Un aiuto in cucina

Gruppo di lavoro

- Mattia Donatelli, 716403, m.donatelli6@studenti.uniba.it
- Donato Ettorre, 717064, d.ettorre2@studenti.uniba.it

https://github.com/Donatelli22/Progetto_ICON_Donatelli_Ettorre

AA 2022-23

Indice

Introduzione

Questo progetto nasce dal desiderio di fornire degli strumenti a utenti qualunque per trovare in comodità e con facilità piatti in base ai loro desideri. Abbiamo notato nei vari siti in cui ci siamo andati a documentare per quanto fornite di ricette, scarse opzioni di ricerca e a volte un po' povere nella descrizione della proprietà dei piatti. Quindi siamo andati a realizzare un sistema esperto che vada a fornire dei piatti in base alle proprietà richieste e una serie di sistemi di apprendimento supervisionato per andare a trovare quali dati sono importanti nella predizione di un attributo.

Elenco argomenti di interesse

- **Sistema basato su conoscenza e ontologie**: Utilizzo di Protegé per la costruzione dell'ontologia, utilizzo di Owlready2 per la manipolazione di ontologia tramite query SPARQL
- **Apprendimento Supervisionato**: Utilizzo dei modelli di classificazione: alberi di decisione, Random Forest, AdaBoost, GradientBoosting, K-NN, Naive Bayes Gaussiano.

Sistema basato su conoscenza e ontologie

Sommario

La creazione di questo Sistema Esperto è nata dalla voglia di facilitare la creazione di menu in base ai propri gusti e desideri. Per fare ciò non abbiamo trovato Dataset che ci soddisfino perciò abbiamo deciso di usare le informazioni presenti nel sito "https://www.giallozafferano.it/ "per creare un'ontologia e andarla ad ampliare con informazioni non presenti sul sito.
Il programma permette la creazione di menu da tre portate: primo piatto, secondo piatto e dessert chiedendo all'utente la dieta che segue, i suoi gusti e le sue capacità.

Strumenti utilizzati

Per questo progetto abbiamo utilizzato diversi strumenti:

- **Protegé**: Un editor per ontologie con cui abbiamo scritto la nostra in formato RDF, andando a definire classi, proprietà e individui;
- **Pyhton**: Linguaggio di programmazione ad altro livello con molte librerie che vanno a semplificare il lavoro;
- **Owlready2**: Libreria Pyhton che permette la manipolazione, l'importazione e l'utilizzo di ontologie scritte in RDF/XML tramite anche l'utilizzo di SPARQL query;
- **Pyswip**: Libreria Python che permette l'utilizzo di query Prolog direttamente da codice o da file esterni integrabili tramite il comando "consult()".

Decisioni di Progetto

La nostra ontologia è composta da tre classi:

- **dieta**: Che a sua volta è stata suddivisa in altre tre sottoclassi che sono: **noglutine** per indicare una dieta che richiede l'assenza del glutine, **nolattosio** per indicare una dieta che richiede l'assenza del lattosio e **vegano** per una dieta vegana che quindi richiede l'assenza di carne, pesce e derivati;
- **ingrediente**: Classe che rappresenta gli ingredienti principali che saranno presenti nei Piatti;
- **piatti**: Classe che rappresenta i piatti che è divisa in tre sottoclassi che indicano le portate presenti nel menu: **primopiatto**, **secondopiatto** e **dessert.**

Va specificato che un piatto può contenere sia più ingredienti che entrare in più diete (ad esempio un piatto che rientra nella dieta vegano rientra in automatico anche nella dieta nolattosio data l'assenza di derivati)

Figura 1: Classi dell'ontologia

Per quanto riguarda le relazioni ne sono state definite due:

- **piatto** Contiene **ingrediente**: Relazione definita tramite Object Property che definisce che in un piatto siano presenti determinati ingredienti;
- menu (**dieta**, **ingrediente**): Relazione definita tramite **Prolog**, controlla se l'ingrediente è accettato dalla dieta.

