Riorganizzazione dei telefoni cellulari

Componenti del gruppo

- Nicolò Pino, [MAT. 735405], n.pino@studenti.uniba.it

Link GitHub: https://github.com/Pinuz2001/MobileClassification.git

A.A 2023 - 2024

Indice

Introduzione

Questo progetto ha come obiettivo principale quello di riorganizzare dei telefoni cellulari presenti in un dataset secondo le loro caratteristiche, ovvero le features. Oltre a realizzare questa riorganizzazione, il sistema sarà capace di suggerire in quale categoria dovrà essere inserito un telefono cellulare che non è mai stato archiviato.

Il sistema è stato realizzato in Python. Grazie a questo linguaggio si è potuto fare largo uso di molte librerie utili allo scopo del progetto:

- **matplotlib**: libreria che offre la possibilità di stampare a video dei grafici
- **numpy**: libreria che gestisce gli array
- **pandas**: libreria che gestisce dei dataset
- **scikit_learn**: libreria che offre funzioni per l'apprendimento
- **imblearn**: libreria che offre la possibilità di effettuare oversampling SMOTE su dataset
- **pgmpy**: libreria che gestisce la rete bayesiana
- **pyswip**: libreria che offre servizi Prolog

In particolare, l'IDE utilizzato è PyCharm.

Architettura del progetto

Il progetto si costituisce de seguenti file.py, ognuno implementa una parte fondamentale del progetto (i file vengono esplicitati nell'ordine in cui devono essere eseguiti):

1. "preprocessing.py" → si occupa della pulizia del dataset e della rimozione delle features irrilevanti
2. "unsupervisedLearning.py" → si occupa di implementare la parte relativa all'apprendimento non supervisionato
3. "supervisedLearning.py" → si occupa di implementare la parte relativa all'apprendimento supervisionato
4. "supervisedLearningOversampling.py" → si occupa di effettuare l'oversampling del dataset e implementare nuovamente l'apprendimento supervisionato
5. "bayesianNetwork.py" → si occupa di implementare la parte di rete bayesiana ma lavorando su valori continui nel dataset
6. "bayesianNetworkDiscretized.py" → si occupa di implementare la parte di rete bayesiana con valori discreti nel dataset
7. "knowledgeBase.py" → si occupa della costruzione e interrogazione della KB

Gestione del dataset

Il dataset di partenza è stato trovato dal seguente link: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification/data?select=train.csv.

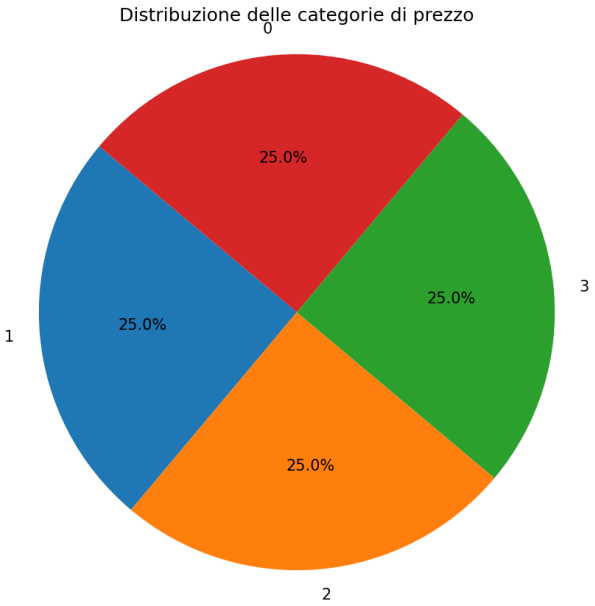
Il dataset in questione è formato da circa 2000 righe e dalle seguenti features:

- battery_power: rappresenta la potenza della batteria del telefono cellulare (espressa in mhA)
- blue: può assumere solamente i valori:
    - 0: se il telefono cellulare non è munito di bluetooth
    - 1: se il telefono cellulare è munito di bluetooth
- clock_speed: rappresenta la velocità con la quale il microprocessore esegue le istruzioni
- dual_sim: può assumere solamente i valori:
    - 0: se il telefono cellulare non ha una slot per la seconda sim
    - 1: se il telefono cellulare ha una slot per la seconda sim
- fc: rappresenta i megapixel della fotocamera frontale del telefono cellulare
- four_g: può assumere solamente i valori:
    - 0: se il telefono cellulare non appartiene alla generazione 4G
    - 1: se il telefono cellulare appartiene alla generazione 4G
- int_memory: rappresenta la memoria interna del telefono cellulare espressa in Gigabyte
- m_dep: rappresenta lo spessore del telefono cellulare espresso in cm
- mobile_wt: rappresenta il peso del telefono cellulare
- n_cores: rappresenta il numero di cores che il telefono cellulare possiede
- pc: rappresenta i pixel della fotocamera principale del telefono cellulare
- px_height: contiene la risoluzione in altezza espressa in pixel
- px_width: rappresenta la risoluzione in lunghezza espressa in pixel del telefono cellulare
- ram: rappresenta la misura della RAM del telefono cellulare espressa in Megabyte
- sc_h: rappresenta l'altezza dello schermo del telefono cellulare espressa in cm
- sc_w: rappresenta la lunghezza dello schermo del telefono cellulare espressa in cm
- talk_time: rappresenta la durata massima del telefono cellulare in chiamata espressa in ore
- three_g: può assumere solamente i valori:
    - 0: se il telefono cellulare non appartiene alle generazione 3G
    - 1: se il telefono cellulare appartiene alla generazione il 3G
- touch_screen: può assumere solamente i valori:
    - 0: se il telefono cellulare non è munito di touch screen
    - 1: se il telefono cellulare è munito di touch screen
- wifi: può assumere solamente i valori:
    - 0: se il telefono cellulare non possiede il Wi-Fi
    - 1: se il telefono cellulare possiede il Wi-Fi
- price_range: può assumere solamente i valori:
    - 0: se il telefono cellulare appartiene alla fascia di prezzo "Low cost"
    - 1: se il telefono cellulare appartiene alla fascia di prezzo "Medium cost"
    - 2: se il telefono cellulare appartiene alla fascia di prezzo "High cost"
    - 3: se il telefono cellulare appartiene alla fascia di prezzo "Very high cost"

Una volta caricato il dataset tramite la libreria pandas, il sistema procede con la pulizia del dataset e la rimozione di alcune features irrilevanti. In particolare:

1. Sono state rimosse le righe che contenevano dati mancanti e i duplicati
2. Essendo tutte le features numeriche, il sistema le normalizza utilizzando un oggetto MinMaxScaler in modo tale da ottenere numeri compresi tra 0 e 1. La normalizzazione Min-Max è molto semplice: si sottrae al valore di una feature il minimo tra i valori che assume la feature e si divide per la differenza tra il massimo e il minimo della feature.
3. Vengono rimosse le feature 'blue', 'dual_sim', 'four_g', 'three_g', 'touch_screen', 'wifi', 'talk_time', 'sc_h', 'sc_w' per ridurre la complessità del dataset e rendere più semplice la sua interpretazione e le sue previsioni.

Inizialmente i telefoni cellulari sono categorizzati in base alla loro fascia di prezzo (feature 'price_range') in questo modo:



Apprendimento non supervisionato

L'apprendimento non supervisionato è un tipo di apprendimento automatico dove il computer impara dai dati senza essere istruito esplicitamente su quali siano gli output desiderati. Ci sono principalmente due tipi di apprendimento non supervisionato:

- **Clustering**: Questo tipo di apprendimento cerca di raggruppare insieme dati simili in cluster o gruppi.
- **Riduzione della dimensionalità**: Qui, l'obiettivo è di ridurre il numero di variabili o caratteristiche nei dati mantenendo comunque le informazioni più importanti.

