),nl,move(X,0,[(X,0)|Z]). move(X,Y,Z):-P
is X+Y,P>=4,Y>0,K is 4-X,M is Y-
K,\+member((4,M),Z),write("pour from 3jug to 4jug"),nl,move(4,M,[(4,M)|Z]).
move(X,Y,Z):-P is X+Y,P>=3,X>0,K is 3-Y,M is X-
K,\+member((M,3),Z),write("pour from 4jug to 3jug"),nl,move(M,3,[(M,3)|Z]).
move(X,Y,Z):-K is X+Y,K<4,Y>0,\+member((K,0),Z),write("pour from 3jug to
4jug"),nl,move(K,0,[(K,0)|Z]).
move(X,Y,Z):-K is X+Y,K<3,X>0,\+member((0,K),Z),write("pour from 4jug to
3jug"),nl,move(0,K,[(0,K)|Z]).
```

- **Output**:

```
fill 4 jug
fill 3 jug
pour 4 jug
pour 3 jug
fill 4 jug
pour from 4jug to 3jug
pour 3 jug
pour from 4jug to 3jug
fill 4 jug
pour from 4jug to 3jug
pour 3 jug
done
```

- **Conclusion**: We implemented Water Jug Problem in GNU Prolog.

1.10 Write a program to perform following operations on lists in prolog

- ✓ **Create a List**
- ✓ **Write in List**
- ✓ **Check Membership**
- ✓ **Length of a List**
- ✓ **Reverse a List**
- ✓ **Concatenation**
- ✓ **Add an item**
- ✓ **Delete an item**
- ✓ **Sub list**
- ✓ **Permutations**
- ✓ **Append list**

Finding nth element

- Software requirements: GNU Prolog
- Theory:
  In Prolog, lists are inbuilt data structures. Lists can be used to represent sets, stacks, queues, linked lists, and several complex data structures such as trees, graphs, etc.

Basic Notation and Properties of Lists:

- A list in Prolog is an ordered collection of items denoted as [i1, i2, …, in].
- Unlike arrays in other programming languages where we can directly access any element of the array, prolog lists allow direct access of the first element only which is denoted as Head. Therefore we can write a prolog list as: [Head | Rest], where Rest is the rest of the list excluding the first element Head.
- Prolog lists allow nonhomogeneous data types of list items.
- Nested lists of arbitrary depths are also allowed in prolog.

A list can be either **empty** or **non-empty**. In the first case, the list is simply written as a Prolog atom, []. In the second case, the list consists of two things as given below −

- The first item, called the **head** of the list;
- The remaining part of the list, called the **tail**.

Suppose we have a list like: [red, green, blue, white, dark]. Here the head is red and tail is [green, blue, white, dark]. So the tail is another list.

Basic Operations on Lists

Following table contains various operations on prolog lists −

Operations	Definition
Membership Checking	During this operation, we can verify whether a given element is member of specified list or not?
Length Calculation	With this operation, we can find the length of a list.
Concatenation	Concatenation is an operation which is used to join/add two lists.
Delete Items	This operation removes the specified element from a list.
Append Items	Append operation adds one list into another (as an item).
Insert Items	This operation inserts a given item into a list.

- **Code:**

```
list_mem(X,[X|_]).
list_mem(X,[_|TAIL]) :- list_mem(X,TAIL).

list_len[],0).
list_len([_|TAIL],N) :- list_len(TAIL,N1), N is N1 + 1.

list_con([],L,L).
list_con([X1|L1],L2,[X1|L3]) :- list_con(L1,L2,L3).

list_remove(X, [X], []).
list_remove(X,[X|L1], L1).
list_remove(X, [Y|L2], [Y|L1]) :- list_remove(X,L2,L1).

list_append(A,T,T) :- list_mem(A,T),!.
list_append(A,T,[T|A]).
```

- **Output:**

```
compiling C:/GNU-Prolog/bin/p1(10).pl for byte code...
C:/GNU-Prolog/bin/p1(10).pl compiled, 1 lines read - 426 bytes written, 12 ms
| ?- list_mem(b,[a,b,c]).

true ?

yes
| ?- list_mem(b,[a,[b,c]]).

no
| ?- list_mem([b,c],[a,[b,c]]).

true ?

yes
| ?- list_mem(d,[a,b,c]).

no
| ?-
```

```
compiling C:/GNU-Prolog/bin/p1(10).pl for byte code...
C:/GNU-Prolog/bin/p1(10).pl compiled, 6 lines read - 929 bytes written, 11 ms
| ?- list_len([],Len).

Len = 0

yes
| ?- list_len([a,b,c,d,e,f,g,h,i,j],Len).

Len = 10

yes
| ?- list_len([[a,b],[c,d],[e,f]],Len).

Len = 3

yes
| ?-
```

```
compiling C:/GNU-Prolog/bin/p1(10).pl for byte code...
C:/GNU-Prolog/bin/p1(10).pl compiled, 11 lines read - 1825 bytes written, 12 ms
| ?- list_rev([a,b,c,d,e],NewList).

NewList = [e,d,c,b,a]

yes
| ?- list_rev([a,b,c,d,e],[e,d,c,b,a]).

yes
| ?- list_rev([a,b,c,d,e],[e,d,c,b,x]).

no
| ?-
```

```
NewList = [1,2,a,b,c]

yes
| ?- list_con([],[a,b,c],NewList).

NewList = [a,b,c]

yes
| ?- list_concat([[1,2,3],[p,q,r]],[a,b,c],NewList).
uncaught exception: error(existence_error(procedure,list_concat/3),top_level/0)
| ?- list_con([[1,2,3],[p,q,r]],[a,b,c],NewList).

NewList = [[1,2,3],[p,q,r],a,b,c]

yes
| ?-
```

```
NewList = [a,e,i,o,u]

yes
| ?- list_append(e,[e,i,o,u],NewList).

NewList = [e,i,o,u]

yes
| ?- list_append([a,b],[e,i,o,u],NewList).

NewList = [[a,b],e,i,o,u]

yes
| ?-
```

```
| ?- list_delete(a,[a,e,i,o,u],NewList).

NewList = [e,i,o,u] ?

yes
| ?- list_delete(a,[a],NewList).

NewList = [] ?

yes
| ?- list_delete(X,[a,e,i,o,u],[a,e,o,u]).

X = i ? ;

no
| ?-
```

```
| ?- list_insert(a,[e,i,o,u],NewList).

NewList = [a,e,i,o,u] ? a

NewList = [e,a,i,o,u]

NewList = [e,i,a,o,u]

NewList = [e,i,o,a,u]

NewList = [e,i,o,u,a]

NewList = [e,i,o,u,a]

(15 ms) no
| ?-
```

```
| ?- list_perm([a,b,c,d],X).

X = [a,b,c,d] ? a

X = [a,b,c,d]

X = [a,b,d,c]

X = [a,b,d,c]

X = [a,b,d,c]

X = [a,b,d,c]

X = [a,c,b,d]

X = [a,c,b,d]

X = [a,c,d,b]

X = [a,c,d,b]

X = [a,c,d,b]

X = [a,c,d,b]

X = [a,d,b,c]
```

```
X = [d,b,c,a]

X = [d,c,a,b]

X = [d,c,a,b]

X = [d,c,b,a]

X = [d,c,b,a]

X = [d,c,b,a]

X = [d,c,b,a]

X = [d,c,a,b]

X = [d,c,a,b]

X = [d,c,b,a]

X = [d,c,b,a]

X = [d,c,b,a]

X = [d,c,b,a]

(93 ms) no
```

- **Conclusion:** We implemented various operations.

1.11 Write a program to demonstrate cut and fail in prolog.

- Software requirements: GNU Prolog
- Theory:
  When one statement is true, another one must be false. In such cases we can use the cut.
  We can also define a predicate where we use the two cases using disjunction (OR logic). So when first one satisfies, it does not check for the second one, otherwise, it will check for the second statement.
  The fail predicate simply fails the rule.

  The fail forces backtracking in an attempt to unify with another clause.

- **Code:**

```
max(X,Y,X) :- X >= Y, !.
max(X,Y,Y) :- X < Y.
max_find(X,Y,Max) :- X>=Y,!, Max = X; Max = Y.
a(X):- b(X),c(X),fail.
a(X):-d(X).
b(1).
b(4).
c(1).
c(3).
d(4).
```

- **Output:**

```
?- max(10,20,Max).

Max = 20

yes
?- max(20,10,Max).

Max = 20

(16 ms) yes
?-
```

```
Compiling C:/Users/dell/Desktop/SEM 6/AI/P1/p1(11).pl for byte code...
C:/Users/dell/Desktop/SEM 6/AI/P1/p1(11).pl compiled, 11 lines read -
| ?- a(X).

X = 4

yes
| ?-
```

- **Conclusion:** We leaned cut and fail function.

1.12 Write a program to solve Monkey Banana problem in Prolog

- Software requirements: GNU Prolog
- Theory:

Suppose the problem is as given below −

- A hungry monkey is in a room, and he is near the door.
- The monkey is on the floor.
- Bananas have been hung from the center of the ceiling of the room.
- There is a block (or chair) present in the room near the window.
- The monkey wants the banana, but cannot reach it.

So if the monkey is clever enough, he can come to the block, drag the block to the center, climb on it, and get the banana. Below are few observations in this case −

- Monkey can reach the block, if both of them are at the same level.
- If the block position is not at the center, then monkey can drag it to the center.
- If monkey and the block both are on the floor, and block is at the center, then the monkey can climb up on the block. So the vertical position of the monkey will be changed.
- When the monkey is on the block, and block is at the center, then the monkey can get the bananas.

We will create some predicates:

We have some predicates that will move from one state to another state, by performing action.

- When the block is at the middle, and monkey is on top of the block, and monkey does not have the banana (i.e. ***has not*** state), then using the ***grasp*** action, it will change from ***has not*** state to ***have*** state.
- From the floor, it can move to the top of the block (i.e. ***on top*** state), by performing the action ***climb***.
- The **push** or **drag** operation moves the block from one place to another.
- Monkey can move from one place to another using **walk** or **move** clauses.

  Another predicate will be canget(). Here we pass a state, so this will perform move predicate from one state to another using different actions, then perform canget() on state 2. When we have reached to the state '**has>**', this indicates '**has banana**'. We will stop the execution.

- **Code:**

```
move(state(middle,onbox,middle,hasnot), grasp, state(middle,onbox,middle,has)).
move(state(P,onfloor,P,H), climb, state(P,onbox,P,H)).
move(state(P1,onfloor,P1,H), drag(P1,P2), state(P2,onfloor,P2,H)).
move(state(P1,onfloor,B,H), walk(P1,P2), state(P2,onfloor,B,H)).
canget(state(_,_,_,has)).
canget(State1) :- move(State1,_,State2), canget(State2).
```

- **Output:**

```
| ?- canget(state(atdoor, onfloor, atwindow, hasnot)).

true ?

yes
| ?- trace
.
The debugger will first creep -- showing everything (trace)

yes
{trace}
| ?- canget(state(atdoor, onfloor, atwindow, hasnot)).
      1    1  Call: canget(state(atdoor,onfloor,atwindow,hasnot)) ?
      2    2  Call: move(state(atdoor,onfloor,atwindow,hasnot),_71,_111) ?
      2    2  Exit: move(state(atdoor,onfloor,atwindow,hasnot),walk(atdoor,_99),state(_99,onfloor,atwindow,hasnot)) ?
      3    2  Call: canget(state(_99,onfloor,atwindow,hasnot)) ?
      4    3  Call: move(state(_99,onfloor,atwindow,hasnot),_129,_169) ?
      4    3  Exit: move(state(atwindow,onfloor,atwindow,hasnot),climb,state(atwindow,onbox,atwindow,hasnot)) ?
      5    3  Call: canget(state(atwindow,onbox,atwindow,hasnot)) ?
      6    4  Call: move(state(atwindow,onbox,atwindow,hasnot),_184,_224) ?
      6    4  Fail: move(state(atwindow,onbox,atwindow,hasnot),_184,_212) ?
      5    3  Fail: canget(state(atwindow,onbox,atwindow,hasnot)) ?
      4    3  Redo: move(state(atwindow,onfloor,atwindow,hasnot),climb,state(atwindow,onbox,atwindow,hasnot)) ?
      4    3  Exit: move(state(atwindow,onfloor,atwindow,hasnot),drag(atwindow,_157),state(_157,onfloor,_157,hasnot)) ?
      5    3  Call: canget(state(_157,onfloor,_157,hasnot)) ?
      6    4  Call: move(state(_157,onfloor,_157,hasnot),_187,_227) ?
      6    4  Exit: move(state(_157,onfloor,_157,hasnot),climb,state(_157,onbox,_157,hasnot)) ?
      7    4  Call: canget(state(_157,onbox,_157,hasnot)) ?
      8    5  Call: move(state(_157,onbox,_157,hasnot),_242,_282) ?
      8    5  Exit: move(state(middle,onbox,middle,hasnot),grasp,state(middle,onbox,middle,has)) ?
      9    5  Call: canget(state(middle,onbox,middle,has)) ?
      9    5  Exit: canget(state(middle,onbox,middle,has)) ?
      7    4  Exit: canget(state(middle,onbox,middle,hasnot)) ?
      5    3  Exit: canget(state(middle,onfloor,middle,hasnot)) ?
      3    2  Exit: canget(state(atwindow,onfloor,atwindow,hasnot)) ?
      1    1  Exit: canget(state(atdoor,onfloor,atwindow,hasnot)) ?
```

- **Conclusion:** We solved the monkey banana problem with prolog.

1.13 Implement Missionaries and Cannibals Problem Solution in Prolog Missionaries and Cannibals is a problem in which 3 missionaries and 3 cannibals want to cross from the left bank of a river to the right bank of the river. There is a boat on the left bank, but it only carries at most two people at a time (and can never cross with zero people). If cannibals ever outnumber missionaries on either bank, the cannibals will eat the missionaries.

- **Theory:**

  In the missionaries and cannibals problem, three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board.

- **Code:**

```
path(Ini, Ini, _, []).

path(Ini, Fin, Visited, [move(M, C, Dir)|Path]):-      move(M, C, Dir),

mov(move(M, C, Dir), Ini, Aux),

\+ not_valid(Aux),

 \+ member(Aux, Visited),

path(Aux, Fin, [Aux|Visited], Path).
```

- **Output:**

```
compiling C:/GNU-Prolog/bin/pl_13.pl for byte code...
C:/GNU-Prolog/bin/pl_13.pl compiled, 30 lines read - 4749 bytes written, 9 ms

yes
| ?- path(state(0,0,right),state(3,3,_),[],Path).