Infine, siamo andati a definire gli individui come segue:

- Le istanze di **ingrediente**: istanze che vanno a rappresentare gli ingredienti la cui unica proprietà è il "**Nome"**. Sono stati definiti undici individui per andare a coprire gli ingredienti principali presenti in un piatto. Gli ingredienti scelti sono i seguenti: Carne, Cioccolata, Crema, Frutta, Latte, Legumi, Pasta, Pesce, Riso, Uova e Verdura.
- Le istanze di **piatto:** in realtà non ci sono delle istanze dirette per piatti ma bensì delle istanze delle sue sottoclassi con le seguenti proprietà: "**Nome"** per indicare il nome del piatto, "**Link**" per fornire all'utente il link alla ricetta per seguire il procedimento, "**Difficolta**" per indicare la difficoltà del piatto (che si divide in facile, medio e difficile), "**Prezzo**" per indicare il costo del piatto (che si divide in basso, medio e alto), "**Sapore**" (Utilizzato soltanto nelle istanze di **primopiatto** e **secondopiatto**) per indicare l'intensità del sapore del piatto (che si divide in leggero, medio, forte e piccante) e infine "**Temperatura**" (Utilizzato soltanto nelle istanze di **dessert**) per indicare la temperatura a cui va servito il piatto (che si divide in freddo, normale e caldo).
- Le istanze di **dieta**: come le istanze di **piatto** non ha sue istanze ma le hanno le sue sottoclassi. Sta anche da specificare che queste istanze sono le stesse condivise dalle sottoclassi di **piatto**.

Per utilizzare questa ontologia abbiamo dovuto ricorrere alla libreria **Owlready2** che ha permesso tramite la funzione “get_ontology ("file://... ").load()” di manipolare e interrogare l’ontologia.

Abbiamo deciso di utilizzare la libreria **Pyswip** per controllare la correttezza degli ingredienti richiesti dall’utente seguendo la dieta precedentemente scelta andando a richiamare un file tramite la funzione **“consult ("..")”**

```
p = Prolog()
onto = get_ontology("file://menu.rdf").load()
p.consult("menu.pl")
```

Figura 2: utilizzo dei metodi "get_ontology" e "consult"

Le domande che abbiamo scelto di porre all’utente sono le seguenti (con le possibili risposte fra parentesi):

- Seguite una delle seguenti diete? (NoLattosio, NoGlutine, Vegano o Nessuno): Domanda che verrà posta soltanto all’inizio dell’interazione e non a ogni singolo piatto come nel caso delle altre domande;
- Tipo del primo piatto? (pasta, riso): Per indicare se come primo piatto preferisca un piatto di pasta o di riso;
- Sapore del primo piatto? (leggero, medio, forte o piccante): Per indicare che tipo di sapore preferisce per il primo piatto;
- Difficolta del primo piatto? (facile, medio o difficile): per indicare il grado di difficolta del primo piatto;
- Costo del primo piatto? (basso, medio o alto): per indicare il costo desiderato per il primo piatto;
- Tipo del secondo piatto? (carne, pesce, verdura, legumi);
- Sapore del secondo piatto? (leggero, medio, forte o piccante);
- Difficolta del secondo piatto? (facile, medio o difficile);
- Costo del secondo piatto? (basso, medio o alto);
- Tipo di dessert? (cioccolata, frutta o crema);
- Temperatura a cui va servito? (caldo, normale, freddo): dato che dare una risposta sul sapore del dessert potrebbe risultare tedioso abbiamo preferito sostituirlo con la temperatura a cui va servito;
- Difficolta del dessert? (facile, medio o difficile);
- Costo del dessert? (basso, medio o alto).