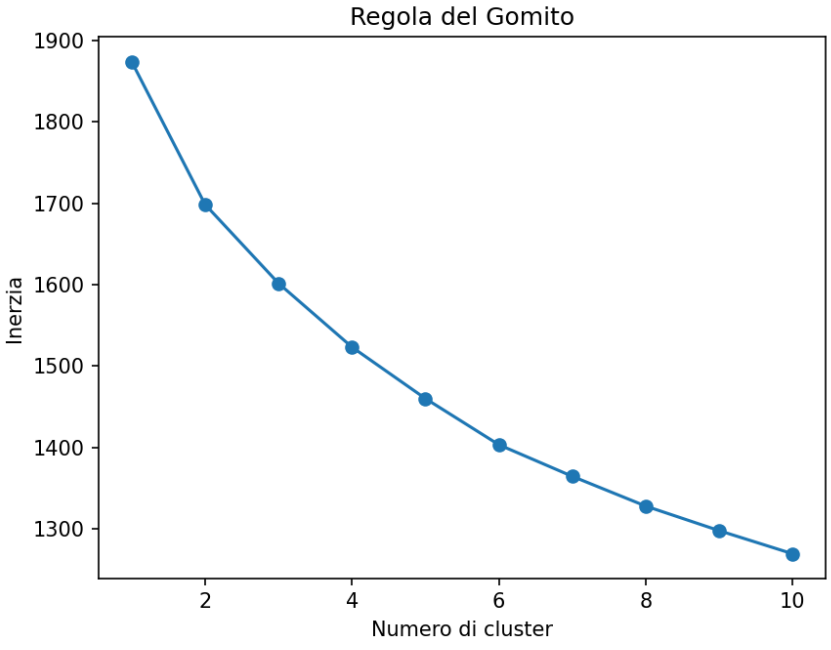
In questo caso, il sistema applica l'apprendimento non supervisionato di tipo clustering, ovvero cerca di catalogare i telefoni cellulari, non più secondo la fascia di prezzo, bensì in base alle caratteristiche (features) simili. Il tipo di clustering che il sistema esegue è detto hard clustering. L'hard clustering associa ad ogni esempio un cluster (insieme di dati simili). Un'alternativa sarebbe stata quella di eseguire il soft clustering, ovvero una tecnica che associa ad ogni esempio la probabilità di appartenenza ad ogni cluster.

La fase di apprendimento non supervisionato è stata supportata dalla libreria sklearn che ha effettuato il clustering e dalla libreria matplotlib per la stampa dei grafici.

L'algoritmo di hard clustering utilizzato dal sistema è il KMeans. Il funzionamento del KMeans può essere così riassunto:

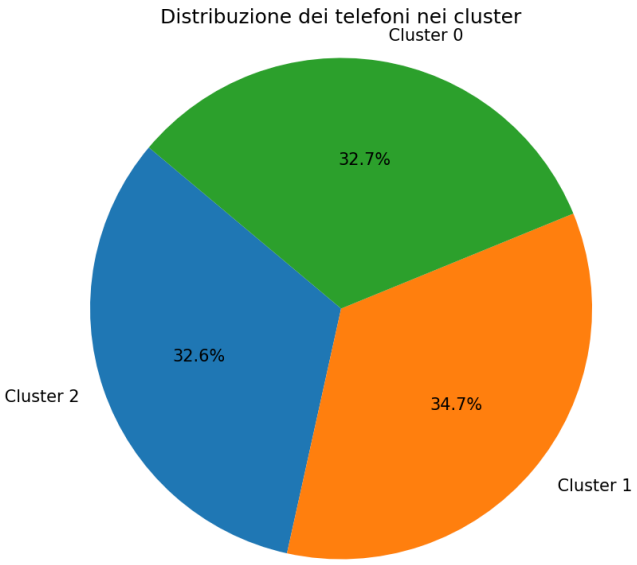
1. Inizializzazione: Si scelgono casualmente K punti all'interno del dataset come centroidi iniziali dei cluster.
2. Assegnazione dei punti ai cluster: Per ciascun punto nel dataset, si calcola la distanza dai centroidi e si assegna il punto al cluster il cui centroide è più vicino.
3. Aggiornamento dei centroidi: Una volta che tutti i punti sono stati assegnati ai cluster, si calcola il nuovo centroide di ciascun cluster come la media dei punti assegnati ad esso.
4. Ripetizione: I passaggi 2 e 3 vengono ripetuti fino a quando i centroidi non cambiano significativamente o fino a quando si raggiunge un numero massimo di iterazioni.

Per trovare il numero ideale di cluster da fornire in input all'algoritmo, il sistema segue una tecnica nota come "Regola del gomito":



Il numero ideale di cluster è 3.

L’hard clustering che ne segue è il seguente:



Apprendimento supervisionato

L'apprendimento supervisionato è una tecnica di apprendimento automatico dove un algoritmo impara a fare predizioni o a prendere decisioni basate su dati di input e output etichettati. Durante il processo di addestramento, l'algoritmo riceve un insieme di dati di addestramento che contiene sia le caratteristiche di input (features) che le etichette di output corrispondenti (features target). L'obiettivo è apprendere una funzione che mappa gli input agli output, in modo che possa fare predizioni accurate su nuovi dati.

Ci sono due tipi principali di apprendimento supervisionato: la classificazione, dove l'output è una categoria predefinita, e la regressione, dove l'output è un valore numerico.

Il sistema quando applica l’apprendimento supervisionato esegue le seguenti fasi:

1. Determinazione della migliore combinazione di iper-parametri per ogni modello da utilizzare per la classificazione
2. Addestramento dei modelli e test
3. Valutazione delle prestazioni dei modelli

Durante la fase di apprendimento supervisionato il sistema utilizza le librerie matplot e numpy per la stampa a video dei grafici relativi alle prestazioni dei modelli, la libreria sklearn per le funzioni di apprendimento, la libreria pandas per la gestione dei dataset e la libreria imblearn per l’oversampling del dataset.

La nuova categorizzazione, dopo aver effettuato l’apprendimento supervisionato, non sarà altro che la classificazione dei telefoni cellulari in categorie (o classi), dove queste ultime sono gli indici dei cluster ottenuti durante l’apprendimento non supervisionato.

Il sistema utilizza due modelli di apprendimento automatico:

- DecisionTree: Un albero decisionale è una struttura gerarchica a forma di albero dove ogni nodo interno rappresenta un test su un attributo, ogni ramo corrisponde a un'uscita di questo test e ogni foglia rappresenta una classe di output. L'obiettivo è dividere ricorsivamente il dataset in sottoinsiemi omogenei, rendendo decisioni basate sulle caratteristiche dei dati.
- RandomForest: Una random forest è un insieme di alberi decisionali, dove ciascun albero è costruito su un sottoinsieme casuale dei dati di addestramento e utilizza un sottoinsieme casuale delle caratteristiche. Durante la predizione, ciascun albero vota per la classe prevista e la classe con il maggior numero di voti viene scelta come output del modello.