Path = [move(1,1,left),move(1,1,right),move(0,2,left),move(0,1,right),move(0,2,left),move(0,1,right),move(2,0,left)
,move(1,1,right),move(2,0,left),move(0,1,right),move(0,2,left),move(1,0,right),move(1,1,left)] ?
```

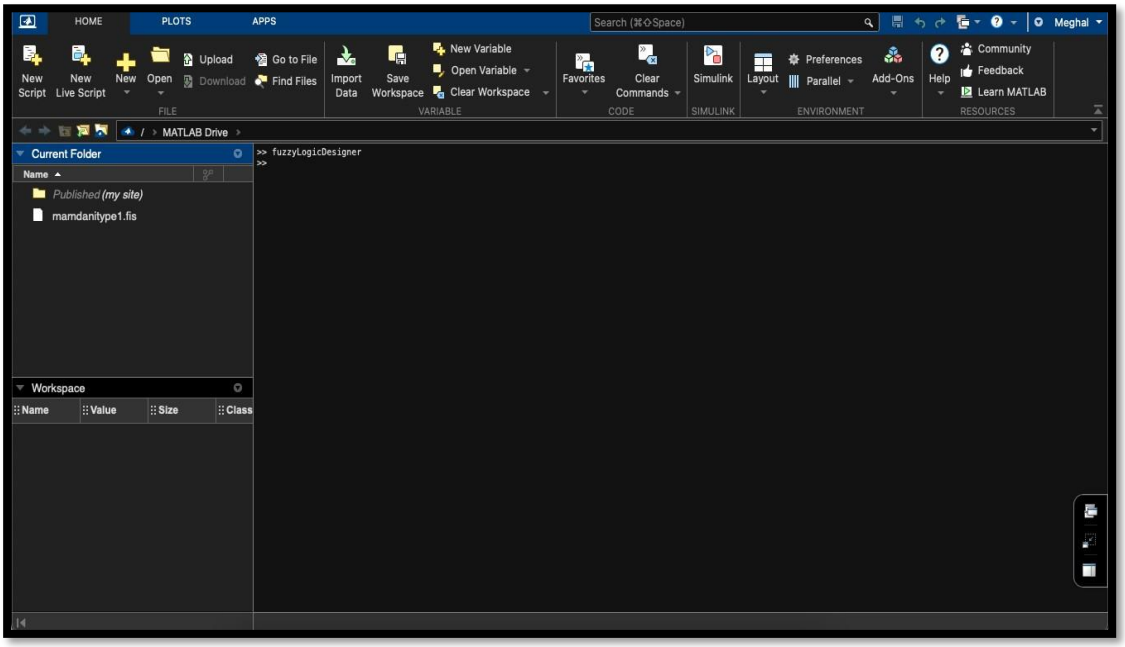
- **Conclusion:** We implemented Missionaries and Cannibals Problem in GNU Prolog.

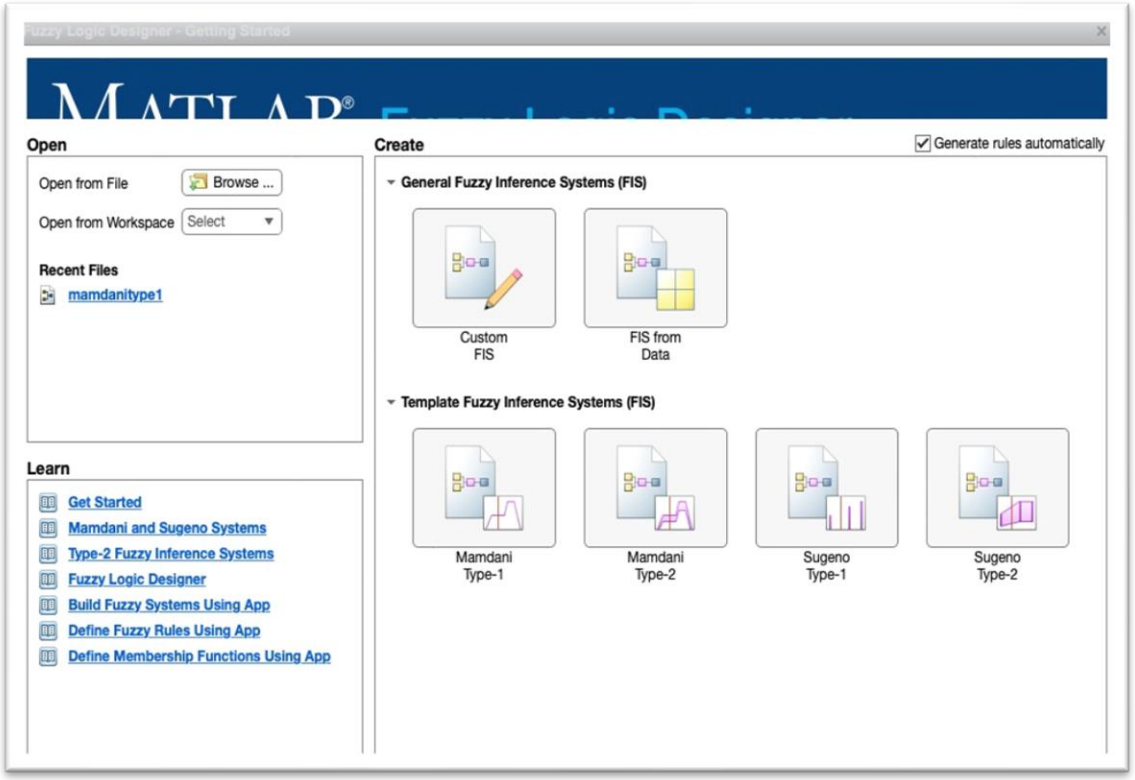
PRACTICAL:2

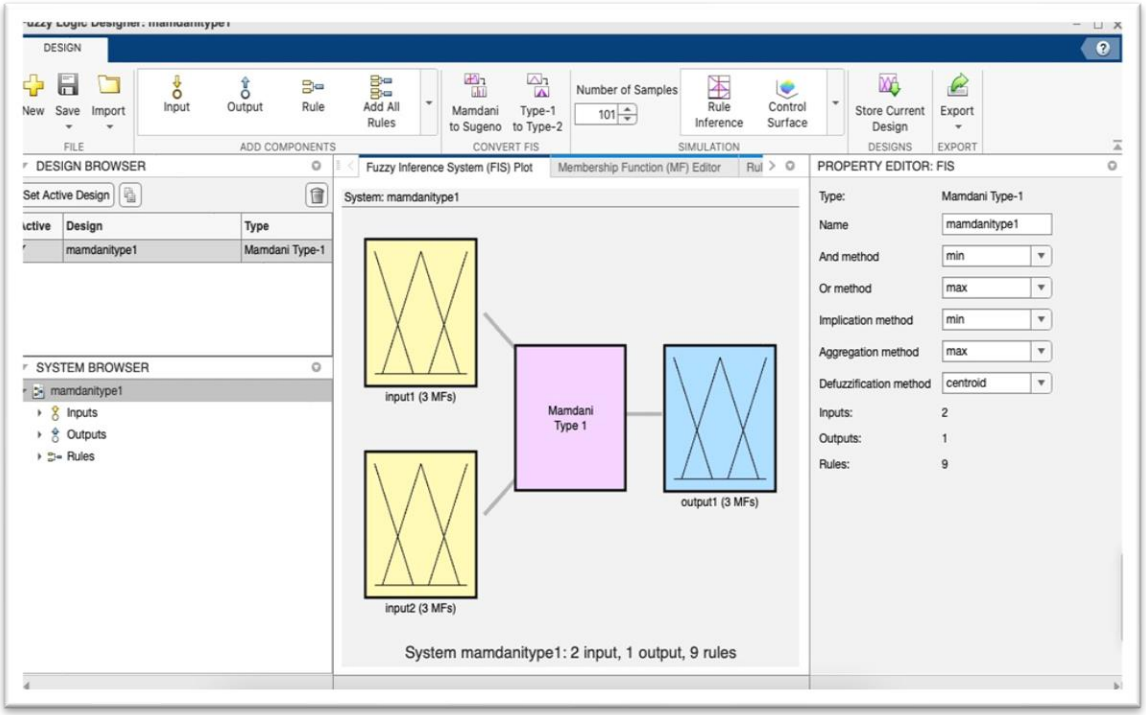
Aim: Design a controller to determine wash time of a domestic washing machine. Assume the input is dirt and grease on clothes. Use three descriptors for input variables and five descriptors for output variables. Derive the set of rules for controller action and defuzzification. The design should be supported by the figure wherever possible. Show that if the clothes are solid to a larger degree the wash time will be more and vice versa.

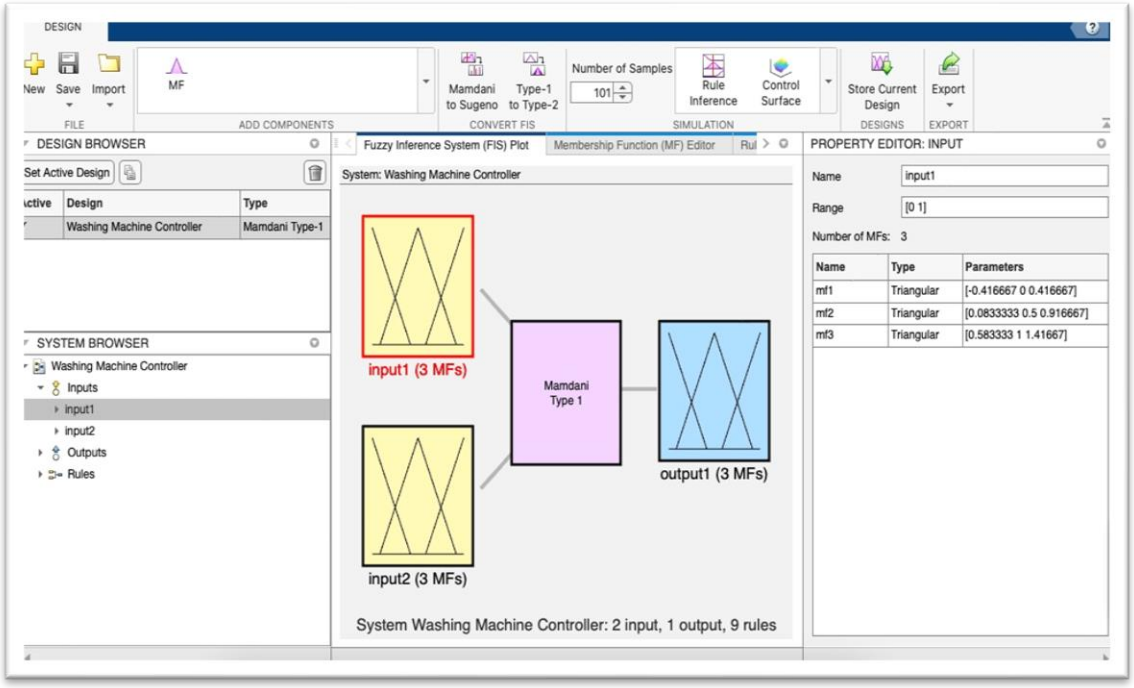
Software requirements: Matlab, Fuzzy Logic Designer

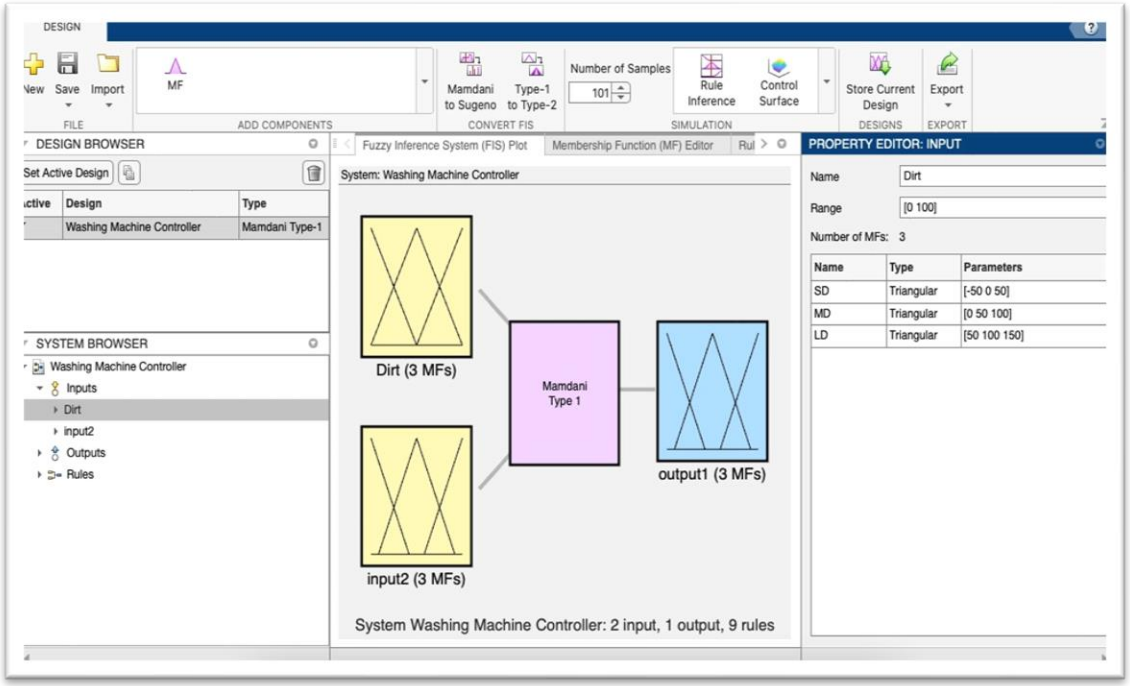
Steps:

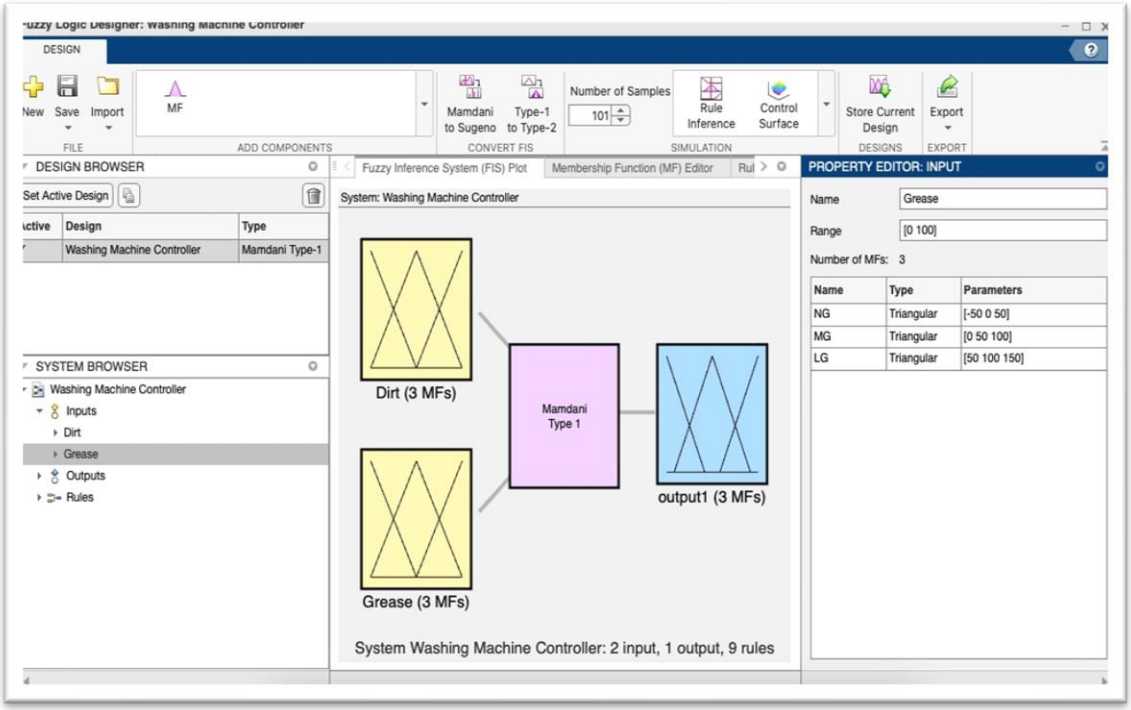


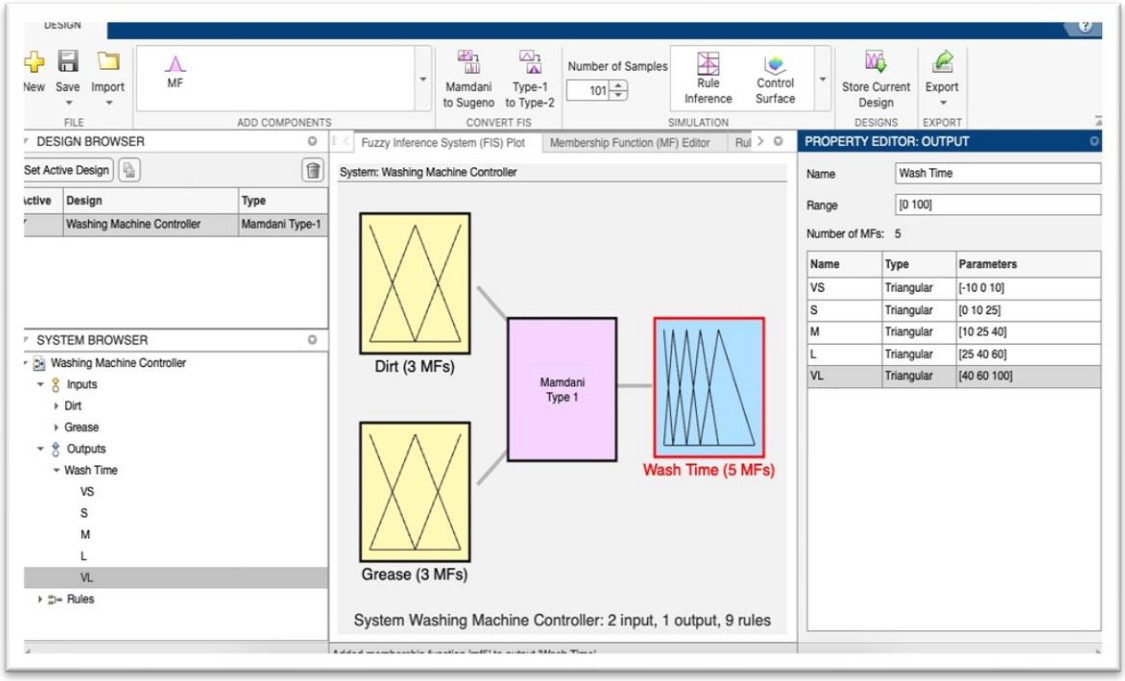

Open
Open from File
Browse ...
Open from Workspace
Select
Recent Files
mamdanitype1
Learn
Get Started
Mamdani and Sugeno Systems
Type-2 Fuzzy Inference Systems
Fuzzy Logic Designer
Build Fuzzy Systems Using App
Define Fuzzy Rules Using App
Define Membership Functions Using App
Create
Generate rules automatically
General Fuzzy Inference Systems (FIS)
Custom FIS
FIS from Data
Template Fuzzy Inference Systems (FIS)
Mamdani Type-1
Mamdani Type-2
Sugeno Type-1
Sugeno Type-2

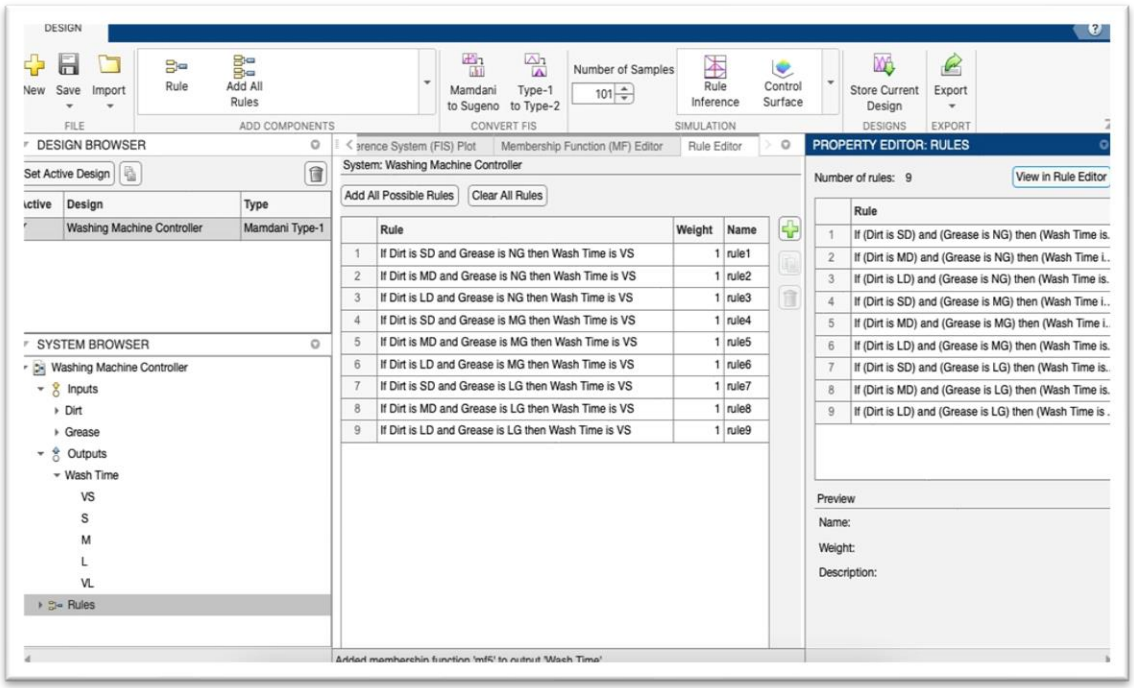

DESIGN
New
Save
Import
Input
Output
Rule
Add All Rules
Mamdani to Sugeno
Type-1 to Type-2
Number of Samples
101
Rule Inference
Control Surface
Store Current Design
Export
DESIGN BROWSER
Set Active Design
Design
Type
mamdanitype1
Mamdani Type-1
SYSTEM BROWSER
mamdanitype1
Inputs
Outputs
Rules
Fuzzy Inference System (FIS) Plot
Membership Function (MF) Editor
System: mamdanitype1
input1 (3 MFs)
input2 (3 MFs)
Mamdani Type 1
output1 (3 MFs)
System mamdanitype1: 2 input, 1 output, 9 rules
PROPERTY EDITOR: FIS
Type: Mamdani Type-1
Name mamdanitype1
And method min
Or method max
Implication method min
Aggregation method max
Defuzzification method centroid
Inputs: 2
Outputs: 1
Rules: 9


DESIGN
New
Save
Import
MF
Mamdani to Sugeno
Type-1 to Type-2
Number of Samples
101
Rule Inference
Control Surface
Store Current Design
Export
FILE
ADD COMPONENTS
CONVERT FIS
SIMULATION
DESIGNS
EXPORT
DESIGN BROWSER
Set Active Design
Design
Type
Washing Machine Controller
Mamdani Type-1
SYSTEM BROWSER
Washing Machine Controller
Inputs
input1
input2
Outputs
Rules
Fuzzy Inference System (FIS) Plot
Membership Function (MF) Editor
System: Washing Machine Controller
input1 (3 MFs)
input2 (3 MFs)
Mamdani Type 1
output1 (3 MFs)
System Washing Machine Controller: 2 input, 1 output, 9 rules
PROPERTY EDITOR: INPUT
Name
input1
Range
[0 1]
Number of MFs: 3
Name
Type
Parameters
mf1
Triangular
[-0.416667 0 0.416667]
mf2
Triangular
[0.0833333 0.5 0.916667]
mf3
Triangular
[0.583333 1 1.41667]

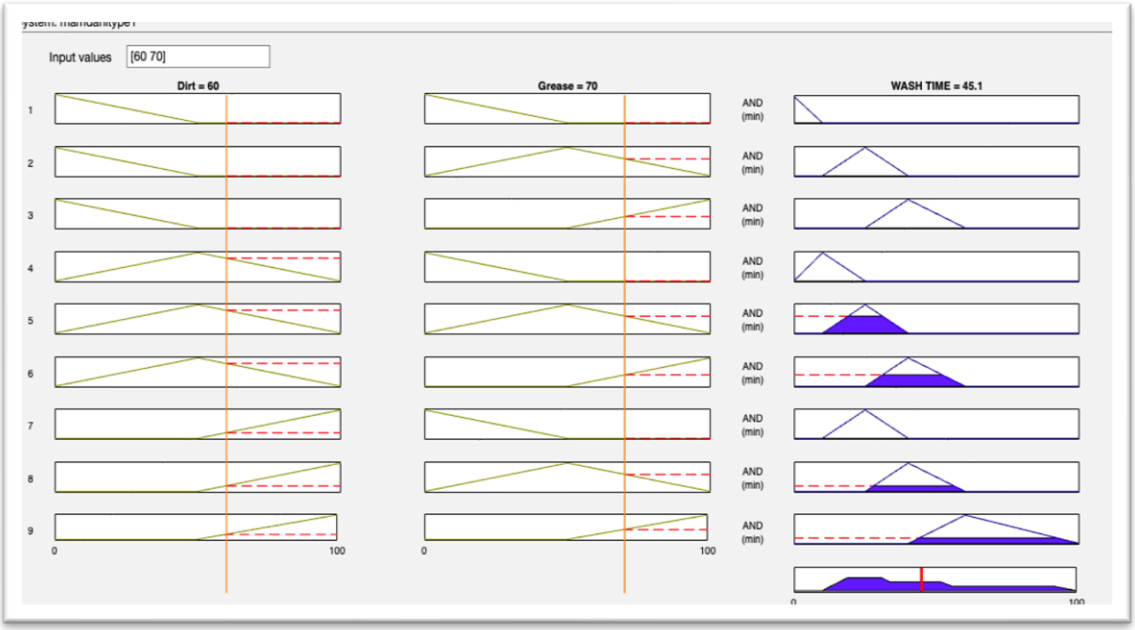

DESIGN
New
Save
Import
MF
Mamdani to Sugeno
Type-1 to Type-2
Number of Samples
101
Rule Inference
Control Surface
Store Current Design
Export
FILE
ADD COMPONENTS
CONVERT FIS
SIMULATION
DESIGNS
EXPORT
DESIGN BROWSER
Set Active Design
Design
Type
Washing Machine Controller
Mamdani Type-1
SYSTEM BROWSER
Washing Machine Controller
Inputs
Dirt
input2
Outputs
Rules
Fuzzy Inference System (FIS) Plot
Membership Function (MF) Editor
System: Washing Machine Controller
Dirt (3 MFs)
input2 (3 MFs)
Mamdani Type 1
output1 (3 MFs)
System Washing Machine Controller: 2 input, 1 output, 9 rules
PROPERTY EDITOR: INPUT
Name
Dirt
Range
[0 100]
Number of MFs: 3
Name
Type
Parameters
SD
Triangular
[-50 0 50]
MD
Triangular
[0 50 100]
LD
Triangular
[50 100 150]


Fuzzy Logic Designer: Washing Machine Controller
DESIGN
New
Save
Import
MF
Mamdani to Sugeno
Type-1 to Type-2
Number of Samples
101
Rule Inference
Control Surface
Store Current Design
Export
FILE
ADD COMPONENTS
CONVERT FIS
SIMULATION
DESIGNS
EXPORT
DESIGN BROWSER
Set Active Design
Design
Type
Washing Machine Controller
Mamdani Type-1
SYSTEM BROWSER
Washing Machine Controller
Inputs
Dirt
Grease
Outputs
Rules
Fuzzy Inference System (FIS) Plot
Membership Function (MF) Editor
System: Washing Machine Controller
Dirt (3 MFs)
Grease (3 MFs)
Mamdani Type 1
output1 (3 MFs)
System Washing Machine Controller: 2 input, 1 output, 9 rules
PROPERTY EDITOR: INPUT
Name
Grease
Range
[0 100]
Number of MFs: 3
Name
Type
Parameters
NG
Triangular
[-50 0 50]
MG
Triangular
[0 50 100]
LG
Triangular
[50 100 150]


DESIGN
New
Save
Import
MF
FILE
ADD COMPONENTS
Mamdani to Sugeno
Type-1 to Type-2
CONVERT FIS
Number of Samples
101
Rule Inference
Control Surface
SIMULATION
Store Current Design
DESIGNS
Export
EXPORT
DESIGN BROWSER
Set Active Design
Design
Type
Washing Machine Controller
Mamdani Type-1
SYSTEM BROWSER
Washing Machine Controller
Inputs
Dirt
Grease
Outputs
Wash Time
VS
S
M
L
VL
Rules
Fuzzy Inference System (FIS) Plot
Membership Function (MF) Editor
System: Washing Machine Controller
Dirt (3 MFs)
Grease (3 MFs)
Mamdani Type 1
Wash Time (5 MFs)
System Washing Machine Controller: 2 input, 1 output, 9 rules
PROPERTY EDITOR: OUTPUT
Name
Wash Time
Range
[0 100]
Number of MFs: 5
Name
Type
Parameters
VS
Triangular
[-10 0 10]
S
Triangular
[0 10 25]
M
Triangular
[10 25 40]
L
Triangular
[25 40 60]
VL
Triangular
[40 60 100]


DESIGN
New
Save
Import
Rule
Add All Rules
FILE
ADD COMPONENTS
Mamdani to Sugeno
Type-1 to Type-2
CONVERT FIS
Number of Samples
101
Rule Inference
Control Surface
SIMULATION
Store Current Design
DESIGNS
Export
EXPORT
DESIGN BROWSER
Set Active Design
Design
Type
Washing Machine Controller
Mamdani Type-1
SYSTEM BROWSER
Washing Machine Controller
Inputs
Dirt
Grease
Outputs
Wash Time
VS
S
M
L
VL
Rules
Membership Function (MF) Editor
Rule Editor
System: Washing Machine Controller
Add All Possible Rules
Clear All Rules
Rule
Weight
Name
1 If Dirt is SD and Grease is NG then Wash Time is VS 1 rule1
2 If Dirt is MD and Grease is NG then Wash Time is VS 1 rule2
3 If Dirt is LD and Grease is NG then Wash Time is VS 1 rule3
4 If Dirt is SD and Grease is MG then Wash Time is VS 1 rule4
5 If Dirt is MD and Grease is MG then Wash Time is VS 1 rule5
6 If Dirt is LD and Grease is MG then Wash Time is VS 1 rule6
7 If Dirt is SD and Grease is LG then Wash Time is VS 1 rule7
8 If Dirt is MD and Grease is LG then Wash Time is VS 1 rule8
9 If Dirt is LD and Grease is LG then Wash Time is VS 1 rule9
PROPERTY EDITOR: RULES
Number of rules: 9
View in Rule Editor
Rule
1 If (Dirt is SD) and (Grease is NG) then (Wash Time is.
2 If (Dirt is MD) and (Grease is NG) then (Wash Time i..
3 If (Dirt is LD) and (Grease is NG) then (Wash Time is.
4 If (Dirt is SD) and (Grease is MG) then (Wash Time i..
5 If (Dirt is MD) and (Grease is MG) then (Wash Time i..
6 If (Dirt is LD) and (Grease is MG) then (Wash Time is.
7 If (Dirt is SD) and (Grease is LG) then (Wash Time is.
8 If (Dirt is MD) and (Grease is LG) then (Wash Time is.
9 If (Dirt is LD) and (Grease is LG) then (Wash Time is .
Preview
Name:
Weight:
Description:

	Rule	Weight	Name
1	If Dirt is SD and Grease is NG then WASH TIME is VS	1	rule1
2	If Dirt is SD and Grease is MG then WASH TIME is M	1	rule2
3	If Dirt is SD and Grease is LG then WASH TIME is L	1	rule3
4	If Dirt is MD and Grease is NG then WASH TIME is S	1	rule4
5	If Dirt is MD and Grease is MG then WASH TIME is M	1	rule5
6	If Dirt is MD and Grease is LG then WASH TIME is L	1	rule6
7	If Dirt is LD and Grease is NG then WASH TIME is M	1	rule7
8	If Dirt is LD and Grease is MG then WASH TIME is L	1	rule8
9	If Dirt is LD and Grease is LG then WASH TIME is VL	1	rule9



- **Conclusion:** The final Wash time = 45.1 minutes.

Practical – 3

Constraint Satisfaction Problem

3.1: Constraint Satisfaction Problem –Crypt Arithmetic

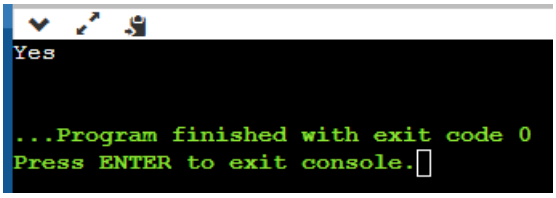
[SEND + MORE = MONEY]

Code:

```
def isSolvable(words, result):
    mp = [-1]*(26)
    used = [0]*(10)
    Hash = [0]*(26)
    CharAtfront = [0]*(26)
    uniq = ""
    for word in range(len(words)):
        for i in range(len(words[word])):
            ch = words[word][i]
            Hash[ord(ch) - ord('A')] += pow(10, len(words[word]) - i - 1)
            if mp[ord(ch) - ord('A')] == -1:
                mp[ord(ch) - ord('A')] = 0
                uniq += str(ch)
            if i == 0 and len(words[word]) > 1:
                CharAtfront[ord(ch) - ord('A')] = 1
    for i in range(len(result)):
        ch = result[i]
        Hash[ord(ch) - ord('A')] -= pow(10, len(result) - i - 1)
        if mp[ord(ch) - ord('A')] == -1:
            mp[ord(ch) - ord('A')] = 0
            uniq += str(ch)
        if i == 0 and len(result) > 1:
            CharAtfront[ord(ch) - ord('A')] = 1
    mp = [-1]*(26)
    return True
def solve(words, i, S, mp, used, Hash, CharAtfront):
    if i == len(words):
        return S == 0
```

```
    ch = words[i]
    val = mp[ord(words[i]) - ord('A')]
    if val != -1:
        return solve(words, i + 1, S + val * Hash[ord(ch) - ord('A')], mp, used, Hash,
CharAtfront)
    x = False
    for l in range(10):
        if CharAtfront[ord(ch) - ord('A')] == 1 and l == 0:
            continue
        if used[l] == 1:
            continue
        mp[ord(ch) - ord('A')] = l
        used[l] = 1
        x |= solve(words, i + 1, S + l * Hash[ord(ch) - ord('A')], mp, used, Hash,
CharAtfront)
        mp[ord(ch) - ord('A')] = -1
        used[l] = 0
    return x
arr = [ "SIX", "SEVEN", "SEVEN" ]
S = "TWENTY"
if isSolvable(arr, S):
    print("Yes")
else:
    print("No")
```

Output:



Conclusion: In this practical we learned about . Constraint Satisfaction problem in ai.

Practical – 4

Natural Language Processing

4.1: Regular Expression Perform Natural Language Processing Tasks [Text Reading, Text Analysis, Text Pre-processing, Text Classification, EDA, Stemming, Lemmatization] using NLTK using Python Programming.

Code:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

data = pd.read_csv("Corona_NLP_train.csv",encoding='latin1')
df = pd.DataFrame(data)
df.head()

plt.figure(figsize=(10,5))
sns.countplot(x='Sentiment', data=df, order=['Extremely Negative', 'Negative', 'Neutral', 'Positive', 'Extremely Positive'], )

df.info()

reg = re.compile("(@[A-Za-z0-9]+)|(#[A-Za-z0-9]+)|([^0-9A-Za-z t])|(w+://S+)")
tweet = []
for i in df["OriginalTweet"]:
    tweet.append(reg.sub(" ", i))
df = pd.concat([df, pd.DataFrame(tweet, columns=["CleanedTweet"])], axis=1, sort=False)


df.head()
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
stop_words = set(stopwords.words('english'))     # make a set of stopwords
vectoriser = TfidfVectorizer(stop_words=None)

X_train = vectoriser.fit_transform(df["CleanedTweet"])
# Encoding the classes in numerical values
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
y_train = encoder.fit_transform(df['Sentiment'])
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(X_train, y_train)

test_data = pd.read_csv("Corona_NLP_test.csv",encoding='latin1')
test_df = pd.DataFrame(test_data)
test_df.head()

reg1 = re.compile("(@[A-Za-z0-9]+)|(#[A-Za-z0-9]+)|([^0-9A-Za-z t])|(w+://S+)")

tweet = []

for i in test_df["OriginalTweet"]:
    tweet.append(reg1.sub(" ", i))
test_df = pd.concat([test_df, pd.DataFrame(tweet, columns=["CleanedTweet"])], axis=1,
sort=False)
test_df.head()


X_test = vectoriser.transform(test_df["CleanedTweet"])
y_test = encoder.transform(test_df["Sentiment"])
# Prediction
y_pred = classifier.predict(X_test)

len(y_pred)

pred_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
pred_df.head()

from sklearn import metrics
# Generate the roc curve using scikit-learn.
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred, pos_label=1)
plt.plot(fpr, tpr)
```