Valutazione

Il sistema dopo aver raccolto le risposte dell'utente fornirà il menu che sarà composto da: un primo, un secondo e un dessert; di questi avrà il nome del piatto e il link che lo ricondurrà alla pagina dove è presente la ricetta con preparazione e ingredienti necessari. Questo è possibile grazie al utilizzo delle query SPARQL che utilizzerà le risposte date dal utente come parametri per fornire risposte tramite il comando "**default_world.sparql()**"

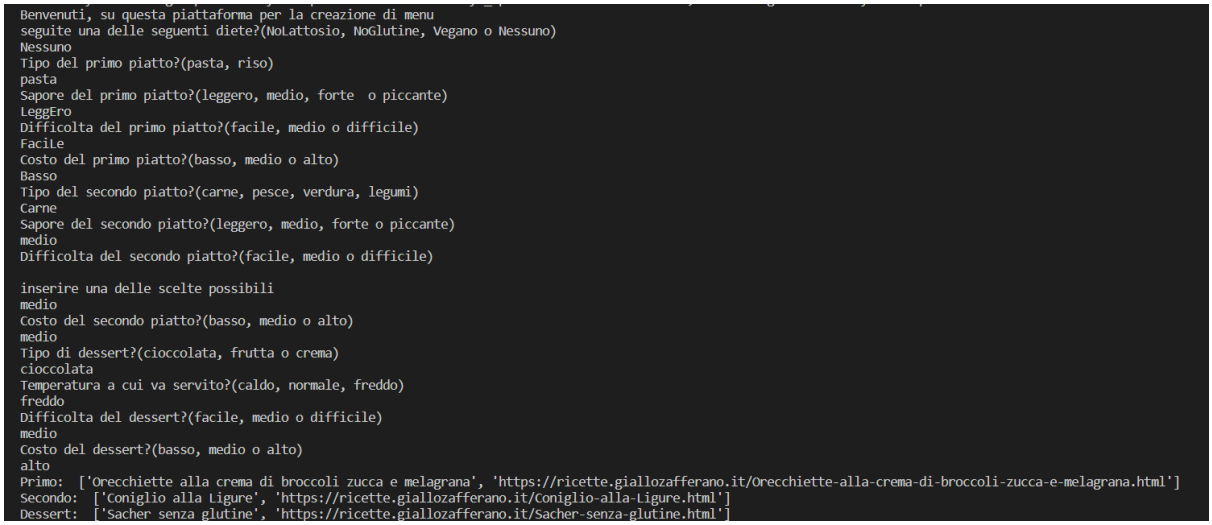


Figura 3: risultato della interazione con l'utente con risposta positiva

Come mostrato in figura (Figura 3) avremo a seguito delle risposte, una lista che andrà a fornire ciò che desiderato dall'utente. Va però fatto notare che non abbiamo trovate abbastanza ricette per coprire tutti i possibili parametri ma in caso l'utente ricada in uno di questi possibili casi sarà avvisato dal sistema (Figura 4)

