Laurea Triennale in informatica-Università di Salerno
Corso di *Intelligenza artificiale* - Prof. F.Palomba

Documentazione
UniNotes

Team members

Nome	Ruolo nel progetto	Acronimo	Informazioni di contatto
Damiana Buono	Team Member	DB	d.buono10@studenti.unisa.it
Alessandra Parziale	Team Member	AP	a.parziale8@studenti.unisa.it
Otino Pio Santosuosso	Team Member	OS	o.santosusso@studenti.unisa.it

Link al repository GitHub : https://github.com/AlessandraParziale/UniNotesFIA.git

Sommario

1. Introduzione: sistema proposto

Molte volte nella scelta della laurea magistrale si prediligono aspetti relativi alla carriera, mettendo in secondo piano quella che dovrebbe essere il più importante, ovvero le nostre predilezioni e i nostri interessi ed in questo ci viene in aiuto Dona.

Si è pensato di andare a realizzare il sistema UniNotes, il quale oltre a formare quella che è una vera e proprio community di studenti, darà ad ogni studente la possibiltà di interagire con un agente intelligente il quale riuscirà a consigliare, attraverso un questionario proposto agli studenti iscritti al sistema, una futura magistrale.

2. Descrizione dell'agente

2.1 Obiettivi

Lo scopo finale del progetto è quello di realizzare un agente intelligente che sia in grado di :

- Consigliare una scelta magistrale, sulla base di informazioni raccolte da studenti magistrali e preferenze espresse dal singolo studente triennale.

2.2 Specifica PEAS

La specifica PEAS dell'agente è :

PEAS	
Performance	La misura di performance del modulo si basa sull'accuratezza di associazione tra le risposte date dallo studente e la futura magistrale.
Environment	Dato che il modulo fa parte del sistema UniNotes, il suo ambiente è formato da tutte le funzionalità offerte dalla piattaforma. L'ambiente è : - ***Completamente osservabile***, i sensori dell'agente gli permettono di ricevere sempre lo stato completo dell'ambiente, in quanto verrà attivato solo dopo aver risposto a tutte le domande del questionario. - ***Stocastico***, dato che la risposta dell'agente è un semplice consiglio lo stato dell'ambiente, ovvero come l'utente decide di interagire con UniNotes dipende dalle sue decisioni. Per tale motivazioni, lo stato dell'ambiente non è completamente determinato dalle azioni dell'agente. - ***Episodico,*** l'agente delibera il risultato solo quando viene richiesto dell'utente in maniera esplicita. - **Statico,** durante il calcolo della risposta il controller della web application mette l'utente in attesa del risultato. - **Discreto,** l'insieme delle possibili percezioni sono le domande e le risposte stabilite in maniera preliminare, quindi un numero limitato.

	- **Agente Singolo,** nell'ambiente è presente un singolo agente.*
Actuators	Gli attuatori consistono in tutto ciò che fa parte della logica di business della web application, ossia le funzioni che permettono il calcolo del risultato.
Sensor	I sensori sono rappresentati dai bottoni per richiedere la compilazione del questionario e dagli elementi visuali utilizzati per rispondere alla domanda.

*Volevamo sottolineare che nel punto 7 della documentazione abbiamo utilizzato due agenti singolarmente addestrati per testare entrambi gli algoritmi proposti.

2.3 Analisi del problema

E' stato deciso di affrontare il problema mediante tecniche di machine learning, in particolare con tecniche di clustering e classificazione, andando con spirito critico a fare delle conclusioni finali sulla base dei risultati riscontrati dalle due tecniche utilizzate.

In particolare con il clustering abbiamo previsto di utilizzare KMeans che ci permetterà di fare una suddivisione in gruppi, dove ogni gruppo ha un certo grado di omogeneità.
Abbiamo deciso di utilizzarlo in quanto eravamo già a conoscenza del numero di cluster che volevamo in output.

Invece per la classificazione abbiamo previsto di utilizzare il Decision Tree il quale va a lavorare con quello che è il concetto di Information Gain (ovvero il grado di purezza di un attributo).
Abbiamo deciso di utilizzarlo in quanto volevamo utilizzare un algoritmo che si basava sulle caratteristiche e quindi sul concetto di entropia.

3. Raccolta e analisi dei dati

3.1 Scelta del dataset

Data la natura del problema per la raccolta dei dati si è scelto di collezionare le risposte alle domande di un questionario sottoposto a studenti che hanno già preso questa decisione.

Il questionario contiene 22 domande a risposta multipla. Le domande sono state strutturate in modo da coprire tutte le magistrali di informatica offerte dall'università degli studi di Salerno.

In questo problema per la raccolta dei dati ci sono 2 possibili opzioni:

1. Somministrazione di un questionario agli studenti.

   Dopo la somministrazione del questionario però ci siamo resi conto che il dataset aveva un numero troppo piccolo di informazioni (30), quindi era difficile andare a realizzare un modello utilizzando questo dataset per l'allenamento.

