Dipartimento Di Informatica
Università di Pisa

Progetto Data Mining

Studenti:
Luca de Martino
Raffaele Villani

Anno Accademico 2020/2021

List of Contents

1 Data Understanding

Data Understanding è la fase preliminare del processo di analisi dei dati, nella quale si comprendono le caratteristiche principali dei dati, come: distribuzioni, correlazioni delle features, tipo degli attributi, individuazione dei valori mancanti e degli outliers.

Le informazioni ottenute attraverso questo processo possono essere utilizzate per migliorare la qualità dei dati in modo da migliorare anche l'efficienza delle fasi successive.

Il dataset fornitoci consiste di **471910 record**, ciascuno composto da **8 feature**: *BasktetID, BaksetDate, Sale, CustomerID, CustomerCountry, ProdID, ProdDescr, Qta*. Rappresenta, quindi, un possibile database di transizioni eseguite su di un ipotetico e-commerce da clienti di nazioni diverse.

In questa sezione, effettuiamo data understanding sul dataset fornitoci.
In particolare, nella sezione 1.1 analiziamo gli attributi che descrivono ciascun record, ne mostriamo la distribuzione tramite grafici e le relative statistiche.
Nella sezione 1.2 viene effettuata una valutazione sulla qualità dei dati presenti all'interno del dataset.
Nella sezione 1.3 perseguiamo la trasformazione dei valori errati o mancanti.
Nella sezione 1.4 creiamo nuove feature e ne studiamo la correlazione per definire il nosro nuovo dataframe di informazioni.

1.1 Semantica dei dati, distribuzioni e statistiche

Per ottenere una migliore comprensione dei dati presentiamo una breve descrizione per ciascun attributo.

Nome	Descrizione	Tipo	Dominio
BasketID	ID della transizione	Stringa	Codice
BasketDate	rappresenta la data in cui la transizione è stata eseguita	Data Time	[01/03/11 - 31/10/11]
Sale	rappresenta il costo unitario del prodotto acquistato	Stringa	Z
CustomerID	Identificativo dell'utente che ha useguito la transizione	Float	$Q_{>0}$
CustomerCountry	rappresenta la nazione da cui é stata effettuata la transizione	Stringa	Paesi
ProdID	Identificativo del prodotto	Stringa	Codice
Qta	Quantità del prodotto acquistato	intero	Z

Table 1: Descrizione degli attributi del dataset

In particolar modo, in questa sezione ci soffermeremo sulla distribuzione dell'attributo categorico *CustomerCountry* e sulle date (BasketDate). La distribuzione delle variabili continue sarà affrontata nella sezione 1.2, per comprendere a pieno la qualità dei dati, siccome sono presenti caratteristiche particolarmente rilevanti. Abbiamo notato fin da subito che il dataset era sbilanciato per quanto concerne il valore *United Kingdom* di CustomerCountry, per affermare ciò

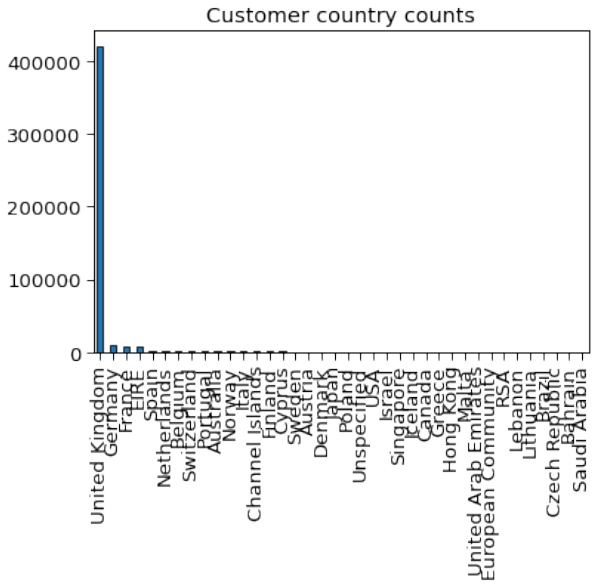


Figure 1: Distribuzione CustomerCountry con UK

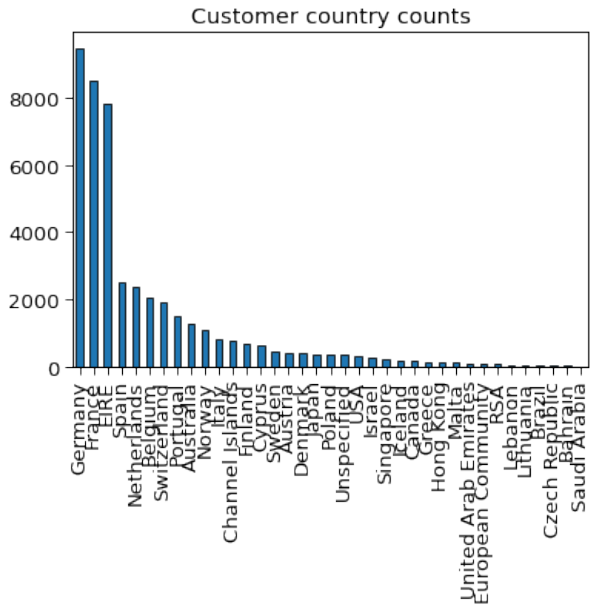


Figure 2: Distribuzione CustomerCountry senza UK

mostriamo sia la distribuzione 1 che la tabella 2 (top dieci delle transizioni in percentuale per nazione). Attraverso questa informazione possiamo notare che il negozio da cui vengono effettuate le transizioni potrebbe essere situato in UK e che è particolarmente diffuso negli UK rispetto alle altre nazioni.

CustomerCountry	Percentuale
United Kingdom	88.9%
Germany	2.3%
France	2.1%
EIRE	1.8%
Spain	0.6%
Netherlands	0.6%
Belgium	0.5%
Switzerland	0.5%
Portugal	0.4%
Australia	0.3%

Table 2: Prime 10 nazioni per numero di transazioni

La distribuzione di *BasketDate* può essere approssimata dall'istogramma 3.

La data più vecchia è *2010-01-12 08:26:00* mentre quella più recente è *2011-12-10 17:19:00*, anche in questo caso il dataset risultata essere sbilanciato rispetto gli anni, come confermato dalla figura 4.

Presentiamo il numero di transizioni eseguite nell'anno 2011 (fig. 5), che ci permette di comprendere che apparentemente abbiamo un declino delle vendite nel mese di Dicembre poiché abbiamo solo i primi 9 giorni di quest'ultimo, tuttavia il suo andamento lasciava presagire un risultato simile ai mesi strettamente precedenti, seguendo, quindi, un pattern in costante crescita sul numero di oggetti venduti.

Visualizziamo l'andamento per mese, settimana e ora delle vendite dai grafici 7,8,9 e possiamo notare che per quanto riguarda le vendite effettuate durante il giorno gli acquisti vengono effettuati a partire dalle ore 6:00 fino ad arrivare alle 20:00, dimostrando che l'utente tende a non effettuare acquisti durante la notte.

Avendo analizzato anche i BasketID vogliamo accennare una particolarità notata, ossia che alcuni di questi presen-

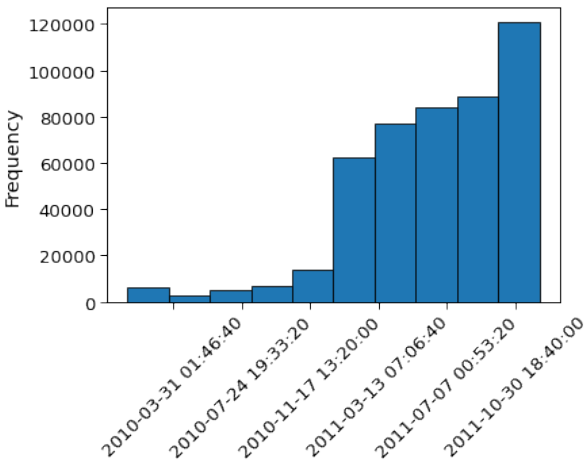


Figure 3: Distribuzione dei BasketDate

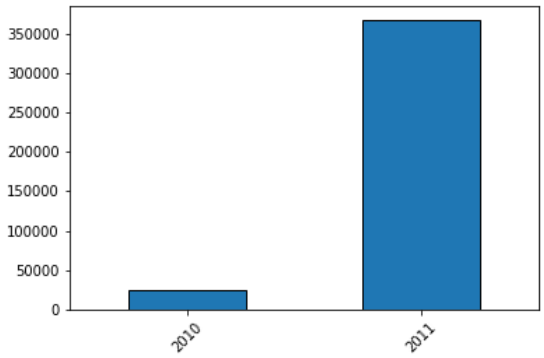


Figure 4: Transazioni per anno

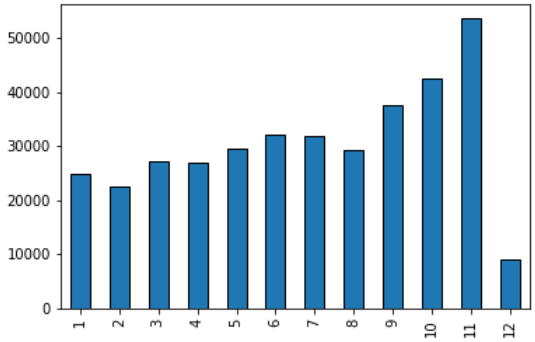


Figure 5: Transazioni per mese

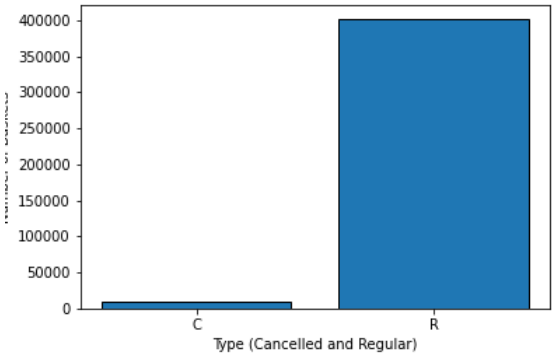


Figure 6: BasketID inizianti per "C" VS non inizianti per "C"