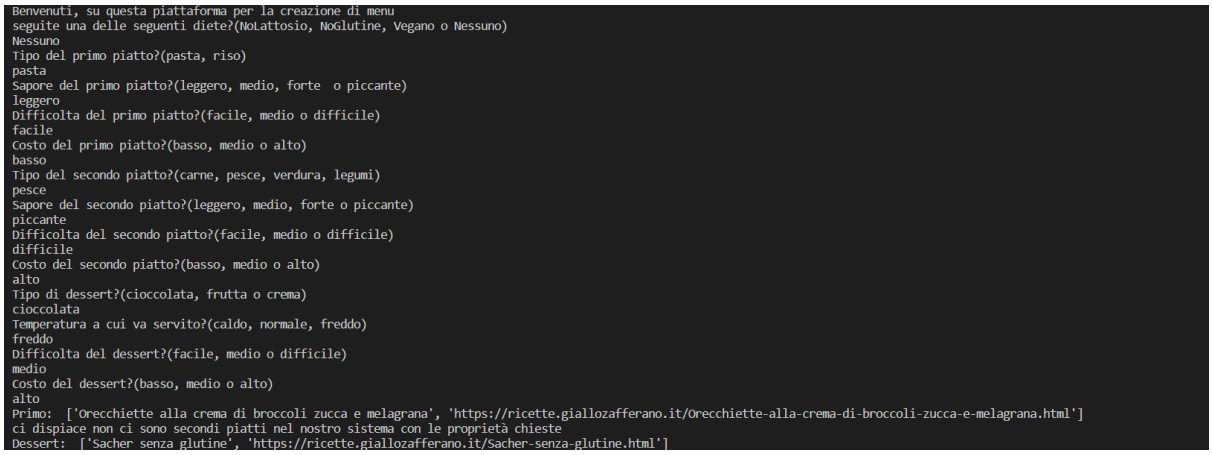


Figura 4: risultato della interazione con l'utente con risposta negativa

Classificazione Difficoltà

Sommario

Una sezione a parte è stata dedicata ad un esperimento indipendente dal Sistema Esperto presentato: si è trovato un metodo per costruire un dataset di diversi piatti a partire dalle informazioni presenti nelle pagine del sito "https://www.giallozafferano.it/ " e si è pensato di utilizzare diversi **modelli di classificazione** per cercare di individuare quali feature presenti nel dataset sono utili per prevedere un possibile target per il task: la feature **Difficoltà,** che per un dato piatto può essere:

- Molto Facile
- Facile
- Media
- Difficile
- Molto Difficile

Strumenti utilizzati

Per questo progetto abbiamo utilizzato i seguenti strumenti:

- **Python**: Linguaggio di programmazione ad alto livello con molte librerie che vanno a semplificare il lavoro, tra le più importanti utilizzate citiamo le seguenti:
    - **Graphviz**: è un software open source per la visualizzazione di grafi
    - **Pandas**: è uno strumento open source per l'analisi e la manipolazione dei dati. È stata usata principalmente per importare il dataset all'interno di python
    - **Json**: un formato adatto all'interscambio di dati fra applicazioni client/server
    - **Urllib3:** un potente e user-friendly client HTTP per python
    - **Csv:** un modulo che implementa classi per la lettura e scrittura di dati tabulari nel formato CSV
    - **Sklearn**: strumento semplice ed efficiente per l'analisi predittiva dei dati. Di questa libreria sono state utilizzate diverse metriche (accuracy_score, f1_score, ecc.) e diversi modelli (DecisionTreeClassifier, GradientBoostingClassifier, ecc.)
    - **Matplotlib**: è una libreria utile nel creare visualizzazioni statistiche, animate ed interattive un python

Decisioni di progetto

La nostra ontologia è stata sviluppata prendendo spunto da un'idea trovata su *github.com* il quale obbiettivo è quello di creare un dataset basato su una vasta gamma di ricette culinarie presenti sul sito web *giallozafferano.it* andando a riportare il nome, gli ingredienti

ed il procedimento di preparazione dei diversi piatti. Il codice originale dal quale ci siamo ispirati e il relativo repository si trovano all'indirizzo web *https://github.com/lorcalhost/RNN-ICR*.

Il nostro dataset è invece composto da cinque classi:

- **Nome**: rappresenta il nome del piatto
- **Difficoltà**: rappresenta il livello di difficoltà nel preparare quel piatto, che può essere:
    - Molto Facile
    - Facile
    - Medio
    - Difficile
    - Molto Difficile
- **Costo**: rappresenta il costo del piatto, che può essere:
    - Molto Basso
    - Basso
    - Medio
    - Alto
    - Elevato
    - Molto Elevato
- **Diete**: rappresenta il tipo di dieta seguita dal piatto, che può essere:
    - Nessuna
    - Vegetariana
    - Light
    - No Lattosio
    - No Glutine

      (ovviamente per un singolo piatto possono essere indicate più diete contemporaneamente)
- **Ingredienti**: rappresenta l'insieme di ingredienti necessari per preparare tale piatto, che possono essere:
    - Uova
    - Burro
    - Latte
    - Pesce
    - ecc…

      (non andiamo a riportare tutti gli ingredienti presenti perché troppi)

