FOOD LABELING SYSTEM NUTRI-SCORE

Affuso Alessio, Amoruso Giuseppe, Perruggini Andrea

INDICE

1. INTRODUZIONE

Il sistema è in grado di analizzare i valori nutrizionali basilari (kCal, Proteine, Grassi e Carboidrati) di un alimento in input e stimare un valore di Nutri-score che va dalla A alla E, predire se l'alimento sia salutare o meno e consigliare degli alimenti simili nutrizionalmente in base ai valori forniti dall'utente.

Il Nutri-score è un sistema di etichettatura dei prodotti alimentari che serve a semplificare l'identificazione dei valori nutrizionali di un alimento, utilizzando due scale correlate:

- Una scala cromatica divisa in cinque gradazioni;
- Una scala alfabetica dalla A alla E;

Il calcolo del punteggio tiene conto di sette diversi parametri di informazioni nutritive per 100g di cibo e 100ml di bevande.
Gli aspetti positivi includono il contenuto di frutta, verdura, legumi, noci, alcuni oli, fibre alimentari e proteine facendo tendere la scala verso la gamma di colore verde. Al contrario, quanto più zucchero, sale, acidi grassi saturi e valore energetico contiene un alimento, più il punteggio tende verso la gamma rossa.

2. DATASET

E' stato utilizzato un dataset scaricato dal seguente sito: Open Food Facts .
In seguito, è stato necessario modificare questo dataset allo scopo di rimuovere i duplicati e cercare di ottenere una copertura su ogni categoria di alimento, ottenendo un totale di 648 alimenti.

3. STRUMENTI

È stato utilizzato Pycharm come Ide e Python come linguaggio di programmazione.
Sono state utilizzate le seguenti **librerie**:

- **Sklearn:** costruzione del KNN classifier, Random Forest per il task di classificazione, K Means per l'individuazione degli alimenti simili;

- **Pgmpy:** creazione di una Rete Bayesiana per il calcolo probabilistico sulla bontà in termini nutrizionali dell'alimento;

- **Pandas:** nella programmazione per computer, Pandas `e una libreria software scritta per il linguaggio di programmazione Python per la manipolazione e l'analisi dei dati. In particolare, offre strutture dati e operazioni per manipolare tabelle numeriche e serie temporali.

4 OTTIMIZZAZIONE DEGLI IPERPARAMETRI

Al fine di rendere notevolmente alta l'accuratezza di ciascun classificatore utilizzato è stato seguito un procedimento di ottimizzazione degli iperparametri. Partiamo dal presupposto che ciascun modello di classificazione prevede la presenza di parametri opportunamente passati in fase di costruzione di un determinato modello; dunque, ciascun valore associato a questi ultimi prendi il nome di iperparametro. Se non esplicitati, ai parametri verranno associati valori di default, che molto spesso non permettono al modello di esaltare la sua massima accuratezza.

- **Exhaustive grid search:** Tale metodo, fornito da GridSearchCV, genera in maniera esaustiva i possibili candidati (iperparametri) attraverso una griglia di valori specificata opportunamente dal parametro "param_grid", caratterizzato da un range di valori per ogni singolo parametro specificato dall'utente. In maniera del tutto

automatica, vengono valutate tutte le possibili combinazioni di assegnazioni degli iperparametri e viene mantenuta la combinazione migliore. Al termine di tale processo, verranno mostrati quelli che sono gli iperparametri migliori per un determinato modello di classificazione. Delle tante procedure utili ai fini di tale topic, sono state scelte proprio queste due in quanto la prima si basa su un procedimento del tutto "manuale" e fortemente esplicativo, visto l'utilizzo di un grafico, il secondo invece è del tutto "automatico", tentando ogni combinazione, sulla base dell'accuratezza raggiunta in ogni singolo tentativo.

4.1 CROSS VALIDATION

La cross-validation è una tecnica statistica utilizzabile in presenza di una buona numerosità del campione osservato. In particolare, la convalida incrociata cosiddetta k-fold consiste nella suddivisione dell'insieme di dati totale in k parti di uguale numerosità e, a ogni passo, la k^a parte dell'insieme di dati viene a essere quella di convalida, mentre la restante parte costituisce sempre l'insieme di addestramento. Così si allena il modello per ognuna delle k parti, evitando quindi problemi di sovradattamento, ma anche di campionamento asimmetrico (e quindi affetto da distorsione) del campione osservato, tipico della suddivisione dei dati in due sole parti (ossia addestramento/convalida).