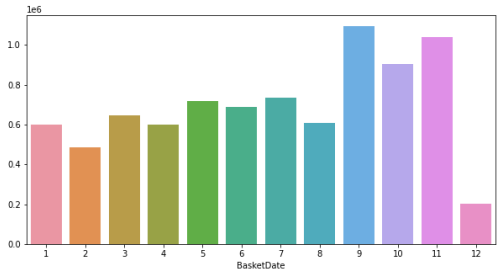


Figure 7: Mesi più proficui

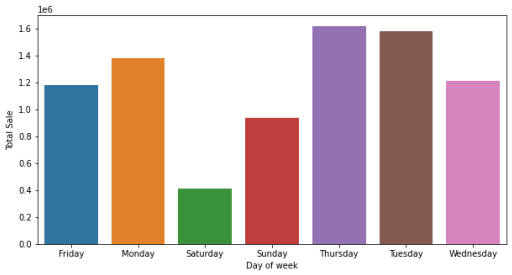


Figure 8: Settimane più proficue

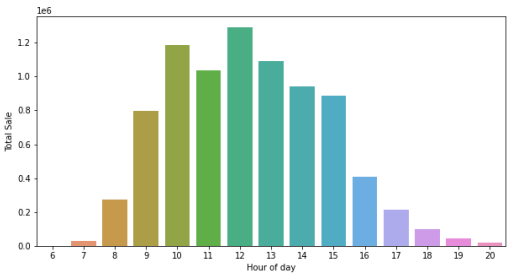


Figure 9: Ore più proficue

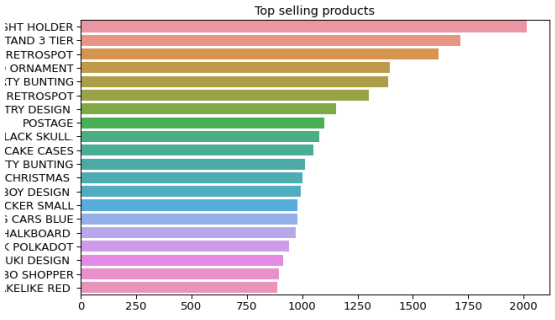


Figure 10: Prodotti più venduti

tano un identificativo che inizia con la lettera *C*. Attualmente riportiamo solo un grafico che mostra le distribuzioni di questi id speciali rispetto a quelli regolari (fig 6) e nei prossimi paragrafi approfondiremo l'argomento.

Possiamo capire meglio il comportamento dei compratori attraverso alcune distribuzioni calcolate, come ad esempio quella che possiamo vedere nella figura 10 che descrive quali sono le tipologie di prodotti più acquistati.

1.2 Valutazione della qualità dei dati

Nella seguente sezione analizzeremo la qualità dei dati presenti all'interno del nostro dataset. La fig. 11 ci fornisce una prima informazione riguardo le features aventi dei missing value e la posizione di quest'ultimi.

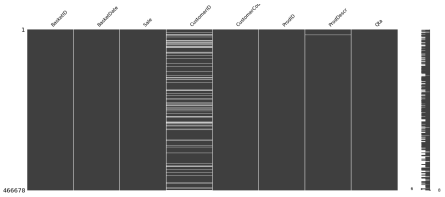


Figure 11: Distribuzione Missing Value

Questa ci permette di osservare come le uniche features incriminate di avere missing values sono *CustomerID* con 65080 valori mancanti e *ProdDescr* con 753 su un totale di 471910 righe. Questo però non garantisce il ritrovamento di tutti i missing value poiché abbiamo delle features di tipo numerico che possono avere dei vincoli sul dominio che se non rispettati possono essere visti come valori mancanti.
In particolare, facciamo delle assunzioni iniziali su *Sale* che non dovrebbe avere valori minori o uguali 0 (in quanto un prezzo unitario deve avere un valore, poiché un prodotto non può essere gratis o scalare denaro) e su *Qta* che anch'esso dovrebbe avere valori solo superiori a 0. Quest'ultima assunzione in particolare sarà poi smentita da un'analisi più approfondita.
Per quanto riguarda la qualità dei dati, abbiamo notato che il campo *Sale* nel dataset originale risulta essere di tipo Object mentre ci aspettiamo debba essere un valore numerico, in quanto rappresenta un prezzo. Discorso inverso vale per il campo *CustomerID* che nel dataset originale è presente come tipo float ma che ha più senso come tipo Object, in quanto rappresentate di una stringa univoca.

Siamo poi passati alla fase di *outlier detection*, per andare ad eliminare quei dati che potrebbero portare a risultati errati nelle analisi successive.
Possiamo vedere dalla figura 12 che sia il campo *Sale* che il campo *Qta* hanno numerosi outliers, quindi passiamo all'analisi di questi ultimi attraverso la tecnica *interquartile range* e rimuoviamo quelli in eccesso ottenendo cosí le distribuzioni mostrate in figura 13 e figura 14.

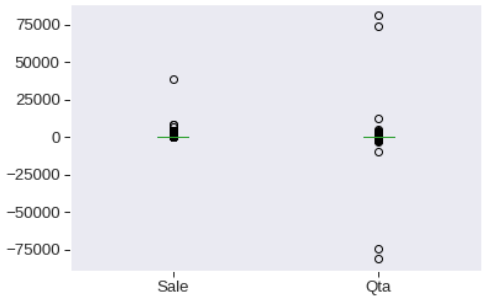


Figure 12: Outlier in Sale e Qta

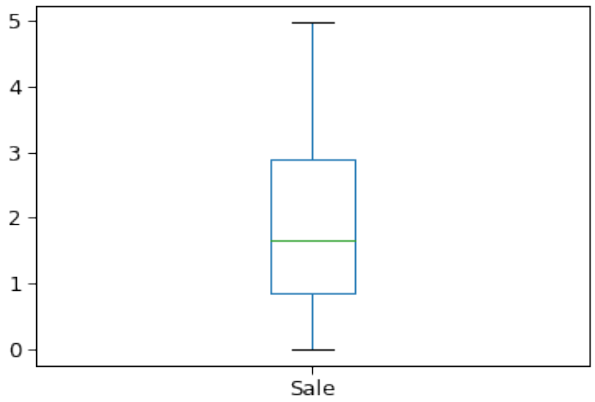


Figure 13: Box Plot Sale senza outlier

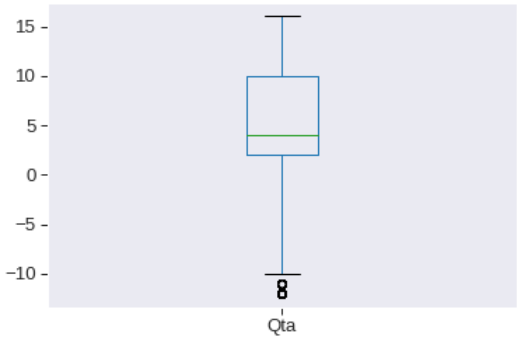


Figure 14: Box Plot Qta senza outlier

1.3 Gestione e trasformazione degli attributi mancanti o errati

In questa fase ci siamo occupati della gestione degli attributi, come prima verifica abbiamo effettuato un'analisi sui dati per capire se fossero presenti dei dati duplicati, abbiamo trovato 5232 dati duplicati, che abbiamo deciso di eliminare dal dataset siccome non fornivano nessuna informazione aggiuntiva e potevano modificare la distribuzione degli attributi. Per le motivazioni del paragrafo 1.2 abbiamo deciso di trasformare il *Sale* in un tipo Float, così da poter ottenere una migliore analisi.
Per quanto concerne la gestione dei dati mancanti, preso atto che le uniche features aventi valori nulli (non valori che non rispettino un vincolo) sono *CustomerID* e *ProdDescr*, le analizziamo nello specifico. Come possiamo vedere dalla tabella 3 i valori mancanti per il CustomerID sono quasi il 15% dei dati, siccome è una buona parte dei dati che abbiamo a disposizione, abbiamo provato varie tecniche per tentare di ripristinarli. Abbiamo controllato se questi valori fossero collegati a un *BasketID* che a sua volta contenesse un *CustomerID* valido in altre righe, tuttavia non siamo riusciti a recuperare nessun *CustomerID*, quindi abbiamo deciso di rimuoverne i record associati.

Feature	Missing Value	% Percentage Value
CustomerID	65073	13.9%
ProdDescr	653	0.2%

Table 3: Missing value

Per quanto riguarda i valori mancanti del ProdDescr abbiamo provato a recuperarli usando il ProductID degli altri record con un successo del 100%. In definitiva, non abbiamo più missing value per il ProdDescr.

Approfondendo l'analisi abbiamo inoltre compreso che le Qta negative erano associate a BasketID con valore che iniziava sempre per lettera "C" come mostrato dalla fig. 6. Abbiamo così dedotto che questi possano corrispondere a resi di uno o più prodotti fatte da un utente, per tal motivo le Qta negative sono state mantenute.