	Nome	Difficolta	Costo	Diete	Ingredienti
0	Focaccia (fagassa) alla genovese	Media	Basso	Senza lattosio; Vegetariano;	Farina 00; Farina Manitoba; Acqua; Sale fino; ...
1	Arancini di riso	Media	Basso	NaN	Zafferano; Burro; Riso vialone nano; Sale fino...
2	Polpettine di tonno e ricotta	Facile	Basso	NaN	Tonno sott'olio; Ricotta vaccina; Acciughe (al...
3	Piadina Romagnola	Facile	Molto basso	Senza lattosio;	Farina 00; Strutto; Acqua; Sale fino; Bicarbon...
4	Casatiello napoletano	Difficile	Medio	NaN	Acqua; Farina 00; Lievito di birra fresco; Sal...
...	...	...	...	...	...
4396	Piadina con erbette patate e lardo di Colonnata	Facile	NaN	NaN	Piadine; Erbette; Patate; Lardo di Colonnata; ...
4397	Insalata con polpette di ceci e bacon	Facile	Basso	Senza lattosio;	Ceci precotti; Bacon; Rosmarino; Uova; Parmigi...
4398	Piadina con gamberi, fiori di zucca e insalata	Facile	NaN	NaN	Piadine; Fiori di zucca; Gamberi; Ravanelli; R...
4399	Burger di riso e barbabietola con salsa di Gra...	Media	NaN	Vegetariano;	Barbabietole; Riso Ribe; Panini integrali; Lat...
4400	Burrito	Facile	Medio	NaN	Farina 00; Acqua; Strutto; Lievito istantaneo ...

Figura 5:esempio dataset

Data la presenza di diversi tipi di features, si sono adottati tre approcci (tutti basati sulla libreria di python Scikit Learn):

- Il primo approccio è stato scelto in mancanza di una conoscenza approfondita delle caratteristiche utili per la classificazione. In questo caso si sono sperimentati modelli di classificazione basati su **alberi decisionali**, in modo tale da effettuare una selezione automatica delle feature più importanti
    - In Scikit Learn, gli alberi di decisione supportano unicamente feature numeriche. Per cui, è stato necessario modificare il dataset prima di eseguire la classificazione. Sono stati quindi utilizzati diversi modelli come DecisionTreeClassifier, RandomForest, AdaBoost e GradientBoostingClassifier
- Una volta apprese quali erano le feature più rilevanti, ovvero il Costo e il Numero di Ingredienti (specificheremo dopo da dove proviene questa feature), si è deciso di procedere con l'utilizzo dell'algoritmo **K-Nearest-Neighbors** per scoprire se i piatti con lo stesso costo e lo stesso numero di ingredienti ne condividono la difficoltà
- Infine, si è provato ad addestrare un classificatore **Naive Bayes Gaussian** con le feature rilevanti. Questo tipo di classificazione considera separatamente i valori possibili di ogni caratteristica per il calcolo delle probabilità condizionate, non richiedendo che sia definita una relazione d'ordine

Le metriche per la valutazione dei diversi modelli sono tutte mediate sul numero di modelli utilizzati.

- In alcuni casi si è studiato l'andamento delle prestazioni dei modelli al variare di qualche iperparametro: per ogni valore dell'iperparametro è stata effettuata una misurazione ed è stata calcolata la media delle prestazioni

Prima di passare alla spiegazione dei diversi approcci alla task di classificazione, spieghiamo come il dataset è stato modificato in modo tale da renderlo ottimale per i diversi modelli.

Modifica Dataset

Per prima cosa è stata eliminata la feature del **Nome** in quanto inutile per la classificazione. Successivamente, le caratteristiche **Difficoltà** e **Costo** sono state modificate rendendole

numeriche anziché alfabetiche. Per la feature **Difficoltà** è stato assegnato un valore numerico in base al livello indicato precedentemente, nella seguente maniera:

- Molto Facile = 0
- Facile = 1
- Media = 2
- Difficile = 3
- Molto Difficile = 4

Per la feature **Costo** è stata eseguita la stessa modifica nel seguente modo:

- Molto Basso = 0
- Basso = 1
- Medio = 2
- Alto = 3
- Elevato = 4
- Molto Elevato = 5

Per la feature **Diete** si è optato nel creare delle caratteristiche booleane per ogni tipo di dieta ed assegnare il valore '1' nel caso in cui il piatto presentasse quella specifica dieta, altrimenti '0'. Successivamente la feature originale **Diete** è stata cancellata.