1. Determinazione della migliore combinazione di iper-parametri per ogni modello da utilizzare per la classificazione

Ancora prima di iniziare l'addestramento dei modelli, è buona norma fissare quelli che sono gli iper-parametri dei modelli. Gli iper-parametri non vengo appresi durante la fase di addestramento ma devono essere necessariamente appresi prima in quanto la loro scelta influisce sulle prestazioni e sulla complessità del modello.
Per la scelta degli iper-parametri il sistema utilizza una tecnica di K-Fold Cross Validation. In particolare, la strategia adottata è la GridSearch con Cross Validation che restituisce la migliore combinazione possibile.

Gli iper-parametri in questione per quanto riguarda DecisionTree sono:

- Max_depth: rappresenta l'altezza massima dell'albero
- Min_samples_split: rappresenta il numero minimo di esempi affinché possa essere inserito un criterio di split. Se il numero è minore viene innestata una foglia
- Min_samples_leaf: rappresenta il numero minimo di esempi per poter creare una foglia

```
param_grid_dt = {
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

Gli iper-parametri in questione per quanto riguarda RandomForest sono:

- N_estimators: rappresenta il numero di alberi nella foresta
- Max_depth: come DecisionTree
- Min_samples_split: come DecisionTree
- Min_samples_leaf: come DecisionTree

```
param_grid_rf = {
    'n_estimators': [10, 20, 50],
    'max_depth': [5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

Gli iper-parametri migliori per i due modelli sono i seguenti:

```
Migliori iperparametri per Decision Tree: {'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5}
Migliori iperparametri per Random Forest: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 20}
```

2. Addestramento dei modelli e test

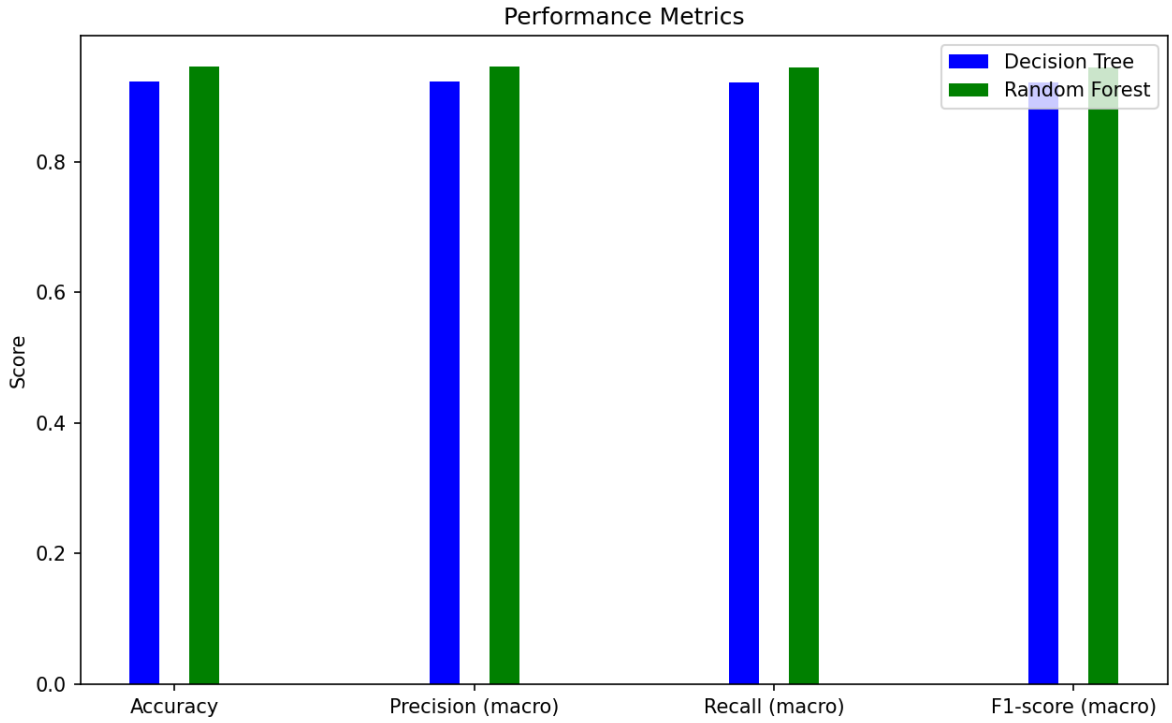
In questa fase i modelli sono stati addestrati utilizzando una K-Fold Cross Validation con k=5.

3. Valutazione delle prestazioni dei modelli

Per valutare le prestazioni del sistema si prendono in considerazione le metriche:

- Accuracy: La proporzione di predizioni corrette rispetto al totale delle predizioni fatte dal modello. È calcolata come il rapporto tra il numero di predizioni corrette e il numero totale di predizioni.
- Precision macro: È una media delle precisioni per ciascuna classe. La precisione per una classe specifica è il rapporto tra il numero di predizioni corrette per quella classe e il numero totale di predizioni fatte per quella classe.
- Recall macro: È una media dei richiami per ciascuna classe. Il richiamo per una classe specifica è il rapporto tra il numero di predizioni corrette per quella classe e il numero totale di istanze di quella classe nel dataset.
- F1-score macro: È una media armonica delle precisioni e dei richiami per ciascuna classe. È calcolato come il doppio del prodotto tra precisione e richiamo diviso per la somma di precisione e richiamo per ogni classe, e quindi media su tutte le classi.

I valori di ogni metrica per ciascun modello sono i seguenti:




```
Decision Tree:
Accuracy: 0.92 (+/- 0.02)
Precision (macro): 0.92 (+/- 0.02)
Recall (macro): 0.92 (+/- 0.02)
F1-score (macro): 0.92 (+/- 0.02)

Random Forest:
Accuracy: 0.95 (+/- 0.02)
Precision (macro): 0.95 (+/- 0.02)
Recall (macro): 0.94 (+/- 0.02)
F1-score (macro): 0.95 (+/- 0.02)
```

I valori per ogni metrica sono abbastanza buoni: questo potrebbe essere un segnale di overfitting. Per verificare la presenza di problemi di sovradattamento occorre valutare altre variabile come, per esempio, la varianza e la deviazione standard. In ogni caso possiamo osservare il tutto attraverso i grafici relativi alle curve di apprendimento per ogni modello.

L'overfitting si verifica quando un modello si adatta troppo ai dati di addestramento, catturando rumore o pattern casuali che sono rappresentativi della relazione generale tra le caratteristiche e l'output desiderato. Questo porta a una scarsa capacità di

generalizzazione del modello su nuovi dati non visti durante l'addestramento. In sostanza, il modello "impara troppo" dai dati di addestramento, memorizzandoli invece di comprendere i concetti sottostanti.
Tra le nuove variabili da considerare abbiamo:

- La varianza: La varianza è una misura della dispersione dei dati attorno alla loro media. Indica quanto i dati si discostano, in media, dal valore medio.
- La deviazione standard: La deviazione standard è la radice quadrata della varianza ed è un'altra misura della dispersione dei dati attorno alla loro media.

