CHIARA BERETTA
1033576

k-means

Ai e machine learning per il marketing

Obiettivi del progetto:

Comprendere e sperimentare il funzionamento degli algoritmi di machine learning attraverso la creazione di un modello di segmentazione clienti con k-means attraverso l’utilizzo dell’intelligenza artificiale

Strumenti utilizzati:

- ChatGPT e Cloude
- JupyterLab
- Streamlit
- GitHub

Perchè il k–medie ?

Il k-medie è un algoritmo non supervisionato che suddivide i dati in gruppi omogenei in base alla distanza da un centroide, rendendolo particolarmente adatto alla segmentazione dei clienti nel marketing.

Per rappresentare questa logica, ho scelto di lavorare su un esempio vicino alla realtà: un e-commerce dedicato al pet care e alla cura degli animali.

Ogni cluster riflette un diverso profilo d’acquisto e l’algoritmo consente di assegnare dinamicamente nuovi utenti al gruppo più coerente con le loro abitudini, simulando il comportamento di un sito che utilizza i dati raccolti per proporre strategie su misura

Come funziona il k–medie ?

L’algoritmo delle k-medie raggruppa un insieme di dati in k gruppi distinti (cluster), basandosi sulla vicinanza tra i punti.

I passaggi:

- Si scelgono il numero di cluster (k) da creare
- Vengono inizializzati casualmente k centroidi (punti centrali di ciascun cluster)
- Ogni punto del dataset viene assegnato al centroide più vicino
- Si calcolano nuovi centroidi per ogni gruppo usando la media dei punti assegnati.
- I punti vengono riassegnati ai nuovi centroidi.
- Il processo si ripete finché i centroidi non cambiano più

Alla fine, ogni punto è associato al cluster il cui centroide è più vicino

Fase 1: creazione del database con Chatgpt e Cloude

PROMT:

"Crea un esempio in python di segmentazione clienti usando l'algoritmo k-means. I dati devono rappresentare 4 gruppi di clienti, con caratteristiche sintetiche ma realistiche: spesa media mensile e numero di acquisti. Visualizza i cluster ottenuti e assegna un nuovo cliente al gruppo più vicino. Genera un codice eseguibile su JupyterLab, con una visualizzazione grafica chiara che evidenzi anche i centroidi e il nuovo cliente."

Codice creato:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

Impostazione della librerie che permettono di gestire dati numerici, visualizzarli graficamente ed eseguire la segmentazione tramite l'algoritmo K-Means

```python
np.random.seed(42)

cluster1 = np.column_stack((
    np.random.normal(loc=160, scale=6, size=50),
    np.random.normal(loc=4, scale=1.2, size=50)
))

cluster2 = np.column_stack((
    np.random.normal(loc=100, scale=6, size=50),
    np.random.normal(loc=10, scale=1.2, size=50)
))

cluster3 = np.column_stack((
    np.random.normal(loc=130, scale=6, size=50),
    np.random.normal(loc=5, scale=1.2, size=50)
))

cluster4 = np.column_stack((
    np.random.normal(loc=115, scale=6, size=50),
    np.random.normal(loc=13, scale=1.2, size=50)
))

X = np.vstack((cluster1, cluster2, cluster3, cluster4))
X[:, 1] = np.clip(X[:, 1], a_min=1, a_max=None)

nuovo_cliente = np.array([[120, 6]])
```

Generazione dei dati: simulazione di clienti con caratteristiche diverse

```python
kmeans = KMeans(n_clusters=4, random_state=0)
kmeans.fit(X)
```

Addestramento del modello: applicazione dell'algoritmo K-Means

```python
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
cluster_assegnato = kmeans.predict(nuovo_cliente)

print(f"Il nuovo cliente {nuovo_cliente[0]} è stato assegnato al cluster:", cluster_assegnato[0])
```