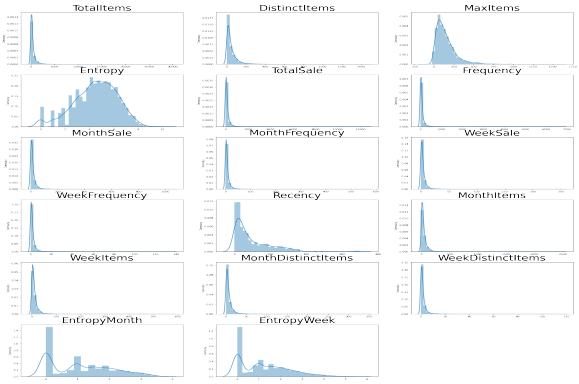
Per quanto riguarda i valori negativi relativi al *Sale*, questi sono stati rimossi in automatico nel momento in cui abbiamo eliminato i record senza CustomerID recuperabile. Per quanto concerne, invece, i 1279 valori di Sale con valore pari a 0 abbiamo deciso di tentarne il recupero tramite il ProdID. Abbiamo assunto di usare la media del prezzo associato nel caso in cui un prodotto con stesso ProdID avesse più prezzi unitari validi. Anche in questo caso il successo è stato del 100% permettondoci di eliminare i "missing value" precedenti.

1.4 Creazione di nuove feature

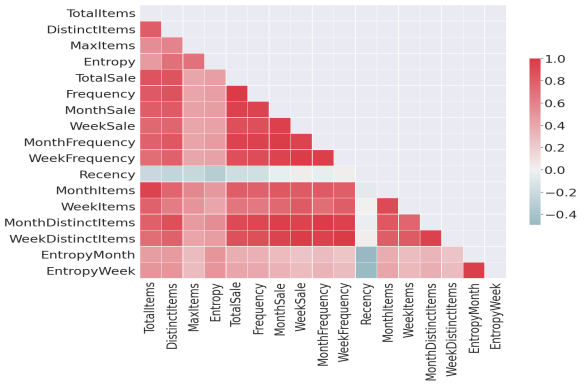
In questa sezione ci siamo dedicati nella creazione di indici provenienti dal dataset precedente al fine di poterle utilizzare per le successive analisi. Le feature create sono le seguenti:

- ***TotalSale*** che corrisponde al saldo totale di un utente durante tutto il periodo di osservazione;
- ***TotalItems*** che corrisponde al numero di oggetti acquistati da un utente durante tutto il periodo di osservazione;
- ***DistinctItems*** che corrisponde al numero di oggetti *distinti* comprati da un utente durante tutto il periodo di osservazione;
- ***MaxItems*** che corrisponde al numero massimo di oggetti comprati da un utente durante tutto il periodo di osservazione;
- ***Entropy*** che corrisponde alla Shannon entropy sul productID, per misurare la varietà di prodotti acquistati da un utente durante tutto il periodo di osservazione;
- ***Frequency*** che corrisponde al numero di volte in cui un untente ha acquistato qualcosa;
- ***CustomerCountry*** che abbiamo deciso di mantenere in quanto informativa sulla nazione di acquisto dell'utente;
- ***Recency*** che corrisponde a quanto tempo è passato dall'ultimo acquisto dell'utente;
- ***MonthSale*** che corrisponde alla media del saldo di un utente durante un mese di osservazione;
- ***WeekSale*** che corrisponde alla media del saldo di un utente durante una settimana di osservazione;
- ***MonthFrequency*** che corrisponde alla media del numero di volte in cui un untente ha acquistato qualcosa durante un mese di osservazione;
- ***WeekFrequency*** che corrisponde alla media del numero di volte in cui un untente ha acquistato qualcosa durante una settimana di osservazione;
- ***MonthItems*** che corrisponde alla media del numero di oggetti acquitati da un utente qualcosa durante un mese di osservazione;
- ***WeekItems*** che corrisponde alla media del numero di oggetti acquitati da un utente qualcosa durante una settimana di osservazione;
- ***WeekDistinctItems*** che corrisponde alla media del numero di oggetti distinti acquitati da un utente qualcosa durante una settimana di osservazione;
- ***EntropyMonth*** che corrisponde alla Shannon entropy sul productID, per misurare la varietà di prodotti acquistati in media da un utente durante un mese di osservazione;
- ***EntropyWeek*** che corrisponde alla Shannon entropy sul productID, per misurare la varietà di prodotti acquistati in media da un utente durante una settimana di osservazione;

Tuttavia, data l'alta correlazione tra gli attributi per mese, settimana e tutto il periodo di osservazione abbiamo deciso di utilizzare per la parte relativa al clustering soltanto gli attributi relativi al perdiodo temporale di un mese, perché ci sembrava un buon compromesso tra i dati e la generalizzazione del comportamento di un utente. La distribuzione dei sopracitati attributi è consultabile nella fig. 15a mentre la figura 15b mostra la correlazione tra le feature selezionate.



(a) Distribuzione delle nuove feature



(b) Matrice di correlazione delle nuove feature

2 Clustering

In questa sezione andremo ad applicare tre delle principali tecniche per la clustering analysis (K-Means, DB-Scan, Hierarchical Clustering) e ne analizzeremo i relativi risultati.

2.1 Normalizzazione

Come possiamo dalla figura 15a le distribuzioni dei nuovi attributi correlati sono tutte simili ad una distribuzione *skewed*, per tale motivo, consci del fatto che una tale distribuzione potrebbe portare a risultati non ottimali optiamo per l'utilizzo della normalizzazione per ottenere distribuzioni quanto più vicine alla distribuzione *gaussiana* possibile. Processiamo tutti gli attributi attraverso la funzione *logScale*, per poi normalizzarli tramite lo *standard z-score*, ottenendo i risultati mostrati nella figura 16.

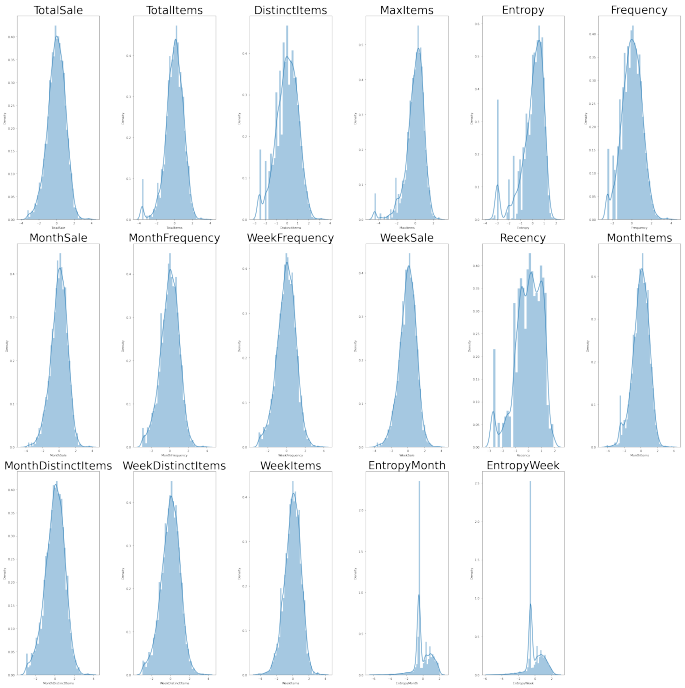


Figure 16: Distribuzione nuove feature scalate

2.2 K-means

La sezione seguente presenterà tutte le scelte effettuate per l'analisi dell' algoritmo K-means e i risultati di quest'ultimo.

2.2.1 Selezione degli attributi

Per quanto riguarda gli attributi utilizzati, abbiamo deciso di non utilizzarli tutti siccome come si può vedere in figura 15b i nuovi attributi calcolati sono molto correlati tra loro. Abbiamo individuato quelli più rilevanti per l'analisi in **MonthSale, MonthFrequency, MonthItems, MonthDistinctItems**.

2.2.2 Identificazione del miglior K

Per scegliere il miglior valore di k per il K-means, abbiamo deciso di utilizzare come parametri di valutazione l'andamento dell'*SSE* e della *Silhouette* attraverso l'utilizzo del *Knee Method* per $k \in [2, 10]$ e con 20 come numero di inizializzazioni randomiche dei centroidi, come possiamo vedere nelle figure 17, 18.

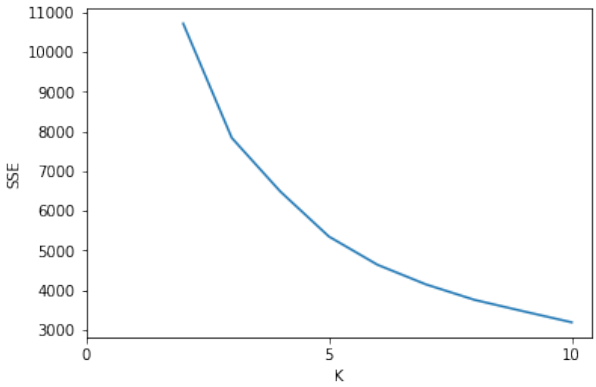


Figure 17: **SSE** al variare di k

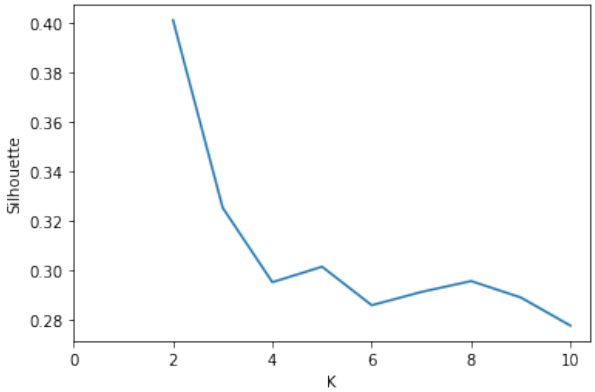


Figure 18: **Silhouette** al variare di k

Per valutare il miglior valore di k abbiamo quindi sia utilizzato l'*Elbow Method heuristic* che rappresenta un buon trade-off tra l'intepretabilità dei dati e l'SSE, che ci fornisce come miglior k il valore 4, sia l' average silhouette method che indica un k=2. Siccome può succedere che non si abbia un numero univoco di cluster proprio perché i dati non sono effettivamente ben clusterizzati, inoltre in base ai vari esperimenti effettuati, optiamo per una via che sia nel mezzo tra le due soluzioni e decidiamo di utilizzare un k=3 con 50 inizializzazioni randomiche dei centroidi.