Per il DecisionTree:

```
Train Error Std: 0.004805798906009275, Test Error Std: 0.011592023119369639
```

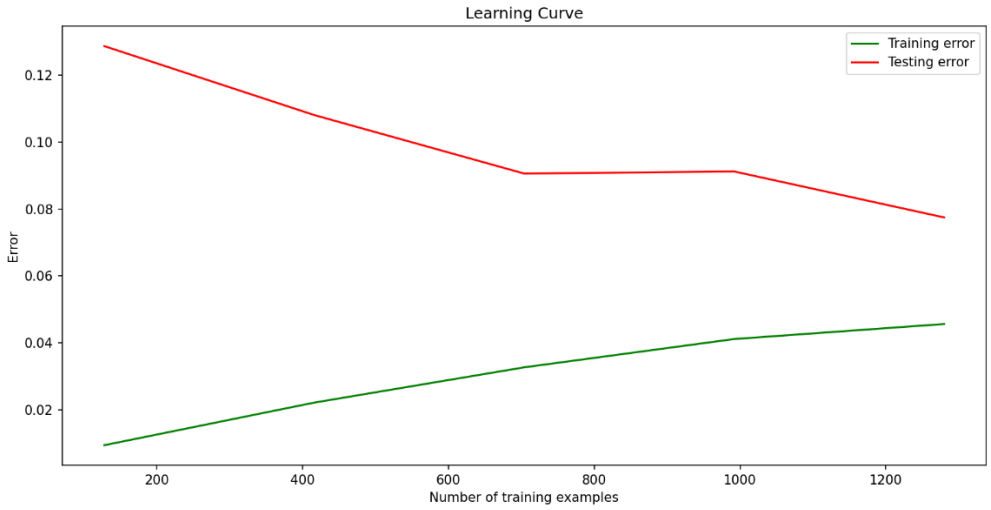
```
Train Error Var: 2.309570312499995e-05, Test Error Var: 0.00013437500000000022
```

Per il RandomForest:

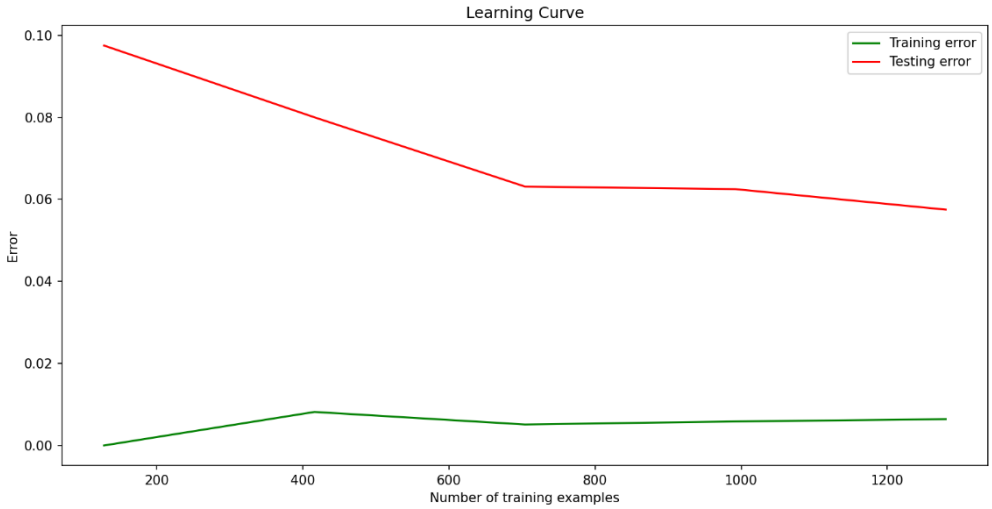
```
Train Error Std: 0.0022857404434886985, Test Error Std: 0.0119569540435681
```

```
Train Error Var: 5.2246093749999125e-06, Test Error Var: 0.00014296874999999951
```

Curva di apprendimento per il DecisionTree:



Curva di apprendimento per il RandomForest:

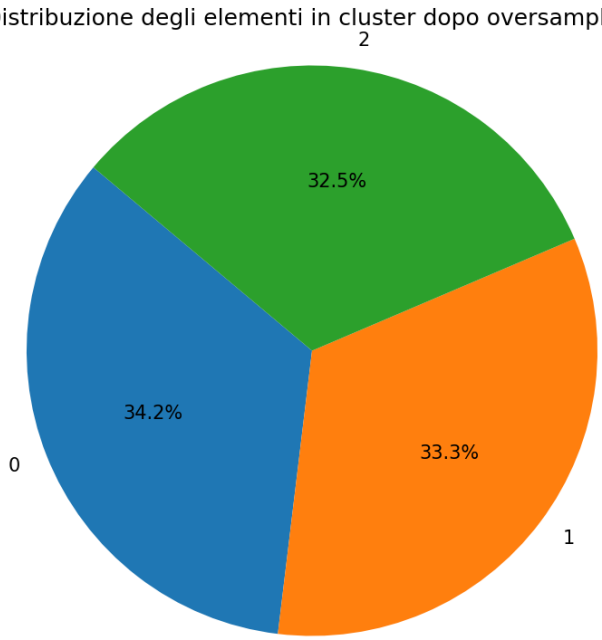


Analisi dei risultati

- Decision Tree → si può notare che la deviazione standard e la varianza hanno valori decisamente bassi. Tuttavia, il valore della varianza del test error è molto alto rispetto a quello del train error. Questa differenza potrebbe significare overfitting.
- Random Forest → i valori di deviazione standard e varianza sono bassi, ma ancora una volta c'è una netta differenza tra varianza del test error e varianza del train error. Questo potrebbe portare alla presenza anche qui di overfitting.

Una possibile soluzione all'overfitting potrebbe essere quella di effettuare over-sampling. L'over-sampling ha come obiettivo principale quello di bilanciare le classi. Questo perché l'overfitting potrebbe dipendere proprio da uno sbilanciamento iniziale delle classi.

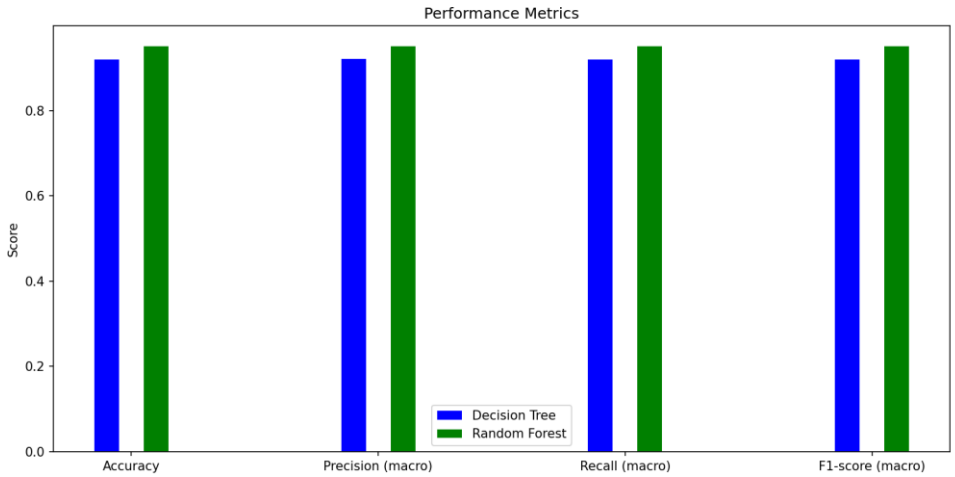
L'over-sampling consiste nel generare nuovi campioni per la classe sottorappresentata in modo che il dataset diventi più bilanciato. Ciò permette al modello di apprendere meglio i pattern delle classi meno rappresentate e ridurre l'overfitting sulla classe maggiormente rappresentata. L'oversampling può essere realizzato duplicando i campioni esistenti o generando nuovi campioni sintetici tramite tecniche come SMOTE (Synthetic Minority Over-sampling Technique).