Predizione: assegnazione del cluster al nuovo cliente

```python
plt.figure(figsize=(10, 7))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='Accent', edgecolor='k', s=60, alpha=0.75)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=250, label='Centroidi')
plt.scatter(nuovo_cliente[0, 0], nuovo_cliente[0, 1],
            facecolors='white', edgecolors='gray',
            marker='*', s=300, linewidths=3, label='Nuovo Cliente')
plt.xlabel("Spesa media mensile (€)")
plt.ylabel("Numero di acquisti mensili")
plt.title("Segmentazione clienti con K-Means")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Visualizzazione: rappresentazione grafica dei cluster, dei centroidi e del nuovo punto

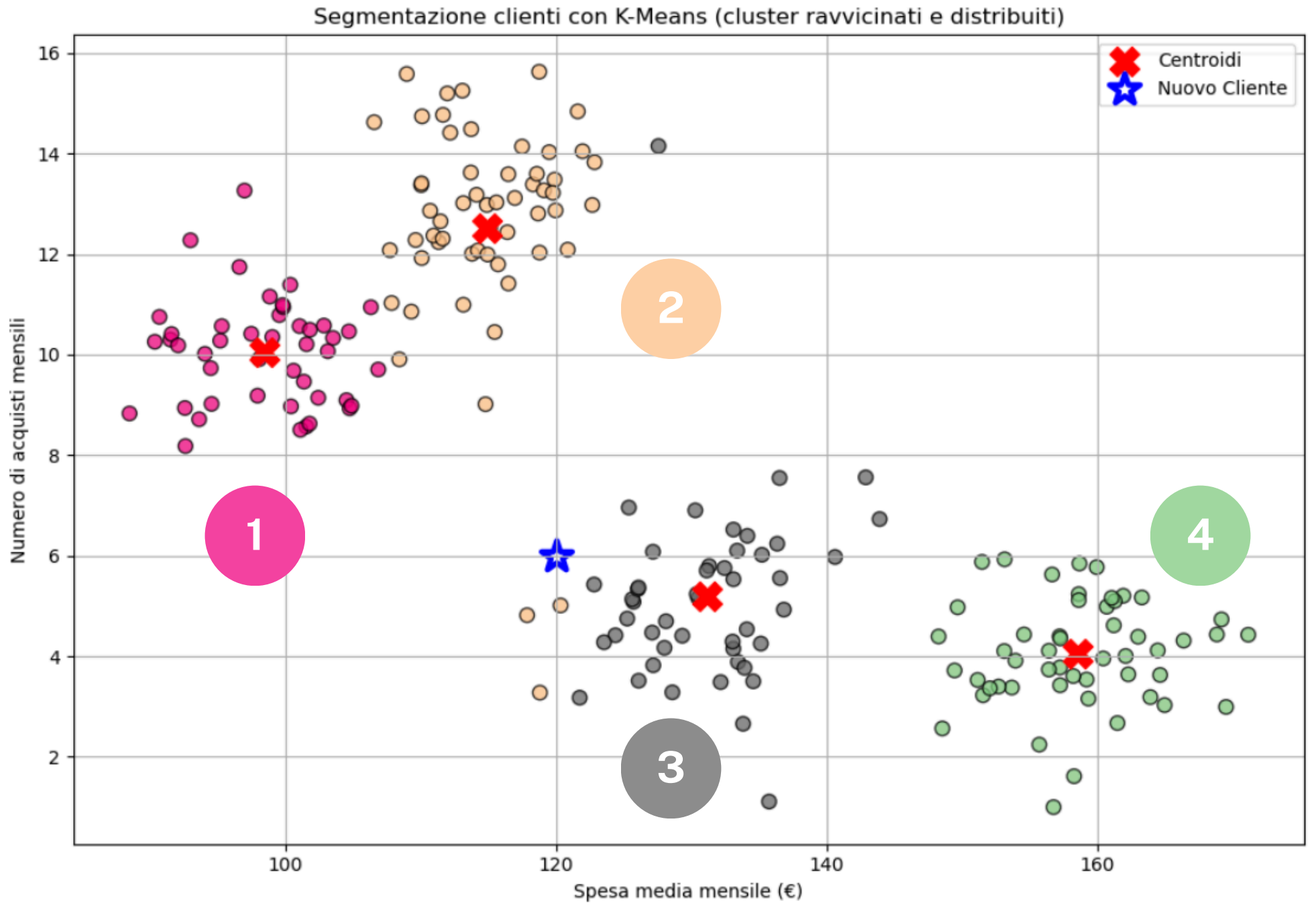
Campione:

Formato da 200 osservazione

Variabili:

- X = spesa media mensile (€)
- Y = numero di acquisti mensili

Visualizzazione:



Visualizzazioe con JupyterLab

Descrizione cluster indivisuati:

1 Cacciatori di offerte

Spesa media mensile: €100 circa
Numero di acquisti mensili: 10 + σ
Descrizione: clienti che effettuano acquisti frequenti ma di poco valore (ex. puntando a offerte e promozioni su prodotti essenziali come cibo secco, lettiere e snack). Sono fidelizzati perchè fanno molto acquisti ma molto attenti al risparmio

2 Pet lover organizzati

Spesa media mensile: €115 circa
Numero di acquisti mensili: 13 + σ
Descrizione: clienti molto attivi che effettuano acquisti molto spesso. Alterneranno quindi prodotti base ad articoli extra (ex. giochi, snack funzionali, antiparassitari), pianificando gli ordini con attenzione.

3 Animali affezionati

Spesa media mensile: €130 circa
Numero di acquisti mensili: 5 + σ
Descrizione: clienti con un comportamento regolare, che acquistano cibo, prodotti e per la cura. Rappresentano la base stabile e prevedibile dell'e-commerce.

4 Amici animali premium

Spesa media mensile: €160 circa
Numero di acquisti mensili: 4 + σ
Descrizione: clienti che acquistano meno frequentemente ma scelgono prodotti di alta qualità (ex. crocchette biologiche, accessori premium, integratori specifici ...)

Strategie per ogni cluster:

1 Cacciatori di offerte

- Newsletter settimanale con sconti mirati
- Bundle promozionali (ex. 2+1 su snack)
- Programma fedeltà a punti → premia la quantità di acquisti
- Sezione “offerte della settimana” in homepage

2 Pet lover organizzati

- Abbonamenti personalizzati
- Reminder per riacquisto (ex. email o notifica: "Sta per finire la scorta di...")