Per la feature **Ingredienti**, dato il gran numero di ingredienti differenti presenti nel dataset, si è scelto di creare una nuova caratteristica chiamata **NumIngredienti** la quale è stata riempita con il valore del numero di ingredienti presenti per ogni piatto. Successivamente, la feature originale **Ingredienti** è stata cancellata.

	Difficolta	Costo	NumIngredienti	Dieta_Nessuna	Dieta_Light	Dieta_Vegetariano	Dieta_NoLattosio	Dieta_NoGlutine
0	2	2	9	0	0	1	1	0
1	2	2	19	1	0	0	0	0
2	1	2	12	1	0	0	0	0
3	1	1	6	0	0	0	1	0
4	3	3	11	1	0	0	0	0
...	...	...	...	...	...	...	...	...
4396	1	0	10	1	0	0	0	0
4397	1	2	12	0	0	0	1	0
4398	1	0	14	1	0	0	0	0
4399	2	0	11	0	0	1	0	0
4400	1	3	23	1	0	0	0	0

Figura 6:Esempio del dataset modificato

Alberi di decisione

Qui di seguito sono presentati i valori di importanza delle diverse features derivate dall'addestramento degli alberi. La performance dei classificatori è stata misurata al variare della profondità dell'albero utilizzato.

Feature importance (la somma dell'importanza di tutte le feature è circa 1)

1. NumIngredienti: 0.371571
2. Costo: 0.367695
3. Dieta_NoLattosio: 0.058123
4. Dieta_Vegetariano: 0.055694
5. Dieta_NoGlutine: 0.049437
6. Dieta_Light: 0.035894
7. Dieta_Nessuna: 0.028253

Random Forest

Si è sperimentata la tecnica di bagging RandomForest al variare del numero di alberi appresi (con profondità massima 30) e si è sperimentato anche l'andamento dell'accuratezza predittiva al variare della profondità degli alberi

Feature importance

1. NumIngredienti: 0.51633
2. Costo: 0.28763
3. Dieta_NoGlutine: 0.042553
4. Dieta_NoLattosio: 0.037331
5. Dieta_NoLattosio: 0.036393
6. Dieta_Light: 0.023409
7. Dieta_Nessuna: 0.023021

AdaBoost

Si è sperimentato sulla base del numero di weak learner (alberi) utilizzati nell'apprendere il modello di boosting

Feature importance

1. NumIngredienti: 0.605583
2. Costo: 0.189543
3. Dieta_Light: 0.062134
4. Dieta_NoLattosio: 0.057967
5. Dieta_Vegetariano: 0.037643
6. Dieta_NoGlutine: 0.013797
7. Dieta_Nessuna: 0.00000

Gradient Boosting

Anche in questo caso si è sperimentato sulla base del numero di weak learner (alberi) utilizzati nell'apprendere il modello di boosting

Feature importance

1. Costo: 0.575197
2. NumIngredienti: 0.277802
3. Dieta_NoLattosio: 0.039218
4. Dieta_NoGlutine: 0.033443
5. Dieta_Nessuna: 0.023063
6. Dieta_Light: 0.012479
7. Dieta_Vegetariano: 0.005465

K-NN

Dai risultati degli alberi di decisione si osserva che le feature più importanti per tutti i modelli sono il **Costo** e il **NumIngredienti**, quindi il K-NN è stato sostanzialmente eseguito sul dataset avente 3 features:

- Features di input: *Costo*, *NumIngredienti*
- Feature di output: ***Difficoltà***

Si sono misurate le prestazioni del classificatore al variare del suo unico parametro, k.

Naive Bayse Gaussiano

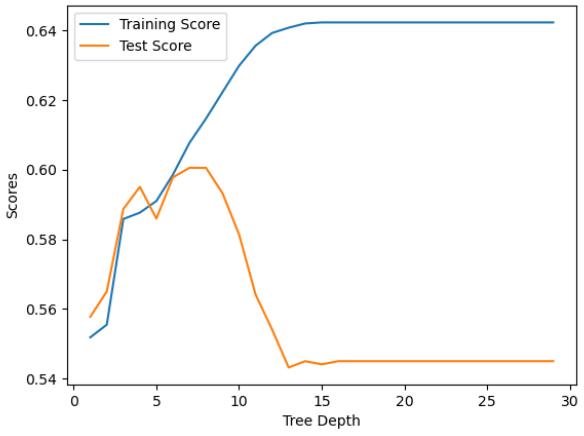
Per l'addestramento del **Naive Bayse Gaussiano** si sono utilizzati sempre e solo le due features più importanti, come nel caso del K-NN.