Iperparametri dopo l’over-sampling:

```
Migliori iperparametri per Decision Tree: {'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}
Migliori iperparametri per Random Forest: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 50}
```

Valori delle metriche per ogni modello dopo l’over-sampling:



```
Decision Tree:
Accuracy: 0.92 (+/- 0.02)
Precision (macro): 0.92 (+/- 0.02)
Recall (macro): 0.92 (+/- 0.02)
F1-score (macro): 0.92 (+/- 0.02)

Random Forest:
Accuracy: 0.95 (+/- 0.03)
Precision (macro): 0.95 (+/- 0.03)
Recall (macro): 0.95 (+/- 0.03)
F1-score (macro): 0.95 (+/- 0.03)
```

Per il Decision Tree:

```
Train Error Std: 0.002678419416875515, Test Error Std: 0.010695837918210702
```

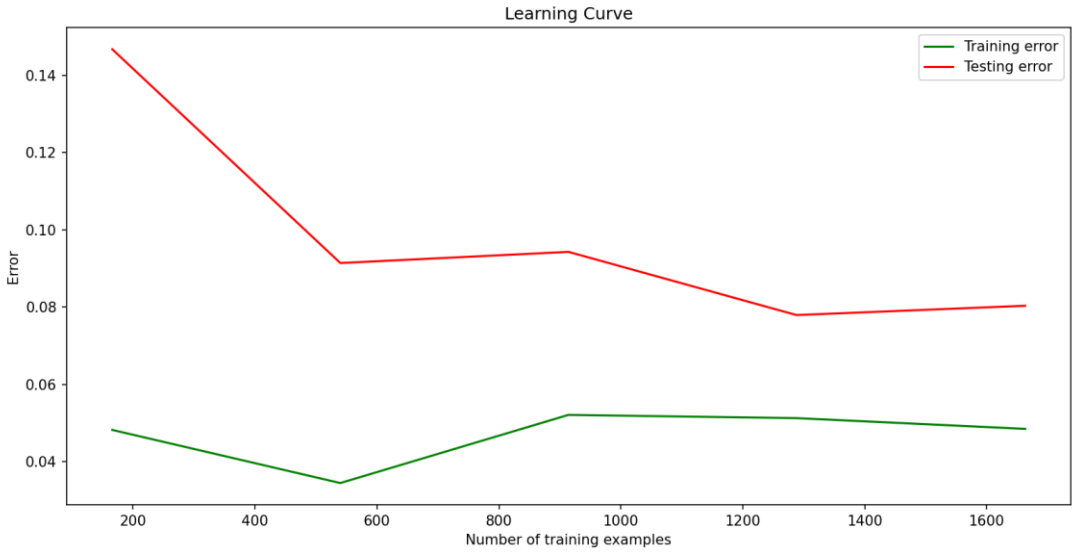
```
Train Error Var: 7.173930572695774e-06, Test Error Var: 0.00011440094877263384
```

Per il Random Forest:

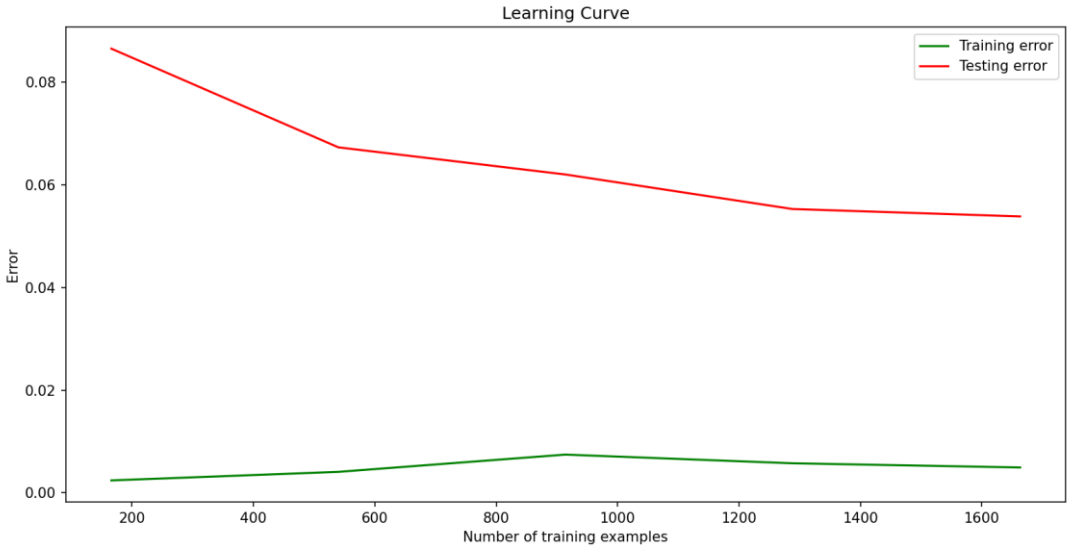
```
Train Error Std: 0.001839694352468835, Test Error Std: 0.013109568237048588
```

```
Train Error Var: 3.3844753105057256e-06, Test Error Var: 0.00017186077936183323
```

Curva di apprendimento per il Decision Tree dopo l'over-sampling:



Curva di apprendimento per il Random Forest dopo l'over-sampling:



Conclusioni

Si può notare che i valori relativi alla deviazione standard e alla varianza siano diminuiti, come lo è adesso la differenza tra varianza del test error e varianza del train error. In generale, si può dire che i valori delle metriche sono rimasti quasi del tutto invariati e restano comunque molto buoni.

Rete bayesiana

In questa fase del progetto, il focus del sistema è quello di applicare il ragionamento probabilistico. A tal proposito, il sistema applica quest'ultimo costruendo una rete bayesiana. Una rete bayesiana è un modello grafico probabilistico che rappresenta una insieme di variabili casuali e le loro dipendenze condizionali tramite un grafo diretto aciclico (DAG). Nelle reti bayesiane ogni nodo rappresenta una dipendenza condizionale. Queste ultime sono utilizzate per modellare sistemi complessi. Le probabilità condizionali associate a ciascun nodo sono calcolate usando il teorema di Bayes.

Per prima cosa, il sistema cerca di individuare quali sono le variabili da inserire nella rete bayesiana su cui poter lavorare. In questo caso, vengono prese in considerazioni le features del dataset che risultano essere:

- battery_power
- clock_speed
- fc
- int_memory
- m_dep
- mobile_wt
- n_cores
- pc

- px_height
- px_width
- ram
- cluster

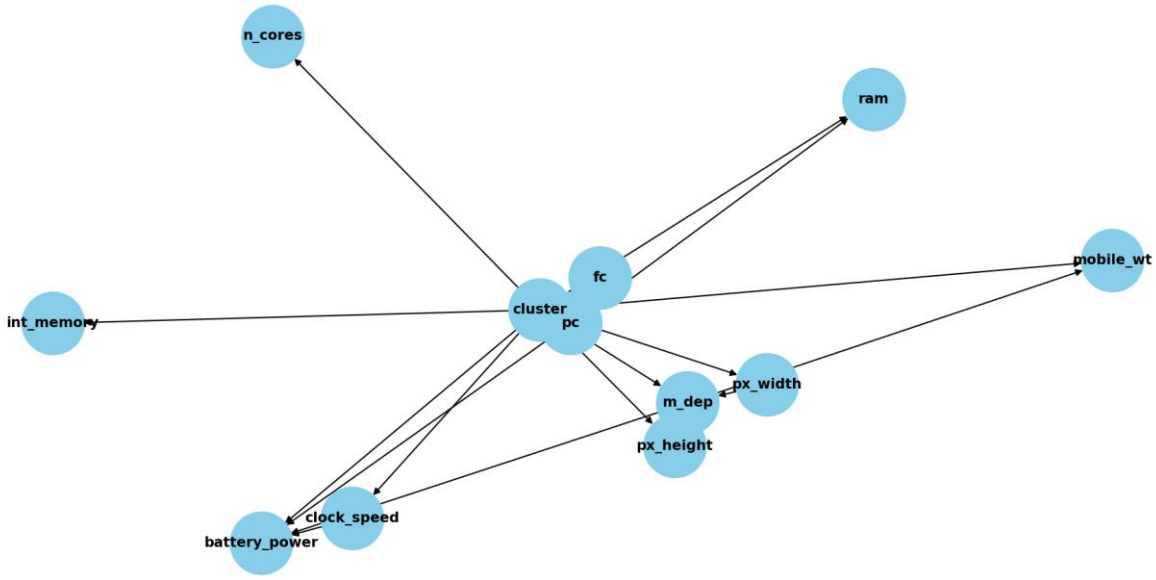
Per formare la struttura della rete bayesiana il sistema divide le variabili in variabili dipendenti e variabili indipendenti. Nelle variabili dipendenti abbiamo le features del dataset tranne la features di output “cluster”. Per individuare le correlazioni tra le variabili della rete bayesiana, il sistema utilizza una matrice di correlazione:

	battery_power	clock_speed	fc	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	cluster
battery_power	1.000000	0.023523	0.044915	0.035429	-0.007142	0.053348	-0.114405	0.051256	0.053917	0.039670	0.052707	-0.039395
clock_speed	0.023523	1.000000	0.000257	0.008567	-0.087436	0.025238	0.010674	-0.040037	-0.009660	0.001374	-0.002509	0.377404
fc	0.044915	0.000257	1.000000	-0.042563	0.040990	-0.037537	-0.055530	0.651747	0.056573	-0.033650	-0.008670	-0.618446
int_memory	0.035429	0.008567	-0.042563	1.000000	0.085309	0.033224	-0.020956	-0.068035	0.042827	0.040387	0.070989	0.137431
m_dep	-0.007142	-0.087436	0.040990	0.085309	1.000000	-0.012711	-0.023814	0.035288	-0.004873	0.044382	-0.024157	-0.106043
mobile_wt	0.053348	0.025238	-0.037537	0.033224	-0.012711	1.000000	0.004504	-0.024470	0.020026	-0.006460	-0.045356	0.037181
n_cores	-0.114405	0.010674	-0.055530	-0.020956	-0.023814	0.004504	1.000000	-0.023064	0.022758	0.003955	0.009478	-0.030110
pc	0.051256	-0.040037	0.651747	-0.068035	0.035288	-0.024470	-0.023064	1.000000	0.046895	-0.052951	-0.017386	-0.683632
px_height	0.053917	-0.009660	0.056573	0.042827	-0.004873	0.020026	0.022758	0.046895	1.000000	0.511683	-0.048275	0.004627
px_width	0.039670	0.001374	-0.033650	0.040387	0.044382	-0.006460	0.003955	-0.052951	0.511683	1.000000	0.013926	0.060932
ram	0.052707	-0.002509	-0.008670	0.070989	-0.024157	-0.045356	0.009478	-0.017386	-0.048275	0.013926	1.000000	0.000682
cluster	-0.039395	0.377404	-0.618446	0.137431	-0.106043	0.037181	-0.030110	-0.683632	0.004627	0.060932	0.000682	1.000000

La matrice di correlazione mostra la correlazione tra tutte le coppie di variabili nel dataset. Valori vicini a 1 indicano una forte correlazione positiva, valori vicini a -1 indicano una forte correlazione negativa, e valori vicini a 0 indicano una debole o nessuna correlazione.

Ho deciso, a questo punto, di utilizzare solamente le correlazioni che hanno valori al di sopra di 2%.

La struttura finale della rete bayesiana che costruisce il sistema è la seguente:



```python
# Crea gli archi della rete bayesiana (correlazioni tra variabili)
for column in dataset:
    if column != 'cluster':
        edges.append(('cluster', column))
edges.append(('clock_speed', 'battery_power'))
edges.append(('fc', 'pc'))
edges.append(('m_dep', 'battery_power'))
edges.append(('m_dep', 'mobile_wt'))
edges.append(('pc', 'ram'))
edges.append(('px_width', 'm_dep'))
edges.append(('px_height', 'm_dep'))
edges.append(('pc', 'battery_power'))
```

Una volta stabiliti gli archi all'interno della rete bayesiana, il sistema procede con il suo addestramento, osservando i valori presenti nel dataset. Per ridurre i tempi di esecuzione del programma in sede di addestramento della rete e calcolo delle CPD, il sistema prende in considerazione le prime 500 righe del dataset.

Per quanto riguarda l'addestramento della rete bayesiana è stato passato come parametro del metodo fit l'estimatore "BayesianEstimator".