- Promo early-access: “prenota ora le novità prima degli altri”

3 Animali affezionati

- Email mensile personalizzata con consigli
- Sconti alle ricorrenze (ex. anniversario primo ordine, compleanno pet ...)
- Up-selling mirato
- Test nuovi prodotti gratuiti

4 Amici animali premium

- Upselling mirato
- Packaging premium e personalizzato
- Storytelling dei prodotti via e-mail
- Programma VIP esclusivo

Un nuovo cliente

Il nuovo cliente creato casualemente da ChatGPT ha assunto delle variabili casuali.

In base a questi dati, l'algoritmo k-means ha assegnato il cliente al cluster 2, denominato "Pet lover organizzati". Questo comportamento dopo un po' di mesi di osservazione suggerisce che il cliente ha già iniziato ad acquistare con regolarità sul nostro sito dimostrando coinvolto e soddisfazione della propria esperienza. Quindi anche su di esso si posso attivare le strategie relative al cluster 2.

Fase 2: quale strategia marketing è più adatta per il singolo?

Dal modello statico all'app interattiva

Dopo la simulazione e visualizzazione dei cluster con JupyterLab, l'obiettivo è stato quello di simulare una reale applicazione dell'algoritmo k-means con altri dati esterni al database inseriti casualmente su un'app web

Perché? questa applicazione potrebbe simulare il reale funzionamento di un sistema automatico di clusterizzazione e raccomandazione quando si raccolgono abbastanza dati si un nuovo cliente

Dalla staticità all’interatività

Basandomi sul database scritto in Python in precedenza si sono presi i centroidi di ogni cluster insieme alla deviazione standard di ognuno

Dopo ho utilizzato Streamlit per trasformare il modello in un'app:

- Teoricamente il sito dopo un determinato periodo di utilizzo di un nuovo utente elabora due valori (spesa e acquisti) ma all’interno dell’app web lo faremo noi
- Analizzando questi due valori con il modello k-means lo si assegna al cluster più vicino
- L’app restituisce:
  - ➜ il profilo assegnato
  - ➜ le strategie consigliate
  - ➜ il grafico dinamico con il punto evidenziato per verificare la coerenza

Pubblicazione online

Strumenti utilizzati per la messa online dell’app: GitHub + Streamlit Cloud

Per condividere pubblicamente l’app ho seguito questi passaggi:

- Caricato il codice su GitHub (app.py + requirements.txt)
- Collegato il repository a Streamlit Cloud
- Lanciato il sito web disponibile a tutti:

https://app-petcare-5zytjelaypr4p2glklgeql.streamlit.app/

Pubblicazione online: app.py