5. MODELLI DI CLASSIFICAZIONE UTILIZZATI

Al fine di ottenere una predizione sui nuovi esempi, sono stati applicati modelli di classificazione basati su apprendimento supervisionato, derivati dalla libreria sklearn. L'idea di utilizzare più modelli ha avuto lo scopo di valutare l'accuratezza di ogni singolo modello in fase di test.

- **K-Nearest Neighbors**: Dopo una fase di ottimizzazione degli iperparametri, è stato effettuato il training del modello, tramite k-folds cross-validation (26 folds).

Sono state utilizzate 25 folds per il training e la restante per il testing, per un totale di 26 risultati di accuracy. I valori così ottenuti sono stati mediati.

Il Knn è un algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti basandosi sulle caratteristiche dei k oggetti più vicini a quello considerato. Un oggetto è classificato in base alla maggioranza dei voti dei suoi k vicini. k è un intero positivo tipicamente non molto grande. La scelta di k dipende dalle caratteristiche dei dati. Generalmente all'aumentare di k si riduce il rumore che compromette la classificazione. Al fine dell'apprendimento lo spazio multidimensionale viene partizionato in regioni in base alle posizioni e alle caratteristiche degli oggetti di apprendimento, rappresentati come vettori. Un oggetto è assegnato alla classe C se questa è la più frequente fra i k esempi più vicini all'oggetto sotto esame, la vicinanza si misura in base alla distanza fra punti. I vicini sono presi da un insieme di oggetti per cui è nota la classificazione corretta.

- **Random Forest**: *Anche in questo caso abbiamo ottimizzato i parametri, con la stessa tecnica vista precedentemente, ed abbiamo effettuato la k-fold cross validation, per mediare i vari risultati di accuracy. Le fold utilizzate sono e stesse utilizzate nel Knn.*
  Abbiamo poi variato il contenuto delle folds per 11 volte, ricalcolando i risultati ottenuti da entrambi gli algoritmi, e mediato ulteriormente i risultati ottenuti.

  È un modello d'insieme ottenuto dall'aggregazione tramite bagging di alberi di decisione. Esso è un meta-stimatore che si adatta ad una serie di alberi decisionali addestrati su vari sotto-campioni del dataset e utilizza la media di ogni singolo output di ogni albero per migliorare l'accuratezza predittiva e il controllo del sovradattamento. Il Random Forest deve essere dotato di due matrici: una matrice X sparsa che contiene i campioni di addestramento e una matrice Y di dimensioni che contiene i valori target.

Dopo aver analizzato i risultati ottenuti dalla precision e dal recall dei 2 classificatori, abbiamo deciso di utilizzare il Knn.

6. RETE BAYESIANA

La rete Bayesiana viene utilizzata per predire se l'alimento che viene inserito in input dall'utente sia salutare o meno, con una certa probabilità. Le dipendenze tra le feature sono state individuate utilizzando la matrice di correlazione.

Una rete bayesiana è un modello grafico probabilistico che rappresenta un insieme di variabili stocastiche con le loro dipendenze condizionali attraverso l'uso di un grafo aciclico diretto (DAG).

Formalmente le reti Bayesiane sono grafi diretti aciclici i cui nodi rappresentano variabili casuali in senso Bayesiano: possono essere quantità osservabili, variabili latenti, parametri sconosciuti o ipotesi. Gli archi rappresentano condizioni di dipendenza; i nodi che non sono connessi rappresentano variabili che sono condizionalmente indipendenti tra di loro. Ad ogni nodo è associata una funzione di probabilità che prende in input un particolare insieme di valori per le variabili del nodo genitore e restituisce la probabilità della variabile rappresentata dal nodo.

7. ALGORITMO DI CLUSTERIZZAZIONE

L'algoritmo tiene in considerazione le seguenti features: calorie, carboidrati, proteine, grassi, sale e suddivide il dataset in cluster.
Dopo aver acquisito il nuovo alimento dall'utente, il sistema dovrà restituire una serie di alimenti simili a quello dato in base al cluster.