2.2.3 Clustering Analysis

Per comprendere meglio il risultato ottenuto presentiamo il grafico che rappresenta i centroidi in coordinate parallele 19. Si può notare che tutti i centroidi sono ben separati in tre classi bassa, media e alta per tutti attributi utilizzati.

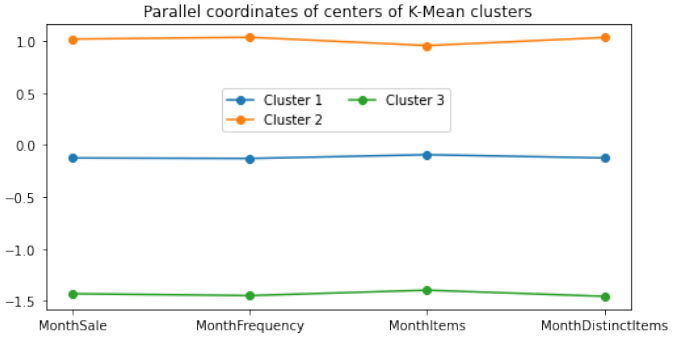
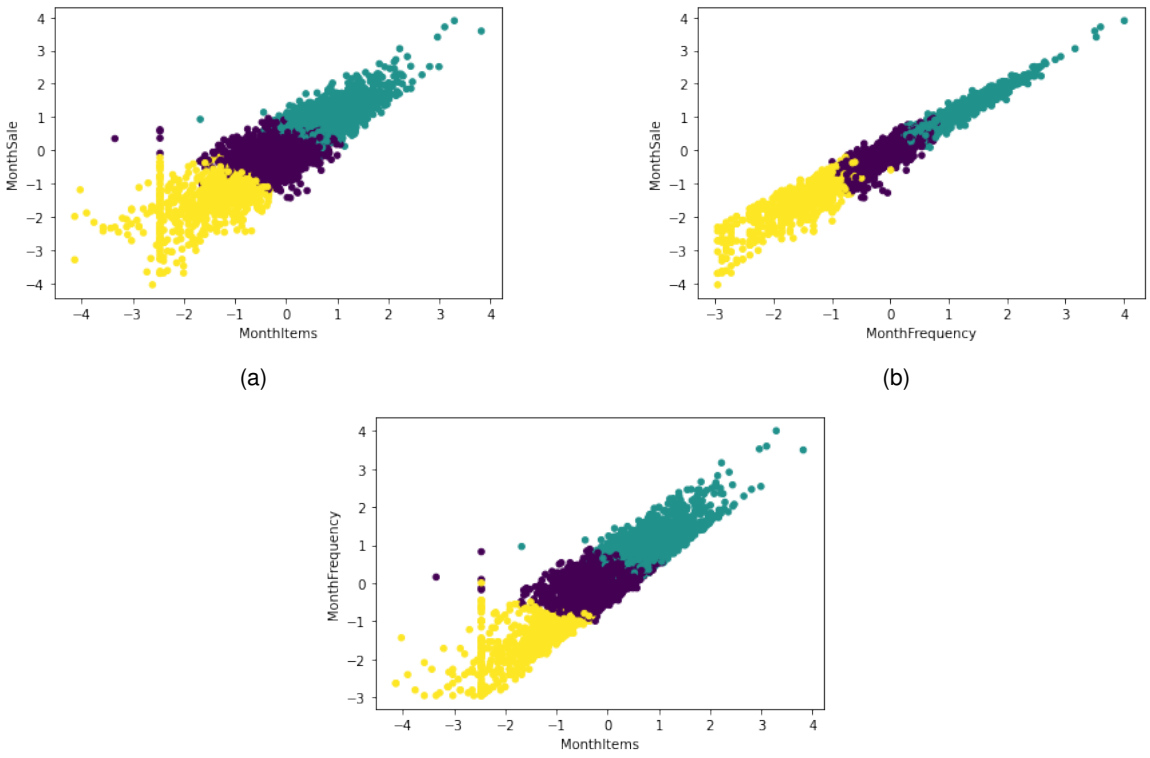


Figure 19: Risultati K-Means

Dagli scatter plot 20 si evince la presenza di tre cluster distinti per colori in cui si ha che nel cluster:



(a) (b)

(c)

Figure 20: K-Means Scatter Plot

- **Giallo**: si hanno gli utenti che acquistano sporadicamente spendono poco e acquistano mediamente meno oggetti;
- **Viola**: si hanno gli utenti che acquistano con una frequenza media, quindi nella norma e che hanno anche una spesa media;
- **Verde**: si hanno gli utenti che acquistano spesso e molti oggetti e che tendono a spendere molto;

Infine il cluster risultante sembra essere molto bilanciato, siccome per il cluster 1 abbiamo il 46% dei dati, per il cluster 2 il 34% mentre per il cluster 3 il 20% dei dati.

2.3 DB-Scan

2.3.1 Scelta degli attributi e della metrica per la distanza

In DB-Scan come per K-Means abbiamo deciso di utilizzare lo stesso set di attributi **MonthSale, MonthFrequency, MonthItems, MonthDistinctItems** mentre la metrica per la distanza utilizzata è stata l'*euclidean distance*.

2.3.2 Scelta dei parametri

Il principale problema con DBSCAN è di trovare i valori corretti principalmente per due parametri:

- **Eps**: il density estimator.
- **MinPts**: fornisce il numero minimo di punti che devono essere a distanza Eps dal punto considerato affinché sia considerato un punto centrale.

Per trovare una buona stima abbiamo usato ancora una volta il *Knee Method*. Dal grafico a sinistra possiamo vedere che tutte le curve hanno valore ottimale vicino a 3700, ed il corrispondente epsilon per questa x è 0,4.

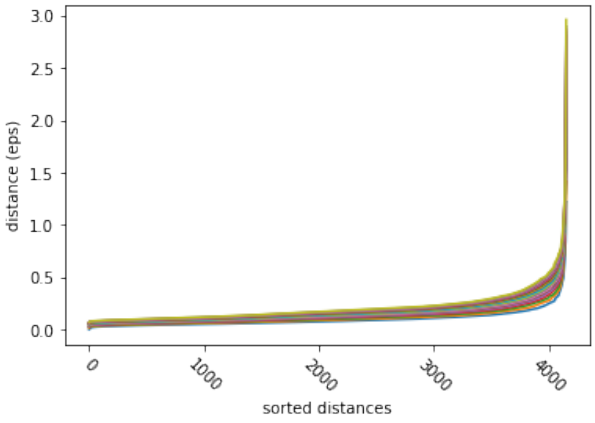


Figure 21: DBScan Distance

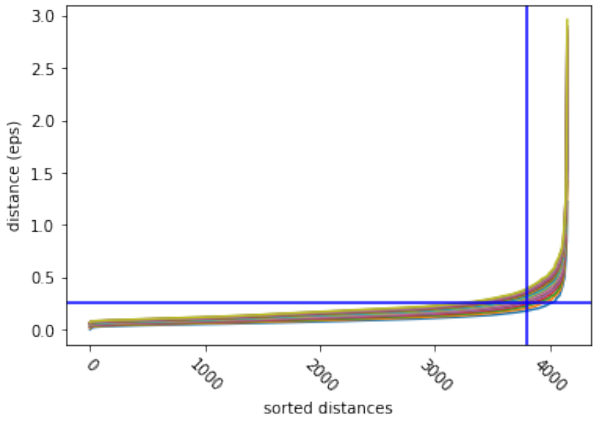


Figure 22: DBScan Distance

Per scegliere il miglior valore di minPts, fissiamo l'epsilon a 0.4 e $minPts \in [2, 20]$, rappresentiamo i valori della silhouette. Decidiamo quindi di fissare il minPts a 14 dato lo studio effettuato dalla figura e attraverso alcuni test pratici con varie combinazioni (fig. 23).