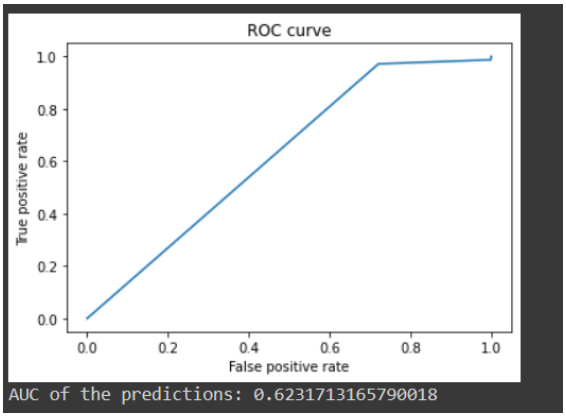
```
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.show()
# Measure the area under the curve. The closer to 1, the "better" the predictions.
print("AUC of the predictions: {0}".format(metrics.auc(fpr, tpr)))

import matplotlib.pyplot as plt
import numpy
from sklearn import metrics



confusion_matrix = metrics.confusion_matrix(y_test, y_pred)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix)

cm_display.plot()
plt.show()

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

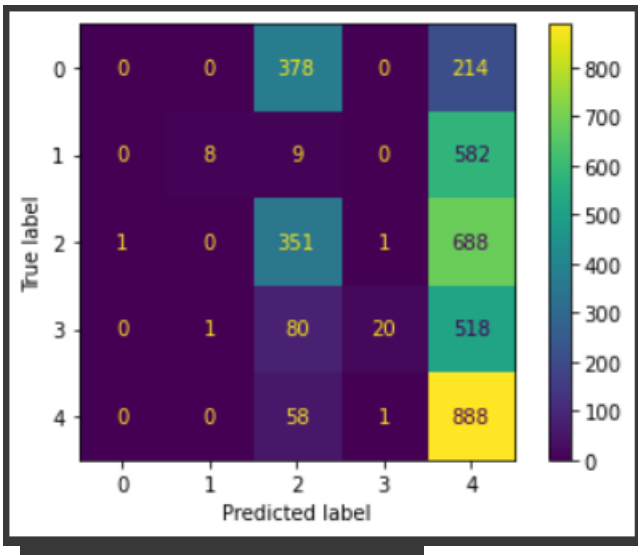
Output:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment	CleanedTweet
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral	Gahan https t co iFz9FAn2Pa and https ...
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive	advice Talk to your neighbours family to excha...
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive	Coronavirus Australia Woolworths to give elde...
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive	My food stock is not the only one which is emp...
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative	Me ready to go at supermarket during the ou...

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment	CleanedTweet
0	1	44953	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative	TRENDING New Yorkers encounter empty supermar...
1	2	44954	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive	When I couldn t find hand sanitizer at Fred Me...
2	3	44955	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive	Find out how you can protect yourself and love...
3	4	44956	Chicagoland	02-03-2020	#Panic buying hits #NewYork City as anxious sh...	Negative	buying hits City as anxious shoppers stock...
4	5	44957	Melbourne, Victoria	03-03-2020	#toiletpaper #dunnypaper #coronavirus #coronav...	Neutral	19 One week everyone...

	Actual	Predicted
0	0	4
1	4	4
2	1	4
3	2	2
4	3	2