```python
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Palette colori e descrizioni
palette = {
    "Cacciatori di offerte": "deeppink",
    "Pet lover organizzati": "sandybrown",
    "Animali affezionati": "dimgray",
    "Amici animali premium": "mediumseagreen"
}

strategie = {
    "Cacciatori di offerte": [
        "📰 Newsletter settimanale",
        "🎁 Bundle 2+1 snack",
        "🏆 Programma punti",
        "🛒 Offerte in homepage"
    ],
    "Pet lover organizzati": [
        "📦 Abbonamenti personalizzati",
        "⏰ Reminder riacquisto",
        "✨ Promo early-access"
    ],
    "Animali affezionati": [
        "📧 Email mensile personalizzata",
        "🎉 Sconti ricorrenze",
        "💎 Up-selling mirato",
        "🧪 Test nuovi prodotti"
    ],
    "Amici animali premium": [
        "💎 Upselling mirato",
        "🎁 Packaging premium",
        "📖 Storytelling via email",
        "👑 Programma VIP"
    ]
}

# Simulazione dati coerente con il tuo grafico
np.random.seed(42)
cluster_specs = {
    "Cacciatori di offerte": {"mean": (100, 10), "std": (5, 1), "count": 50},
    "Pet lover organizzati": {"mean": (115, 13), "std": (5, 1), "count": 50},
    "Animali affezionati": {"mean": (130, 5), "std": (5, 1), "count": 50},
    "Amici animali premium": {"mean": (160, 4), "std": (5, 1), "count": 50},
}

data = []
for cluster, specs in cluster_specs.items():
    x = np.random.normal(specs["mean"][0], specs["std"][0], specs["count"])
    y = np.random.normal(specs["mean"][1], specs["std"][1], specs["count"])
    for xi, yi in zip(x, y):
        data.append({
            "Spesa": xi,
            "Acquisti": yi,
            "Cluster": cluster
        })

df = pd.DataFrame(data)

# Centroidi stimati
centroids = {
    "Cacciatori di offerte": (100, 10),
    "Pet lover organizzati": (115, 13),
    "Animali affezionati": (130, 5),
    "Amici animali premium": (160, 4)
}

# Funzione per assegnare cluster
def assegna_cluster(x, y):
    distanze = {k: np.sqrt((x - cx) ** 2 + (y - cy) ** 2) for k, (cx, cy) in centroids.items()}
    return min(distanze, key=distanze.get)

# Streamlit UI
st.title("\U0001F43E Scopri il tuo profilo cliente PetCare!")
st.markdown("""
Inserisci la **spesa media mensile** e il **numero di acquisti** per scoprire a quale cluster appartieni e quali strategie sono più adatte a te.
""")

spesa = st.number_input("\U0001F4B0 Spesa media mensile (€)", min_value=0.0, value=100.0, step=1.0)
acquisti = st.number_input("\U0001F6CD Numero di acquisti mensili", min_value=0, value=5, step=1)

profilo = assegna_cluster(spesa, acquisti)
st.markdown(f"### 🎯 Il tuo cluster è: **{profilo}**")
st.markdown("**\U0001F4CC Strategie consigliate per questo profilo:**")
for s in strategie[profilo]:
    st.write(f"- {s}")

# Grafico
fig, ax = plt.subplots(figsize=(10, 6))
for cluster in df["Cluster"].unique():
    subset = df[df["Cluster"] == cluster]
    ax.scatter(subset["Spesa"], subset["Acquisti"], label=cluster, color=palette[cluster], edgecolors='black')

for cluster, (x, y) in centroids.items():
    ax.scatter(x, y, color=palette[cluster], marker='X', s=200, edgecolors='black', label=None)

# Nuovo punto utente
ax.scatter(spesa, acquisti, color="gray", marker='*', s=200, edgecolors='black', label="Nuovo Cliente")

ax.set_xlabel("Spesa media mensile (€)")
ax.set_ylabel("Numero di acquisti mensili")
ax.set_title("Segmentazione clienti con K-Means")
ax.legend()
st.pyplot(fig)
```

Codice Python per la visualizzazione dell’applicazione

Pubblicazione online: requirements.txt

Installazione delle librerie:

1	`scikit-learn` ⟶	Per eseguire il clustering con l’algoritmo k-means
2	`streamlit` ⟶	Per costruire l'interfaccia interattiva dell'app web
3	`numpy` ⟶	Per creare e manipolare i dati simulati
4	`matplotlib` ⟶	Per visualizzare i cluster e i centroidi su grafico

Senza questo file, l’app non potrebbe funzionare correttamente online

Cosa ho imparato

- Ho compreso passo per passo la logica delle k-medie, approfondendo come si formano i cluster, cosa rappresentano i centroidi e in che modo un nuovo punto viene automaticamente assegnato al gruppo più vicino

- Ho generato un database realistico e visualizzato i gruppi con grafici chiari, grazie a un utilizzo consapevole dell'intelligenza artificiale generativa e delle librerie Python per la simulazione e l'analisi dei dati

- Ho sviluppato una web app interattiva con Streamlit e GitHub, trasformando il modello in un'interfaccia semplice, accessibile da browser e in grado di simulare ciò che farebbe un e-commerce reale dopo aver raccolto i dati dei clienti.

Grazie per l’attenzione !