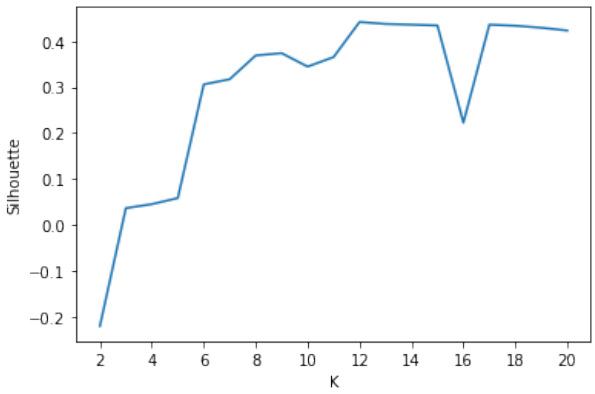


Figure 23: **Silhouette** al variare di k

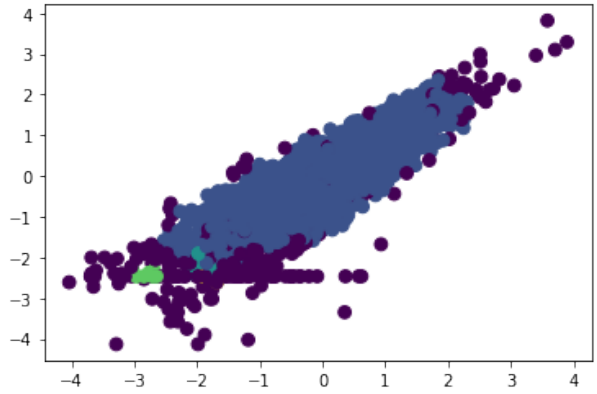


Figure 24: DBScan scatterplot

Il numero stimato di cluster risulta essere 4 e il numero stimato di noise points 261 figura 24.
Nonostante abbiamo provato con diverse combinazioni di parametri, i rusultati sembrano non cambiare.
Il motivo per cui DB-Scan sembra andare così male sul nostro dataset può essere dovuto dal fatto che il nostro dataset è composto da moltissimi punti vicini tra loro (alta densità), pertanto non riesce a trovare alcun cluster, ma solo le zone altamente popolate.

2.4 Hierarchical Clustering

2.4.1 Scelta degli attributi e della metrica per la distanza

Nello Hierarchical Clustering abbiamo deciso di utilizzare come per gli atri 2 algoritmi gli attributi **MonthSale, MonthFrequency, MonthItems, MonthDistinctItems**.
Abbiamo deciso di utilizzare come metriche sia la *euclidean distance* che la *manhattan distance* e per ogni metrica abbiamo utilizzato ward, single, complete e avarage linkages (con l'eccezione dell'utilizzo della manhattan distance con ward linkage siccome non consentito utilizzarla).
Abbiamo calcolato il valore della silhouttes al variare del numero di cluster con $k \in [2, 10]$ e come possiamo vedere dai grafici 25, 26 possiamo notare come le silhouettes iniziano a scendere dopo il terzo cluster, pertanto abbiamo deciso di performare l'analisi con 3 clusters.

Dall'analisi dei risultati abbiamo trovato che gli unici risultati interessanti, in termini di silhouette e bilanciamento del clustering, si ottengono utilizzando:

- euclidean distance e ward linkage;
- manhattan distance e complete linkage;

I relativi dendogrammmi sono in figura 27, 28.

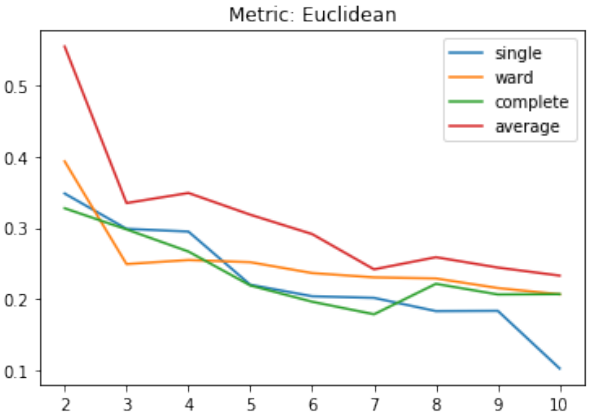


Figure 25

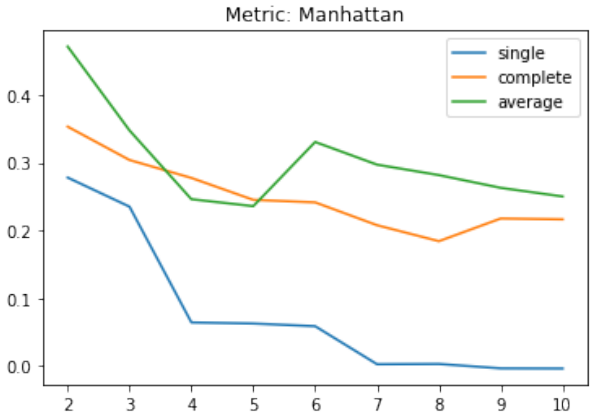


Figure 26

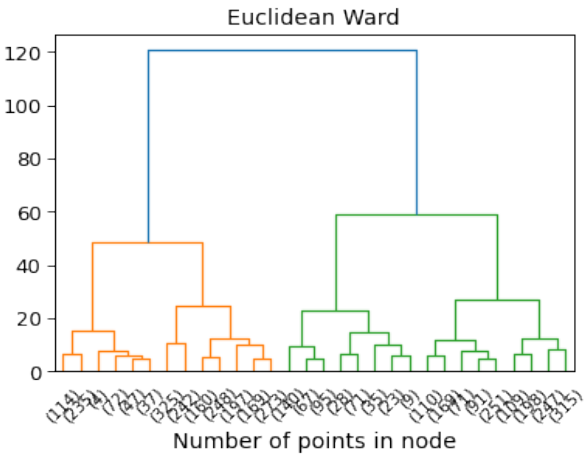


Figure 27

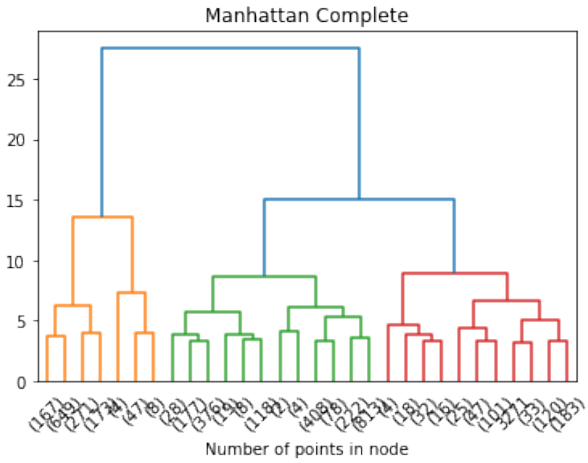


Figure 28

2.5 Confronto dei risultati

Dopo aver utilizzato tutti e tre gli algoritmi di clustering abbiamo notato che l'algoritmo che ha portato ai miglior risultati è stato K-Means. DB-scan ha riportato dei risultati nettamente inferiori rispetto alle altre due tecniche, non riuscendo a creare un buon clustering nonostante il tantativo di variare più volte i valori di *epsilon* e *min-pts*. Hierarchical clustering, invece, ottiene anch'esso dei risultati soddisfacenti quando si utilizza la euclidean distance con ward linkage anche se comunque leggermente sbilanciati, mentre nel caso si utilizzi la manhattan distance con complete linkage si ottengono dei cluster altamente sbilanciati.

2.6 Librerie utilizzate

Per quanto riguarda l'analisi dei cluster abbiamo utilizzato diverse librerie, nelle sezioni precedenti sono riportati i risultati ottenuti attraverso la libreria scikit-learn [Pedregosa et al. (2011)].

Con scikit-learn abbiamo effettuato tutta la parte di sperimentazione, abbiamo poi deciso di utilizzare anche un'altra libreria per mettere a confronto i risultati ottenuti e verificare la presenza di eventuali incogruenze. La libreria in questione è pyclustering [Novikov (2019)], i risultati ottenuti sono pressocché gli stessi (per ulteriori informazioni sui risultati ottenuti da questa libreria, consultare il notebook presente al seguente link github. Come ultima osservazione ci teniamo a precisare che l'utilizzo della libreria pyclustering risulta molto intuitiva (probabilmente anche più di scikit-learn) ma possiede meno funzionalità che rendono la fase di analisi più lunga e contorta, proprio per questo motivo abbiamo deciso di utilizzare questa libreria al fine di validare i risultati ottenuti con scikit-learn.
Inoltre, attraverso la libreria pyclustering abbiamo provato altri due algoritmi di clustering, xmeans e optics. I risultati di xmeans come possiamo vedere in figura 29 sono molto simili ai risultati ottenuti con kmeans.

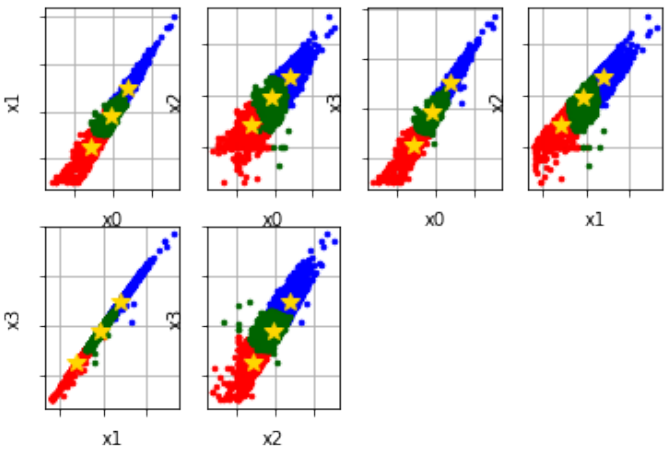


Figure 29: Risultati XMeans

Per optics invece, siccome è un algoritmo molto simile a DB-scan, che risolve la principale debolezza di DB-scan, cioé il problema di trovare clustering significativi in dati con densità variabile [Wikipedia], ci aspettavamo di trovare dei risultati leggermente migliori, mentre come si può vedere in figura 31 i risultati sono pressocché identici a DB-scan, nonostante anche qui abbiamo provato con diversi valori di *radius* e *neighbors*.