```
0.33359662980516064
```

Conclusion: In this practical we learned about NLP in AI.

Practical – 5

Chatbot Application

5.1: Chatbot Building with SAP Conversational AI Track.

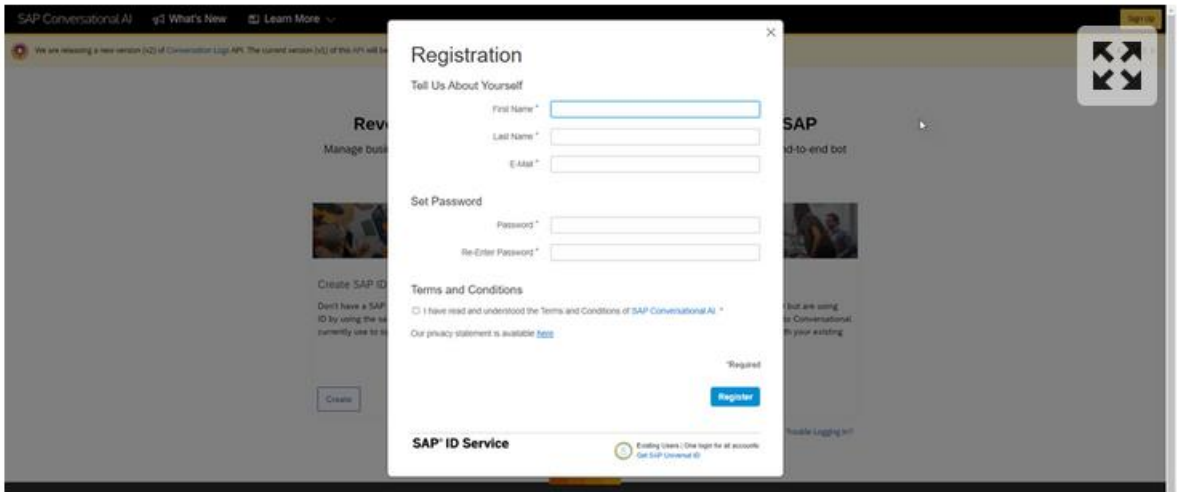
Output:

Step 1

Create SAP Conversational AI account

Go to https://cai.tools.sap/, and click Sign Up in the upper-right corner.

Follow the instructions for creating an account.



Step 2

Create new bot project

Click **New Bot**.

Fill in the following:

- Select **Perform Actions**.

  With a standard **Perform Actions** bot, the developer is responsible for creating entities and intents with expressions. The developer is also responsible for building and managing the conversational flow that pulls information from back-end systems to help simplify processes for the chatbot end user.

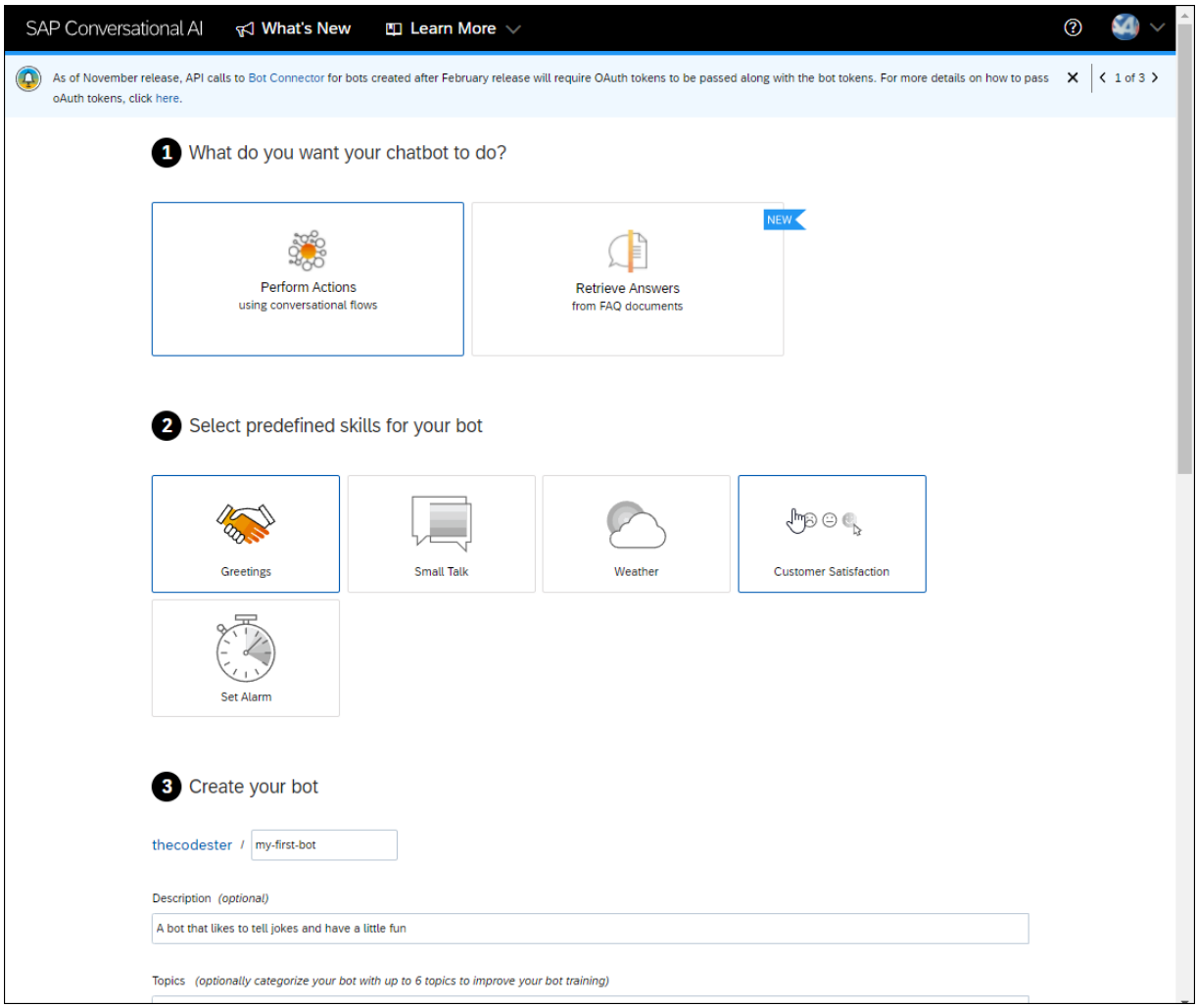
  An **FAQ** bot retrieves answers to users' questions from one or more documents (.csv files) that you upload. The document must include predefined pairs of questions and answers. This allows your bot to map the user's query to the best match and retrieve an answer without interpreting the intent of the question.

  To ease the complexity of the FAQ bot, the intents and entities are predefined and hidden, and the bot includes a set of predefined skills. However, you can design the bot responses as per your business needs.

- For the predefined skills, choose **Greetings**.
- In the **Create your bot** section, enter the following:

Field Name	Value
Bot name	**my-first-bot**
Description	**A bot that likes to tell jokes and have a little fun**

- For **Data Policy**, select the following:
  - **Non-personal**.
  - **Store**.
  - **Non-vulnerable**.
- For **Bot visibility**, select **Public**.



Step 3

The 4 stages of a bot's life

There are 4 stages in your bot's life:

- **Train:** Teach your bot what it needs to understand.

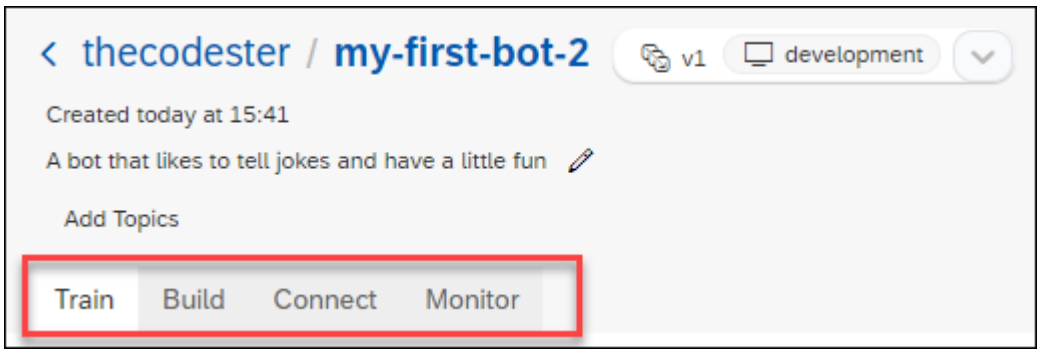
  Here you will create ***intents*** – ideas your bot will recognize – using ***expressions*** that the bot should recognize coming from the user.

- **Build:** Create your conversational flow with our Bot Builder tool. Here you give your bot skills it can perform.

Here you will create ***skills*** – things your bot can do – and define when they will be triggered. Your skills are defined by ***triggers***, ***requirements*** (information your bot must collect), and the ***actions***.

- **Connect:** Ship your bot to one or several messaging platforms.
- **Monitor:** See how users are communicating with your bot, check if the bot is understanding users correctly, and make updates to the bots intents/entities. Monitoring also lets you see what your users want, and gives you ideas of what additional capabilities you could add to the bot.

The 4 phases are represented in the bot project by the 4 tabs.



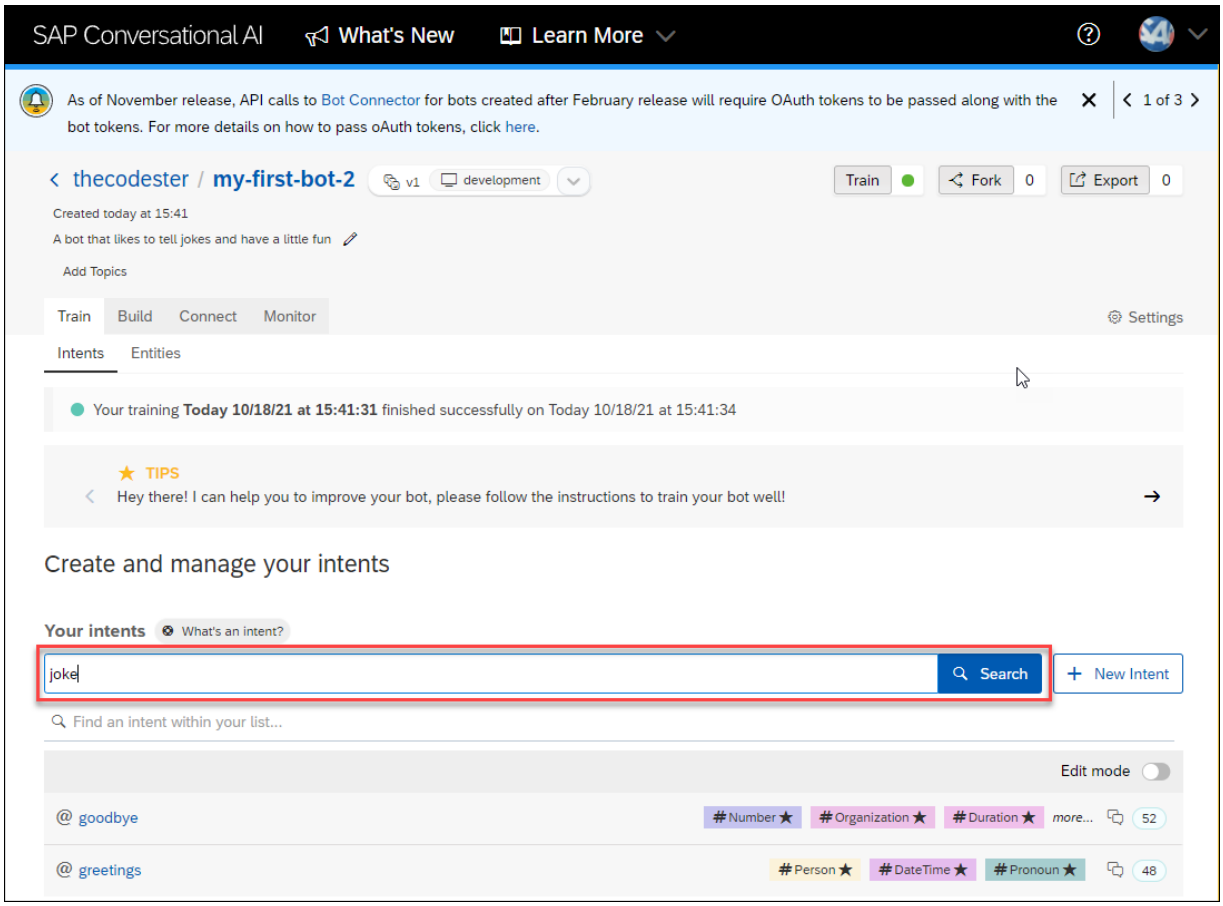
Step 4

Fork Intent

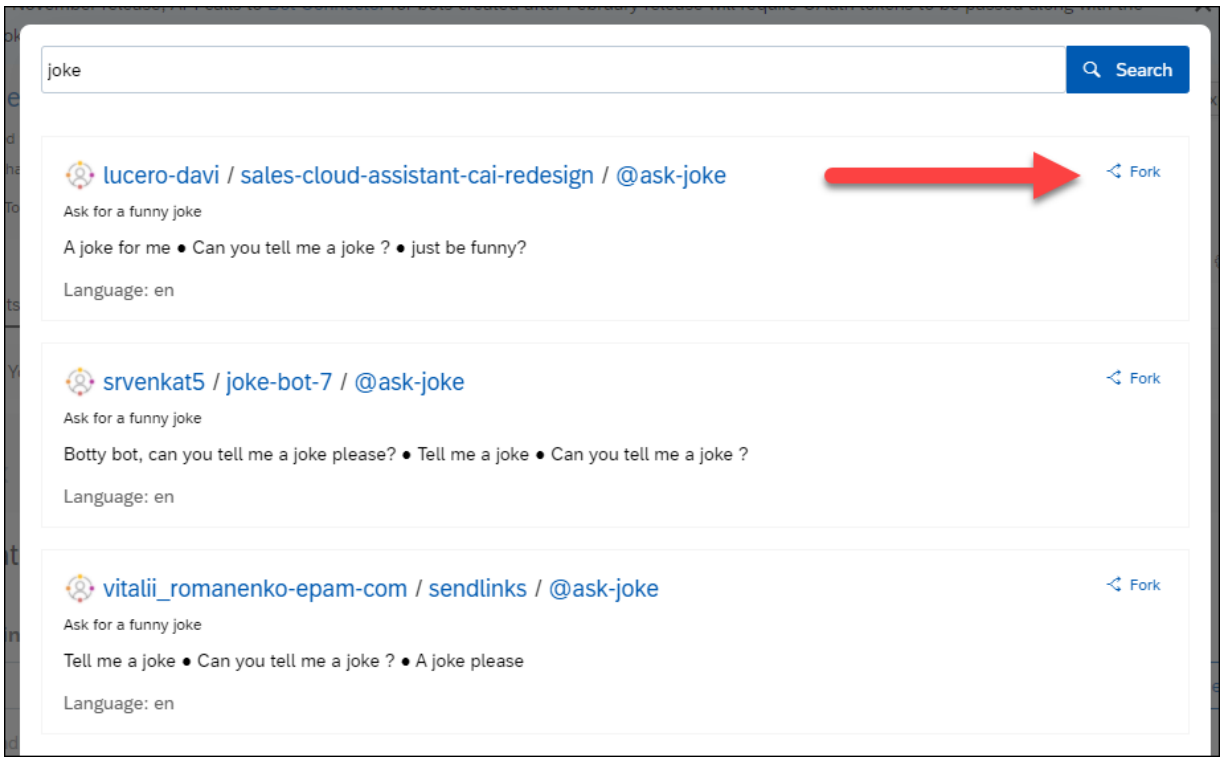
Now we will work in the **Train** tab, where will build intents. Remember, intents are the things people want to tell our bot. Within an intent, we indicate all the possible expressions that a person might use to communicate that intent.

As SAP Conversational AI is collaborative, you do not have to re-create each intent every time. You can "fork" an intent someone has already created and clone it right into your bot.

1. Go to the **Train** tab.
2. In the **Search** box, enter **joke**, and press **Search**.



You will get a list of intents to fork.

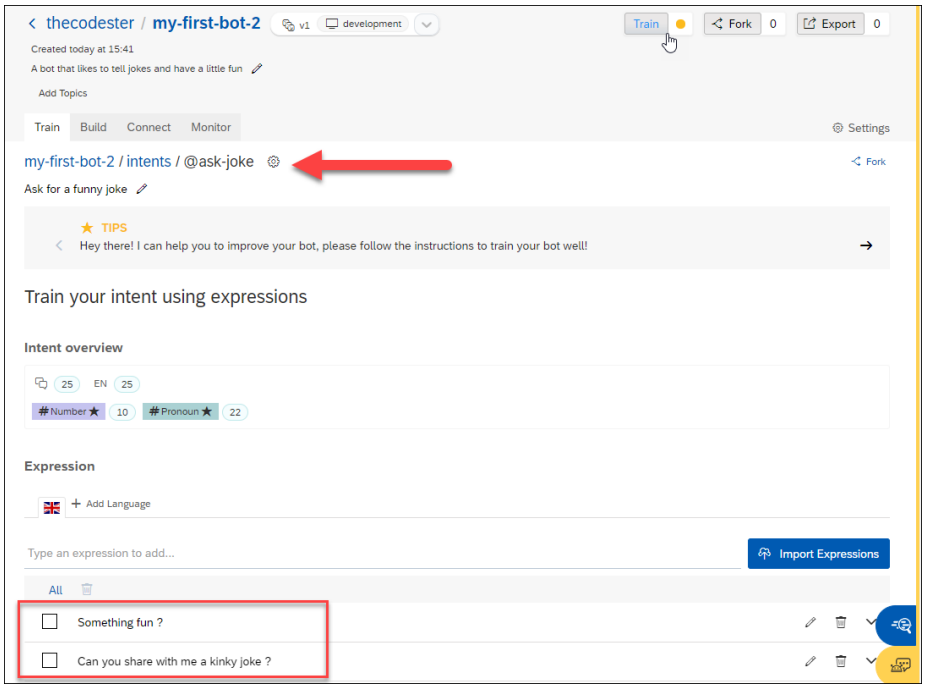


- Click **Fork** for the first @ask-joke intent.

There are many existing intents and the choices are always changing.

- Explore the intent by clicking on it.

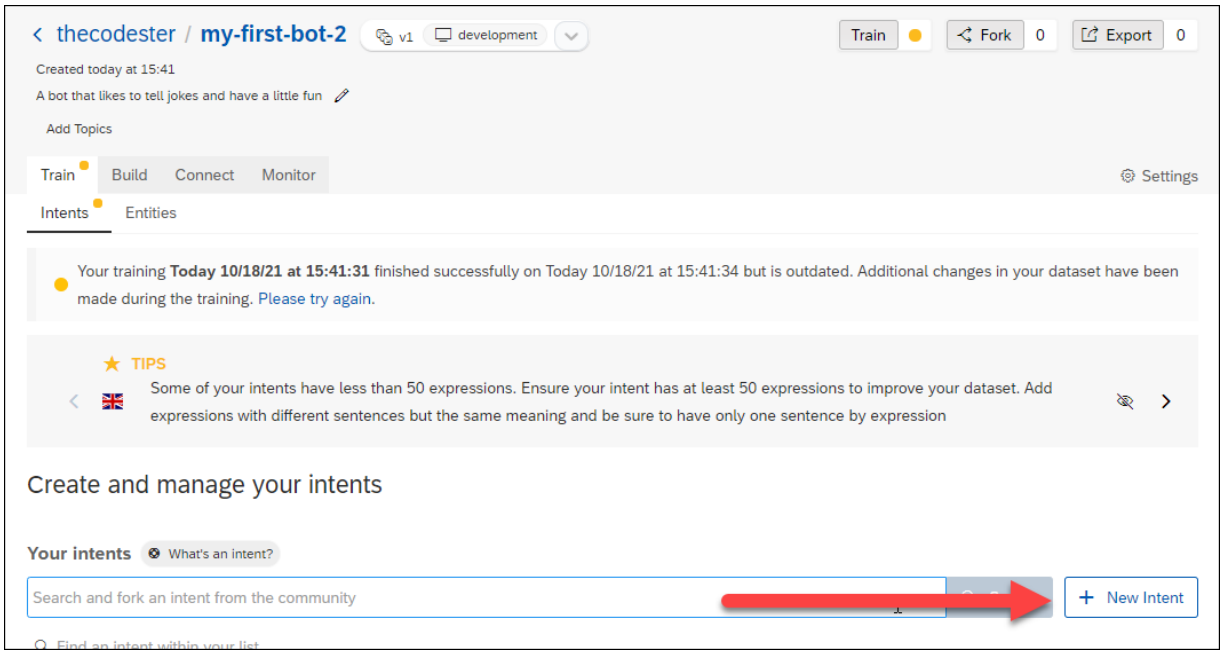
In the intent we forked, there are 2 expressions to discern if someone wants to be told a joke. In yours, you may receive more expressions.



Step 5
Create new intents

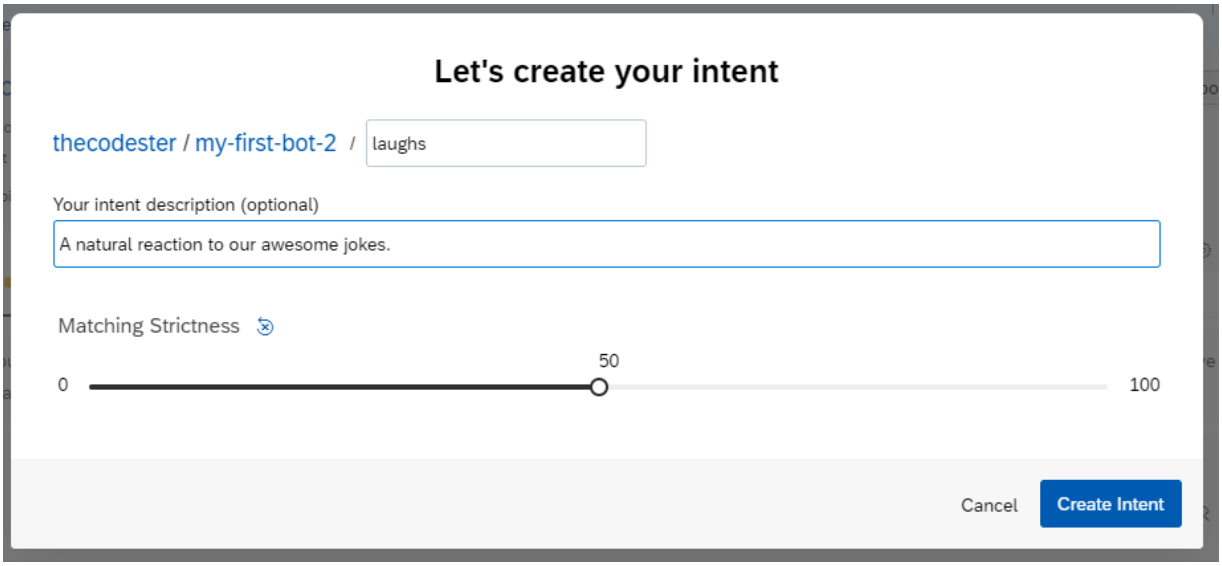
You will be able to reuse many, many intents created by others. But there are times you will want to create your own.

1. Click the **Train** tab.
2. Create an intent for recognizing good reactions to jokes.
    - Click **Create**.



- For the name of the intent, enter **laughs**.
- For the description, enter **A natural reaction to our awesome jokes**.

2.

3. 

   - Click **Create Intent**.
4. Create a second intent, this time for bad reactions to jokes.
   - Click **Create**.
   - For the name of the intent, enter **lame**.

- For the description, enter **You can't succeed every time**.
- Click **Create Intent**.

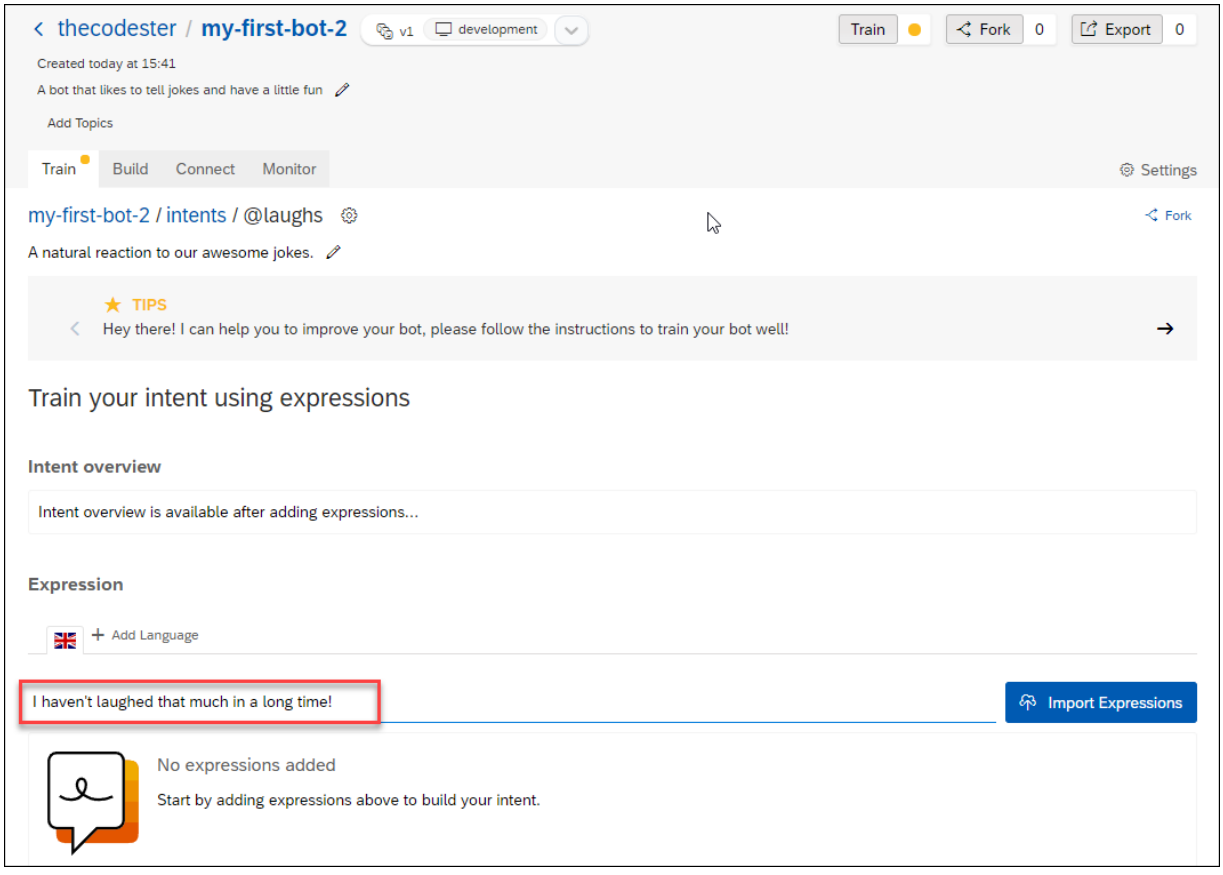
Step 6

Now that you have created 2 intents, we need to add expressions – things people might say if they had that intent. A golden rule would be to add **at least 30 expressions** to an intent, and ideally more than 50.

Put yourself in the shoes of the people talking to your bot. What could they possibly ask?

For this tutorial, you do not have to come up with 50 expressions but you will add 4 to each intent.

1. Click on the @laughs intent.
2. In the expression field, enter a sentence you want your bot to understand, then press **Enter**.



Here are some examples for the laughs intent you added:

Hahaha that's hilarious
ROFL you're good!
That, my friend, was an amazing joke.
I haven't laughed that much in a long time!

 For a production bot, you want 30-50 expressions, all the ways someone is likely to express their intent. In addition, after deploying the bot, you can review what users are writing and add those.

 Do the same for the @lame intent (you'll have to navigate to the intent by click **Train** tab or navigating the breadcrumb).

Here are some examples for the lame intent:

You have no sense of humor whatsoever.
That's both terrible and offensive.
What the heck was that?
Try harder, that was a very bad joke.

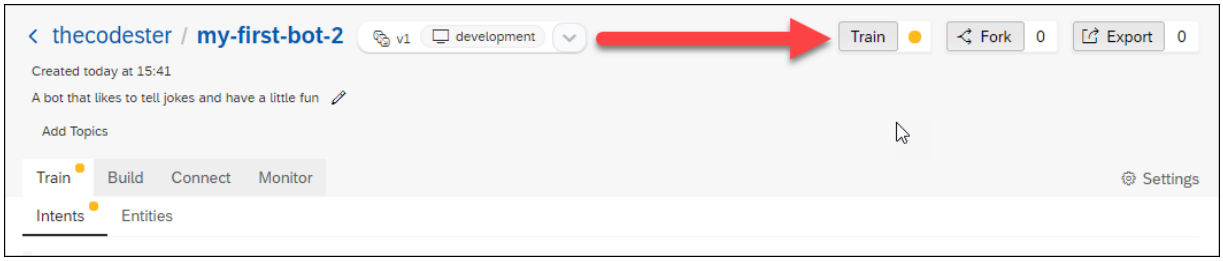
3.

Step 7

We want to now test to see if the bot can detect when a user says something matching the intent we created.

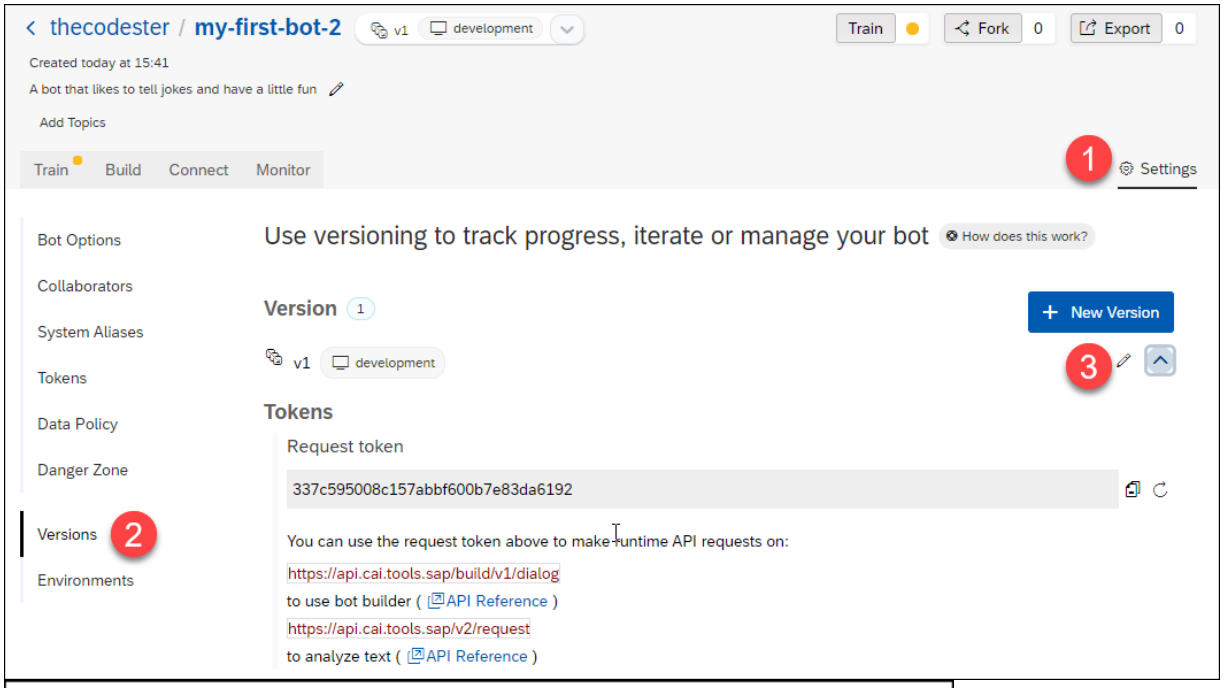
1. Click the **Train** button at the top-right, so that the bot trains itself based on the latest intents and expressions (sometimes referred as your dataset).

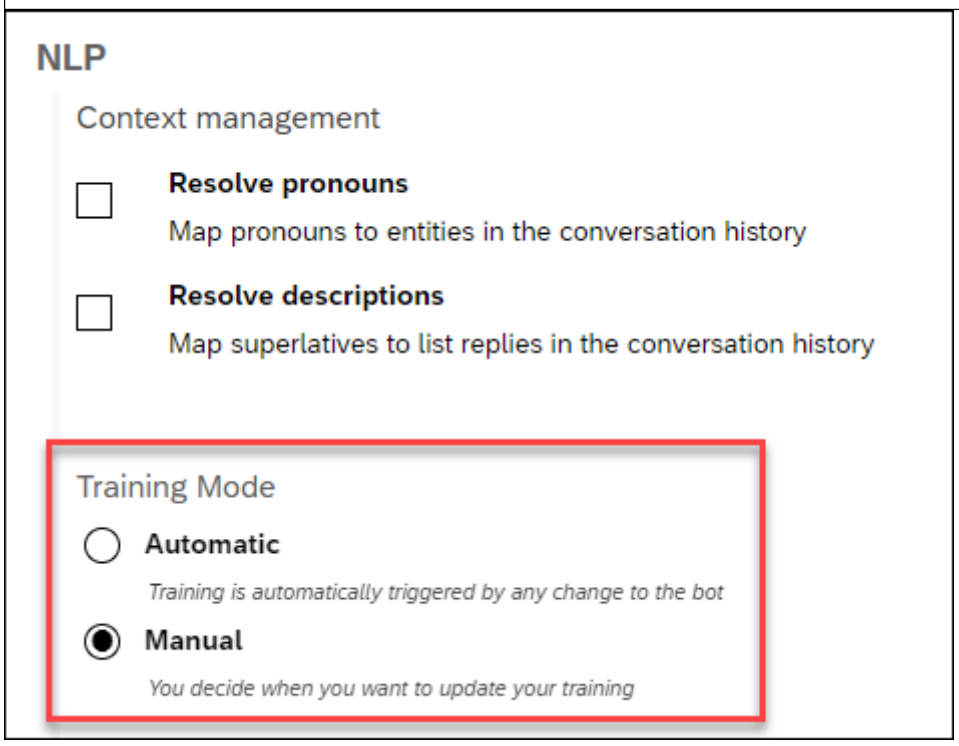
   The button indicates the current status of training: Yellow is out of date, blue is training, green is up to date..



By default, your bot does not train itself but you must click Train for it to understand the latest intents and expressions.

You can change the setting so the bot automatically trains itself at **Settings** > **Version** > **(current version dropdown)**, and scroll down to .





- At the bottom of the panel, open the **Expression Analysis** tab.

Enter something that you think should match the intent, like:

Text
Copy
Botty bot, can you tell me a joke please?

If the intent is successfully trained, the test will show that the bot recognized the intent (as well as entities within the expression).

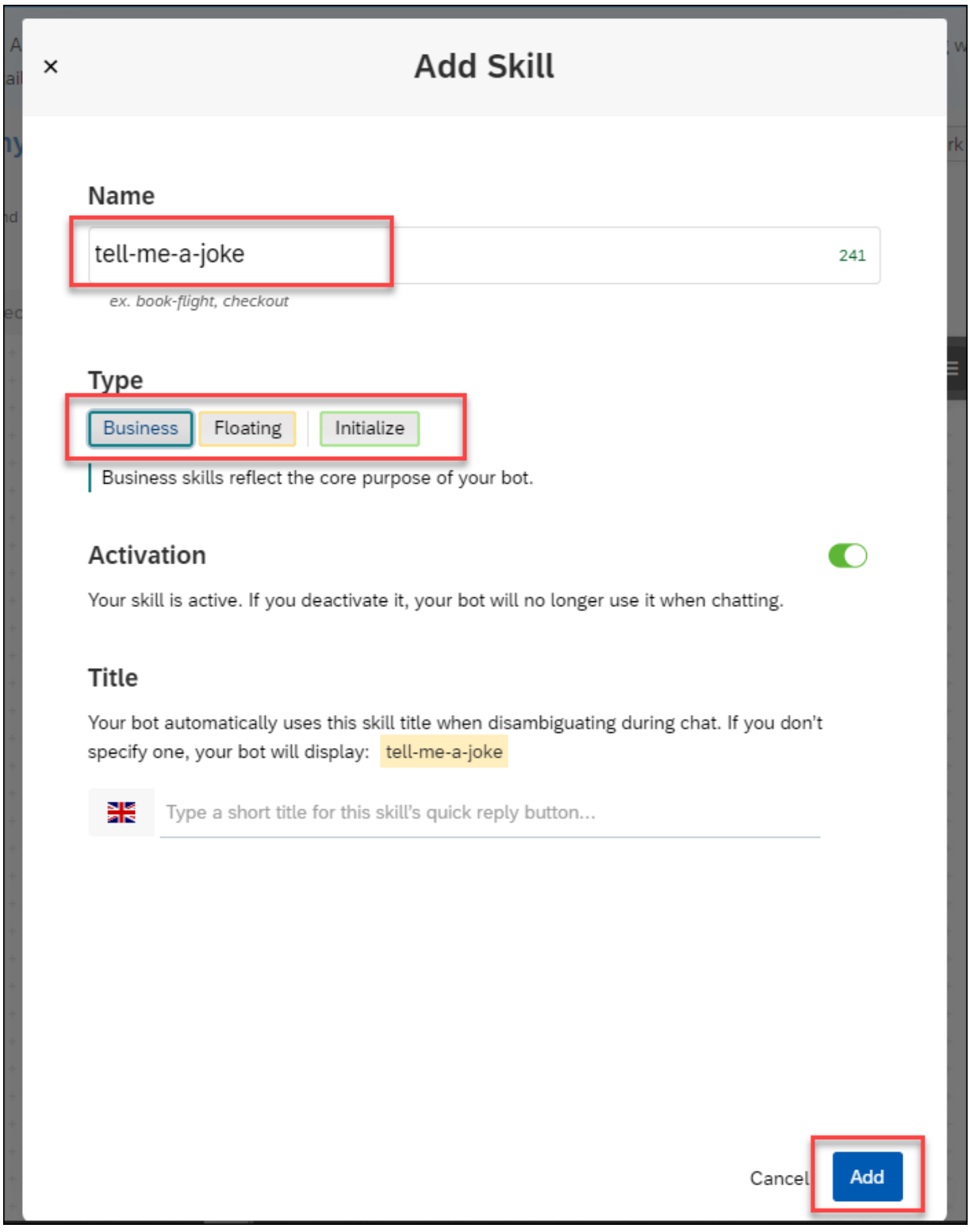
Step 9

Now let's build a skill to tell a joke.

1. Go back to the **Build** tab – where you see the canvas.
2. Click **Add skill**.

   There are several types of skills:

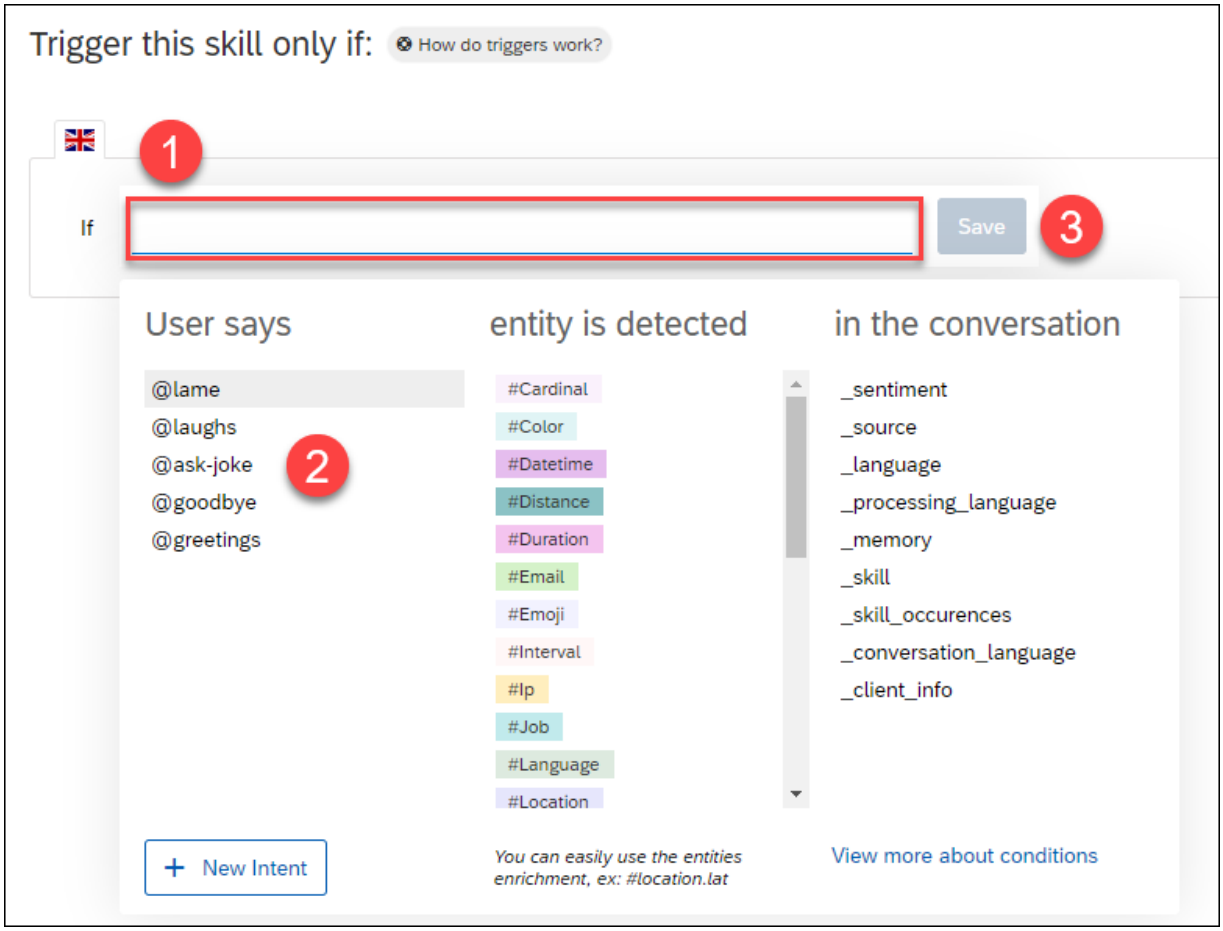
   - **Business** skills reflect the core purpose of your bot.
   - **Floating** skills complement your bot's core business skills (for example, small talk).
   - The **Initialize** skill (there can only be one in a chatbot) is triggered when the conversation with the user starts.
3. Call your skill **tell-me-a-joke**, set the type as **Business**, and click **Add**.



- Click the new **tell-me-a-joke** skill, and open the **Triggers** tab.

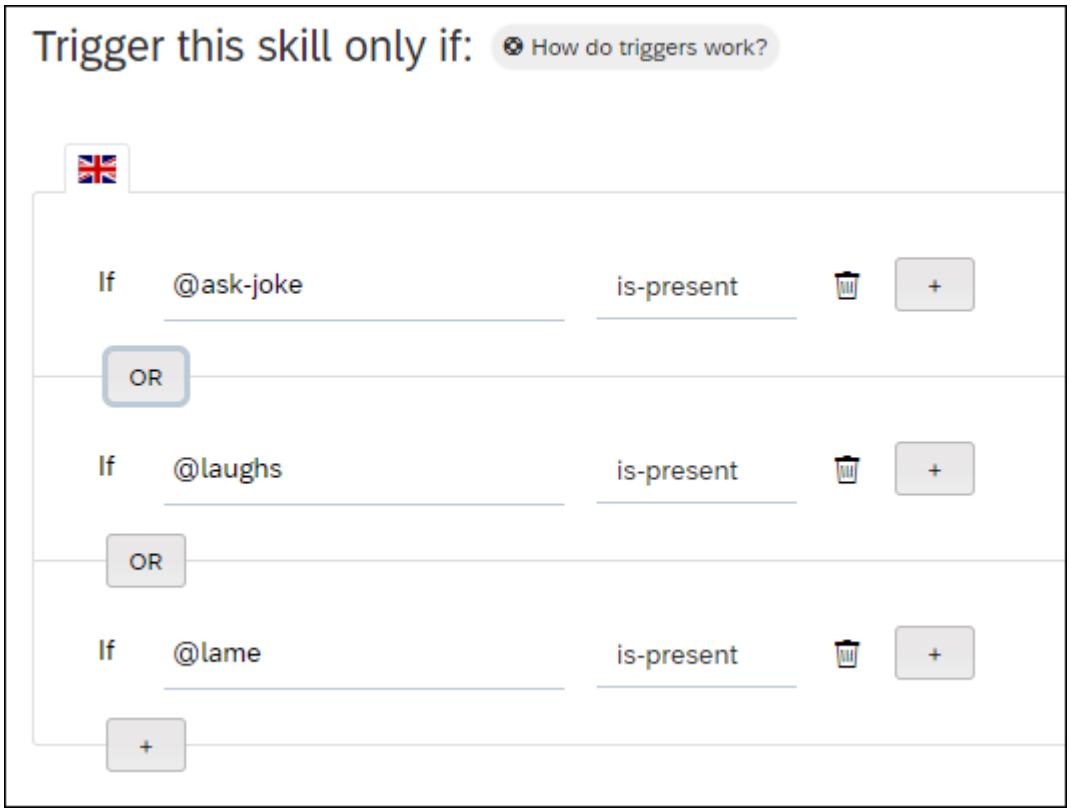
- Add 3 **If** statements, one for each of the @ask-joke, @laugh, and @lame intents.

  - Click in the empty space right after If, select the @ask-joke intent, and then click **Save**.



  - Click on the + sign – **Add a new list of conditions** – and repeat the above for @laughs and @lame intents.
  - Change the logical conditions to Or between each If statement, simply by clicking And.

It should now look like this:



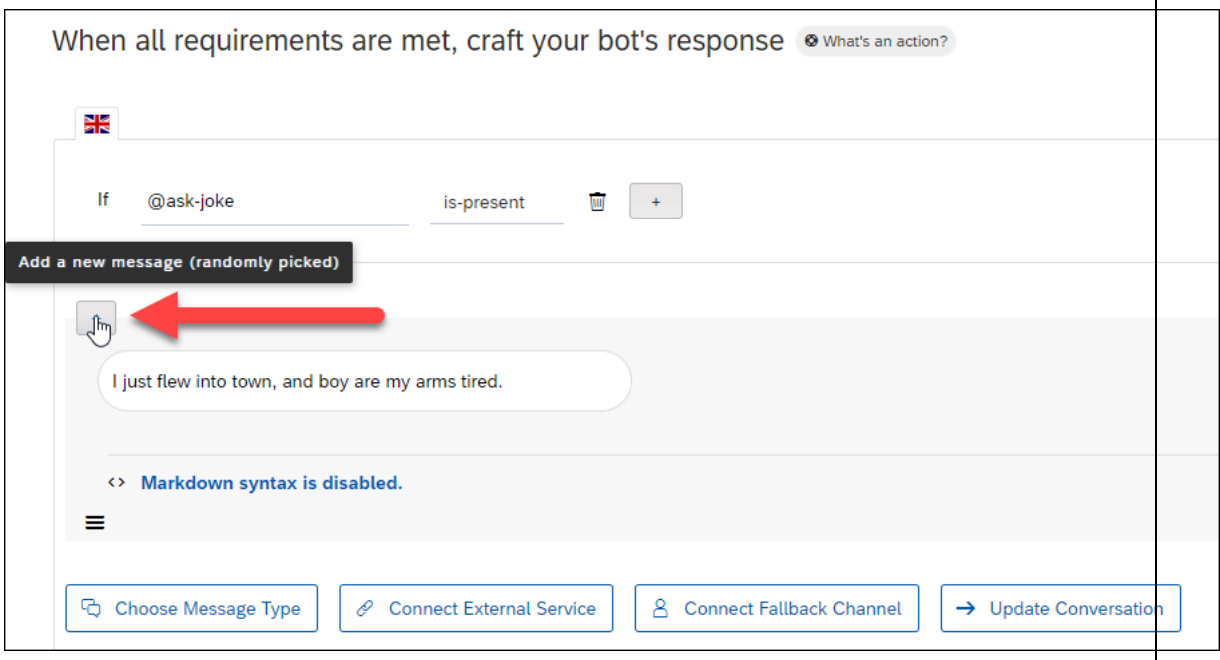
- Go to the **Actions** tab.
- Click **New Action Group**, then **Add Condition**.

After the If, select the @ask-joke intent, and click **Save**.

- Click **Choose Message Type**, choose the **Text** format, and type in a really good joke, like:

Joke
Copy
I just flew into town, and boy are my arms tired.

You can define additional jokes, and one of the jokes from the set will be displayed randomly.

8. 

   You can also set a delay (optional) between two messages, up to 5 seconds. This might be useful when the messages your bot sends are quite long and need time to be read by the user.

9. Click **Save**.

Step 10

Before releasing your bot to your audience, chat with it in real situation. This will help show you how the bot will behave in a "real" conversation.

Double-check if there are any errors in the conversation flow (e.g., bad answers, fallback errors, weird behavior) and correct them before sending the bot out.

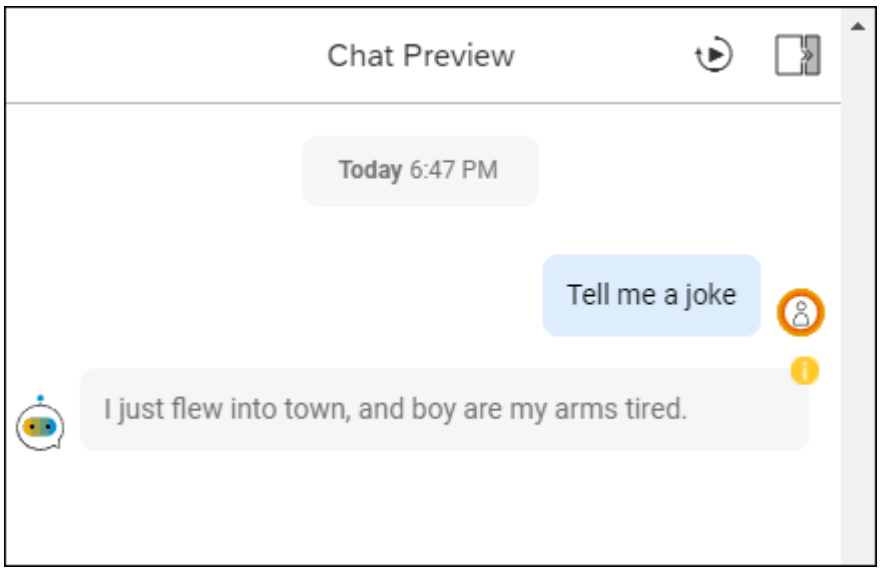
Chat with your bot as often as you can while you build it, which will make it easier to find problems.

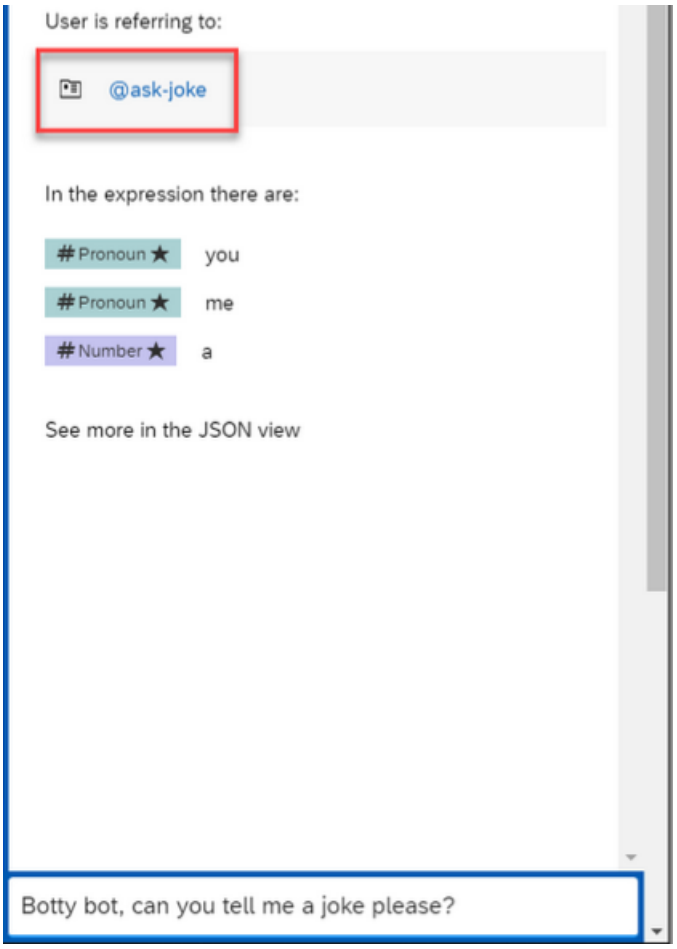
Click on the bottom-right yellow button **Chat Preview** and start sending messages. Use the clear icon

at the top of the panel to refresh the chat.

- If you say **Tell me a joke**, the bot will tell you a joke.

- Try other phrases and see how the bot responds. If the bot does not respond properly, then try improving your intent with additional expressions.



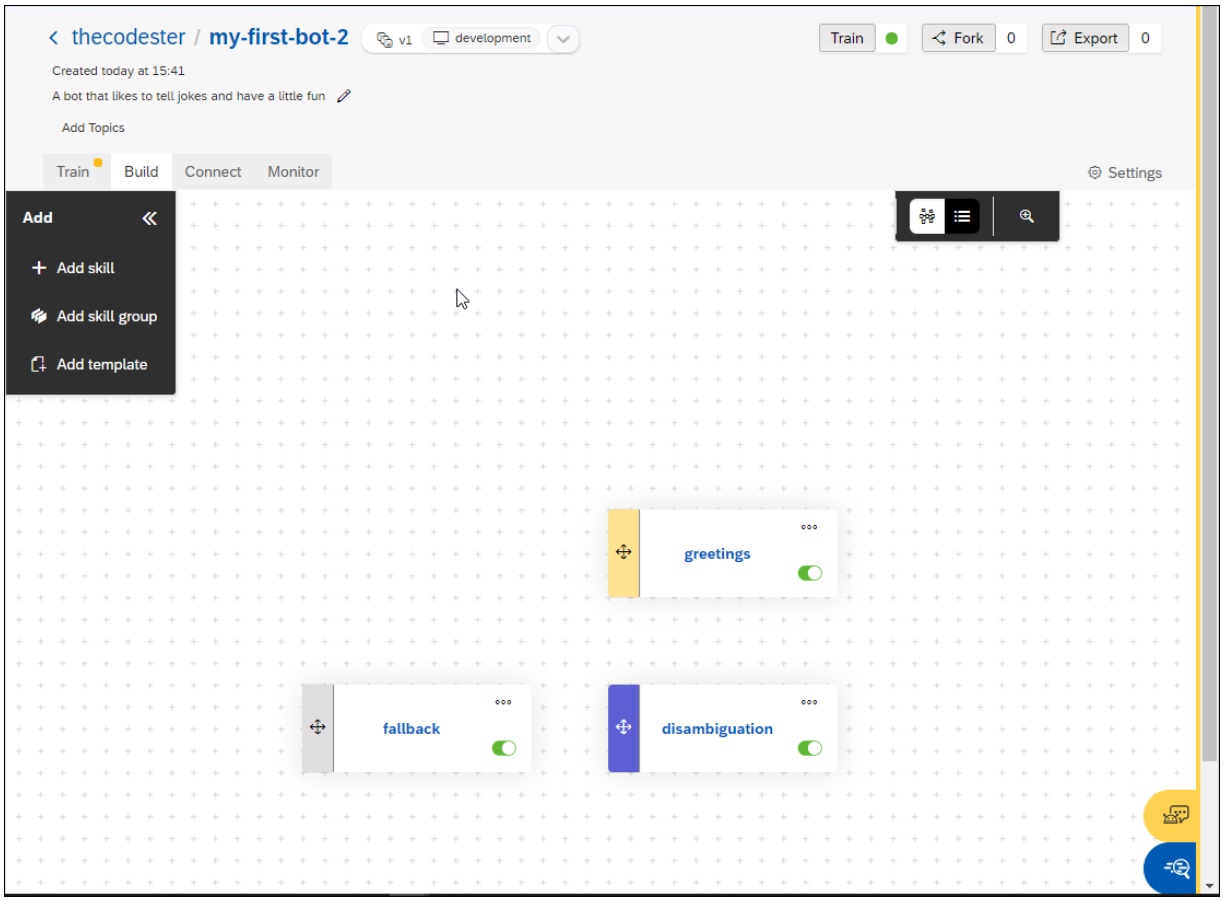
3. 

If the bot did not recognize the intent (or selected the wrong intent), you will need to train your bot some more by adding additional expressions (Step 6).

Step 8

Now that your bot knows how to understand people who talk to it, it's time to give your robot some skills.

Open the **Build** tab.



Inside, you'll find the **Bot Builder**, which helps you construct the conversation flow of your bot.

What is a skill?

Each skill represents one thing that your bot knows how to do. Your skill can be complicated (e.g., manage payment by credit card) or quite simple (e.g., answer a basic question).

Just like intents, you can create a skill from scratch or inherit skills from other bots you've created or from other people's bots.

In our project, choose the predefined skill **Greetings**.

The **Greetings** skill – like all skills – has 4 tabs:

- **README.md:** A description of the purpose of your skill
- **Triggers:** The conditions that must occur – generally the intents that the user must express – for the skill to be executed

- **Requirements:** Information that must be collected in order for the skill to be executed
- **Actions:** The action to take (basically, this is the skill)

If you navigate through the tabs, you'll see that the **Greetings** skill is structured as follows:

- It is triggered if either the intent @greetings or @goodbye is matched.
- It has no requirements because it does not need to collect additional information. That means that it will execute actions directly after being triggered.
- It has two possible actions: If the @greetings intent is matched, it sends a random welcoming message chosen from a list. If the @goodbye intent is matched, it does the same thing, but picks the message from a different list

Conclusion: In this practical we learned how to create a chatbot using SAP tool online.

Practical – 6

Various Tools for ML Techniques

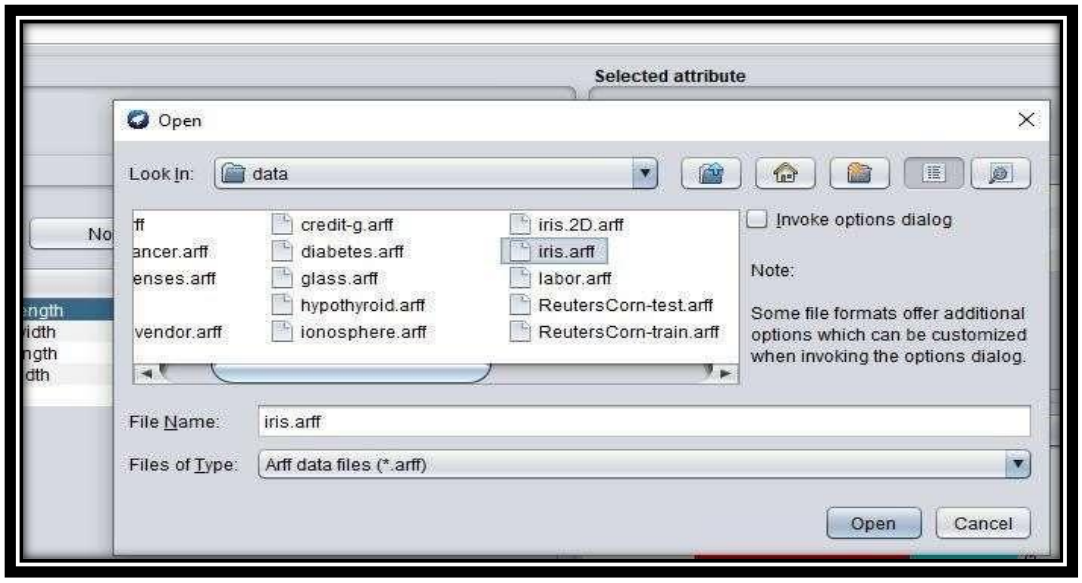
6.1: Perform classification on Iris dataset using neural network tools such as WEKA, ORANGE

Output:

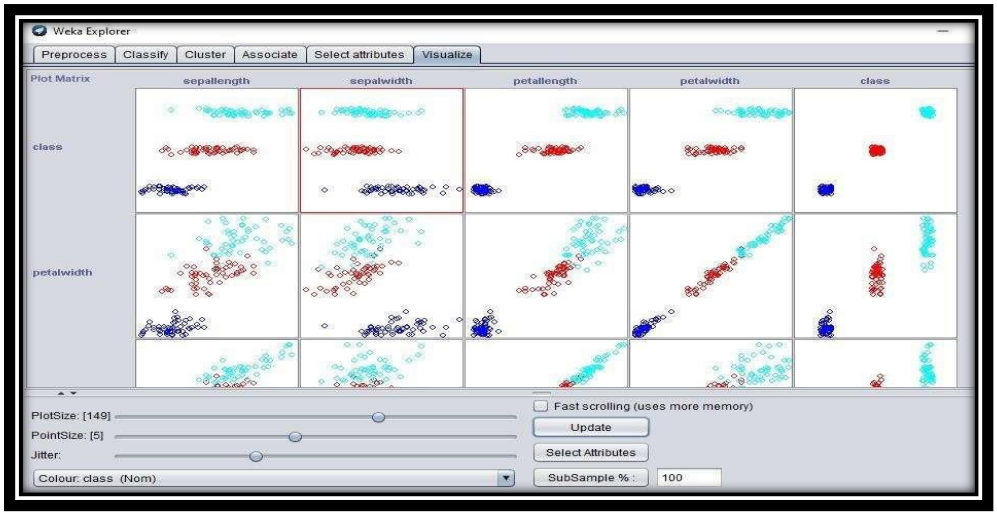
Using WEKA

Classification on Iris Dataset:

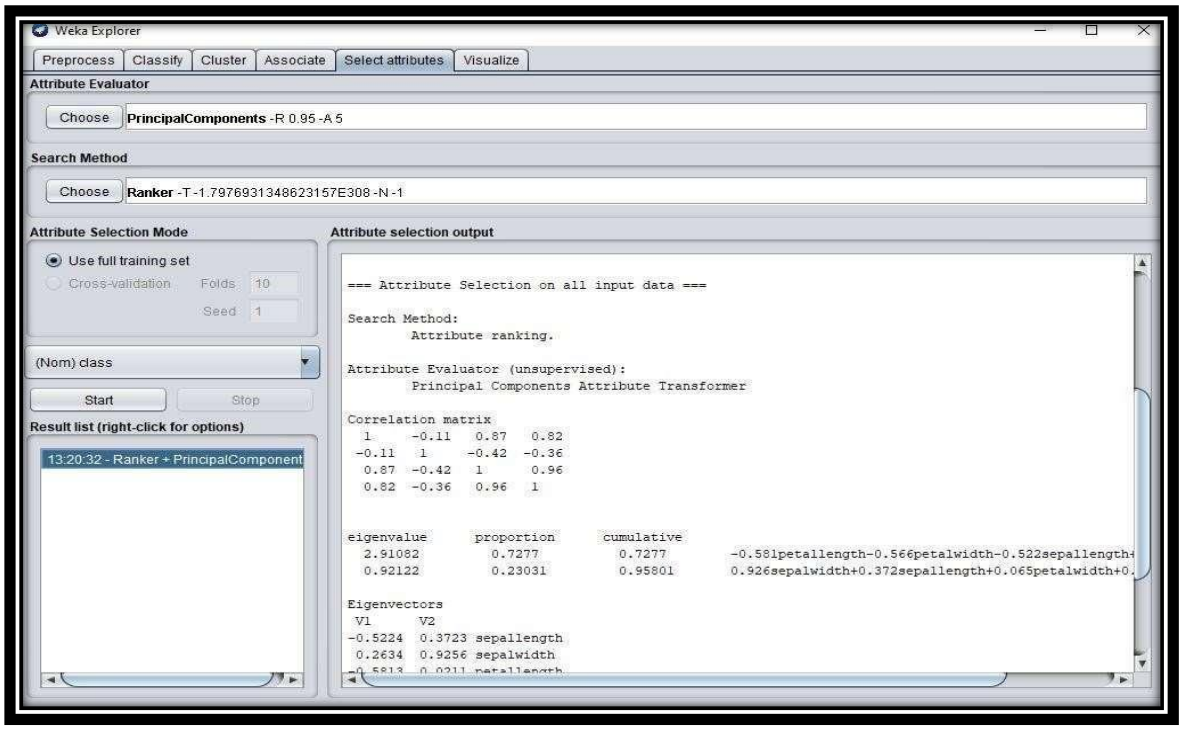
1. Loading the dataset



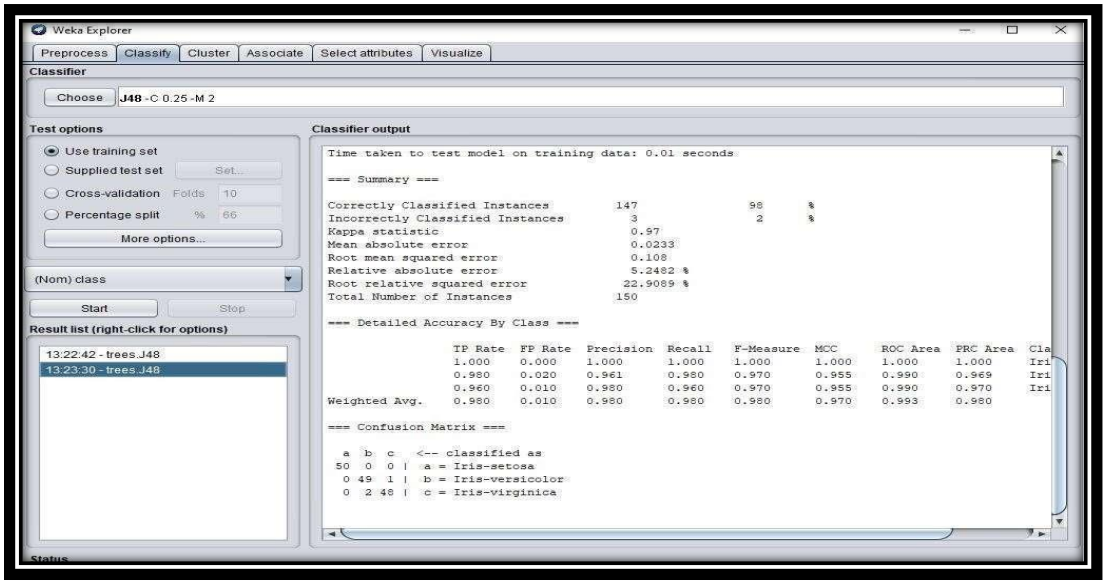
2. Editing the dataset



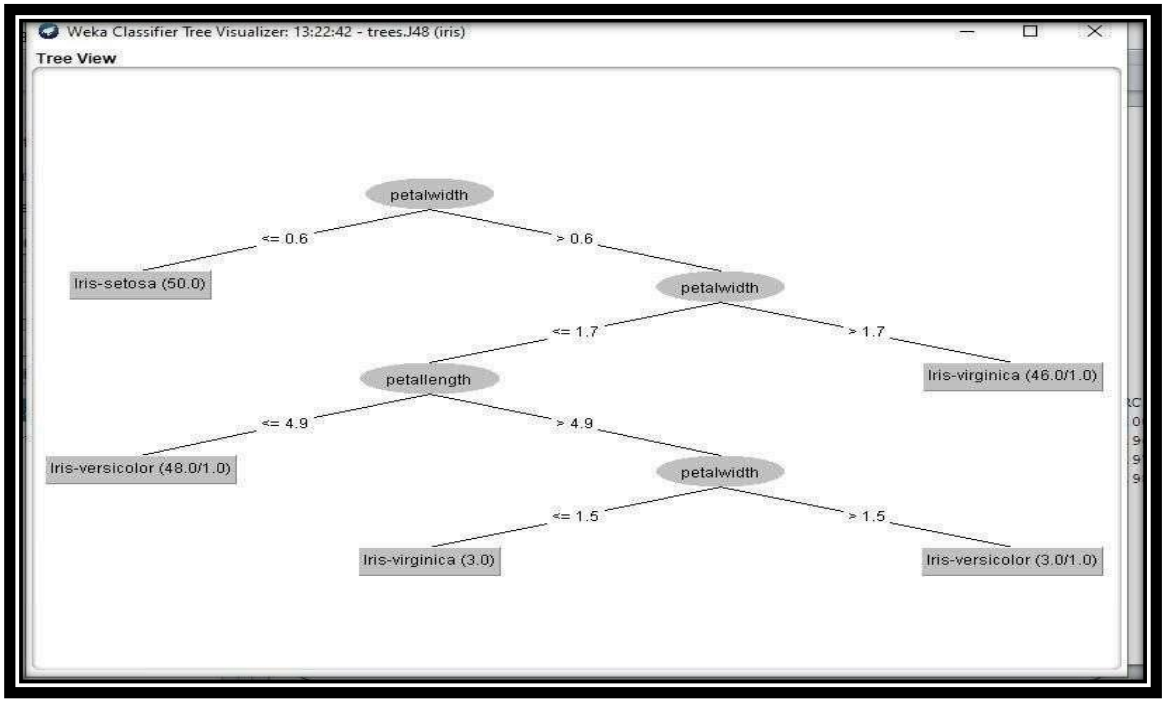
3. Selecting the Attributes



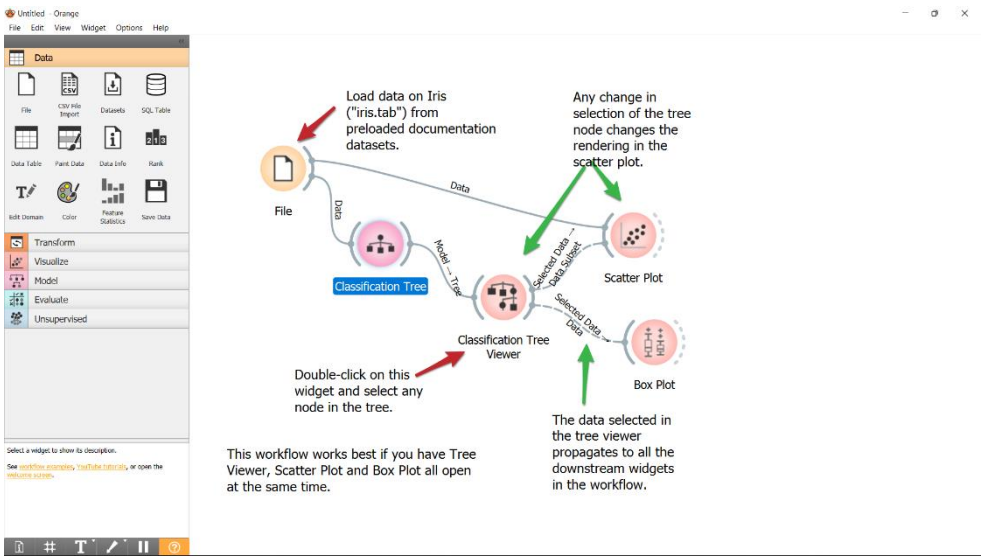
4. Classifying the dataset - J48 – C 0.25 – M 2

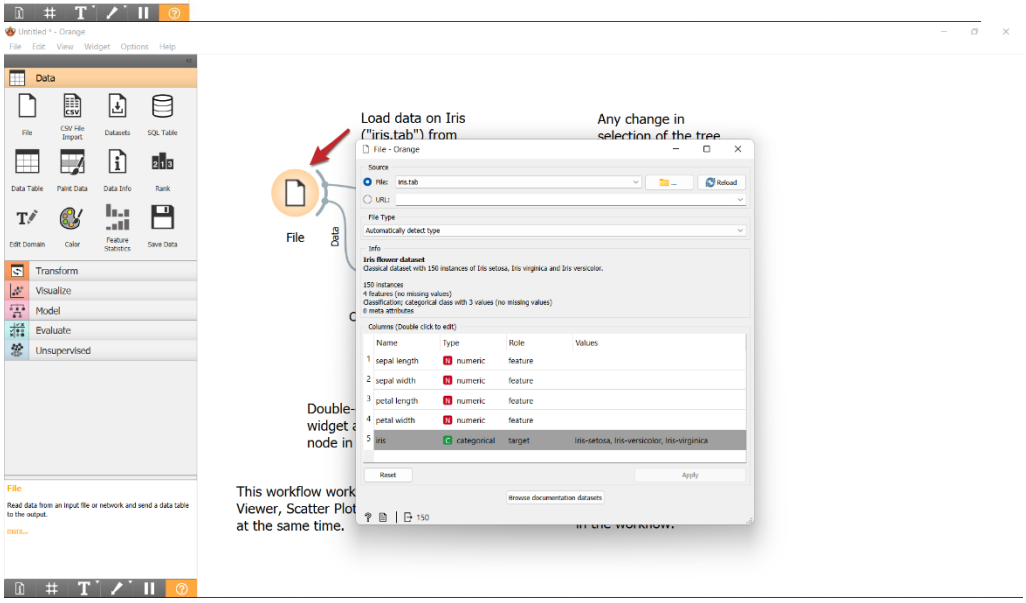


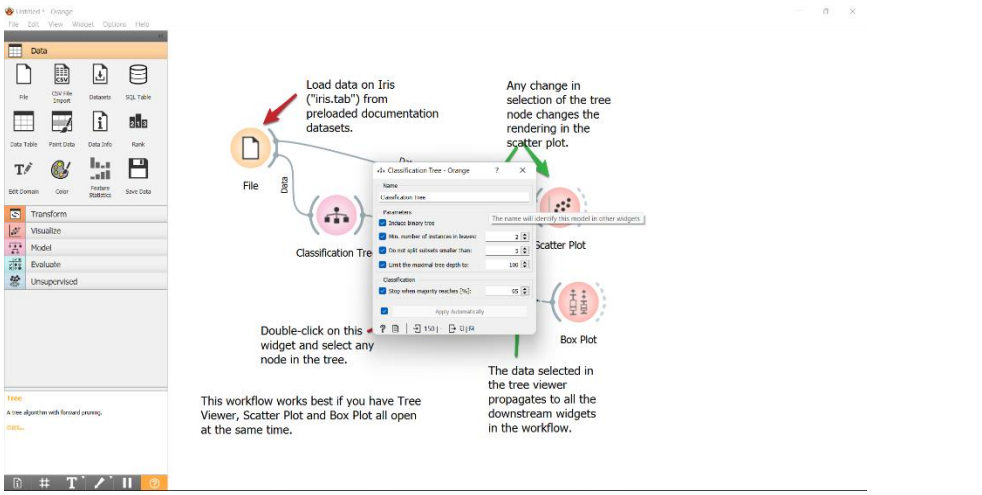
5. Classification of Iris Dataset in Tree View

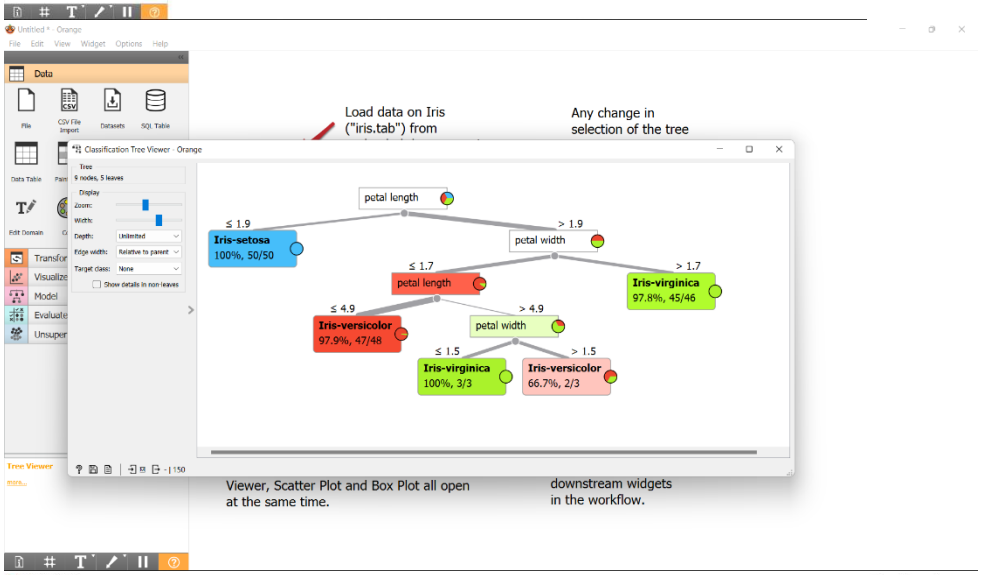


Using Orange:

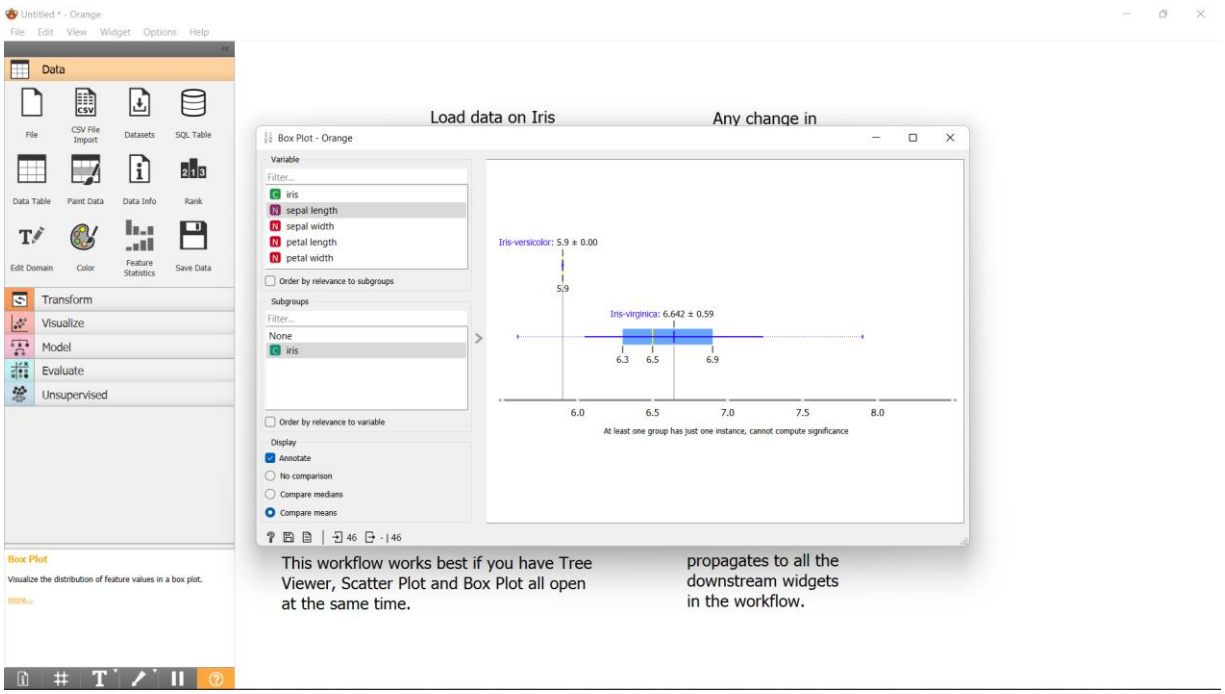


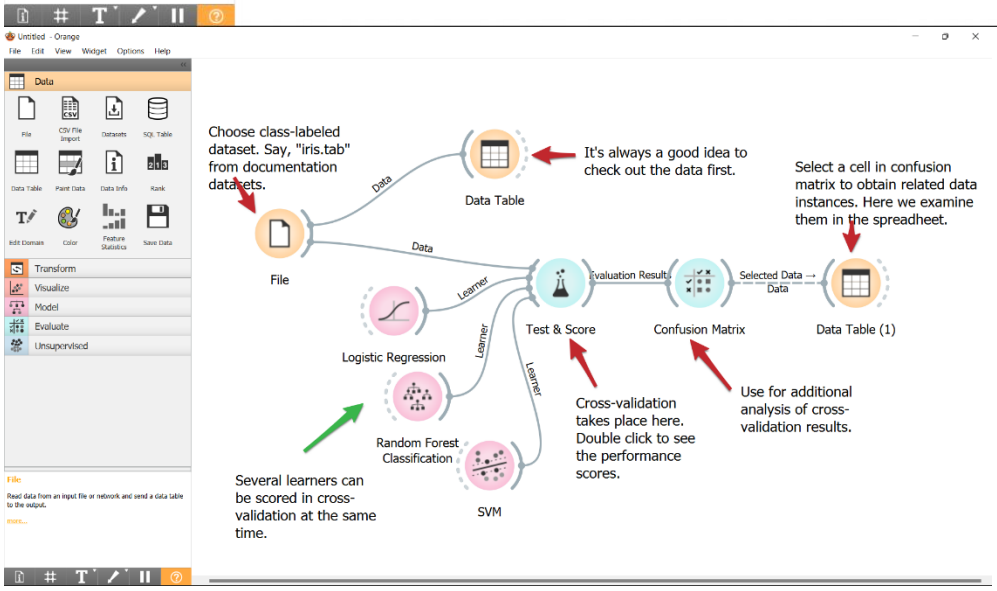


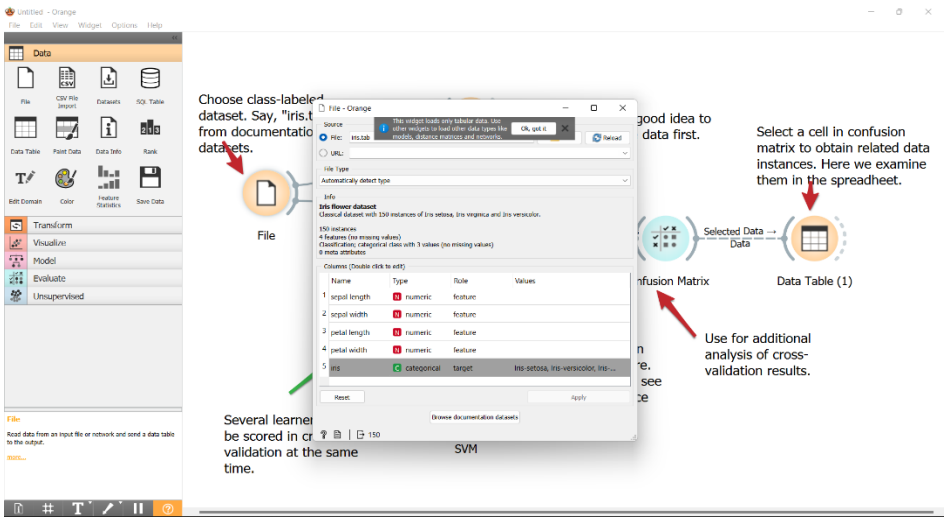

Load data on Iris ("iris.tab") from preloaded documentation datasets.
Any change in selection of the tree node changes the rendering in the scatter plot.
File
Classification Tre
Scatter Plot
Box Plot
Double-click on this widget and select any node in the tree.
The data selected in the tree viewer propagates to all the downstream widgets in the workflow.
This workflow works best if you have Tree Viewer, Scatter Plot and Box Plot all open at the same time.

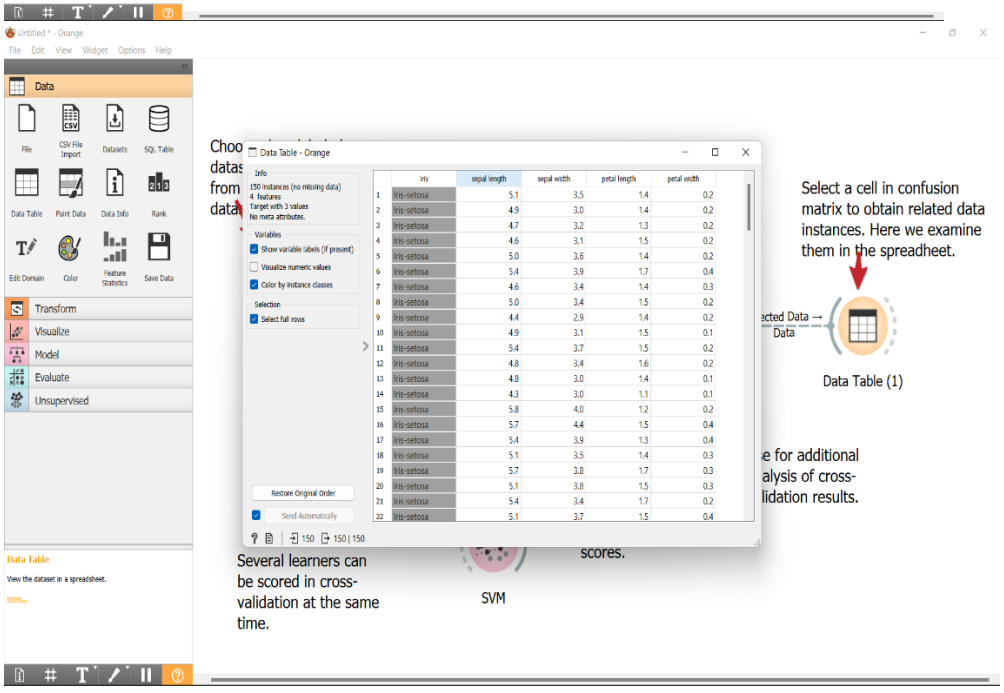

Load data on Iris ("iris.tab") from
Any change in selection of the tree
petal length
≤ 1.9
> 1.9
Iris-setosa
100%, 50/50
petal width
≤ 1.7
> 1.7
petal length
Iris-virginica
97.8%, 45/46
≤ 4.9
> 4.9
Iris-versicolor
97.9%, 47/48
petal width
≤ 1.5
> 1.5
Iris-virginica
100%, 3/3
Iris-versicolor
66.7%, 2/3
Viewer, Scatter Plot and Box Plot all open at the same time.
downstream widgets in the workflow.

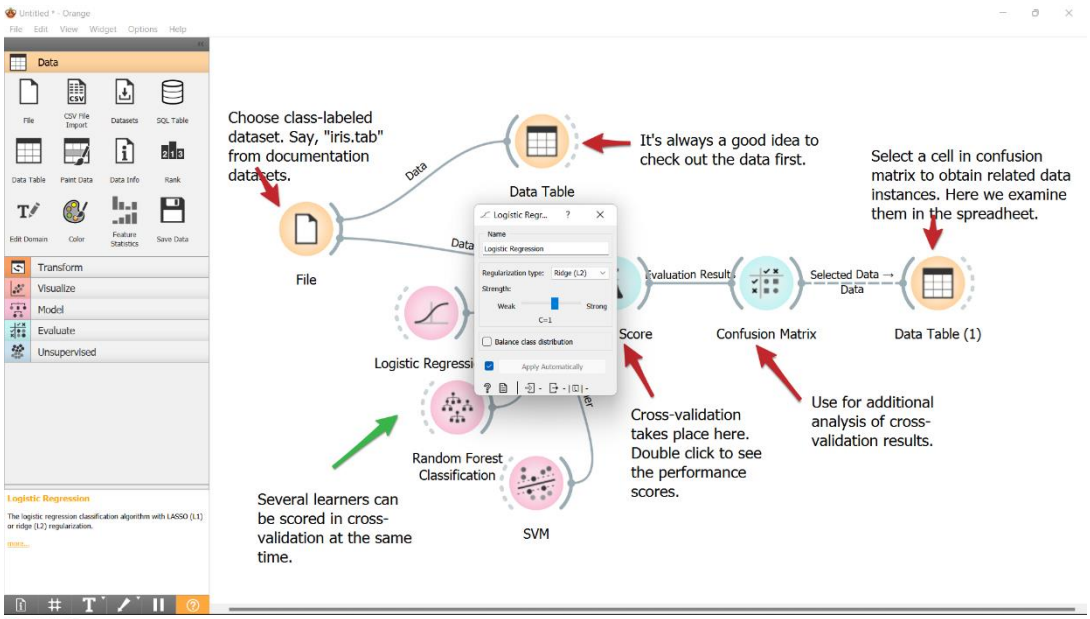

Viewer, Scatter Plot and Box Plot all open at the same time.
downstream widgets in the workflow.

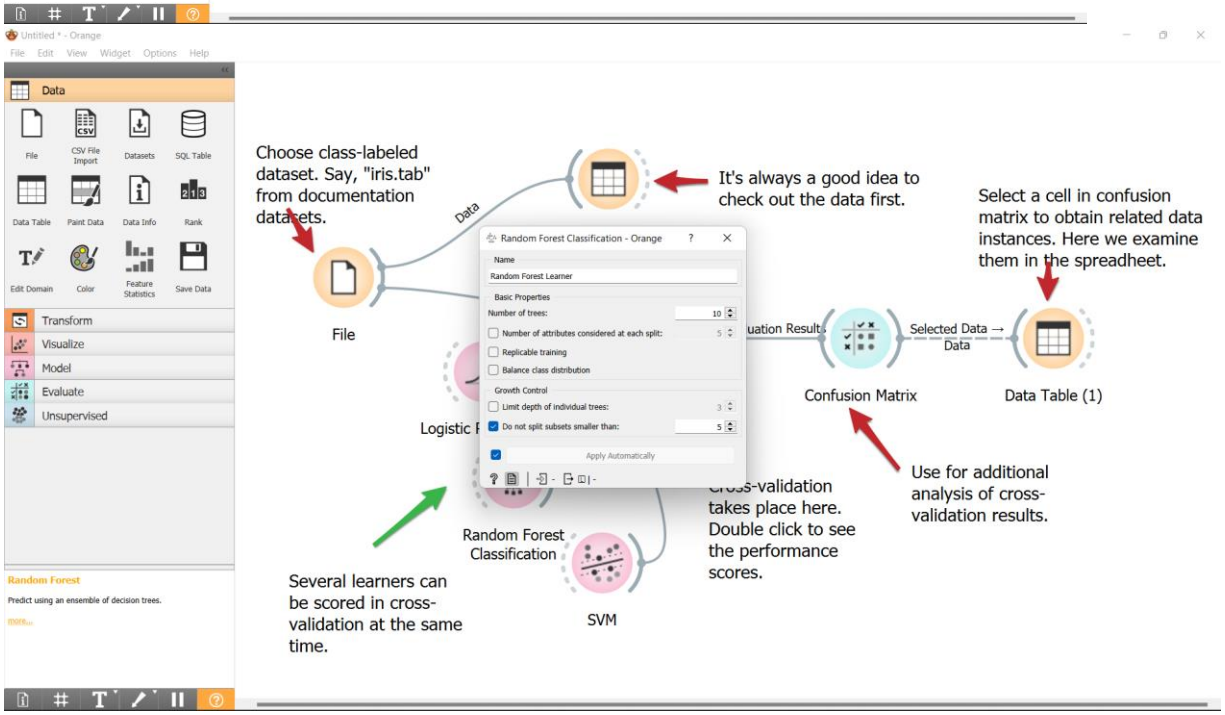

Load data on Iris
Any change in
Box Plot - Orange
Iris-versicolor: 5.9 ± 0.00
Iris-virginica: 6.642 ± 0.59
At least one group has just one instance, cannot compute significance
This workflow works best if you have Tree Viewer, Scatter Plot and Box Plot all open at the same time.
propagates to all the downstream widgets in the workflow.

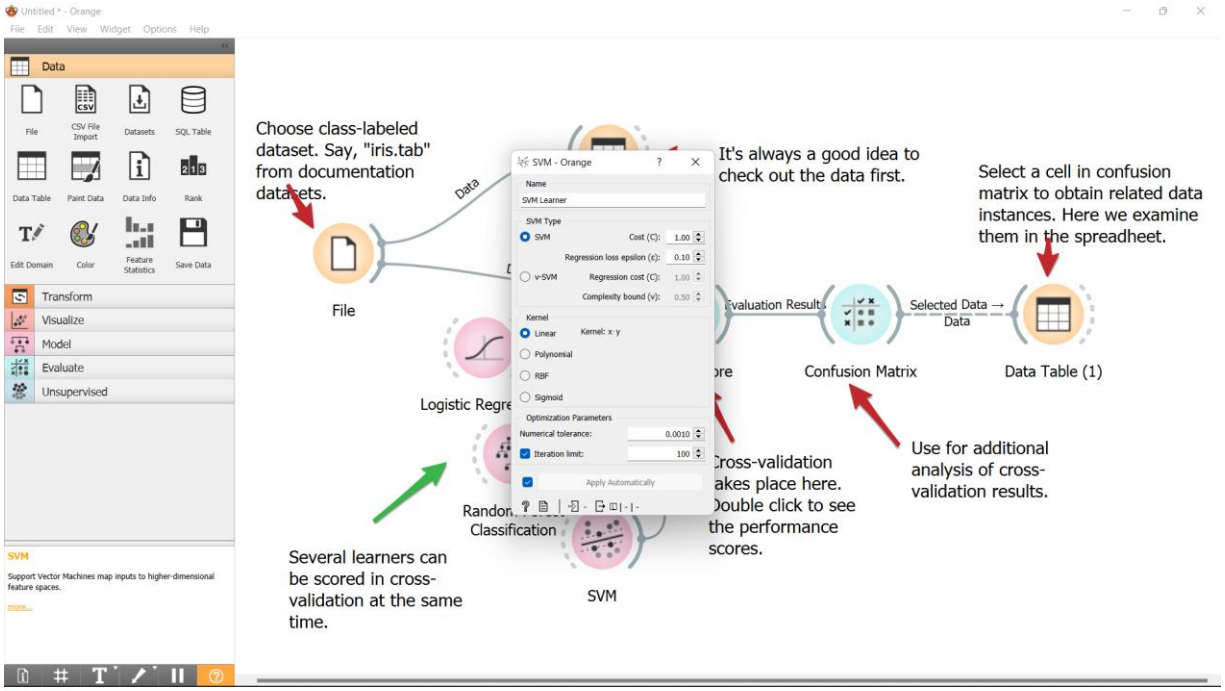

Choose class-labeled dataset. Say, "iris.tab" from documentation datasets.
It's always a good idea to check out the data first.
Data Table
File
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
Logistic Regression
Random Forest Classification
SVM
Test & Score
Confusion Matrix
Data Table (1)
Cross-validation takes place here. Double click to see the performance scores.
Use for additional analysis of cross-validation results.
Several learners can be scored in cross-validation at the same time.

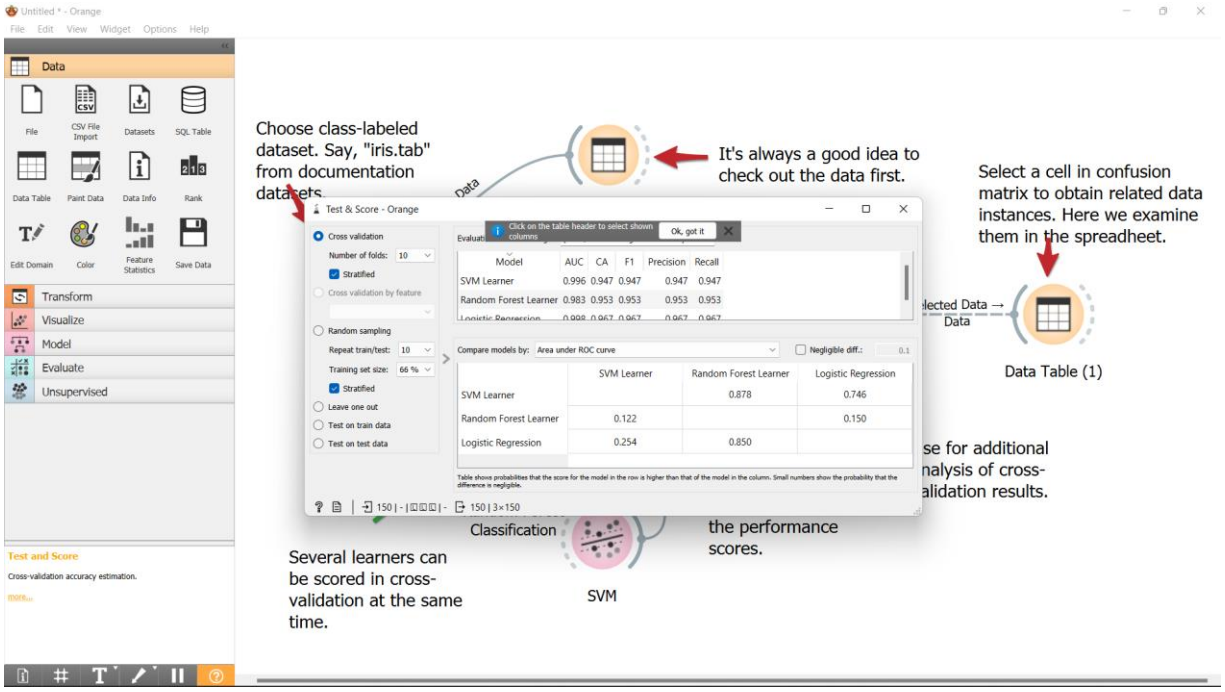

Data
Transform
Visualize
Model
Evaluate
Unsupervised
File
Choose class-labeled dataset. Say, "iris.t
from documentatio
datasets.
good idea to
data first.
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
Selected Data → Data
Data Table (1)
nfusion Matrix
Use for additional analysis of cross-validation results.
Several learne
be scored in cr
validation at the same time.
SVM

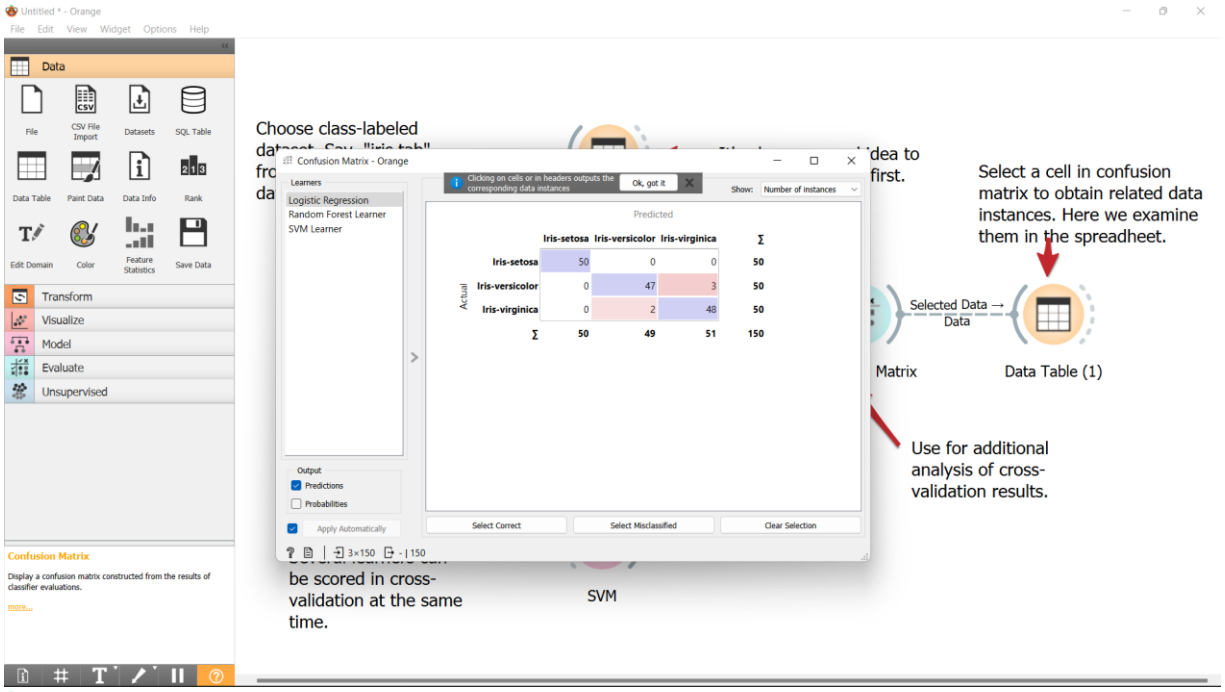

Data
Transform
Visualize
Model
Evaluate
Unsupervised
Choo
datas
from
data
Data Table - Orange
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
ected Data → Data
Data Table (1)
e for additional
alysis of cross-
lidation results.
scores.
Several learners can be scored in cross-validation at the same time.
SVM

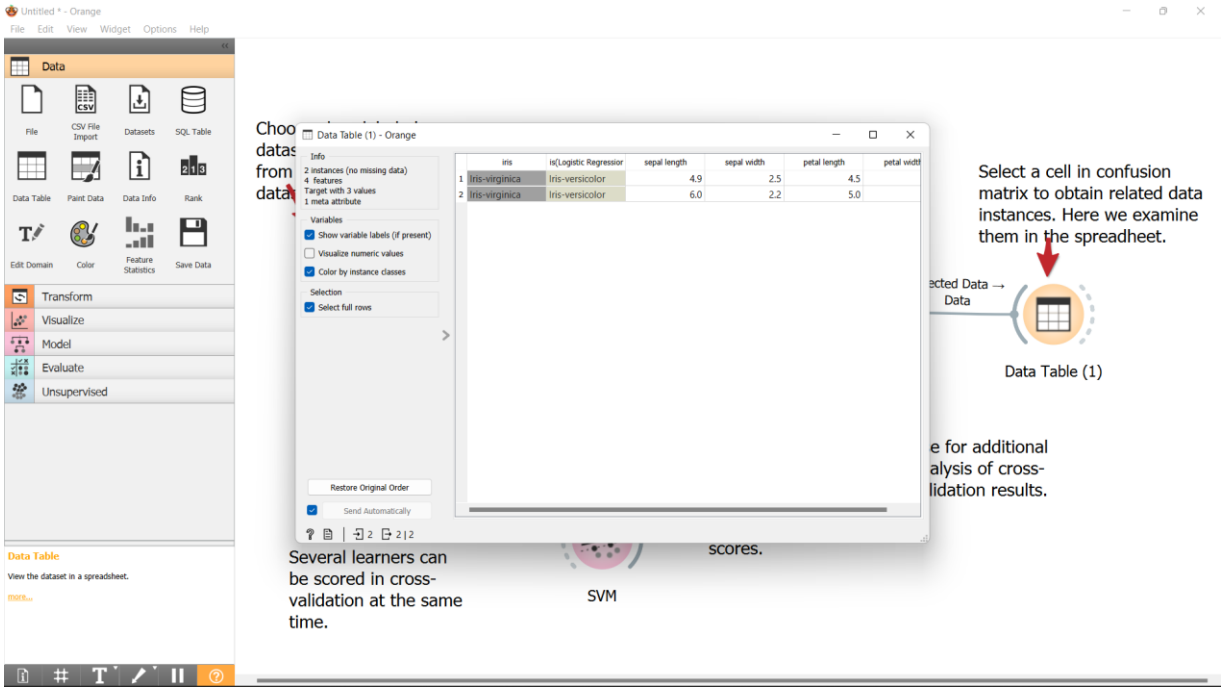

Choose class-labeled dataset. Say, "iris.tab" from documentation datasets.
It's always a good idea to check out the data first.
Data Table
File
Logistic Regression
Random Forest Classification
SVM
Score
Evaluation Results
Confusion Matrix
Selected Data
Data Table (1)
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
Cross-validation takes place here. Double click to see the performance scores.
Use for additional analysis of cross-validation results.
Several learners can be scored in cross-validation at the same time.

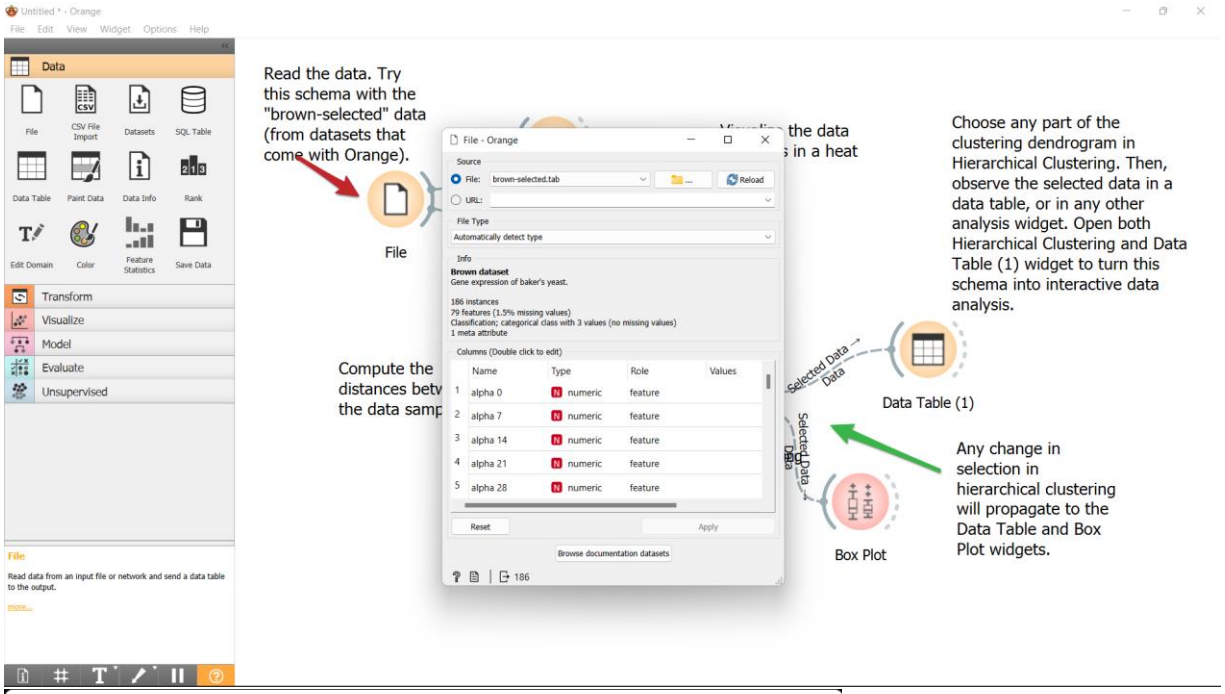

Choose class-labeled dataset. Say, "iris.tab" from documentation datasets.
It's always a good idea to check out the data first.
File
Random Forest Classification
SVM
Confusion Matrix
Data Table (1)
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
Use for additional analysis of cross-validation results.
Several learners can be scored in cross-validation at the same time.

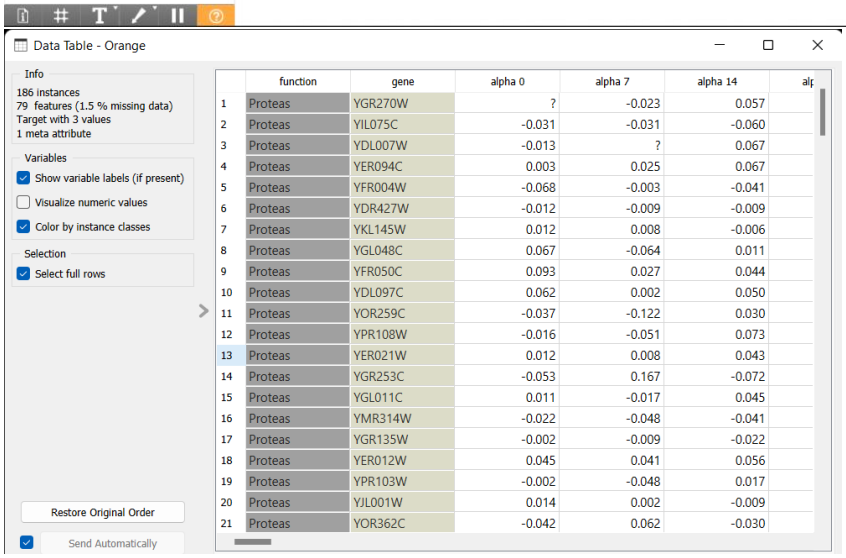

Choose class-labeled dataset. Say, "iris.tab" from documentation datasets.
File
It's always a good idea to check out the data first.
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
Confusion Matrix
Data Table (1)
Use for additional analysis of cross-validation results.
Several learners can be scored in cross-validation at the same time.
SVM


Choose class-labeled dataset. Say, "iris.tab" from documentation datasets.
It's always a good idea to check out the data first.
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
Data Table (1)
Several learners can be scored in cross-validation at the same time.
SVM

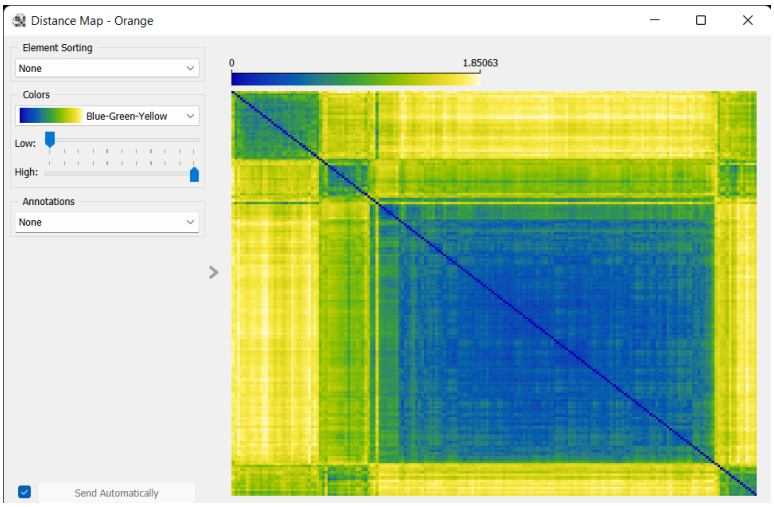