anni	magistrale	eim	ciim	li	liim	llu	ftu	dim	aor	Paro	cs	ds	rsm	ocm	occm	clt	eva	pim	ap	at	ars
0	2	0	0	1	2	2	1	0	1	0	0	3	0	0	5	4	0	2	2	2	3
0	2	0	0	2	2	1	0	0	1	3	1	3	0	0	5	4	1	2	2	4	1
0	4	3	2	1	3	1	1	3	1	0	3	4	1	0	5	8	1	2	2	2	1
0	2	0	2	1	2	1	0	0	0	1	1	4	0	0	2	5	1	2	2	2	2
0	4	1	0	1	2	1	0	0	1	4	0	1	2	1	5	2	1	1	5	5	2
0	2	0	0	1	3	1	1	0	1	0	0	3	0	0	5	8	0	2	5	3	0
0	2	1	0	1	3	0	0	0	1	2	1	1	0	0	5	5	1	2	2	6	0
0	1	0	0	2	0	0	1	0	0	1	0	1	0	0	5	8	1	0	5	7	2
0	2	0	0	1	3	1	0	2	0	1	0	3	0	0	5	2	0	2	2	4	2
0	0	2	0	1	3	2	0	0	1	5	0	7	0	0	5	5	0	0	5	0	2
0	2	0	0	1	1	1	0	0	0	1	0	3	0	0	2	6	1	2	2	4	3
0	4	0	0	1	3	2	0	2	1	0	0	1	1	0	5	2	0	0	5	4	1
0	3	0	3	2	2	2	0	1	1	2	0	3	0	0	5	2	1	0	5	3	2
0	1	0	0	1	0	2	0	0	0	6	2	3	1	0	5	7	1	0	5	1	0
0	2	0	0	2	2	1	0	0	1	7	1	3	0	1	5	8	0	2	2	4	1
0	3	0	2	1	2	1	0	0	1	0	2	1	2	3	2	0	1	2	4	4	3
0	1	1	1	1	1	1	0	0	1	0	0	3	0	0	5	7	1	0	5	1	2
0	1	0	1	1	2	1	0	0	1	0	0	3	0	0	5	7	0	1	5	1	2
0	0	1	1	1	2	1	0	0	1	2	1	1	1	1	5	6	0	0	5	2	0
0	1	0	0	1	0	1	0	0	1	8	0	1	0	0	5	0	0	0	5	8	0
0	0	0	0	1	2	2	0	0	1	0	0	3	1	1	1	3	1	1	5	4	0
0	2	1	1	1	3	0	0	1	0	1	0	8	1	1	5	3	1	2	2	0	2
0	1	0	0	1	3	2	0	0	1	0	3	3	0	0	5	2	0	1	1	9	3
0	1	0	0	1	1	1	0	0	1	0	0	3	0	0	5	7	1	0	1	1	2
1	0	0	0	2	2	2	1	0	1	9	0	3	0	0	5	6	1	2	0	0	2
0	3	0	3	2	2	1	0	0	1	0	1	1	1	1	2	5	0	0	5	4	1
1	1	0	0	1	1	1	0	0	0	0	0	3	0	0	5	7	0	0	5	10	0
0	1	0	1	1	2	1	0	0	1	0	0	0	1	0	5	7	0	0	5	11	3
0	1	0	2	1	0	0	0	0	0	1	2	0	2	2	0	7	0	0	5	5	0
1	1	1	2	1	2	2	0	2	1	0	2	1	2	1	4	7	1	0	5	13	0

Questa tabella è stata realizzata per la comprensione degli acronimi usati per le colonne del dataset.

Acronimo	Domanda
anni	Quanti anni hai ?
magistrale	Quale corso magistrale hai scelto ?
eim	Quanto la tua età ha influenzato la scelta del corso magistrale ?
ciim	Quanto la tua conoscenza della lingua inglese ha influenzato la scelta del corso magistrale ?
li	Qual è il tuo livello della lingua inglese ?
lim	Quanto lo sbocco lavorativo e la relativa retribuzione futura ha influenzato la scelta del corso magistrale ?
llu	Il luogo in cui vivi è lontano dall'università ?
ftu	Per frequentare il corso magistrale da te scelto ti sei trasferito/a nei pressi dell'università ?
dim	Quanto la distanza dall'università ha influito sulla scelta del corso magistrale ?
aor	Desideravi allontanarti o restare nei pressi dell'università?
Paro	Perché ?
cs	Se non dovessi considerare più questi fattori cambieresti la tua scelta ?
ds	C'è qualcosa che ti ha deluso della tua scelta ?
rsm	Ad oggi hai un ripensamento sulla tua scelta magistrale ?

ocm	Ad oggi quanto cambieresti la scelta della magistrale ?
occm	Ad oggi con quale corso magistrale cambieresti la tua scelta ?
clt	Quale corso tra quelli scelti durante la laurea triennale ti ha condotto alla scelta della magistrale ?
eva	È stato l'esame dove hai conseguito uno dei voti più alti ?
pim	Quanto i progetti organizzati dall'università di Salerno hanno influenzato la scelta della magistrale ?
ap	Di che argomento trattava il progetto organizzato dall'università di Salerno che ha influenzato la scelta della magistrale ?
at	Su quale argomento si è incentrata la tua tesi triennale ?
ars	Quanto questo argomento è stato rilevante per la tua scelta ?

2. Aumentare il dataset ottenuto utilizzando un tool online per la generazione di nuovi dati, basandosi sulle domande risposte.
   Bisogna tener presente che si tratta di dati non del tutto veritieri.

