

# Modulo di IA

## Progetto combinato con IS: NewDM

## Sezione: Apprendimento

## Autori: Cirillo Franco, Cirillo Luigi, Fusco Ciro, Aiello Vincenzo

# 1. Introduzione

## **Ambito del modulo**

Il nostro modulo è utilizzato nel prodotto software NewDM, una Piattaforma Desktop per la gestione di un punto vendita di un supermercato. Il modulo è stato progettato per fornire un supporto all’attività del magazziniere, che tra le tante mansioni deve anche gestire le richieste di rifornimento per il supermercato.

## **Scopo del modulo**

Il suddetto modulo di IA nasce allo scopo di fornire delle previsione sulle quantità di prodotti realmente utili per il magazzino, al fine di dare dei consigli per l’acquisto di nuove forniture. In particolare il magazziniere andrà ad inserire i dati richiesti e il sistema restituirà la quantità da acquistare per quel dato prodotto.

## **Specifica PEAS**

Per una rappresentazione schematica dell’ambiente e dell’agente utilizziamo la specifica PEAS (*Performance, Environment, Actuators, Sensors*) che si basa sull’indicazione di quattro elementi:

* ***Performance*** (La misura di prestazione adottata per valutare l’operato dell’agente):

buona previsione delle quantità necessarie al magazzino, riduzione dei costi e maggiori entrate per il magazzino.

* ***Environment*** (Descrizione degli elementi che formano l’ambiente):

magazzinieri, rifornimenti.

* ***Actuators*** (Gli attuatori disponibili all’agente per intraprendere le azioni):

mostra delle quantità consigliate da rifornirsi.

* ***Sensors*** (I sensori attraverso i quali riceve gli input percettivi):

input da tastiera

# 2. Analisi dei requisiti

Come prima fase abbiamo analizzato il comportamento che dovrebbe avere il nostro modulo, vogliamo fornire determinati input e vogliamo ottenere una previsione basata su di essi. Per fare ciò l’agente intelligente dovrebbe apprendere da un insieme di dati e quindi si dovrebbe addestrare su di essi per poter meglio reagire a nuovi dati.

L’insieme di dati (*dataset*) dovrebbe fornirgli sia input e sia l’output da dare, in modo da imparare al meglio possibile. Il nostro problema rientra precisamente nell’insieme dei problemi di Apprendimento supervisionato: apprendere una funzione che descrive il fenomeno studiato partendo da input e output noti. In sostanza si conoscono i dati e si conosce precisamente cosa essi rappresentano. Infatti il nostro è principalmente un problema di *supply forecast* che sono soliti essere risolti con algoritmi supervisionati.

Lo step successivo, che è anche quello di maggiore interesse e su cui si basa l’ intero modulo, consiste nel trovare le giuste *feature*(caratteristiche su cui l’agente andrà lavorare e costituiscono l’input), *etichette*(risultato da dare in output) e quindi soprattutto trovare il dataset che servirà per l’addestramento.

Sono state effettuate numerose ricerche sul web al fine di trovare un dataset che potessimo utilizzare nel nostro modulo, ma non è stato trovato nulla di realmente adeguato per il nostro scopo.

Si è optato per la costruzione di un dataset da zero, essendo a conoscenza che sarebbe stata un’attività abbastanza complessa che ci avrebbe occupato molto tempo, ma che ci avrebbe dato un buon risultato finale, perfettamente adattato al nostro scopo.

Attraverso un brainstorming abbiamo proposto numerose feature e alla fine abbiamo selezionato le migliori, cioè quelle che a nostro avviso meglio caratterizzavano i dati. Invece per l’output abbiamo deciso di utilizzare un numero discreto di etichette, che rappresentano delle approssimazioni del quantitativo di rifornimento. Ulteriori spiegazioni verranno fornite nel paragrafo relativo al dataset.

A questo punto non ci resta che decidere il modello di apprendimento più adeguato al nostro utilizzo. Il modello in questione dovrebbe supportare l’apprendimento supervisionato e dalle nostre scelte(di output) dovrebbe essere un classificatore, quindi gestire output discreti.