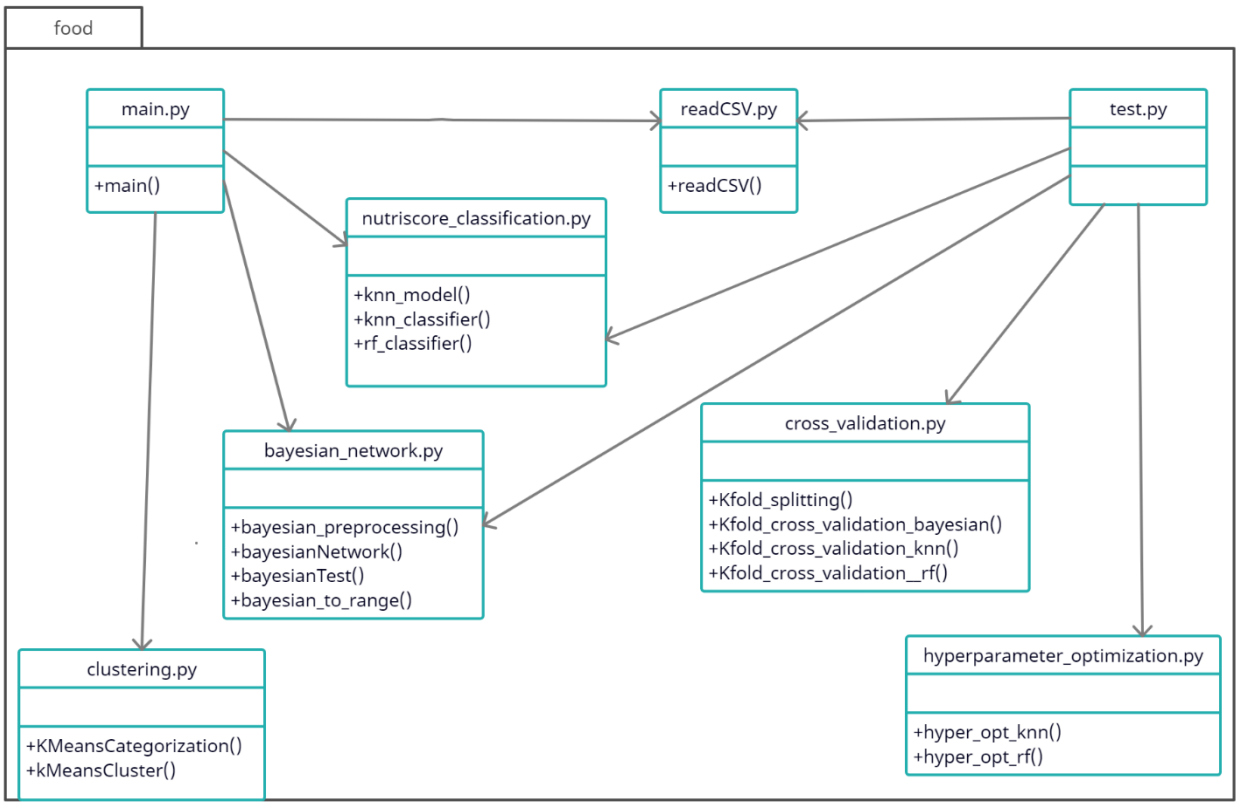
Nell'apprendimento non supervisionato non si ha a disposizione un training set con una feature-target e si necessita di ricostruire un classificatore naturale dei dati.
Per fare ciò si utilizza il Clustering. Una forma generale dell'algoritmo in questione:

- Partiziona gli esempi in classi (cluster);
- Ogni classe predice i valori delle feature per gli esempi contenuti;

Ogni raggruppamento (clustering) ha un suo errore di predizione associato. Il migliore è quello con l'errore minimo.

Noi utilizziamo il K-MEANS:
K-means si basa sui cosiddetti centroidi. Il centroide è un punto appartenente allo spazio delle feature che media le distanze tra tutti i dati appartenenti al cluster ad esso associato. Rappresenta quindi una sorta di baricentro del cluster ed in generale, proprio per le sue caratteristiche, non è uno dei punti del dataset.

8. UML



La figura mostra l'utilizzo dei vari moduli da parte dei moduli main e del test, con l'elenco dei
metodi ad ognuno associati.
Di seguito sono riportate i vari metodi e la rispettiva descrizione.