   Questa decisione è stata presa semplicemente perchè si voleva andare ad applicare le tecniche di Data Preparation, per andare a pulire i dati e bilanciare il dataset.

come output abbiamo ricavato circa 641 righe.

magistrale	anni	ciim	li	liim	clt	eva	ap	at
2	0	0	1	2	4	0	2	2
2	0	0	2	2	4	1	2	4
4	0	2	1	3	8	1	2	2
2	0	2	1	2	5	1	2	2
4	0	0	1	2	2	1	5	5
...	...	...	..	...	...	...	..	..
2	0	0	1	2	4	0	5	2
2	0	0	1	3	4	0	2	4
1	0	0	1	2	7	0	5	2
0	0	0	1	3	2	0	2	4
2	0	0	1	2	2	0	2	1

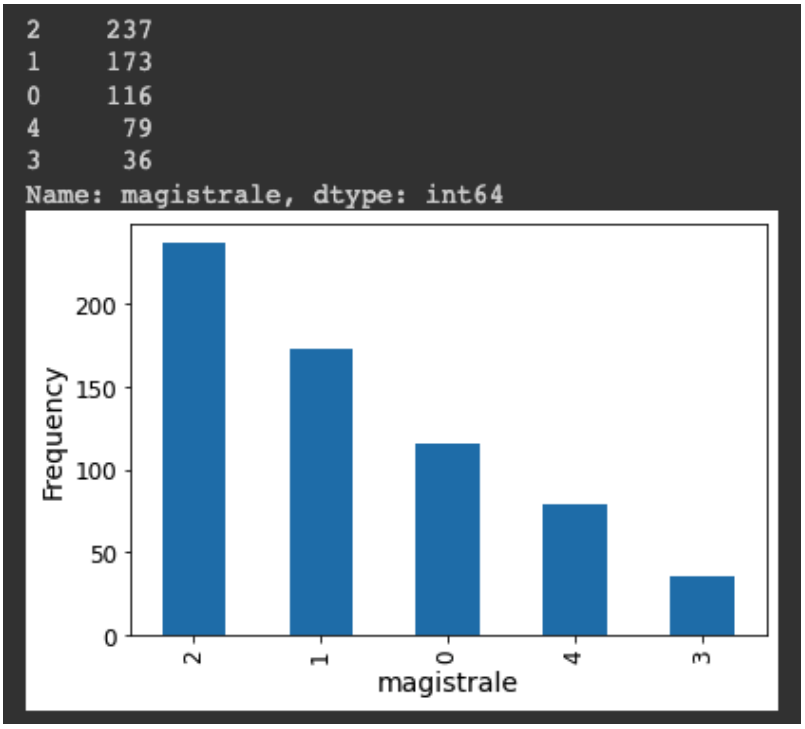
3.2 Operazione sui dati

Abbiamo utilizzato colab per andare a realizzare tutte le fasi di data preparation, andando a sfruttare la potenza dei server di google.
Le seguenti operazioni sono state svolte su entrambi i dataset.

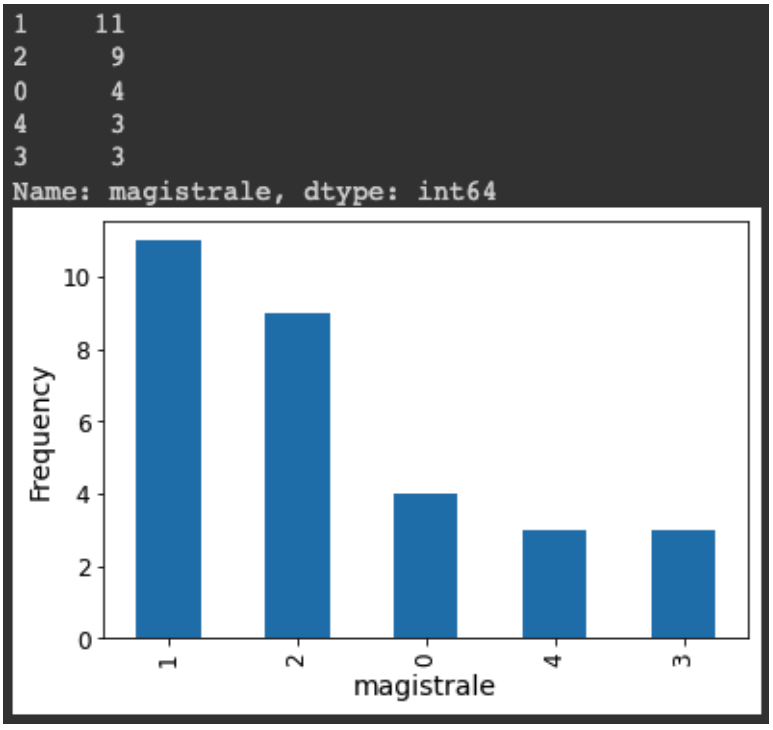
Prima di tutto abbiamo capito come è bilanciato il dataset utilizzando questo script in python

```
pd.value_counts(dataset['magistrale']).plot.bar()
plt.xlabel('magistrale')
plt.ylabel('Frequency')
dataset['magistrale'].value_counts()
```

Dataset Artificiale :



Dataset Reale :



Divisione del Dataset

Sappiamo bene che per andare ad allenare un modello dobbiamo avere anche un dataset di test, per ottenerlo abbiamo deciso di dividere il dataset andando ad utilizzare l'80% per il training e il restante 20% per il test.

Qui sotto è riportato lo script in python per fare ciò.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)
```

Ricordiamo che è stato applicato sempre ad entrambi i dataset.

Bilanciamento del dataset

Per il bilanciamento del dataset andiamo ad utilizzare SMOTE, ovvero una tecnica di oversampling, la quale ci permette di portare il numero delle istanze di minoranza al numero delle istanze di maggioranza.
Bisogna ricordare che il bilanciamento del dataset va fatto **solo** sul dataset di training e **non** su quello di test.