Valutazioni

Alberi di decisione

Qui di seguito sono presentati i grafici relativi all'accuratezza dei classificatori, valutata tramite lo score di Scikit Learn, sia sui dati di training (linea blu) che sui dati di test (linea arancione). Non sono state rilevate differenze nelle prestazioni dei modelli al variare dei criteri di selezione (**log_loss**, **gini**, **entropy**) utilizzati per lo split. In seguito, per comodità, verrà sempre utilizzato il criterio di selezione '**gini**'.

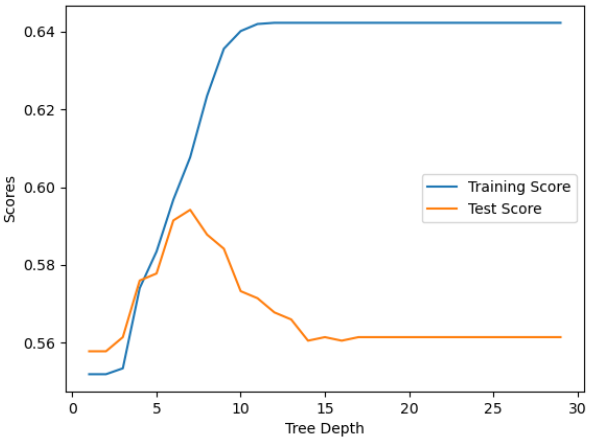
Accuratezza	0.5415832575
Precisione	0.0185926599
Richiamo	0.0333333333
F1-score	0.0238707165



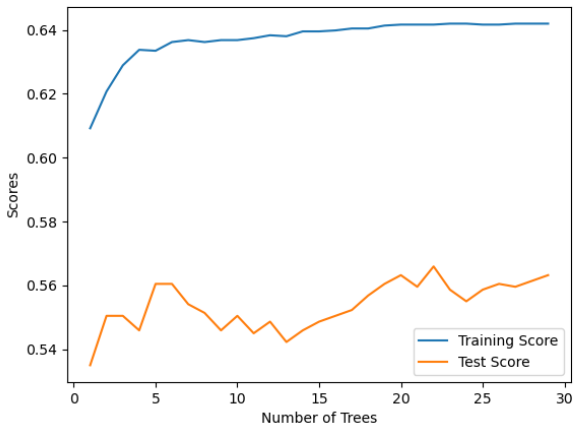
Conclusioni: il modello rimane su una accuratezza discreta e all'aumentare della profondità tende unicamente a overfittare i dati.

Random Forest

Accuratezza	0.5353958143
Precisione	0.0170731096
Richiamo	0.0112106681
F1-score	0.0124485711



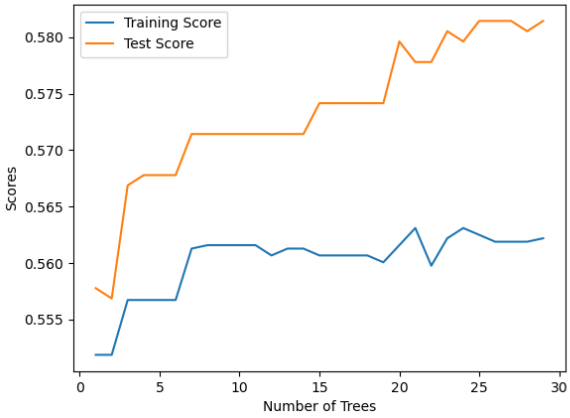
Accuratezza	0.5482559902
Precisione	0.0185926599
Richiamo	0.0333333333
F1-score	0.0238707165



Conclusioni: il modello delle Random Forest risulta simile agli alberi decisionali con un overfitting leggermente minore all'aumentare del numero di alberi