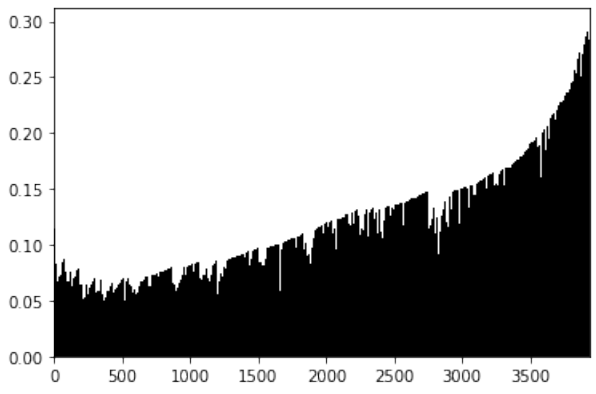


Figure 30: Optics Dendogrammi

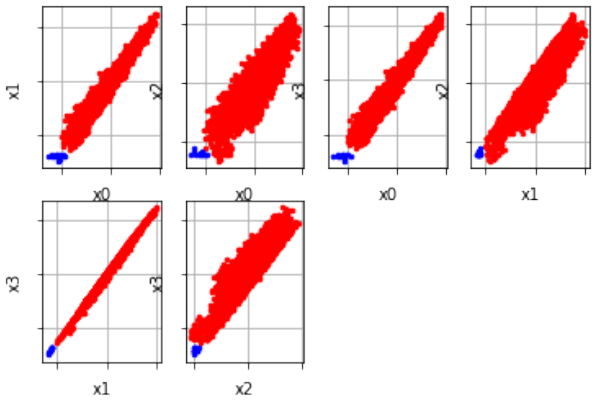


Figure 31: Optics Scatter Plot

3 Classificazione

In questa sezione ci siamo occupati di confrontare diversi classificatori al fine di predirre il comportamento di un utente in tre classi *high-spending customer*, *medium-spending customer* o *low-spending customer*.

3.1 Creazione label obiettivo

Per definire un profilo utente che consentisse la classificazione degli utenti in base alla sua attitudine alla spesa si è deciso di utilizzare l'attributo *MonthSale*, analizzare la sua distribuzione e in base a quest'ultima identificare 2 threshold in modo da definire un utente di tipo *high-spending customer*, *medium-spending customer* o *low-spending customer*. La figura 32 mostra la distribuzione dell'attributo, per decidere a quale categoria l' utente facesse parte abbiamo deciso di utilizzare il t-percentile su questa distribuzione, e abbiamo deciso di assegnare agli utenti presenti nel primo 35-percentile la label *low-spending*, tra il 35-percentile e il 75-percentile *medium-spending* e il resto *high-spending*. La distribuzione dei customer risulta abbastanza bilanciata come possiamo vedere in figura 33. Da notare che nonostante il dataset sia piuttosto bilanciato, abbiamo deciso di utilizzare anche le tecniche di oversampling e undersampling sui classificatori per constatare se bilanciando il dataset i clasificatori avessero dei cali o un aumento delle performance.

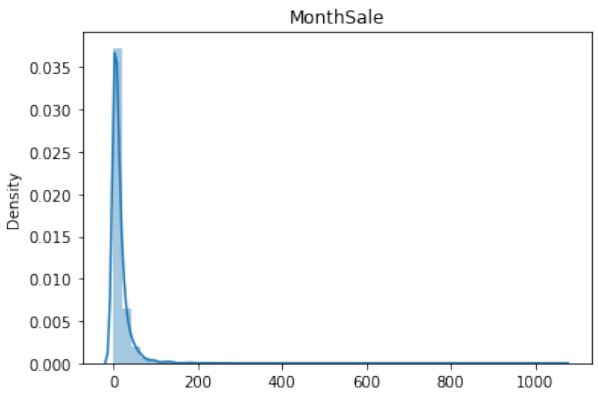


Figure 32: Distribuzione Sale per Mese

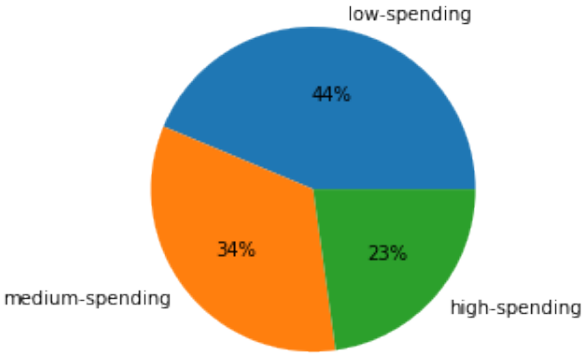


Figure 33: Distribuzione dei label

3.2 Preprocessing del dataset

Ai fini dell'ottenimento di una migliore classificazione, per far si che questa non fosse eccessivamente legata a feature altamente correlate abbiamo deciso di rimuovere le feature: *MonthSale* poiché quest'ultima è stata utilizzata per definire le label e inoltre come detto anche tutte le feature molto correlate con quest'ultima come *TotalSale* *WeekSale*. Come già presentato nella sezione 1.4 il dataset non contiene missing value ma presenta alcuni outlier che abbiamo comunque preferito lasciare per non ridurre di troppo la dimensione del nostro dataset. Abbiamo, inoltre, discretizzato le variabili categoriche presenti *CustomerCountry* e *CustomerLabel*. Il dataset è stato suddiviso in train 70% dei dati e test 30% dei dati. Abbiamo effettuato una gridsearch su tutti i classificatori utilizzando 5 Cross-validation e effettuato una media dei risultati per evitare di overfittare/underfittare.

3.3 Decision Tree

Per effettuare la classificazione degli utenti abbiamo utilizzato il dataset costruito nella sezione 3.2 e per ottenere il miglior modello abbiamo deciso di effettuare una grid search con i parametri in tabella 4:

Parameter	Value
criterion	gini, entropy
max_depth	[2, 5, 10, 15, None]
min_samples_split	[2, 5, 10, 20]
min_samples_leaf	[1, 5, 10, 20]

Table 4: Decision Tree - Grid Search

I migliori risultati si ottengono con un decision tree addestrato con i seguenti parametri: 'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 20, 'min_samples_split': 5. Nella figura 36 è presentata la matrice di confusione ottenuta e nella figura 37 è presentata la ROC curve.

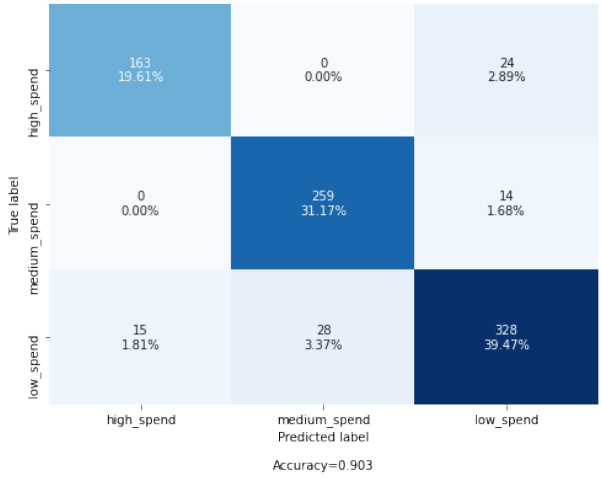


Figure 34: Decision Tree Confusion Matrix

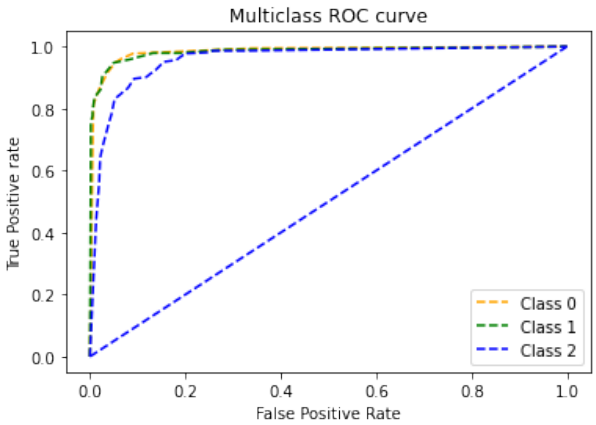


Figure 35: Decision Tree ROC Curve

Come possiamo notare il classificatore ottiene un ottimo score nella predizione di tutte le classi.

3.4 Random Forest

Abbiamo utilizzato il dataset costruito nella sezione 3.2 per effettuare una grid search con i parametri contenuti all'interno della tabella 5.

Parameter	Value
n_estimators	[25, 50, 100, 200, 500, 1000]
criterion	gini, entropy
max_depth	[2,3,5,6,7,10,12,None]
bootstrap	[True, False]

Table 5: Random Forest - Grid Search

I migliori risultati si ottengono con un Random Forest addestrato con i seguenti parametri: 'bootstrap': True, 'criterion': 'entropy', 'max_depth': None, 'n_estimators': 50. Nella figura 36 è presentata la matrice di confusione e nella figura 37 è presenta la ROC curve.