```
from imblearn.over_sampling import SMOTE

print("Prima dell'OverSampling, il numero delle label '0': {}".format(sum(y_train == 0)))
print("Prima dell'OverSampling, il numero delle label '1': {}".format(sum(y_train == 1)))
print("Prima dell'OverSampling, il numero delle label '2': {}".format(sum(y_train == 2)))
print("Prima dell'OverSampling, il numero delle label '3': {}".format(sum(y_train == 3)))
print("Prima dell'OverSampling, il numero delle label '4': {}".format(sum(y_train == 4)))

sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
print("\n")

print("Dopo l'OverSampling, il numero di label '0': {}".format(sum(y_train_res == 0)))
print("Dopo l'OverSampling, il numero di label '1': {}".format(sum(y_train_res == 1)))
print("Dopo l'OverSampling, il numero di label '2': {}".format(sum(y_train_res == 2)))
print("Dopo l'OverSampling, il numero di label '3': {}".format(sum(y_train_res == 3)))
print("Dopo l'OverSampling, il numero di label '4': {}".format(sum(y_train_res == 4)))
```

Output DataSet Artificiale:

```
Prima dell'OverSampling, il numero delle label '0': 97
Prima dell'OverSampling, il numero delle label '1': 140
Prima dell'OverSampling, il numero delle label '2': 185
Prima dell'OverSampling, il numero delle label '3': 30
Prima dell'OverSampling, il numero delle label '4': 60


Dopo l'OverSampling, il numero di label '0': 185
Dopo l'OverSampling, il numero di label '1': 185
Dopo l'OverSampling, il numero di label '2': 185
Dopo l'OverSampling, il numero di label '3': 185
Dopo l'OverSampling, il numero di label '4': 185
```

Output DataSet Reale:

```
Prima dell'OverSampling, il numero delle label '0': 3
Prima dell'OverSampling, il numero delle label '1': 8
Prima dell'OverSampling, il numero delle label '2': 8
Prima dell'OverSampling, il numero delle label '3': 2
Prima dell'OverSampling, il numero delle label '4': 3


Dopo l'OverSampling, il numero di label '0': 8
Dopo l'OverSampling, il numero di label '1': 8
Dopo l'OverSampling, il numero di label '2': 8
Dopo l'OverSampling, il numero di label '3': 8
Dopo l'OverSampling, il numero di label '4': 8
```

Normalizzazione del dataset

Per evitare che l'algoritmo possa sottostimare o sovrastimare l'importanza di una caratteristica abbiamo deciso di normalizzare i dati del dataset.
Per far ciò abbiamo utilizzato la normalizzazione che appunto ci permette di normalizzare i dati specificando un range.

```
from sklearn.preprocessing import MinMaxScaler

scl = MinMaxScaler(feature_range=(0, 1))
X_train_res = scl.fit_transform(X_train_res)
x_test = scl.fit_transform(X_test)
```

In questo caso possiamo vedere che abbiamo inserito come range (0,1) e abbiamo applicato la trasformazione sia sul dataset di training che sul dataset di test.

Feature engineering

In questa fase andiamo a definire quelle che sono le caratteristiche principali.
Abbiamo usato il feature selection, tramite il quale abbiamo selezionato le caratteristiche più correlate al problema in esame.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_regression
from numpy import set_printoptions

fs = SelectKBest(score_func=chi2,k=9)
fs.fit_transform(X_train_res, y_train_res)

X_new_train_res = fs.transform(X_train_res)
X_new_test = fs.transform(X_test)


X.columns[fs.get_support(indices=True)]

X.columns[fs.get_support(indices=True)].tolist()
```

Output DataSet Artificiale:

```
['anni', 'lim', 'ftu', 'dim', 'Paro', 'cs', 'ds', 'ocm', 'ap']
```

Output DataSet Reale:

```
['eim ', 'liim', 'llu', 'ftu', 'dim', 'aor', 'ds', 'occm', 'at']
```

In questa fase abbiamo però riscontrato che le tecniche di feature engeneering ci portavano alla scelta di feature che non erano molto consone al nostro problema(dovuto giustamente alla casualità dei dati presente nel dataset artificiale).

Per ovviare a questo problema abbiamo scelto solo 9 domande più importanti e sensate e abbiamo costruito il nostro dataset su quelle.

Le 9 domande scelte sono:

Domanda	Risposta
Quale corso magistrale hai scelto ?	(0)Cloud Computing (1)Sicurezza Informatica (2)Software Engineering and IT Management (3)IoT (4)Data Science & Machine Learning
Quanti anni hai ?	(0)22 - 26 (1)27 - 31 (2)32 - 36 (3)36 - 40 (4)>40
Quanto la tua conoscenza della lingua inglese ha influenzato la scelta del corso magistrale ?	(0)Per nulla (1)Più no che si (2)Più si che no (3)Moltissimo
Qual è il tuo livello della lingua inglese ?	(0)Principiante (A1-A2) (1)Intermedio (B1-B2) (2)Avanzato (C1-C2)
Quanto lo sbocco lavorativo e la relativa retribuzione futura ha influenzato la scelta del	(0)Per nulla (1)Più no che si

