Rappresentazione della conoscenza

Sistemi esperti in Prolog

Sistemi Esperti

Applicazioni che forniscono prestazioni analoghe a quelle di un **esperto umano**.

Tecnologie

- **basati sulla conoscenza**
- **a scatola nera**: es. reti neurali

Esperto finanziario

Problema: assegnazione di un prestito.

La banca richiede ed acquisisce una serie di informazioni sul soggetto che chiede il prestito e ne valuta la consistenza per decidere se è vantaggioso concedere il prestito.

Il rischio è sia quello di non restituzione della somma sia quello di perdere il cliente.

Elementi per la valutazione

♢ **Garanzie patrimoniali** (collateral). Beni o proprietà che possono costituire una garanzia per la restituzione del debito.

♢ **Situazione finanziaria** (financial record). Fattori che forniscono indicazioni sulla possibilità di pagamento delle rate.

♢ **Guadagno** per la banca (yield). Presumibilmente riferito sia al prestito sia alla posizione complessiva del cliente.

Garanzie patrimoniali

- **Prima classe**: es. depositi in valuta,
- **Seconda classe**: es. azioni e titoli,
- **Non liquidabili**: es. ammortamenti,

Situazione finanziaria

Fattori principali:

- **rendita netta** per bene posseduto
- **guadagni lordi** sulle vendite

Fattori secondari (in ordine di importanza):

- **debito** a breve termine sulle vendite annuali
- **crescita** delle vendite nell'ultimo anno

Guadagno per la banca

Questa informazione è riservata ...

la fornisce direttamente la banca.

L'esperto

Le valutazioni dell'esperto sono del tipo:

> Se le garanzie patrimoniali sono eccellenti e la situazione finanziaria è buona, anche con un guadagno ragionevole per la banca, conviene concedere il prestito

Problema: caratterizzare le valutazioni qualitative.

Struttura del programma

```
credit(Client,Answer) :-
    ok_profile(Client),
    collateral_rating(Client,CollateralRating),
    financial_rating(Client,FinancialRating),
    bank_yield(Client,Yield),
    evaluate(profile(CollateralRating,FinancialRating,Yield),
             Answer),
            !.
```

Valutazione delle garanzie patrimoniali

```
collateral_rating(Client,Rating) :-
   collateral_profile(Client,FirstClass,SecondClass,Illiquid),
   collateral_evaluation(FirstClass,SecondClass,Illiquid,Rating).

collateral_profile(Client,FirstClass,SecondClass,Illiquid) :-
    requested_credit(Client,Credit),
collateral_percent(first_class,Client,Credit,FirstClass),
collateral_percent(second_class,Client,Credit,SecondClass),
collateral_percent(illiquid,Client,Credit,Illiquid).

collateral_percent(Type,Client,Total,Value) :-
    findall(X,(collateral(Collateral,Type),
               amount(Collateral,Client,X)),Xs),
    sumlist(Xs,Sum),
    Value is Sum*100/Total.
```

Regole di valutazione patrimoniale

```
collateral_evaluation(FirstClass,SecondClass,Illiquid,excellent) :-
   FirstClass >= 100.
collateral_evaluation(FirstClass,SecondClass,Illiquid,excellent) :-
   FirstClass > 70, FirstClass + SecondClass >= 100.
collateral_evaluation(FirstClass,SecondClass,Illiquid,good) :-
   FirstClass + SecondClass > 60,
   FirstClass + SecondClass < 70,
   FirstClass + SecondClass + Illiquid >= 100.
```

Classificazione delle garanzie patrimoniali

```
collateral(local_currency_deposits,first_class).
collateral(foreign_currency_deposits,first_class).
collateral(negotiate_instruments,second_class).
collateral(mortgage,illiquid).
```

Valutazione situazione finanziaria

```
financial_rating(Client,Rating) :-
   financial_factors(Factors),
   score(Factors,Client,0,Score),
   calibrate(Score,Rating).
```

Fattori di peso

```
financial_factors([(net_worth_per_assets,5),
    (last_year_sales_growth,1),
    (gross_profits_on_sales,5),
    (short_term_debt_per_annual_sales,2)  ]).

score([(Factor,Weight)|Factors],Client,Acc,Score) :-
   value(Factor,Client,Value),
   Acc1 is Acc + Weight*Value,
   score(Factors,Client,Acc1,Score).
score([],Client,Score,Score).
```

Regole di valutazione finanziaria

```
calibrate(Score,bad) :- Score =< -500.
calibrate(Score,medium) :- -500 < Score, Score < 150.
calibrate(Score,good) :- 150 =< Score, Score < 1000.
calibrate(Score,excellent) :- Score >= 1000.
```

Valutazione complessiva

```
evaluate(Profile,Answer) :-
rule(Conditions,Answer), verify(Conditions,Profile).

verify([condition(Type,Test,Rating)|Conditions],Profile) :-
   scale(Type,Scale),
   select_value(Type,Profile,Fact),
   compare(Test,Scale,Fact,Rating),
   verify(Conditions,Profile).
verify([],Profile).
```

Le regole di valutazione

```
compare('=',Scale,Rating,Rating).
compare('>',Scale,Rating1,Rating2) :-
   precedes(Scale,Rating1,Rating2).
compare('>=',Scale,Rating1,Rating2) :-
   precedes(Scale,Rating1,Rating2) ; Rating1 = Rating2.
compare('<',Scale,Rating1,Rating2) :-
   precedes(Scale,Rating2,Rating1).
compare('=<',Scale,Rating1,Rating2) :-
   precedes(Scale,Rating2,Rating1) ; Rating1 = Rating2.

precedes([R1|Rs],R1,R2).
precedes([R|Rs],R1,R2) :- R \== R2, precedes(Rs,R1,R2).

select_value(collateral,profile(C,F,Y),C).
select_value(finances,profile(C,F,Y),F).
select_value(yield,profile(C,F,Y),Y).
```

Dati bancari e regole

```
rule([condition(collateral,'>=',excellent),
      condition(finances,'>=',good),
      condition(yield,'>=',reasonable)],give_credit).
rule([condition(collateral,'=',good),
      condition(finances,'=',good),
      condition(yield,'>=',reasonable)],consult_superior).
rule([condition(collateral,'=<',moderate),
      condition(finances,'=<',medium)], refuse_credit).

scale(collateral,[excellent,good,moderate]).
scale(finances,[excellent,good,medium,bad]).
scale(yield,[excellent,reasonable,poor]).
```

Dati di esempio

```
bank_yield(client1,excellent).
requested_credit(client1,5000).

amount(local_currency_deposits,client1,3000).
amount(foreign_currency_deposits,client1,2000).
amount(bank_guarantees,client1,300).

amount(negotiate_instruments,client1,500).
amount(stocks,client1,900).

amount(mortgage,client1,1200).
amount(documents,client1,1400).
```

Dati di esempio

```
value(net_worth_per_assets,client1,40).
value(last_year_sales_growth,client1,20).
value(gross_profits_on_sales,client1,45).
value(short_term_debt_per_annual_sales,client1,9).

ok_profile(client1).
```

Utilità

```
sumlist(Is,Sum) :- sumlist(Is,0,Sum).
sumlist([I|Is],Temp,Sum) :-
   Temp1 is Temp + I,
   sumlist(Is,Temp1,Sum).
sumlist([],Sum,Sum).
```