Modulo nutriscore_classification.py

knn_model(df, col_list, hypers, values):

"""

predizione tramite classificatore KNN
:param df: dataframe in input
:param col_list: nomi delle features da considerare
:param hypers: iperparametri ottimizzati
:param values: valori da predire
:return: nutriscore predetto
"""

knn_classifier(df, col_list, folds, hyp_opt: bool = False):

"""

testa un classificatore KNN
:param df: dataframe in input
:param col_list: lista di nomi di features da considerare
:param folds: numero di fold
:param hyp_opt: True se ottimizzazione degli iperparametri richiesta, False altrimenti
:return: valore medio di accuracy su tutte le fold
"""

rf_classifier(df, col_list, folds, hyp_opt: bool = False):

"""

testa un classificatore RF
:param df: dataframe in input
:param col_list: lista di nomi di features da considerare
:param folds: numero di fold
:param hyp_opt: True se ottimizzazione degli iperparametri richiesta, False altrimenti
:return: valore medio di accuracy su tutte le fold

Modulo bayesian_network.py

bayesian_preprocessing(food_df, values=None):

"""

operazioni preliminari da effettuare sul dataframe per renderlo idoneo ad una rete bayesiana
:param food_df: dataframe in input
:param values: None se non c'é bisogno di predizione

:return: dataset idoneo ad una rete bayesiana
"""

bayesianNetwork(food_df, values):

"""
previsione tramite rete bayesiana
:param food_df: dataframe in input
:param values: valori da predire
:return: stringa decisionale
"""

bayesianTest(food_df, folds):

"""
test di una rete bayesiana tramite cross-validation
:param food_df: dataframe in input
:param folds: numero di folds
:return: accuracy media
"""

values_to_range(new_food_df, f_old, f_val, i, cont, step):

"""
converte in range da 0 a 4 i valori di un dataframe
:param new_food_df: dataframe
:param f_old: nome precedente delle features
:param f_val: nome aggiornato delle features
:param i: percentuale
:param cont: contatore
:param step: passo
:return: dataframe trasformato
"""

Modulo clustering.py

kMeansCategorization(data, col_list):

"""
clustering del dataframe secondo features in input
:param data: dataframe su cui effettuare il clustering
:param col_list: nome delle features da considerare
:return: valori contenuti in un cluster
"""

kMeansCluster(df, col_list, values):

"""

predizione cluster dei valori in input
:param df: dataframe in input
:param col_list: nomi delle features per il clustering
:param values: valori in input
:return: stringa contenente valori appartenenti al cluster predetto
"""

Modulo readCSV.py

readCSV(path, separ):

"""
legge un file .csv e lo incapsula in un dataframe pandas
:param path: percorso file .csv da leggere
:param separ: separatore (carattere)
:return: dataframe contenente i dati estratti
"""

Modulo hyperparameter_optimization.py

hyper_opt_knn(X, y, folds, classifier: bool = True):

"""
ottimizzazione dei parametri di un modello KNN
:param X: X dataframe - valori noti
:param y: y column(s) - valori da predire
:param folds: numero di folds per la cross-validation
:param classifier: True se classificatore KNN, False se regressore KNN
:return: parametri ottimizzati
"""

hyper_opt_rf(X, y, folds):

"""
ottimizzazione dei parametri di un modello RF
:param X: X dataframe - valori noti
:param y: y column(s) - valori da predire
:param folds: numero di folds per la cross-validation
:return: parametri ottimizzati
"""

Modulo cross_validation.py

KFold_splitting(X, y, splits=10):

"""

divisione del dataset in train e test set
:param X: X dataframe - valori noti
:param y: y column(s) - valori da predire
:param splits: numero di folds da utilizzare
:return: lista delle varie combinazioni di folds (train/test sets)
"""

kFold_cross_validation_bayesian(X, y, splits=10):

"""

cross-validation per la rete bayesiana
:param X: X dataframe - valori noti
:param y: y column(s) - valori da predire
:param splits: numero di folds da utilizzare
:return: valore medio di accuracy
"""

kFold_cross_validation_knn(X, y, hypers, classifier: bool = True, splits=10):

"""

esegue cross validation utilizzando la tecnica K-fold su un knn Classifier o Regressor
:param hypers: valori ottimali degli iperparametri
:param X: X dataframe - valori noti
:param y: y column(s) - valori da predire
:param classifier: True se knn Classifier (default), False se knn Regressor
:param splits: numero di folds
:return: valore medio di accuracy
"""

kFold_cross_validation_rf(X, y, hypers, splits=10):

"""

esegue cross validation utilizzando la tecnica K-fold su un Random Forest
:param hypers: valori ottimali degli iperparametri
:param X: X dataframe - valori noti
:param y: y column(s) - valori da predire
:param splits: numero di folds
:return: valore medio di accuracy
"""

9. CONTATTI

Affuso Alessio	728856	a.affuso4@studenti.uniba.it
Amoruso Giuseppe	697808	g.amoruso43@studenti.uniba.it
Perruggini Andrea	699041	a.perruggini@studenti.uniba.it