corso magistrale ?	(2)Più si che no (3)Moltissimo
Quale corso tra quelli scelti durante la laurea triennale ti ha condotto alla scelta della magistrale ?	(0)Calcolo Scientifico (1)Fisica (2)Fondamenti di Intelligenza Artificiale (3)Grafica ed Interattività (4)Interazione Uomo-Macchina (5)Mobile Programming (6)Programmazione Avanzata (7)Sicurezza (8)Simulazione
Di che argomento trattava il progetto organizzato dall'università di Salerno che ha influenzato la scelta della magistrale ?	(0)Cloud Computing (1)Sicurezza Informatica (2)Software Engineering and IT Management (3)IoT (4)Data Science & Machine Learning (5)Nessuno dei progetti svolti mi ha incentivato a scegliere un determinato corso magistrale
Su quale argomento si è incentrata la tua tesi triennale ?	(0)Cloud Computing (1)Sicurezza Informatica (2)Software Engineering and IT Management (3)IoT (4)Data Science & Machine Learning
È stato l'esame dove hai conseguito uno dei voti più alti ?	(0) No (1)SI

4. Algoritmo di clustering

La scelta dell'algoritmo di clustering è ricaduta sull'algoritmo di Kmeans. L'algoritmo k-means è un algoritmo di tipo non supervisionato, che analizza dei gruppi partizionali che permettono di suddividere un insieme di oggetti in *k* gruppi sulla base dei loro attributi.
Il K-cluster è stato semplice da trovare poiché le magistrali sono 5 (Cloud Computing, Sicurezza Informatica, Software Engineering and IT Management, IoT, Data Science & Machine Learning).

Per l'implementazione dell'algoritmo, sklearn ci offre una libreria che ci permette di creare un modello, specificando il numero di cluster che vogliamo,ci permette di allenarlo e poi su questo fare delle predizioni.

```
from sklearn.cluster import KMeans
import numpy as np
kmeans = KMeans(n_clusters=5, random_state=0).fit(X_new_train_res)
ris = kmeans.predict(X_new_test)
centroids = kmeans.cluster_centers_
```

Verifica del numero di cluster considerati

Per verificare ciò abbiamo utilizzato il punto di gomito che ci permette di identificare il numero di cluster ideali (andando a stabilire un compromesso tra errore e capacità di raggruppamento).

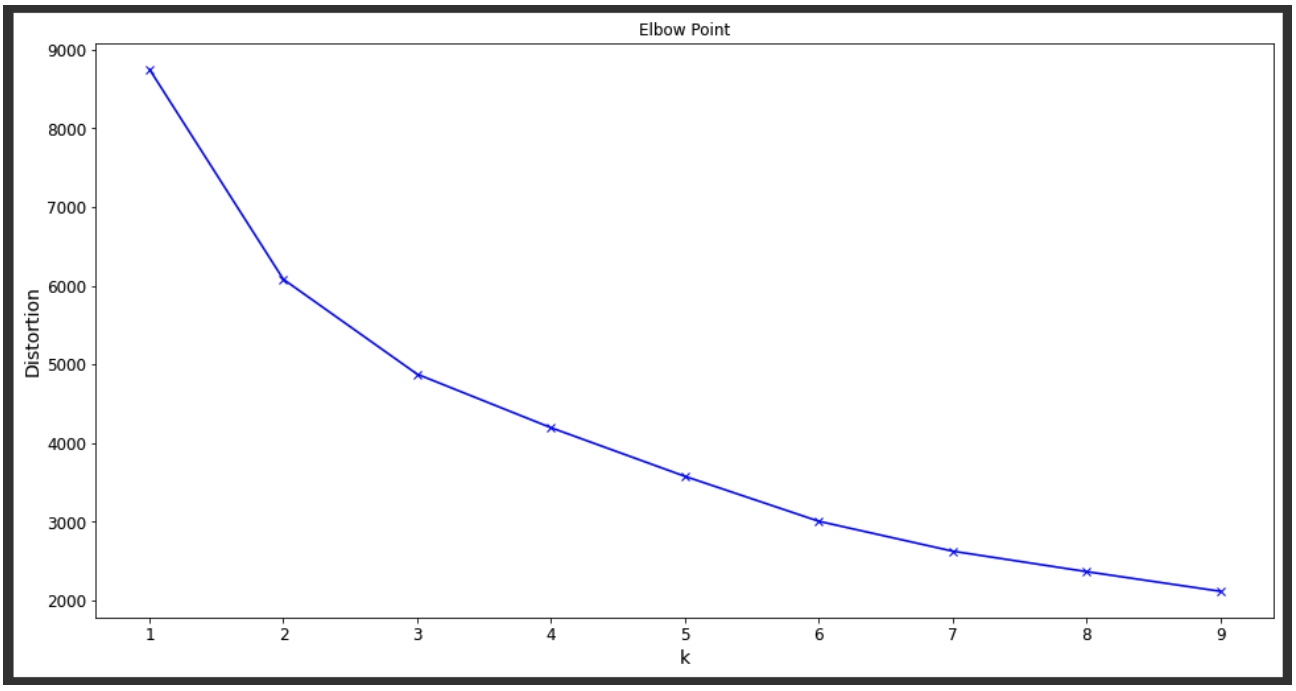
```
#Ottimizzazione per vedere il numero di cluster ideali
# verifico il numero di cluster ideali utilizzando l'elbow-method
# preparo un dizionario dove memorizzerò i valori della somma delle distanze al quadrato
# di ogni valore di K testato
sdq = []
K = range(1,10)
for k in K:
    # istanzio la classe di gestione di K-Means e un numero di cluster pari al k attuale
    # NOTA: utilizzo l'inizializzazione "k-means++" che inizializza i centroidi
    # in modo casuale ma garantendo che non si trovino troppo vicini l'uno con l'altro,
    # per minimizzare i problemi che questo porterebbe, soprattutto con valori di K elevati
    kmeans = KMeans(init="k-means++", n_clusters=k)
    kmeans.fit(X)

    # per il k attuale, memorizzo il valore di ssd
    sdq.append(kmeans.inertia_)
```