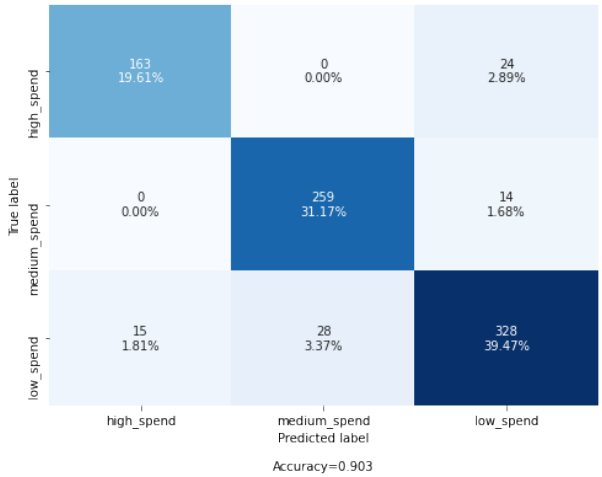


Figure 36: Random Forest Confusion Matrix

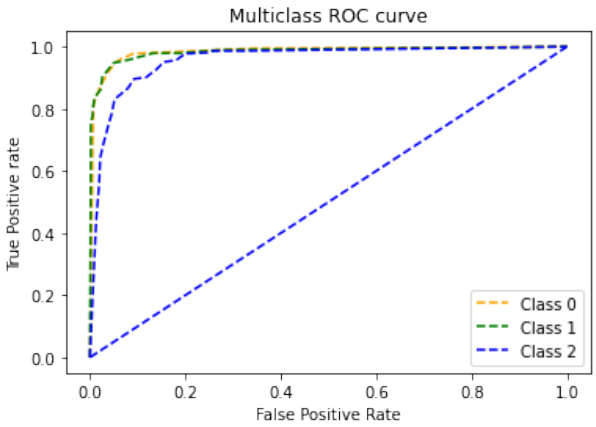


Figure 37: Random Forest ROC Curve

3.5 SVM

Per effettuare la classificazione anche qui abbiamo utilizzato il dataset costruito nella sezione 3.2 per poi effettuare una grid search con i parametri contenuti all'interno della tabella 6.

Parameter	Value
C	[0.1, 1, 10, 100, 1000]
gamma	[1,0.5, 0.1, 0.01, 0.001]
kernel	['rbf', 'sigmoid']

Table 6: SVM - Grid Search

I migliori risultati si ottengono con un SVM addestrato con i seguenti parametri: {'C': 1, 'gamma': 0.001, 'kernel': 'rbf' }. Nella figura 38 è presentata la matrice di confusione e nella figura 39 è presenta la ROC curve.

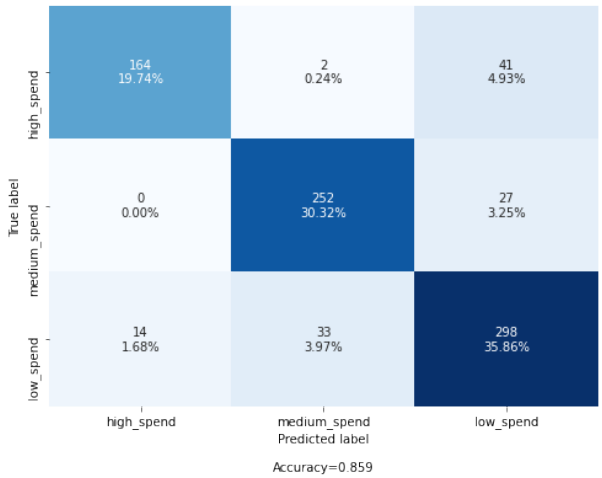


Figure 38: SVM Confusion Matrix

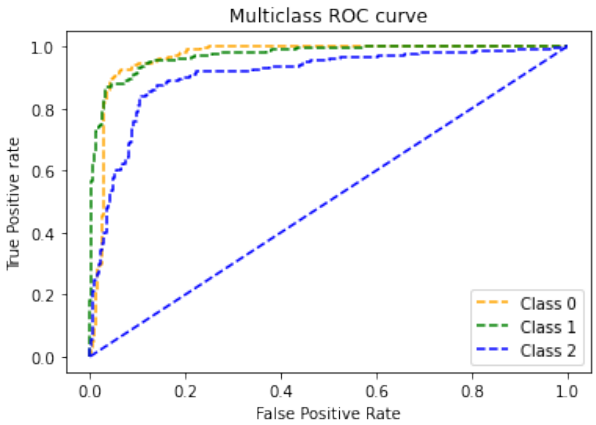


Figure 39: SVM ROC Curve

3.6 Neural Network

Per le reti neurali abbiamo utilizzato una libreria implementata da noi [WaveGrad] che permette di costruire reti deep, in maniera perfettamente compatibile con la libreria scikit-learn.
Abbiamo quindi utilizzato il dataset costruito nella sezione 3.2, ma ci siamo accorti che attraverso l'utilizzo delle reti neurali quest'ultimo non produceva dei risultati soddisfacenti. Per ovviare al problema abbiamo deciso di normalizzare e scalare il dataset, aumentando così notevolmente le performance. L' articolo [University] presenta una spiegazione adeguata sul perché l'utilizzo di queste tecniche aumenta le performance delle reti neurali.
Per testare la rete neurale abbiamo utilizzato i seguenti parametri 7.

Parameter	**Value**
HiddenLayer	[5, 10, 20]
L2 regularization	[0.1, 0.01, 0.001, 0.0001]
Lerning rate	[0.1, 0.01, 0.001, 0.0001]
Optimizer	SGD with momentum

Table 7: Neural Network - Grid Search

I migliori risultati si ottengono con una NN addestrata con i seguenti parametri:'hidden layer' : 10, 'L2' : 0.001, 'learning rage' : 0.001, 'optimizer' : SGD. Le figure 41 e 40 mostrano la loss e l'accuracy della rete neurale per il train e validation set, come si può notare quest'ultima ottiene degli ottimi risultati e non ha problemi di overfitting.

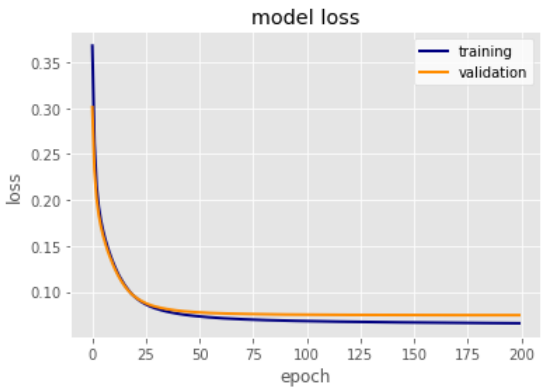


Figure 40: Neural Network Loss

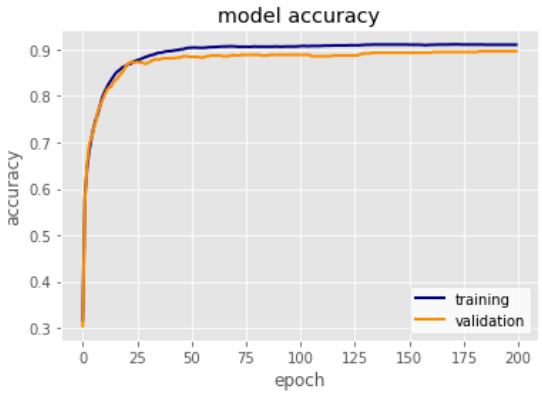


Figure 41: Neural Network Accuracy

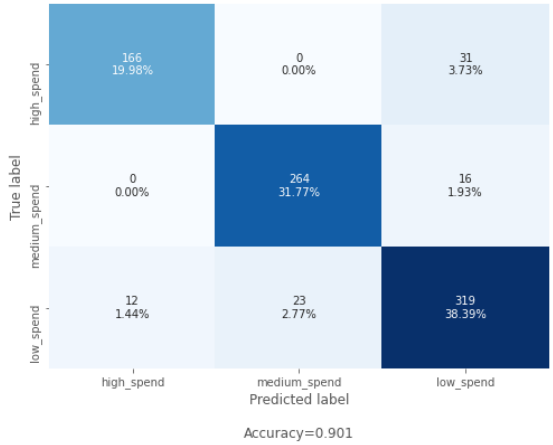


Figure 42: Neural Network Confusion Matrix

Infine, nella figura 42 vediamo come si comporta per ogni classe attraverso la confusion matrix.

3.7 Risultati Finali

Una volta provati tutti gli algoritmi ed aver trovato i miglior parametri da utilizzare, attraverso il training e la validation. Li abbiamo testati sul dataset di test, precedentemente costruito, ottenendo i risultati riportati in tabella 8 l' approccio evidenziato risulta essere il migliore per le metriche utilizzate, ottenendo un ottimo bilanciamento della predizione su tutte le classi e un tempo non alto di training. Inoltre, i classificatori per il quale è stato effettuato l'*Undersampling* sono seguiti dalla lettera U mentre dalla lettera O per l'*Oversampling*.

	Accuracy	Precision				Recall				F1-Score			
		0	1	2	AVG	0	1	2	AVG	0	1	2	AVG
SVM	**85.92%**	77%	89%	83%	**83%**	87%	86%	79%	**84%**	81%	87%	81%	**83%**
Random Forest	**90.85%**	91%	91%	88%	**90%**	89%	92%	89%	**90%**	90%	92%	89%	**90%**
Decision Tree	**90.13%**	92%	92%	87%	**90%**	87%	90%	90%	**89%**	90%	91%	88%	**90%**
Neural Network	**89.53%**	85%	94%	90%	**89%**	93%	92%	87%	**91%**	89%	93%	88%	**90%**
Decision Tree O	**87.36%**	91%	90%	83%	**88%**	89%	88%	86%	**88%**	90%	89%	85%	**88%**
Decision Tree U	**89.53%**	86%	95%	88%	**90%**	95%	91%	86%	**90%**	90%	93%	87%	**90%**
Random Forest O	**90.01%**	90%	93%	89%	**90%**	92%	94%	87%	**91%**	91%	93%	88%	**90%**
Random Forest U	**90.13%**	90%	95%	90%	**92%**	94%	93%	89%	**92%**	92%	94%	90%	**92%**
KNN	**82.67%**	89%	86%	79%	**85%**	84%	87%	81%	**84%**	87%	86%	80%	**84%**
Naive Bayes	**86.64%**	94%	87%	83%	**88%**	82%	92%	85%	**86%**	87%	89%	84%	**87%**

Table 8: Classificazione - Comparativa Risultati Finali

4 Sequential Pattern Mining

Negli algoritmi di Sequential Pattern Mining (SPM) quali AprioriAll, PrefixSpan, CM-SPADE e GSP si prende in input un database di sequenze. Una sequenza è una lista ordinata di transazioni. Lo scopo è quello di trovare una sottosequenza che appaia "spesso" nel set di sequenze, tenendo in considerazione l'ordine sequenziale tra esse. Si decide, inoltre, di selezionare i pattern che abbiano un support, ossia un iperparametro che tiene conto di quante volte il pattern è frequente, sopra un certo valore. Nella nostra analisi abbiamo deciso di utilizzare sia GSP che è basato sul principio *Apriori Principle* che PrefixSpan che esplora la *prefix-projection* riducendo l'effort nella generazione delle sottosequenze (*Han*).