Le probabilità condizionali per ogni variabile apprese dalla rete sono le seguenti:

```
CPD of cluster:
+------------+----------+
| cluster(0) | 0.359736 |
+------------+----------+
| cluster(1) | 0.306271 |
+------------+----------+
| cluster(2) | 0.333993 |
+------------+----------+
```

```
CPD of battery_power:
+-----------------------------------+-----+-----------------------------------+
| clock_speed                       | ... | clock_speed(1.0000000000000002)   |
+-----------------------------------+-----+-----------------------------------+
| cluster                           | ... | cluster(2)                        |
+-----------------------------------+-----+-----------------------------------+
| m_dep                             | ... | m_dep(1.0)                        |
+-----------------------------------+-----+-----------------------------------+
| pc                                | ... | pc(1.0)                           |
+-----------------------------------+-----+-----------------------------------+
| battery_power(0.0013360053440213) | ... | 0.002336448598130837              |
+-----------------------------------+-----+-----------------------------------+
| battery_power(0.0033400133600534) | ... | 0.002336448598130837              |
+-----------------------------------+-----+-----------------------------------+
```

```
CPD of clock_speed:
+-----------------------------------+-----+-----------------------+
| cluster                           | ... | cluster(2)            |
+-----------------------------------+-----+-----------------------+
| clock_speed(0.0)                  | ... | 0.0003800547278808147 |
+-----------------------------------+-----+-----------------------+
| clock_speed(0.0399999999999999)   | ... | 0.0003800547278808147 |
+-----------------------------------+-----+-----------------------+
| clock_speed(0.0799999999999999)   | ... | 0.0003800547278808147 |
+-----------------------------------+-----+-----------------------+
| clock_speed(0.12)                 | ... | 0.0003800547278808147 |
+-----------------------------------+-----+-----------------------+
| clock_speed(0.16)                 | ... | 0.0003800547278808147 |
+-----------------------------------+-----+-----------------------+
| clock_speed(0.2)                  | ... | 0.0003800547278808147 |
+-----------------------------------+-----+-----------------------+
| clock_speed(0.24)                 | ... | 0.0003800547278808147 |
+-----------------------------------+-----+-----------------------+
```

```
CPD of fc:
+-----------------------+-----+----------------------+
| cluster               | ... | cluster(2)           |
+-----------------------+-----+----------------------+
| fc(0.0)               | ... | 0.3681090076971083   |
+-----------------------+-----+----------------------+
| fc(0.0526315789473684) | ... | 0.16059912627418343  |
+-----------------------+-----+----------------------+
| fc(0.1052631578947368) | ... | 0.10131058872477633  |
+-----------------------+-----+----------------------+
| fc(0.1578947368421052) | ... | 0.09538173496983562  |
+-----------------------+-----+----------------------+
| fc(0.2105263157894736) | ... | 0.09538173496983562  |
+-----------------------+-----+----------------------+
| fc(0.2631578947368421) | ... | 0.06573746619513209  |
+-----------------------+-----+----------------------+
| fc(0.3157894736842105) | ... | 0.059808612440191374 |
+-----------------------+-----+----------------------+
| fc(0.3684210526315789) | ... | 0.0183066361556064   |
+-----------------------+-----+----------------------+
```

```
CPD of int_memory:
+---------------------------------+-----+-----------------------+
| cluster                         | ... | cluster(2)            |
+---------------------------------+-----+-----------------------+
| int_memory(0.0)                 | ... | 0.006085701737875648  |
+---------------------------------+-----+-----------------------+
| int_memory(0.0161290322580645)  | ... | 0.0001568479829349393 |
+---------------------------------+-----+-----------------------+
| int_memory(0.032258064516129)   | ... | 0.012014555492816357  |
+---------------------------------+-----+-----------------------+
| int_memory(0.0483870967741935)  | ... | 0.006085701737875648  |
+---------------------------------+-----+-----------------------+
| int_memory(0.064516129032258)   | ... | 0.012014555492816357  |
+---------------------------------+-----+-----------------------+
| int_memory(0.0806451612903225)  | ... | 0.017943409247757065  |
+---------------------------------+-----+-----------------------+
| int_memory(0.0967741935483871)  | ... | 0.017943409247757065  |
+---------------------------------+-----+-----------------------+
```

```
CPD of m_dep:
+---------------------------+-----+-----------------------------+
| cluster                   | ... | cluster(2)                  |
+---------------------------+-----+-----------------------------+
| px_height                 | ... | px_height(0.976530612244898) |
+---------------------------+-----+-----------------------------+
| px_width                  | ... | px_width(0.9986648865153538) |
+---------------------------+-----+-----------------------------+
| m_dep(0.0)                | ... | 0.1                         |
+---------------------------+-----+-----------------------------+
| m_dep(0.1111111111111111) | ... | 0.1                         |
+---------------------------+-----+-----------------------------+
| m_dep(0.2222222222222222) | ... | 0.1                         |
+---------------------------+-----+-----------------------------+
| m_dep(0.3333333333333333) | ... | 0.1                         |
+---------------------------+-----+-----------------------------+
```

```
CPD of mobile_wt:
+-------------------------------+-----+-------------------------+
| cluster                       | ... | cluster(2)              |
+-------------------------------+-----+-------------------------+
| m_dep                         | ... | m_dep(1.0)              |
+-------------------------------+-----+-------------------------+
| mobile_wt(0.0)                | ... | 0.00022522522522522471 |
+-------------------------------+-----+-------------------------+
| mobile_wt(0.0083333333333334) | ... | 0.00022522522522522471 |
+-------------------------------+-----+-------------------------+
| mobile_wt(0.0166666666666667) | ... | 0.00022522522522522471 |
+-------------------------------+-----+-------------------------+
| mobile_wt(0.025)              | ... | 0.00022522522522522471 |
+-------------------------------+-----+-------------------------+
| mobile_wt(0.0333333333333333) | ... | 0.00022522522522522471 |
+-------------------------------+-----+-------------------------+
| mobile_wt(0.0416666666666667) | ... | 0.00022522522522522471 |
+-------------------------------+-----+-------------------------+
```

```
CPD of n_cores:
+-----------------------------+-----+---------------------+
| cluster                     | ... | cluster(2)          |
+-----------------------------+-----+---------------------+
| n_cores(0.0)                | ... | 0.12574110671936758 |
+-----------------------------+-----+---------------------+
| n_cores(0.1428571428571428) | ... | 0.11388339920948616 |
+-----------------------------+-----+---------------------+
| n_cores(0.2857142857142857) | ... | 0.17317193675889328 |
+-----------------------------+-----+---------------------+
| n_cores(0.4285714285714285) | ... | 0.14352766798418973 |
+-----------------------------+-----+---------------------+
| n_cores(0.5714285714285714) | ... | 0.09609683794466403 |
+-----------------------------+-----+---------------------+
| n_cores(0.7142857142857142) | ... | 0.0723814229249012  |
+-----------------------------+-----+---------------------+
```

```
CPD of pc:
+-------------------------+-----+-------------------------+
| cluster                 | ... | cluster(2)              |
+-------------------------+-----+-------------------------+
| fc                      | ... | fc(0.9473684210526316) |
+-------------------------+-----+-------------------------+
| pc(0.0)                 | ... | 0.047619047619047616    |
+-------------------------+-----+-------------------------+
| pc(0.05)                | ... | 0.047619047619047616    |
+-------------------------+-----+-------------------------+
| pc(0.1)                 | ... | 0.047619047619047616    |
+-------------------------+-----+-------------------------+
| pc(0.15)                | ... | 0.047619047619047616    |
+-------------------------+-----+-------------------------+
| pc(0.2)                 | ... | 0.047619047619047616    |
+-------------------------+-----+-------------------------+
| pc(0.25)                | ... | 0.047619047619047616    |
+-------------------------+-----+-------------------------+
| pc(0.3)                 | ... | 0.047619047619047616    |
+-------------------------+-----+-------------------------+
```

```
CPD of px_height:
+--------------------------------+-----+-------------------------+
| cluster                        | ... | cluster(2)              |
+--------------------------------+-----+-------------------------+
| px_height(0.0020408163265306) | ... | 2.3195828462209595e-05 |
+--------------------------------+-----+-------------------------+
| px_height(0.0030612244897959) | ... | 0.005952049583402982   |
+--------------------------------+-----+-------------------------+
| px_height(0.0066326530612244) | ... | 2.3195828462209595e-05 |
+--------------------------------+-----+-------------------------+
| px_height(0.0071428571428571) | ... | 2.3195828462209595e-05 |
+--------------------------------+-----+-------------------------+
| px_height(0.0096938775510204) | ... | 0.005952049583402982   |
+--------------------------------+-----+-------------------------+
| px_height(0.010204081632653)  | ... | 0.005952049583402982   |
+--------------------------------+-----+-------------------------+
```

```
CPD of px_width:
+--------------------------------+-----+-------------------------+
| cluster                        | ... | cluster(2)              |
+--------------------------------+-----+-------------------------+
| px_width(0.0)                  | ... | 0.005951674593104457   |
+--------------------------------+-----+-------------------------+
| px_width(0.0086782376502002) | ... | 2.2820838163744083e-05 |
+--------------------------------+-----+-------------------------+
| px_width(0.0100133511348464) | ... | 2.2820838163744083e-05 |
+--------------------------------+-----+-------------------------+
| px_width(0.0113484646194926) | ... | 2.2820838163744083e-05 |
+--------------------------------+-----+-------------------------+
| px_width(0.0126835781041388) | ... | 2.2820838163744083e-05 |
+--------------------------------+-----+-------------------------+
| px_width(0.0133511348464619) | ... | 0.005951674593104457   |
+--------------------------------+-----+-------------------------+
```

```
CPD of ram:
+-------------------------+-----+----------------------+
| cluster                 | ... | cluster(2)           |
+-------------------------+-----+----------------------+
| pc                      | ... | pc(1.0)              |
+-------------------------+-----+----------------------+
| ram(0.0005344735435595) | ... | 0.002118644067796601 |
+-------------------------+-----+----------------------+
| ram(0.0058792089791555) | ... | 0.002118644067796601 |
+-------------------------+-----+----------------------+
| ram(0.0074826296098343) | ... | 0.002118644067796601 |
+-------------------------+-----+----------------------+
| ram(0.0101549973276322) | ... | 0.002118644067796601 |
+-------------------------+-----+----------------------+
| ram(0.0120256547300908) | ... | 0.002118644067796601 |
+-------------------------+-----+----------------------+
| ram(0.0146980224478888) | ... | 0.002118644067796601 |
+-------------------------+-----+----------------------+
| ram(0.0213789417423837) | ... | 0.002118644067796601 |
+-------------------------+-----+----------------------+
```