Untitled * - Orange
File Edit View Widget Options Help
Data
File
CSV File Import
Datasets
SQL Table
Data Table
Paint Data
Data Info
Rank
Edit Domain
Color
Feature Statistics
Save Data
Transform
Visualize
Model
Evaluate
Unsupervised
Confusion Matrix
Display a confusion matrix constructed from the results of classifier evaluations.
more...
Choose class-labeled
Confusion Matrix - Orange
Learners
Logistic Regression
Random Forest Learner
SVM Learner
Clicking on cells or in headers outputs the corresponding data instances
Ok, got it
Show: Number of instances
Predicted
Actual
Iris-setosa Iris-versicolor Iris-virginica Σ
Iris-setosa 50 0 0 50
Iris-versicolor 0 47 3 50
Iris-virginica 0 2 48 50
Σ 50 49 51 150
Output
Predictions
Probabilities
Apply Automatically
Select Correct
Select Misclassified
Clear Selection
3×150 | 150
idea to
first.
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
Selected Data → Data
Matrix
Data Table (1)
Use for additional analysis of cross-validation results.
be scored in cross-validation at the same time.
SVM

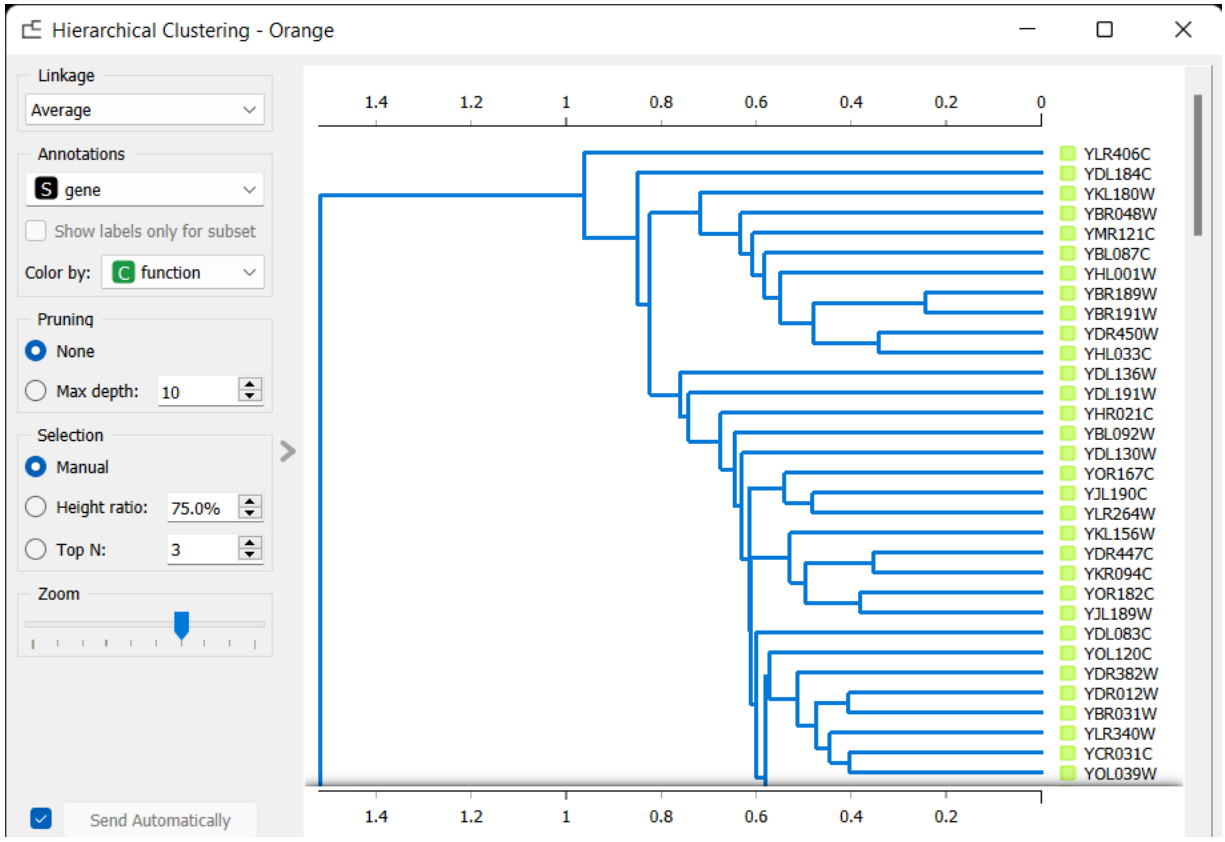

Untitled * - Orange
File Edit View Widget Options Help
Data
File
CSV File Import
Datasets
SQL Table
Data Table
Paint Data
Data Info
Rank
Edit Domain
Color
Feature Statistics
Save Data
Transform
Visualize
Model
Evaluate
Unsupervised
Data Table
View the dataset in a spreadsheet.
more...
Data Table (1) - Orange
Info
2 instances (no missing data)
4 features
Target with 3 values
1 meta attribute
Variables
Show variable labels (if present)
Visualize numeric values
Color by instance classes
Selection
Select full rows
Restore Original Order
Send Automatically
2 | 2
iris
is(Logistic Regression
sepal length
sepal width
petal length
1 Iris-virginica Iris-versicolor 4.9 2.5 4.5
2 Iris-virginica Iris-versicolor 6.0 2.2 5.0
Select a cell in confusion matrix to obtain related data instances. Here we examine them in the spreadheet.
Data Table (1)
e for additional
alysis of cross-
lidation results.
scores.
Several learners can be scored in cross-validation at the same time.
SVM





	function	gene	alpha 0	alpha 7	alpha 14	alp
1	Proteas	YGR270W	?	-0.023	0.057	
2	Proteas	YIL075C	-0.031	-0.031	-0.060	
3	Proteas	YDL007W	-0.013	?	0.067	
4	Proteas	YER094C	0.003	0.025	0.067	
5	Proteas	YFR004W	-0.068	-0.003	-0.041	
6	Proteas	YDR427W	-0.012	-0.009	-0.009	
7	Proteas	YKL145W	0.012	0.008	-0.006	
8	Proteas	YGL048C	0.067	-0.064	0.011	
9	Proteas	YFR050C	0.093	0.027	0.044	
10	Proteas	YDL097C	0.062	0.002	0.050	
11	Proteas	YOR259C	-0.037	-0.122	0.030	
12	Proteas	YPR108W	-0.016	-0.051	0.073	
13	Proteas	YER021W	0.012	0.008	0.043	
14	Proteas	YGR253C	-0.053	0.167	-0.072	
15	Proteas	YGL011C	0.011	-0.017	0.045	
16	Proteas	YMR314W	-0.022	-0.048	-0.041	
17	Proteas	YGR135W	-0.002	-0.009	-0.022	
18	Proteas	YER012W	0.045	0.041	0.056	
19	Proteas	YPR103W	-0.002	-0.048	0.017	
20	Proteas	YJL001W	0.014	0.002	-0.009	
21	Proteas	YOR362C	-0.042	0.062	-0.030	



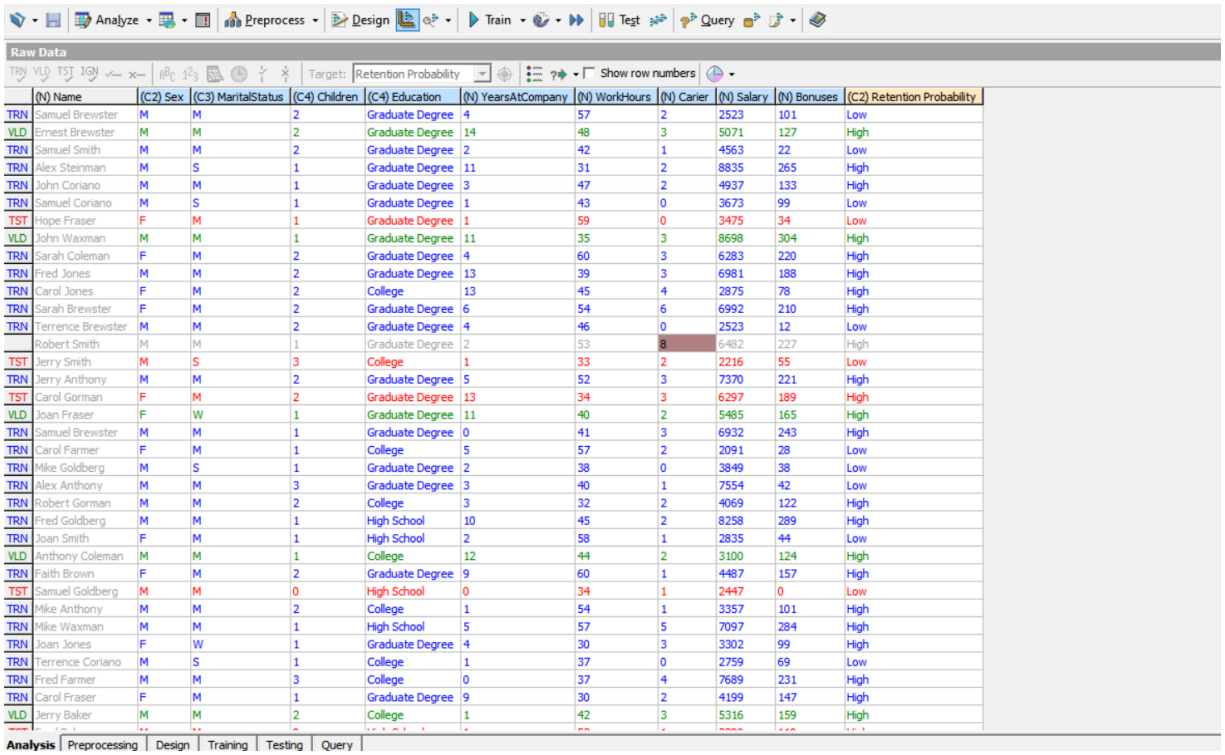


Conclusion: In this practical we performed classification on Iris dataset using neural network tools such as WEKA, ORANGE.

6.2: Show the uses of NEUROINTELLIGENCE to make prediction for faculty retention ratio.

Output:

Analysis:



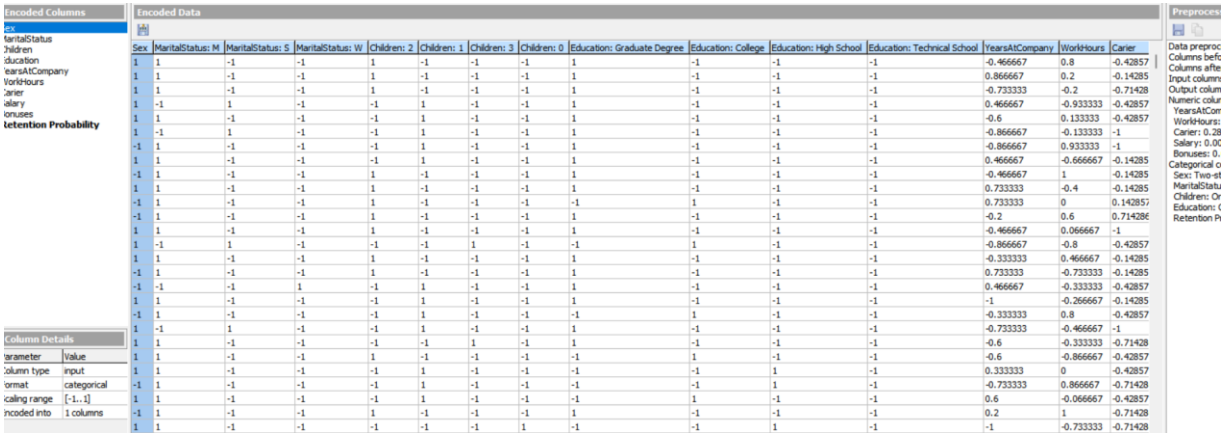
Raw Data

Target: Retention Probability | Show row numbers

	(N) Name	(C2) Sex	(C3) MaritalStatus	(C4) Children	(C4) Education	(N) YearsAtCompany	(N) WorkHours	(N) Carier	(N) Salary	(N) Bonuses	(C2) Retention Probability
TRN	Samuel Brewster	M	M	2	Graduate Degree	4	57	2	2523	101	Low
VLD	Ernest Brewster	M	M	2	Graduate Degree	14	48	3	5071	127	High
TRN	Samuel Smith	M	M	2	Graduate Degree	2	42	1	4563	22	Low
TRN	Alex Steinman	M	S	1	Graduate Degree	11	31	2	8835	265	High
TRN	John Coriano	M	M	1	Graduate Degree	3	47	2	4937	133	High
TRN	Samuel Coriano	M	S	1	Graduate Degree	1	43	0	3673	99	Low
TST	Hope Fraser	F	M	1	Graduate Degree	1	59	0	3475	34	Low
VLD	John Waxman	M	M	1	Graduate Degree	11	35	3	8698	304	High
TRN	Sarah Coleman	F	M	2	Graduate Degree	4	60	3	6283	220	High
TRN	Fred Jones	M	M	2	Graduate Degree	13	39	3	6981	188	High
TRN	Carol Jones	F	M	2	College	13	45	4	2875	78	High
TRN	Sarah Brewster	F	M	2	Graduate Degree	6	54	6	6992	210	High
TRN	Terrence Brewster	M	M	2	Graduate Degree	4	46	0	2523	12	Low
	Robert Smith	M	M	1	Graduate Degree	2	53	8	6482	227	High
TST	Jerry Smith	M	S	3	College	1	33	2	2216	55	Low
TRN	Jerry Anthony	M	M	2	Graduate Degree	5	52	3	7370	221	High
TST	Carol Gorman	F	M	2	Graduate Degree	13	34	3	6297	189	High
VLD	Joan Fraser	F	W	1	Graduate Degree	11	40	2	5485	165	High
TRN	Samuel Brewster	M	M	1	Graduate Degree	0	41	3	6932	243	High
TRN	Carol Farmer	F	M	1	College	5	57	2	2091	28	Low
TRN	Mike Goldberg	M	S	1	Graduate Degree	2	38	0	3849	38	Low
TRN	Alex Anthony	M	M	3	Graduate Degree	3	40	1	7554	42	Low
TRN	Robert Gorman	M	M	2	College	3	32	2	4069	122	High
TRN	Fred Goldberg	M	M	1	High School	10	45	2	8258	289	High
TRN	Joan Smith	F	M	1	High School	2	58	1	2835	44	Low
VLD	Anthony Coleman	M	M	1	College	12	44	2	3100	124	High
TRN	Faith Brown	F	M	2	Graduate Degree	9	60	1	4487	157	High
TST	Samuel Goldberg	M	M	0	High School	0	34	1	2447	0	Low
TRN	Mike Anthony	M	M	2	College	1	54	1	3357	101	High
TRN	Mike Waxman	M	M	1	High School	5	57	5	7097	284	High
TRN	Joan Jones	F	W	1	Graduate Degree	4	30	3	3302	99	High
TRN	Terrence Coriano	M	S	1	College	1	37	0	2759	69	Low
TRN	Fred Farmer	M	M	3	College	0	37	4	7689	231	High
TRN	Carol Fraser	F	M	1	Graduate Degree	9	30	2	4199	147	High
VLD	Jerry Baker	M	M	2	College	1	42	3	5316	159	High

Analysis | Preprocessing | Design | Training | Testing | Query

Pre-processing:



Encoded Columns

Sex
MaritalStatus
Children
Education
YearsAtCompany
WorkHours
Carier
Salary
Bonuses
Retention Probability

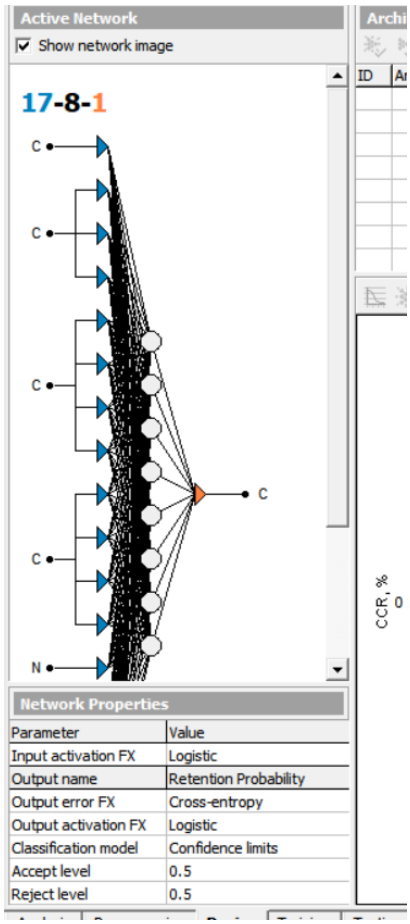
Column Details

Parameter	Value
Column type	input
Format	categorical
Scaling range	[-1..1]
Encoded into	1 columns

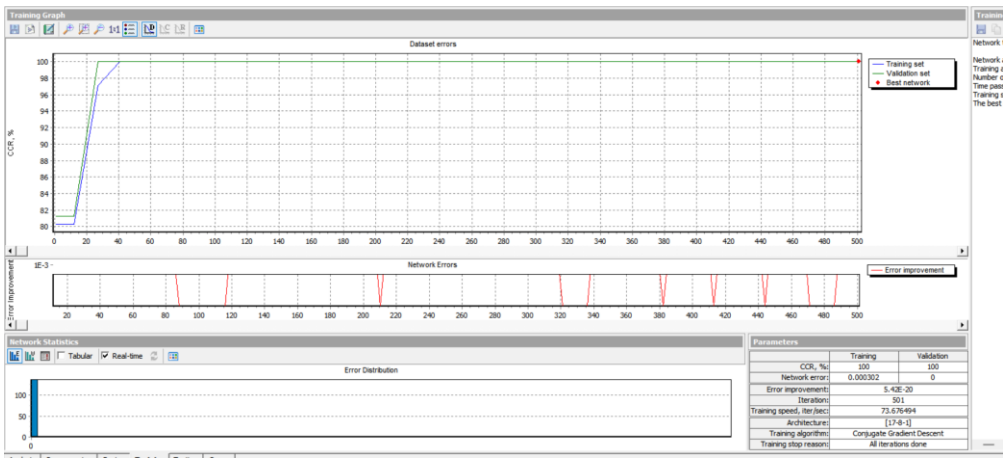
Encoded Data

Sex	MaritalStatus: M	MaritalStatus: S	MaritalStatus: W	Children: 2	Children: 1	Children: 3	Children: 0	Education: Graduate Degree	Education: College	Education: High School	Education: Technical School	YearsAtCompany	WorkHours	Carier
1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-0.466667	0.8	-0.42857
1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	0.866667	0.2	-0.14285
1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-0.733333	-0.2	-0.71428
1	-1	1	-1	-1	1	-1	-1	1	-1	-1	-1	0.466667	-0.933333	-0.42857
1	1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	-0.6	0.133333	-0.42857
1	-1	1	-1	-1	1	-1	-1	1	-1	-1	-1	-0.866667	-0.133333	-1
-1	1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	-0.866667	0.933333	-1
1	1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	0.466667	-0.666667	-0.14285
-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-0.466667	1	-0.14285
1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	0.733333	-0.4	-0.14285
-1	1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	0.733333	0	0.142857
-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-0.2	0.6	0.714286
1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-0.466667	0.066667	-1
1	-1	1	-1	-1	-1	1	-1	-1	1	-1	-1	-0.866667	-0.8	-0.42857
1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-0.333333	0.466667	-0.14285
-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	0.733333	-0.733333	-0.14285
-1	-1	-1	1	-1	1	-1	-1	1	-1	-1	-1	0.466667	-0.333333	-0.42857
1	1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	-1	-0.266667	-0.14285
-1	1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	-0.333333	0.8	-0.42857
1	-1	1	-1	-1	1	-1	-1	1	-1	-1	-1	-0.733333	-0.466667	-1
1	1	-1	-1	-1	-1	1	-1	1	-1	-1	-1	-0.6	-0.333333	-0.71428
1	1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	-0.6	-0.866667	-0.42857
-1	1	-1	-1	-1	1	-1	-1	-1	-1	1	-1	0.333333	0	-0.42857
1	1	-1	-1	-1	1	-1	-1	-1	-1	1	-1	-0.733333	0.866667	-0.71428
1	1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	0.6	-0.066667	-0.42857
-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	0.2	1	-0.71428
1	1	-1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	-0.733333	-0.71428

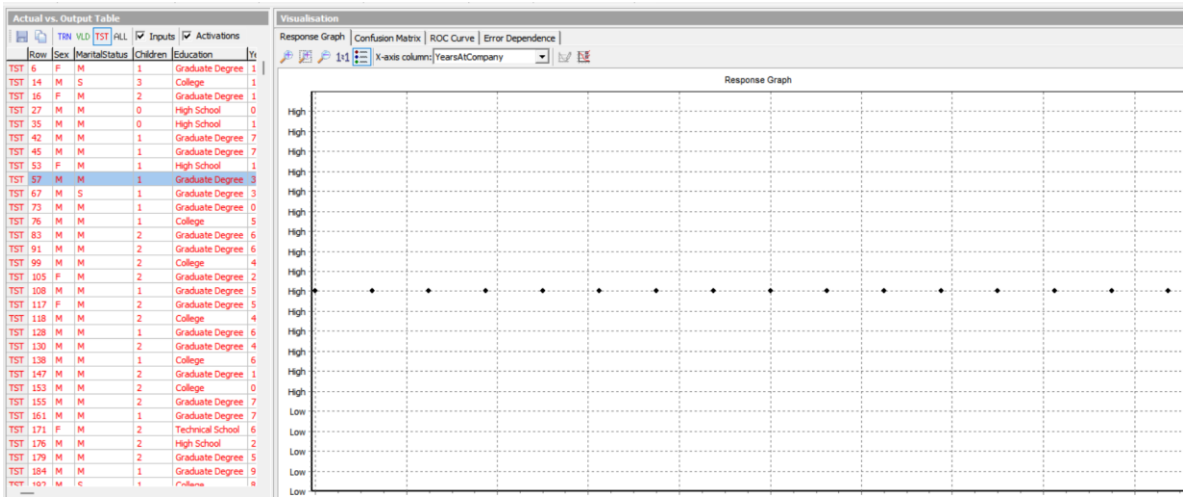
Design:



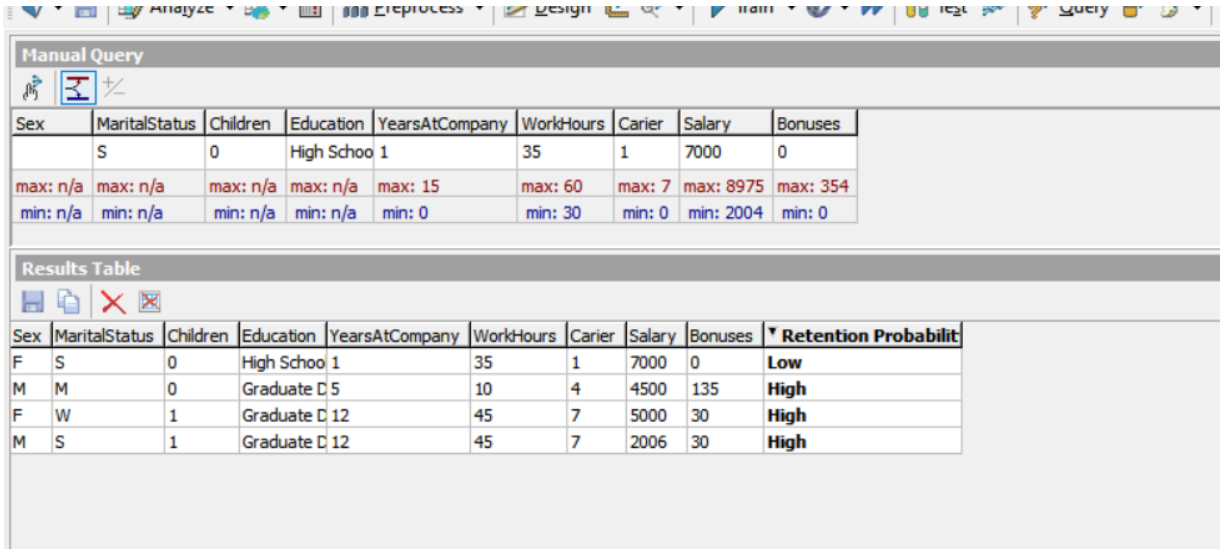
Training:



Testing:



Query:



Conclusion: In this practical we learned about NEUROINTELLIGENCE tool and how to work on it.

Practical –7

Game Playing

7.1: Apply MINIMAX Algorithm to solve Tic-Tac-Toe game in python. Design your solution using alpha-beta pruning.

Code:

```
from random import choice
from math import inf
board = [[0, 0, 0],
         [0, 0, 0],
         [0, 0, 0]]
def Gameboard(board):
    chars = {1: 'X', -1: 'O', 0: ' '}
    for x in board:
        for y in x:
            ch = chars[y]
            print(f'| {ch} |', end='')
        print('\n' + '---------------')
    print('===============')
def Clearboard(board):
    for x, row in enumerate(board):
        for y, col in enumerate(row):
            board[x][y] = 0
def winningPlayer(board, player):
    conditions = [[board[0][0], board[0][1], board[0][2]],
                  [board[1][0], board[1][1], board[1][2]],
                  [board[2][0], board[2][1], board[2][2]],
                  [board[0][0], board[1][0], board[2][0]],
```

```
                [board[0][1], board[1][1], board[2][1]],
                [board[0][2], board[1][2], board[2][2]],
                [board[0][0], board[1][1], board[2][2]],
                [board[0][2], board[1][1], board[2][0]]]

    if [player, player, player] in conditions:
        return True
    return False
def gameWon(board):
    return winningPlayer(board, 1) or winningPlayer(board, -1)
def printResult(board):
    if winningPlayer(board, 1):
        print('X has won! ' + '\n')
    elif winningPlayer(board, -1):
        print('O\'s have won! ' + '\n')
    else:
        print('Draw' + '\n')
def blanks(board):
    blank = []
    for x, row in enumerate(board):
        for y, col in enumerate(row):
            if board[x][y] == 0:
                blank.append([x, y])
    return blank
def boardFull(board):
    if len(blanks(board)) == 0:
        return True
```