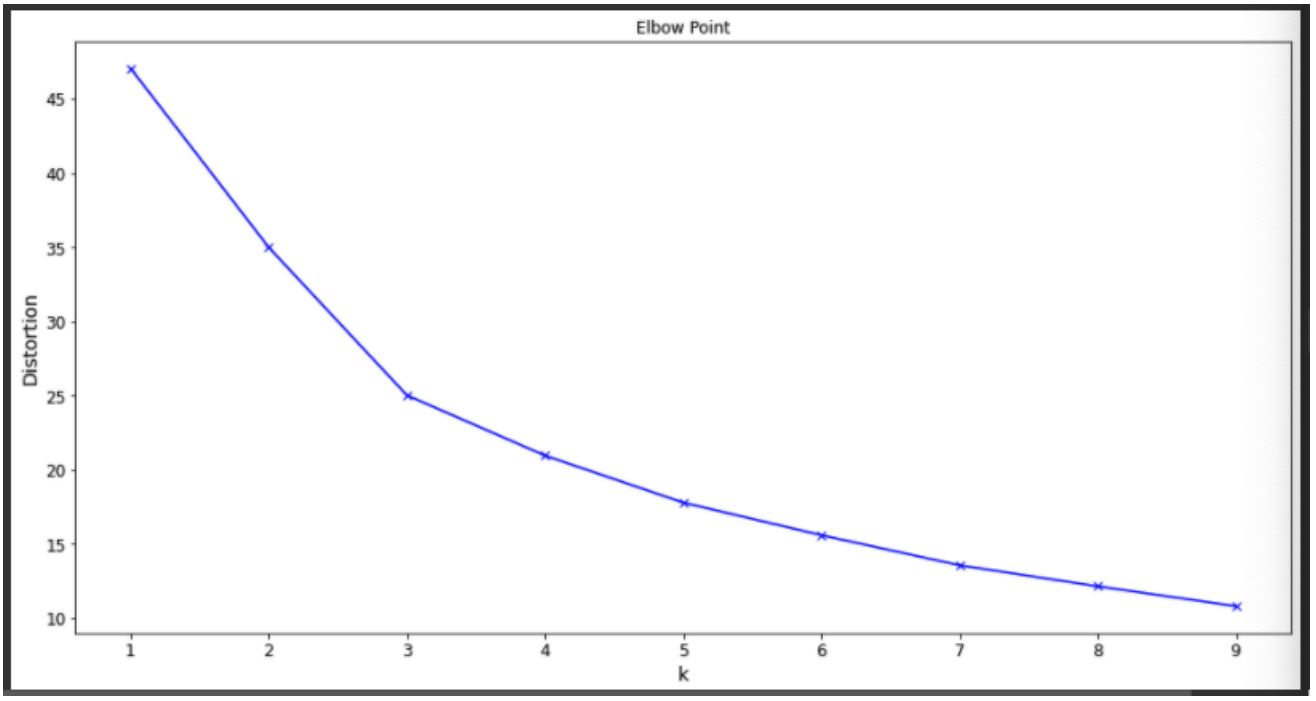
Visualizzazione del punto di gomito

```
plt.figure(figsize=(16,8))
plt.plot(K, sdq, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('Elbow Point')
plt.show()
```

Output DataSet Artificiale :



Output DataSet Reale:



5. Algoritmo di classificazione

Abbiamo deciso di utilizzare anche un algoritmo di classificazione per poter confrontare la bontà tra i due algoritmi.
La scelta dell'algoritmo di classificazione è ricaduta sull'algoritmo del Decision Tree.
L'algoritmo del Decision tree è un algoritmo di tipo supervisionato, che mira a creare un albero i cui nodi rappresentano un sottoinsieme di caratteristiche del problema e i cui archi rappresentano delle decisioni, attraverso l'information gain abbiamo misurato il grado di purezza di un attributo.

Anche in questo caso sklearn ci offre una libreria che ci permette di creare, allenare e testare un modello.

In questo caso va specificato il numero massimo dell'altezza dell'albero.

```
from sklearn.tree import DecisionTreeClassifier

tree_model = DecisionTreeClassifier(max_depth=12, random_state=42)


tree_model.fit(X_train, y_train)

y_pred = tree_model.predict(X_test)
```

Abbiamo deciso anche di far vedere quella che è la bontà della predizione e come possiamo vedere non è affatto ottima.