Si è scelto cosi di utilizzare il classificatore Random Forest, ulteriori spiegazioni sono state fornite nel paragrafo relativo alle motivazioni.

### **Breve spiegazione Random Forest**

Le Random forest sono un metodo di ensamble learning per la classificazione e la regressione, che operano costruendo una moltitudine di alberi decisionali al momento dell'addestramento e fornendo la classe che è l’etichetta (classificazione) o previsione media / media ( regressione) dei singoli alberi.

ALGORITMO

1. A partire dal training set D costruisci B sottoinsiemi di bootstrap(estrazione con reinserimento)
2. Per ciascuno dei B sottoinsiemi costruisci un albero di decisione
3. Ad ogni nodo
   * Seleziona m variabili in maniera random tra le M possibili e fallo in maniera indipendente per ciascun nodo
   * Trova il miglior criterio di split per tale sottoinsieme di m variabili
4. Fa crescere ciascun albero fino alla massima altezza in relazione al criterio di splitting adottato senza effettuare il pruning(taglio dei rami)

Quando un nuovo campione viene dato in input alla foresta, esso passa attraverso ciascun albero separatamente. La classe del nuovo campione verrà scelta per mezzo di un processo di voto o di media delle risposte di tutti gli alberi.

# 3. Soluzione proposta

//libreria sklearn per Random Forest Classifier in python

La soluzione è stata proposta inizialmente attraverso un notebook jupiter scritto in python, al fine di valutarne tutti gli aspetti e l’ accuratezza. Al termine è stato tradotto in java attraverso l’utilizzo di un Porter.

## **Dataset**

Per effettuare la predizione è stato utilizzato il classificatore Random Forest della libreria sklearn. In assenza di dati su cui effettuare l’ addestramento di questo classificatore abbiamo provveduto alla scrittura di un dataset facendolo con la massima accuratezza in modo da renderlo il più realistico possibile.

Il dataset è formato da i seguenti campi:

* tipologia(FruttaVerdura, Pesce, Carne, Casa, Elettronica)
* stagione(Estate, Inverno, Primavera, Autunno)
* zona supermercato(Periferia, Residenziale)
* festività(Feriale, Lavorativo)
* scadenza(Breve, Media, Lunga)
* dimensione confezione(Piccola,Media,Grande)
* costo(prezzo)
* spedizione(prezzo)
* fornitura(20-50-100-150-200-300-500)

Queste rappresentano tutte le variabili su cui deve basarsi il classificatore e che quindi devono essere immesse da input, l’ultima è l’ etichetta che deve restituire il classificatore.

Raggiungibile al link:

## **Notebook**

In allegato il pdf del notebook, dove è presente il codice commentato e tutti i grafici risultanti dai dati per una migliore chiarezza.

In breve

1. caricamento dataset
2. divisione in training set e test set
3. addestramento della Random Forest
4. calcolo dell’accuratezza sui dati di test
5. calcolo dell’importanza di ogni feature e visualizzazione con grafico
6. estrapolazione di un albero dalla Random Forest
7. visione dettagliata di una parte dell’albero
8. traduzione in java

## **Migrazione in java**

//utilizzo del porter

# Risultati ottenuti

//Funzionamento effettivo del modulo

# Motivazioni delle scelte

La scelta è ricaduta sulla Random Forest perché essa risponde bene alle esigenze della situazione, cioè gestire le risposte relative ad una serie di domande al fine di restituire l’ etichetta giusta per i dati in input.

Infatti questo classificatore si presta molto bene ai problemi decisionali ed è un classificatore n-ario completamente automatizzabile. Lavora bene con valori numerici e categorici e non è influenzato da valori anomali, quindi risponde alle esigenze del nostro dataset.

Le foreste casuali generalmente forniscono un'elevata precisione poiché il principio del modello è quello di calcolare la media dei risultati tra i molteplici alberi decisionali che costruisce.

# Conclusioni e sviluppi futuri

//riaddestramento