```
    return False
def setMove(board, x, y, player):
    board[x][y] = player
def playerMove(board):
    e = True
    moves = {1: [0, 0], 2: [0, 1], 3: [0, 2],
             4: [1, 0], 5: [1, 1], 6: [1, 2],
             7: [2, 0], 8: [2, 1], 9: [2, 2]}
    while e:
        try:
            move = int(input('Enter a number between 1-9: '))
            if move < 1 or move > 9:
                print('Invalid Move! Try again!')
            elif not (moves[move] in blanks(board)):
                print('Invalid Move! Try again!')
            else:
                setMove(board, moves[move][0], moves[move][1], 1)
                Gameboard(board)
                e = False
        except(KeyError, ValueError):
            print('Enter a number!')
def getScore(board):
    if winningPlayer(board, 1):
        return 10
    elif winningPlayer(board, -1):
        return -10
    else:
```

```
        return 0
    def abminimax(board, depth, alpha, beta, player):
        row = -1
        col = -1
        if depth == 0 or gameWon(board):
            return [row, col, getScore(board)]
        else:
            for cell in blanks(board):
                setMove(board, cell[0], cell[1], player)
                score = abminimax(board, depth - 1, alpha, beta, -player)
                if player == 1:
                    # X is always the max player
                    if score[2] > alpha:
                        alpha = score[2]
                        row = cell[0]
                        col = cell[1]
                else:
                    if score[2] < beta:
                        beta = score[2]
                        row = cell[0]
                        col = cell[1]
                setMove(board, cell[0], cell[1], 0)
                if alpha >= beta:
                    break
            if player == 1:
                return [row, col, alpha]
            else:
```

```
            return [row, col, beta]
    def o_comp(board):
        if len(blanks(board)) == 9:
            x = choice([0, 1, 2])
            y = choice([0, 1, 2])
            setMove(board, x, y, -1)
            Gameboard(board)
        else:
            result = abminimax(board, len(blanks(board)), -inf, inf, -1)
            setMove(board, result[0], result[1], -1)
            Gameboard(board)

    def x_comp(board):
        if len(blanks(board)) == 9:
            x = choice([0, 1, 2])
            y = choice([0, 1, 2])
            setMove(board, x, y, 1)
            Gameboard(board)
        else:
            result = abminimax(board, len(blanks(board)), -inf, inf, 1)
            setMove(board, result[0], result[1], 1)
            Gameboard(board)
    def makeMove(board, player, mode):
        if mode == 1:
            if player == 1:
                playerMove(board)
            else:
```

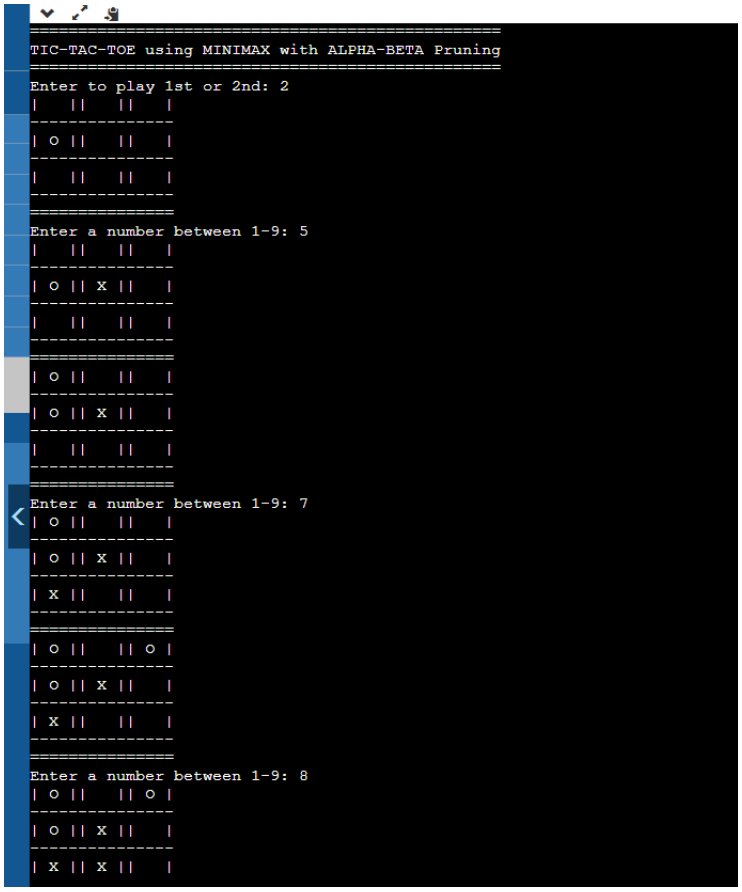
```
            o_comp(board)
    else:
        if player == 1:
            o_comp(board)
        else:
            x_comp(board)
def pvc():
    while True:
        try:
            order = int(input('Enter to play 1st or 2nd: '))
            if not (order == 1 or order == 2):
                print('Please pick 1 or 2')
            else:
                break
        except(KeyError, ValueError):
            print('Enter a number')

    Clearboard(board)
    if order == 2:
        currentPlayer = -1
    else:
        currentPlayer = 1
    while not (boardFull(board) or gameWon(board)):
        makeMove(board, currentPlayer, 1)
        currentPlayer *= -1
    printResult(board)
print("================================================")
```

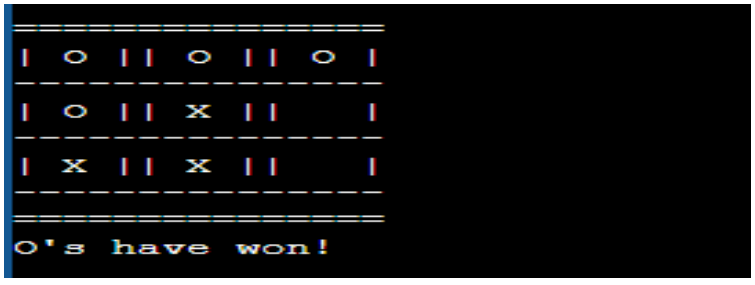
print("TIC-TAC-TOE using MINIMAX with ALPHA-BETA Pruning")

print("==")

pvc()

Output:





Conclusion: In this practical we learned how to Apply MINIMAX Algorithm to solve Tic-Tac-Toe game in python.

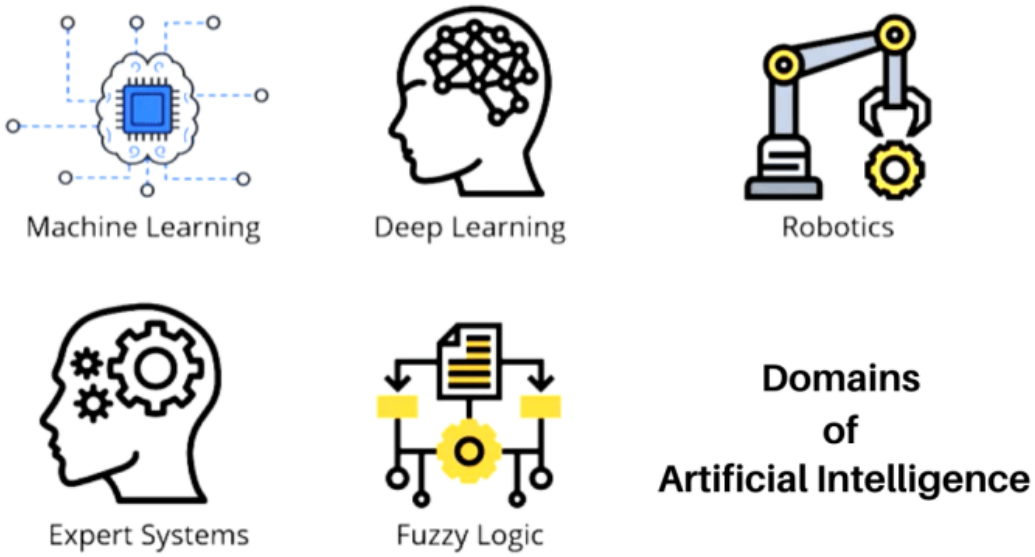
Practical –8

Game Playing

8.1: Presentation on AI Domains, Applications, Problems and its Explanation

Domains of AI:

- Machine Learning
- Deep Learning
- Robotics
- Expert systems
- Fuzzy Logic



Machine Learning:

Machine Learning is basically the science of getting machines to interpret processes and analyze the data to solve real world problems. Machine learning supervised and unsupervised in enforcement learning.

Deep Learning:

Deep Learning is the process of implementing neural networks on high dimensional data to gain insights and form solutions, basically logic behind in the face verification algorithms on Facebook it is the logic behind self-driving cars, virtual assistant and Alexa.

Then we have natural language processing first to science to draw insights from natural human languages and grow the businesses.

Example: Twitter and Amazon

Twitter uses NLP to filter out terrorizing words in the twitter

Amazon uses NLP to understand the consumer behavior and user experiences

Robotics:

Robotics is a branch of artificial intelligence which focuses on different branches and applications of robots. AI robots are artificial environments to produce results by taking some accountable actions.

Example: Sofia robot

Expert Systems:

Expert systems are AI based computer systems that learn and reciprocate the decision-making ability of human experts. Expert systems use IF/Then logics mentioned in order to solve any complex problems. They do not relay conventional proceeding programming.

Expert systems mainly used in information management seem to use virus detection, medical and hospital records and so on.

Fuzzy Logic:

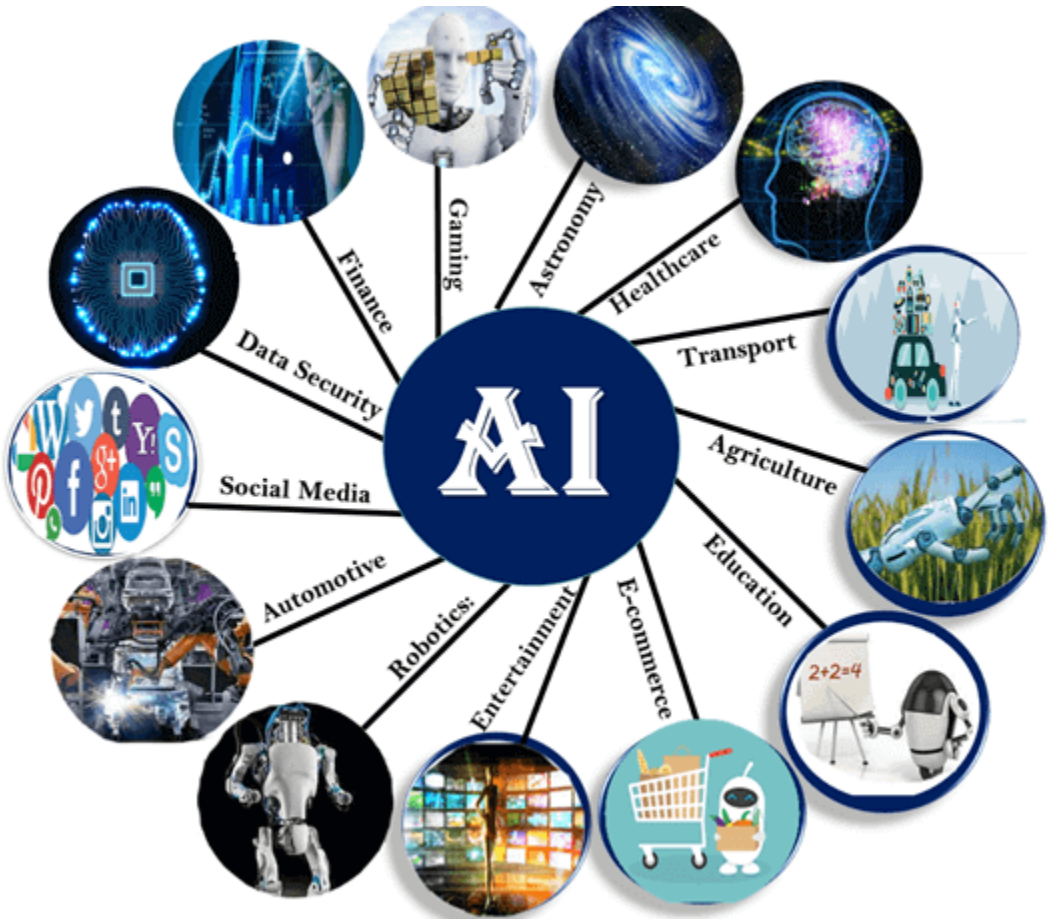
Fuzzy logic is computing approach that is based on principles degree of truth instead of

usual modern logic that we used which is basically Boolean logic. Fuzzy logic used in the medical field to solve complex problems which involve decision making is also used in automated gas systems in cars.

Application of AI

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:



1. AI in Astronomy

- Artificial Intelligence can be very useful to solve complex universe problems. AI technology can be helpful for understanding the universe such as how it works, origin, etc.

2. AI in Healthcare

- In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry.
- Healthcare Industries are applying AI to make a better and faster diagnosis than humans. AI can help doctors with diagnoses and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

3. AI in Gaming

- AI can be used for gaming purpose. The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

4. AI in Finance

- AI and finance industries are the best matches for each other. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

5. AI in Data Security

- The security of data is crucial for every company and cyber-attacks are growing very rapidly in the digital world. AI can be used to make your data more safe and secure. Some examples such as AEG bot, AI2 Platform,are used to determine software bug and cyber-attacks in a better way.

6. AI in Social Media

- Social Media sites such as Facebook, Twitter, and Snapchat contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amounts of data. AI can analyze lots of data to identify the latest trends, hashtag, and requirement of different users.

7. AI in Travel & Transport

- AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

8. AI in Automotive Industry

- Some Automotive industries are using AI to provide virtual assistant to their user for better performance. Such as Tesla has introduced TeslaBot, an intelligent virtual assistant.
- Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

9. AI in Robotics:

- Artificial Intelligence has a remarkable role in Robotics. Usually, general robots are programmed such that they can perform some repetitive task, but with the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed.
- Humanoid Robots are best examples for AI in robotics, recently the intelligent Humanoid robot named as Erica and Sophia has been developed which can talk and behave like humans.

10. AI in Entertainment

- We are currently using some AI based applications in our daily life with some entertainment services such as Netflix or Amazon. With the help of ML/AI algorithms, these services show the recommendations for programs or shows.

11. AI in Agriculture

- Agriculture is an area which requires various resources, labor, money, and time for best result. Now a day's agriculture is becoming digital, and AI is emerging in this field. Agriculture is applying AI as agriculture robotics, solid and crop monitoring, predictive analysis. AI in agriculture can be very helpful for farmers.

12. AI in E-commerce

- AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

13. AI in education:

- AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.
- AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.

Top Common Challenges in AI

There are several Artificial Intelligence problems, and we are going to address these challenges and how to solve them.

1. Computing Power

The amount of power these power-hungry algorithms use is a factor keeping most developers away. Machine Learning and Deep Learning are the stepping stones of this Artificial Intelligence, and they demand an ever-increasing number of cores and GPUs to work efficiently. There are various domains where we have ideas and knowledge to implement deep learning frameworks such as asteroid tracking, healthcare deployment, tracing of cosmic bodies, and much more.

They require a supercomputer's computing power, and yes, supercomputers aren't cheap. Although, due to the availability of Cloud Computing and parallel processing systems developers work on AI systems more effectively, they come at a price. Not everyone can afford that with an increase in the inflow of unprecedented amounts of data and rapidly increasing complex algorithms.

2. Trust Deficit

One of the most important factors that are a cause of worry for the AI is the unknown nature of how deep learning models predict the output. How a specific set of inputs can devise a solution for different kinds of problems is difficult to understand for a layman.

Many people in the world don't even know the use or existence of Artificial Intelligence, and how it is integrated into everyday items they interact with such as smartphones, Smart TVs, Banking, and even cars (at some level of automation).

Must Read: Free nlp online course!

3. Limited Knowledge

Although there are many places in the market where we can use Artificial Intelligence as a better alternative to the traditional systems. The real problem is the knowledge of Artificial Intelligence. Apart from technology enthusiasts, college students, and researchers, there are only a limited number of people who are aware of the potential of AI.

For example, there are many **SMEs (Small and Medium Enterprises)** which can have their work scheduled or learn innovative ways to increase their production, manage resources, sell and manage products online, learn and understand consumer behavior and react to the market effectively and efficiently. They are also not aware of service providers such as Google Cloud, Amazon Web Services, and others in the tech industry.

4. Human-level

This is one of the most important challenges in AI, one that has kept researchers on edge for AI services in companies and start-ups. These companies might be boasting of above 90% accuracy, but humans can do better in all of these scenarios. For example, let our model predict whether the image is of a dog or a cat. The human can predict the correct output nearly every time, mopping up a stunning accuracy of above 99%.

For a deep learning model to perform a similar performance would require unprecedented finetuning, hyperparameter optimization, large dataset, and a well-defined and accurate algorithm, along with robust computing power, uninterrupted training on train data and testing on test data. That sounds a lot of work, and it's actually a hundred times more difficult than it sounds.

One way you can avoid doing all the hard work is just by using a service provider, for they can train specific deep learning models using pre-trained models. They are trained on millions of images and are fine-tuned for maximum accuracy, but the real problem is that they continue to show errors and would really struggle to reach human-level performance.

5. Data Privacy and Security

The main factor on which all the deep and machine learning models are based on is the availability of data and resources to train them. Yes, we have data, but as this data is generated from millions of users around the globe, there are chances this data can be used for bad purposes.

For example, let us suppose a medical service provider offers services to 1 million people in a city, and due to a cyber-attack, the personal data of all the one million users fall in the hands of everyone on the dark web. This data includes data about diseases, health problems, medical history, and much more. To make matters worse, we are now dealing with planet size data. With this much information pouring in from all directions, there would surely be some cases of data leakage.

Some companies have already started working innovatively to bypass these barriers. It trains the data on smart devices, and hence it is not sent back to the servers, only the trained model is sent back to the organization.

Conclusion: In this practical we did in depth analysis in AI.