Output DataSet Artificiale (Prima dell'ottimizzazione):

	precision	recall	f1-score	support
0	0.19	0.32	0.24	19
1	0.52	0.36	0.43	33
2	0.82	0.71	0.76	52
3	0.33	0.50	0.40	6
4	0.30	0.32	0.31	19
accuracy			0.50	129
macro avg	0.43	0.44	0.43	129
weighted avg	0.55	0.50	0.52	129

Accuracy: 0.49612403100775193

Output DataSet Reale (Prima dell'ottimizzazione):

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2
1	0.50	0.50	0.50	4
2	0.00	0.00	0.00	1
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	0
accuracy			0.25	8
macro avg	0.10	0.10	0.10	8
weighted avg	0.25	0.25	0.25	8

Accuracy: 0.25

Abbiamo deciso di applicare un algoritmo di ricerca (GridSearch) per andare a prendere l'altezza ideale dell'albero.

Sotto sono riportati i nuovi dati relativi all'accuratezza.

```
from sklearn.model_selection import GridSearchCV

params = {'max_leaf_nodes': list(range(10, 100)), 'min_samples_split': [9, 10, 11]}
grid_search_cv = GridSearchCV(DecisionTreeClassifier(random_state=42), params, n_jobs=-1, verbose=1, cv=3)

grid_search_cv.fit(X_train, y_train)
```

```
from sklearn.metrics import accuracy_score

y_pred = grid_search_cv.predict(X_test)
accuracy_score(y_test, y_pred)

print(confusion_matrix(y_test, y_pred, labels=labels))
print(classification_report(y_test, y_pred))
```

Output DataSet Artificiale (Dopo l'ottimizzazione):

	precision	recall	f1-score	support
0	0.22	0.42	0.29	19
1	0.53	0.52	0.52	33
2	0.73	0.67	0.70	52
3	0.38	0.50	0.43	6
4	0.00	0.00	0.00	19
accuracy			0.49	129
macro avg	0.37	0.42	0.39	129
weighted avg	0.48	0.49	0.48	129

Output DataSet Reale (Dopo l'ottimizzazione):

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2
1	0.67	0.50	0.57	4
2	0.00	0.00	0.00	1
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	0
accuracy			0.25	8
macro avg	0.13	0.10	0.11	8
weighted avg	0.33	0.25	0.29	8

6. Considerazione dei due algoritmi scelti

Esaminando i due algoritmi scelti e verificando la loro accuratezza abbiamo stabilito che il K-Means raggiunge un'accuratezza di circa il 50% con il DataSetReale e del 30% con il DataSetArtificiale, invece il DecisionTree raggiunge un'accuratezza di circa il 25% con il DataSetReale e del 50% con il DataSetArtificiale.

Accuracy KMeans

DataSet Reale:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	1.00	0.67	0.80	3
2	0.00	0.00	0.00	1
3	1.00	1.00	1.00	1
4	0.00	0.00	0.00	0
accuracy			0.50	6
macro avg	0.40	0.33	0.36	6
weighted avg	0.67	0.50	0.57	6

Accuracy: 0.5

In questo caso anche se si raggiunge il 50% di accuratezza possiamo osservare che "0, 2, 4" non verranno mai predetti dall'agente.

DataSet Artificiale:

	precision	recall	f1-score	support
0	0.10	0.11	0.10	19
1	0.45	0.58	0.51	33
2	0.46	0.31	0.37	52
3	0.13	0.33	0.19	6
4	0.06	0.05	0.06	19
accuracy			0.31	129
macro avg	0.24	0.27	0.24	129
weighted avg	0.33	0.31	0.31	129

Accuracy: 0.31007751937984496

In questo caso raggiungiamo circa il 30% di accuratezza e possiamo osservare che "1" avrà una predizione maggiore rispetto agli altre target.

Accuracy DecisionTree

DataSet Reale:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2
1	0.50	0.50	0.50	4
2	0.00	0.00	0.00	1
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	0
accuracy			0.25	8
macro avg	0.10	0.10	0.10	8
weighted avg	0.25	0.25	0.25	8

Accuracy: 0.25

In questo caso raggiungiamo circa il 25% di accuratezza e possiamo osservare che non vengono predetti mai "0, 2, 3, 4" e per questo non è opportuno utilizzare questo dataset per eseguire predizioni.

DataSet Artificiale:

	precision	recall	f1-score	support
0	0.19	0.32	0.24	19
1	0.52	0.36	0.43	33
2	0.82	0.71	0.76	52
3	0.33	0.50	0.40	6
4	0.30	0.32	0.31	19
accuracy			0.50	129
macro avg	0.43	0.44	0.43	129
weighted avg	0.55	0.50	0.52	129

Accuracy: 0.49612403100775193

In questo caso raggiungiamo circa il 50% di accuratezza e possiamo osservare che vengono predetti maggiormente "2, 3" ma rispetto ai precedenti casi osservati è quello più consono per un eventuale utilizzo.

7. Integrazione con il sistema UniNotes

UniNotes nasce come una web application sviluppata in java, mentre il modulo sviluppato per la predizione è stato sviluppato in Python, per andare ad unire queste due tecnologia abbiamo pensato di utilizzare la potenza dei Web Service i quali si basano sull'xml e sappiamo bene che l'xml è uno degli standard più utilizzati tra piattaforme eterogenee.

Quindi per la comunicazione tra questi moduli abbiamo deciso di utilizzare un linguaggio comune : JSON.