Ada Boost

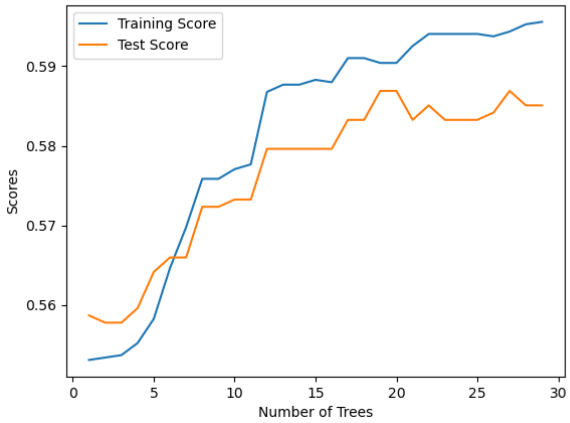
Accuratezza	0.5542917804
Precisione	0.0185926599
Richiamo	0.0333333333
F1-score	0.0238707165



Conclusioni: nel modello AdaBoost l'accuratezza dei dati di test è migliore rispetto a quella dei dati di training al contrario degli altri modelli ma rimane comunque presente l'overfitting.

Gradient Boosting

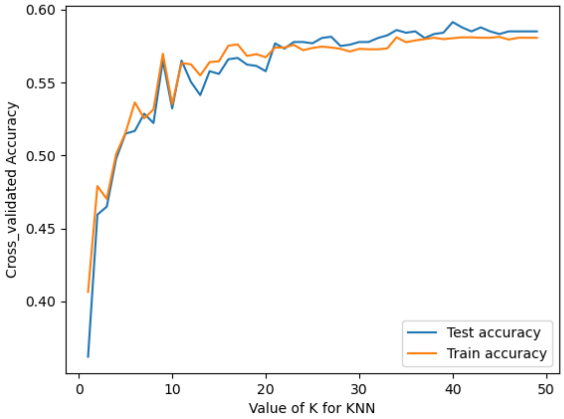
Accuratezza	0.5572945101
Precisione	0.0259714632
Richiamo	0.0222222222
F1-score	0.0202756672



Conclusioni: il modello Gradient Boosting è quello che presenta meno overfitting rispetto a tutti gli altri (dovuto alla minore distanza tra le due linee nel grafico) ed è l'unico modello nel quale varia un minino l'ordine d'importanza delle diverse feature. Ma nonostante tutto, l'accuratezza rimane simile agli altri modelli.

K-NN

Accuratezza	0.549526699
Precisione	0.0051333638
Richiamo	0.0050305666
F1-score	0.0048027258



Conclusioni: Dal grafico possiamo notare come l'overfitting è presente in maniera molto minore rispetto ai modelli degli alberi, inoltre si nota che all'aumentare del valore di K l'accuratezza sui dati di training diventi migliore rispetto a quella sui dati di test. Il K-NN, utilizzando solo due feature, si è rivelato accurato allo stesso livello degli alberi decisionali.

Naive Bayse Gaussiano

Sui dati di training l'accuratezza è 0.5442961165, mentre sui dati di test

Accuratezza	Precisione	Richiamo	F1-score
0.5432211101	0.34741622	0.3404895318	0.2737593532

Conclusioni: Anche qui, le prestazioni solo simili agli altri modelli, tranne per i valori di *Precisione*, *Richiamo* e *F1-score* che sono migliori rispetto a tutti gli altri.

Conclusioni

Il sistema esperto può essere ampliato sia per quanto riguarda le domande da porre all'utente (chiedere tutti gli ingredienti presenti nel piatto ad esempio) o anche inserire altri individui per andare a coprire tutte le possibili opzioni, allargando con informazioni da siti diversi da "giallozafferano.it".

I risultati ottenuti dai modelli di classificazione risultano accettabili, considerando la natura non booleana della feature obbiettivo e della scarsa quantità di feature importanti per la previsione della difficoltà. Si potrebbe provare ad aggiungere una feature per la presenza di ogni singolo ingrediente e ritentare l'addestramento dei modelli e la predizione della feature.