4.1 Dataset Preprocessing

Come prima cosa abbiamo effettuato il preprocessing del dataset per poter utilizzare gli algoritimi di Sequential Pattern Mining, in quanto non utilizzabili direttamente su un dataset di tipo tabulare ma necessitanti di un array di liste.
Il primo passo è stato quello di eliminare, come anche nella fase di data understanding, tutti quei dati che non risultavano utili all'analisi, quindi tutti i valori nulli. Questa volta abbiamo deciso di mantenere gli outlier poiché per questo tipo di analisi possono risultare interessanti e abbiamo rimosso tutti i carrelli contenenti le cancellazioni degli ordini (BasketID inizianti per lettera C). Abbiamo utilizzato solo gli utenti con almeno 5 carrelli durante tutto il periodo di osservazione, per due motivi principali. Il primo è che gli utenti con meno di 5 carrelli avrebbero riportato pattern poco rilevanti nell'ambito di un periodo di osservazione così lungo, il secondo motivo è relativo alle performance dei due algoritmi, che essendo computazionalmente molto dispendiosi lavorano molto più agevolmente con dataset non troppo grandi.
Infine, abbiamo modellato gli utenti come sequenze di basket e poi sostituito il BasketID con i relativi oggetti comprati.

4.2 Algoritmi Utilizzati

Come accennato precedentemente abbiamo deciso di utilizzare due diversi tipi di algoritmi e librerie per questo tipo di task, il primo è Generalized Sequential Pattern algorithm (GSP) che si basa sul *Apriori Principle*, il secondo è PrefixSpan [Gao].

Abbiamo dapprima utilizzato GPS e dopo varie prove abbiamo deciso che il min_support migliore si aggirasse tra 40 e 50, infatti variando il support il numero di pattern trovati cambia notevolmente, dato che con un support basso si ottengono più regole ma al prezzo di un elevato tempo computazionale dell'algoritmo. La distribuzione dei pattern è rappresentata in *scala logaritmica* nella figura 43.

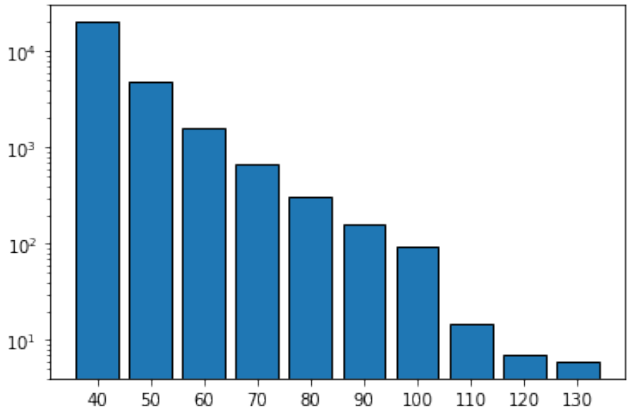


Figure 43: Plot della distribuzione dei pattern in scala logaritmica

Dopodiché abbiamo utilizzato PrefixSpan e abbiamo notato che il tempo computazionale di quest'ultimo era di molto inferiore a quello richiesto da GSP. Abbiamo deciso di riportare i risultati di PrefixSpan in quanto ci ha permesso di sperimentare con diversi support. In **tabella 9** abbiamo riportato il pattern che si ripete piú volte, il pattern con lunghezza maggiore in assoluto e con support più alto e i pattern con elementi distinti e support più alto.

Tipologia	Supporto	Pattern
oggetto periodico	59	['WHITE HANGING HEART T-LIGHT HOLDER', 'WHITE HANGING HEART T-LIGHT HOLDER', 'WHITE HANGING HEART T-LIGHT HOLDER', 'WHITE HANGING HEART T-LIGHT HOLDER', 'WHITE HANGING HEART T-LIGHT HOLDER', 'WHITE HANGING HEART T-LIGHT HOLDER']
Lunghezza Maggiore	43	['JUMBO BAG RED RETROSPOT', 'JUMBO BAG PINK POLKADOT', 'JUMBO BAG RED RETROSPOT', 'JUMBO BAG PINK POLKADOT', 'JUMBO BAG RED RETROSPOT', 'JUMBO BAG PINK POLKADOT', 'JUMBO BAG RED RETROSPOT']
Distinti 2	148	['JUMBO BAG RED RETROSPOT', 'LUNCH BAG RED RETROSPOT']
Distinti 3	106	['LUNCH BAG RED RETROSPOT', 'LUNCH BAG PINK POLKADOT', 'LUNCH BAG BLACK SKULL.']
Distinti 4	75	['LUNCH BAG RED RETROSPOT', 'LUNCH BAG CARS BLUE', 'LUNCH BAG PINK POLKADOT', 'LUNCH BAG BLACK SKULL.']
Distinti 5	57	['LUNCH BAG RED RETROSPOT', 'LUNCH BAG CARS BLUE', 'LUNCH BAG PINK POLKADOT', 'LUNCH BAG SUKI DESIGN ', 'LUNCH BAG BLACK SKULL.']
Distinti 6	41	['LUNCH BAG RED RETROSPOT', 'LUNCH BAG WOODLAND', 'LUNCH BAG PINK POLKADOT', 'LUNCH BAG CARS BLUE', 'LUNCH BAG BLACK SKULL.', 'LUNCH BAG APPLE DESIGN']

Table 9: SPM - Regole Selezionate

Il primo pattern mostra come l'articolo *WHITE HANGING HEART T-LIGHT HOLDER* viene comprato ripetutamente in diverse occasioni da più utenti, questo potrebbe significare che l'articolo seguente viene comprato periodicamente, tale analisi potrebbe essere approfondita attraverso l'introduzione degli attributi temporali nell'algoritmo. Con il secondo pattern notiamo che l'articolo *JUMBO BAG RED RETROSPOT* viene comprato spesso, mostrandoci che quest'ultimo viene comprato a intervalli più o meno regolari seguito da altri tipi di borse di diverso colore, questa analisi mostra come in questo caso sarebbe meglio tenere gli articoli maschili insieme a quelli femminili invece che dividerli, siccome le due borse sembrano essere di categoria maschile e femminile. Le ultime righe invece presentano pattern con oggetti diversi da quali si può notare come vengano comprati oggetti che hanno caratteristiche in comune tra loro o che corrisponono ad un oggetto con stessa funzionalità ma con caratteristiche fisiche differenti come nel caso della riga *distinti 2*.

5 Conclusioni

In conclusione per l'analisi di questo dataset di customer supermarket abbiamo utilizzato diverse tecniche di data mining, provando a capire il comportamento degli utenti.
Nella sezione di **Data Understanding**, abbiamo affrontato il problema relativo alla semantica dei dati e alla loro qualità, ad esempio su come rimuovere dati ridondanti e sostituirne i valori mancanti. Tale analisi è stata fatta per comprendere quanto più possibile le variabili presenti e preparare il dataset per le analisi successive.
Nella sezione **Clustering** abbiamo cercato gruppi (cluster) di customer in modo da capire meglio i dati che avevamo a disposizione, il problema principale è scaturito dalla grande densità di punti che non hanno portato a dei risultati ottimali con alcuni algoritmi, nonostante ciò siamo riusciti ad ottenere dei cluster ben bilanciati e significativi attraverso l'utilizzo di K-Means.
Nel task di **Classificazione** abbiamo provato a predirre il comportamento di un untente(*Basso consumatore, Medio consumatore, Alto consumatore*), per fare ciò abbiamo utilizzato diversi tipi di classificatori con diverse configurazioni e ne abbiamo utilizzato la migliore, selezionata tramite una cross-validation con 5-fold.
Infine, nella sezione di **Sequential Pattern Mining** abbiamo utilizzato GSP e PrefixSpan testando diverse configurazioni di quest'ultimi, al fine di scoprire pattern interessanti e ricorrenti tra gli utenti.

6 Riferimenti

Gao, C. *prefixspan*. Available from: `https://github.com/chuanconggao/PrefixSpan-py`.

Han, J. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth.

Novikov, A., 2019. PyClustering: Data mining library. *Journal of open source software*, 4(36), p.1230. Available from: `https://doi.org/10.21105/joss.01230`.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12, pp.2825–2830.

University, S. *Cs*. Available from: `https://cs231n.github.io/neural-networks-2/#batchnorm`.

WaveGrad. *Wavegrad*. Available from: `https://github.com/vlnraf/WaveGrad`.

Wikipedia. *Optics*. Available from: `https://en.wikipedia.org/wiki/OPTICS_algorithm`.