Per implementare ciò abbiamo realizzato:

1. MagistraleServiceImpl che utilizza i metodi offerti da PythonAdapter.

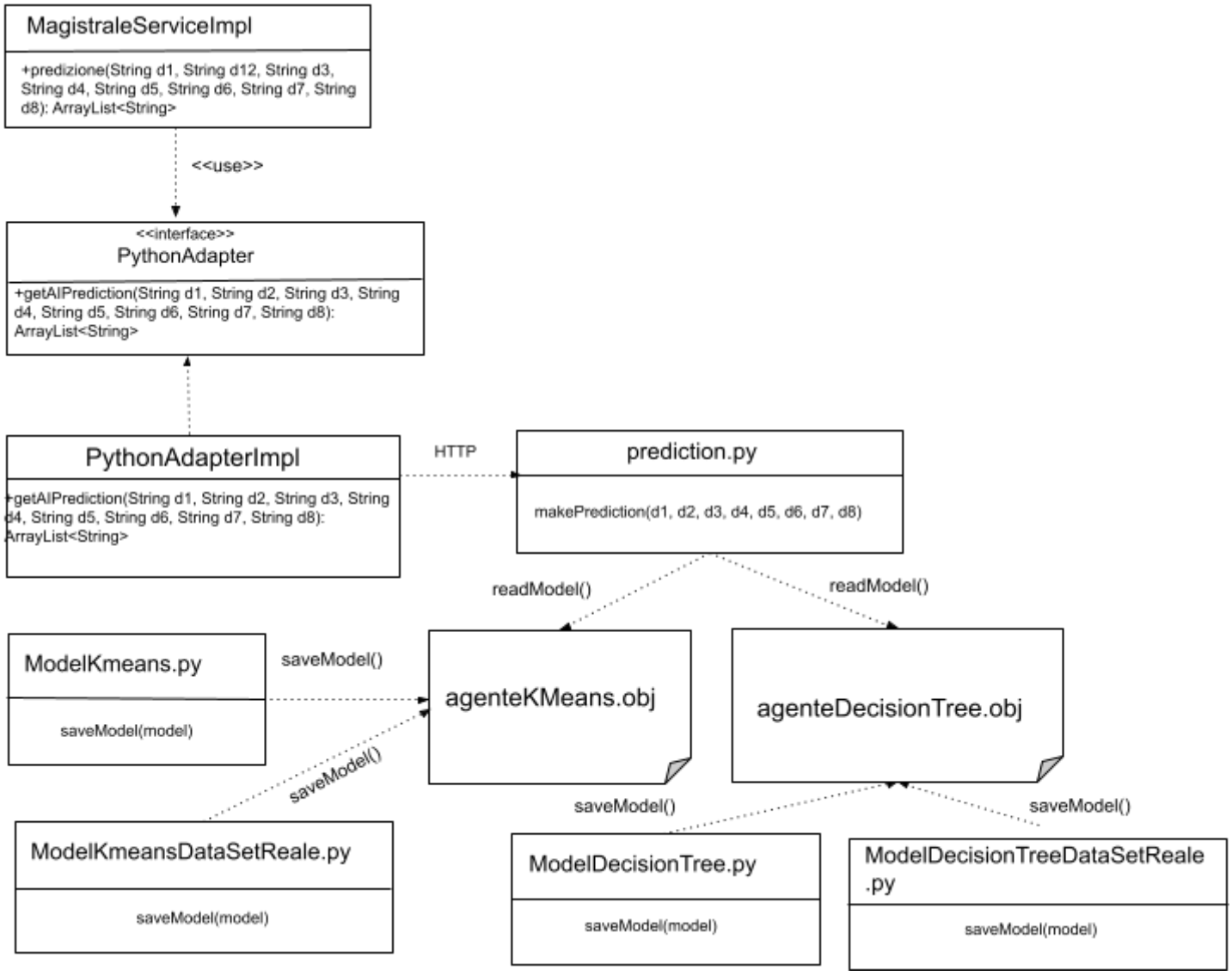
2. PythonAdapter espone il metodo per effettuare la predizione getAIPrediction(...).

3. PythonAdapterImpl che implementa PythonAdapter, e all'interno di questa classe andiamo ad implementare il metodo getAIPrediction(...).
   In questo metodo andiamo a creare la stringa JSON da inviare a Prediction.py e una volta inviata attendiamo il risultato della predizione.

4. Prediction.py è una classe scritta in Python che ha la funzionalità di trattare uno script Python come un Web Service in modo da mettere a disposizione il servizio di predizione.
   Per far ciò abbiamo utilizzato un micro-framework in python,Flask, che ci permette appunto di fornire il servizio di predizione.
   Questo servizio può essere richiamato facendo una richiesta HTTP all'URL localhost:5000/.

5. agenteKMeans e agenteDecisionTree sono i due modelli allenati rispettivamente con l'algoritmo di KMeans e DecisionTree.

6. Infine abbiamo
   a. ModelKMeans.py che va ad allenare il modello con il dataset Artificiale utilizzando KMeans;
   b. ModelKMeansDataSetReale.py che va ad allenare il modello con il dataset reale utilizzando KMeans;
   c. ModelDecisionTree.py che va ad allenare il modello con il dataset Artificiale utilizzando il DecisionTree;
   d. ModelDecisionTreeDataSetReale.py che va ad allenare il modello con il dataset reale utilizzando il DecisionTree;



8. Glossario

- **UniNotes :** Piattaforma web che permette di accedere facilmente a contenuti universitari.
- **DataSet :** Collezione di dati, ovvero i campioni raccolti.
- **Dona** : Nome dell'agente intelligente.
- **Flask** : Micro-framework per offrire Web Service.
- **Colab** : Colab è una piattaforma che ci permette di eseguire codice python sfruttando i server di Google.