Dopo aver appreso le CPD per ogni variabile, il sistema ha come obiettivo quello di predire il cluster di appartenenza di un campione estratto dal dataset. Infatti, di seguito viene riportato un esempio di campione estratto dal dataset con il relativo cluster di appartenenza predetto:

```
     battery_power  clock_speed        fc  int_memory     m_dep  mobile_wt   n_cores   pc  px_height  px_width       ram
145       0.877088         0.36  0.052632    0.112903  0.333333   0.508333  0.142857  0.1   0.385714  0.190921  0.881881
100%|██████████| 1/1 [00:00<00:00, 22.87it/s]
WARNING:pgmpy:BayesianModel has been renamed to BayesianNetwork. Please use BayesianNetwork class, BayesianModel will be removed in future.
WARNING:pgmpy:BayesianModel has been renamed to BayesianNetwork. Please use BayesianNetwork class, BayesianModel will be removed in future.
   cluster
0        1
```

Il sistema, fatto ciò, vuole predire il cluster di appartenenza di un campione che non è presente nel dataset. A tal proposito, se un esempio non appartiene al dataset originale, significa che presenta una combinazione di valori delle variabili che il modello non ha mai visto prima. In questo caso, la rete bayesiana potrebbe non essere in grado di fare una previsione accurata per una determinata caratteristica di quel campione perché non ha informazioni sulla distribuzione di probabilità condizionata di quella caratteristica, data la combinazione specifica di valori delle altre variabili.

Per ovviare a questa limitazione, il sistema non fa altro che discretizzare i valori all'interno del dataset, così da facilitare l'addestramento della rete bayesiana, l'apprendimento di quelle che saranno le nuove CPD e la predizione del cluster di appartenenza di un campione random.

Alcuni esempi di CPD apprese dalla rete bayesiana a seguito della discretizzazione dei valori nel dataset:

CPD of battery_power:

clock_speed	...	clock_speed(9)
cluster	...	cluster(2)
m_dep	...	m_dep(9)
pc	...	pc(9)
battery_power(0)	...	0.09999999999999998
battery_power(1)	...	0.09999999999999998
battery_power(2)	...	0.09999999999999998
battery_power(3)	...	0.09999999999999998

CPD of clock_speed:

cluster	...	cluster(2)
clock_speed(0)	...	0.0009881422924901185
clock_speed(1)	...	0.0009881422924901185
clock_speed(2)	...	0.0009881422924901185
clock_speed(3)	...	0.03063241106719368
clock_speed(4)	...	0.07213438735177866
clock_speed(5)	...	0.1907114624505929

CPD of fc:

cluster	cluster(0)	...	cluster(2)
fc(0)	0.044954128440366975	...	0.5286561264822137
fc(1)	0.07247706422018349	...	0.19664031620553366
fc(2)	0.13853211009174313	...	0.1610671936758894
fc(3)	0.14954128440366973	...	0.07806324110671939
fc(4)	0.14403669724770643	...	0.024703557312252974
fc(5)	0.0889908256880734	...	0.0069169960474308335

CPD of int_memory:

cluster	...	cluster(2)
int_memory(0)	...	0.07213438735177866
int_memory(1)	...	0.10770750988142294
int_memory(2)	...	0.08399209486166008
int_memory(3)	...	0.07213438735177866
int_memory(4)	...	0.09584980237154152
int_memory(5)	...	0.05434782608695652

CPD of m_dep:

cluster	cluster(0)	...	cluster(2)
px_height	px_height(0)	...	px_height(9)
px_width	px_width(0)	...	px_width(9)
m_dep(0)	0.0003322259136212624	...	0.000826446280991736
m_dep(1)	0.1996677740863787	...	0.4966942148760333
m_dep(2)	0.0003322259136212624	...	0.000826446280991736
m_dep(3)	0.0003322259136212624	...	0.4966942148760333

Ecco un esempio di elemento che non è presente già nel dataset:

```
   battery_power  clock_speed  fc  int_memory  m_dep  mobile_wt  n_cores  pc  px_height  px_width  ram
0            0            2   1           4      7          0        3   3          9         6    2
100%|██████████| 1/1 [00:00<00:00, 86.76it/s]
WARNING:pgmpy:BayesianModel has been renamed to BayesianNetwork. Please use BayesianNetwork class, BayesianModel will be removed in future.
WARNING:pgmpy:BayesianModel has been renamed to BayesianNetwork. Please use BayesianNetwork class, BayesianModel will be removed in future.
WARNING:pgmpy:Found unknown state name. Trying to switch to using all state names as state numbers
   cluster
0        1
```

In questa fase di progetto le librerie utilizzate sono:

- pgmpy: gestisce la rete bayesiana
- networkx: permette di visualizzare grafici
- matplotlib: permette di visualizzare grafici

Prolog e Basi di Conoscenza

In questa ultima fase del progetto il sistema applica il ragionamento logico. In particolare, viene costruita una base di conoscenza (Knowledge Base) basata sul dataset contente le features dei telefoni cellulari. Una base di conoscenza è un insieme organizzato di informazioni e dati relativi a un particolare dominio di conoscenza. Essa può includere fatti, regole, relazioni, e altre forme di informazioni strutturate e non strutturate.

La libreria utilizzata per scrivere i fatti e le regole in Prolog è “pyswip”.

In questo caso, la KB contiene dei fatti e delle regole:

- Fatto “telefono” il quale è un’affermazione sempre vera in quanto contiene le informazioni relative ad un singolo telefono, compreso anche il cluster di appartenenza.
- Regola “telefono_cluster(Id, Cluster) :- telefono(Id, _, _, _, _, _, _, _, _, _, _, _, Cluster)”
- Regola “telefono_con_ram(Id, Ram) :- telefono(Id, _, _, _, _, _, _, _, _, _, Ram, _)”
- Regola “telefono_con_battery_power(Id, BatteryPower) :- telefono(Id, BatteryPower, _, _, _, _, _, _, _, _, _, _)”
- Regola “telefono_ad_alte_prestazioni(Id) :- telefono(Id, BatteryPower, _, _, _, _, _, _, _, _, Ram, _), Ram >= 0.8, BatteryPower >= 0.7.”

Con l’utilizzo di queste regole e della libreria “pyswip”, il sistema offre la possibilità di interrogare attraverso delle query la base di conoscenza. Infatti, ecco alcune query che implementa:

```
I telefoni che appartengono al cluster: 2.0
{'Id': 0}
{'Id': 3}
{'Id': 10}
{'Id': 12}
{'Id': 21}
{'Id': 22}
{'Id': 33}
{'Id': 34}
{'Id': 36}
```

```
I telefoni che hanno come ram il valore: 0.6127739176910743
{'Id': 0}
{'Id': 773}
```

```
I telefoni che hanno come battery_power il valore: 0.6305945223780896
{'Id': 8}
{'Id': 201}
{'Id': 1803}
```

```
I telefoni ad alte prestazioni:
{'Id': 21}
{'Id': 30}
{'Id': 37}
{'Id': 41}
{'Id': 46}
{'Id': 67}
```