DATA MINING 2 REPORT

IBM STOCKS AND ABALONES

ANNO ACCADEMICO 2017/2018

A CURA DI:

MARTINA CINQUINI
VIRGINIA MORINI
SAAD SBAI

Il Dataset IBM stocks

IBM (International Business Machines Corporation) è un'azienda statunitense appartenente al settore informatico (software, hardware e servizi). Costituita nel 1911, quotata in borsa nel 1959, attualmente vanta un fatturato di 80 miliardi e 12 miliardi di utile netto.

Il dataset *IBM stocks* fornisce i dati finanziari di IBM dal 1962 all'aprile 2018 ed è composto da 14168 istanze descritte da 7 colonne di attributi:

- **Date** nel formato mese/giorno/anno
- **Open:** asta(prezzo) di apertura
- **High:** valore più alto di ogni singolo giorno
- **Low:** valore più basso di ogni singolo giorno
- **Close:** asta di chiusura
- **Adj close:** (adjusted closing price) prezzo di chiusura di un'azione modificato per includere eventuali distribuzioni e operazioni societarie prima dell'apertura del giorno successivo.
- **Volume:** volume totale di ogni singolo giorno

L' analisi seguente si riferisce solamente al prezzo di apertura (*open*).

In un primo momento si è osservato che, nel dataset, per ogni anno sono presenti 250 istanze e non 365, questo a causa del fatto che il sabato e la domenica le borse sono chiuse.

Sulla pagina web dell'azienda[1] sono presenti una serie di paragrafi riguardanti gli eventi salienti anno per anno. Questa fonte verrà utilizzata per svolgere un'analisi non solo prettamente numerica, in modo da ottenere delle interpretazioni ragionevoli.

Nella seguente figura (Figura 1) è mostrato l'andamento dei prezzi di apertura dal 1962 al 2018, prima di effettuare qualsiasi operazione di preprocessing.

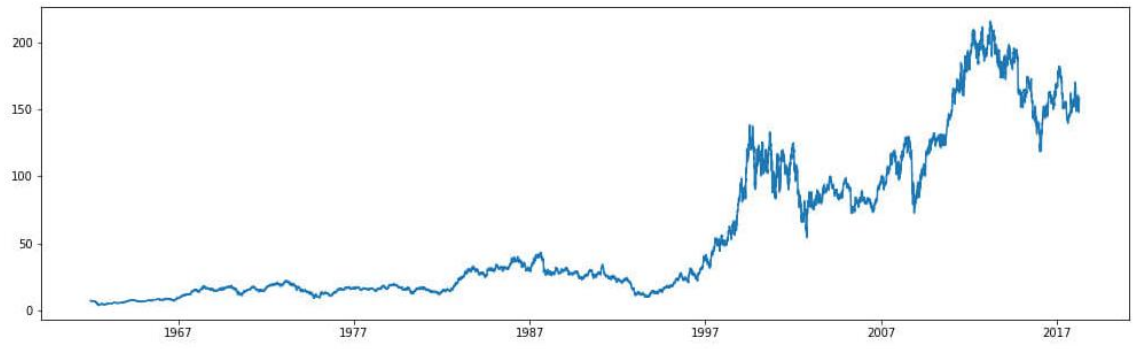

Figura 1. Andamento Time series

[1] https://www-03.ibm.com/ibm/history/

Time Series Preprocessing

Nella fase di preprocessing, come illustrato dalla figura 2, la time series originale è stata suddivisa in anni, ottenendo 56 time series, una per ogni anno. Si sono sostituiti i valori nulli delle time series incomplete con il valore -3 (il valore minimo post-normalizzazione).

Inoltre, abbiamo usato una funzione di *smooth* con una finestra di 10 per eliminare il noise presente nella serie.

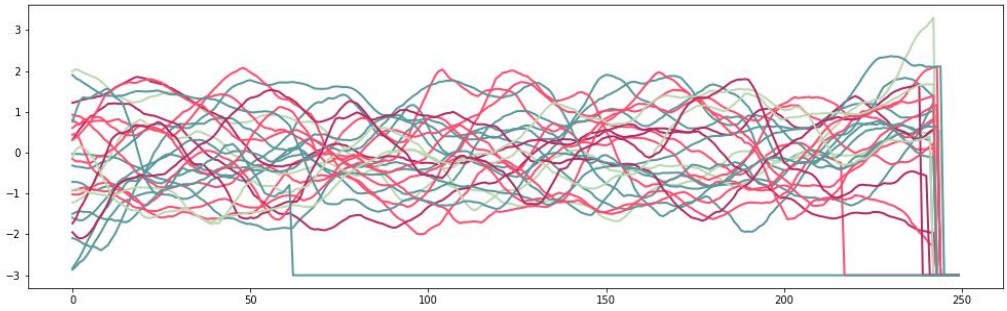

Figura 2. Time series suddivise temporalmente

Time Series Clustering

In questa fase si è scelto di utilizzare 3 algoritmi di clustering: *Dbscan, K-means* e *Hierarchical*. Di seguito, vengono indicate le considerazioni effettuate e i relativi risultati.

Dbscan

Si è testato l'algoritmo con diversi valori di *eps*, ma nonostante ciò non risulta essere particolarmente efficace, infatti, DBSCAN tende ad escludere quasi la metà delle time series classificandole come noise.

Il risultato migliore ottenuto è con *eps=8.8* con cui si sono individuati 3 cluster con una struttura omogenea rappresentanti pattern molto semplici e con *silhouette=0.105*.

Di seguito, si è scelto di mostrare soltanto i cluster più interessanti.

1. **Cluster 1:**
    - Time series include: 1963,1965,1967,1969,1982,1983,
      1994,1995,1997,1998,2006,2009,
      2010,2011,2016
    - Andamento sempre crescente con picchi nella seconda metà dell'anno.

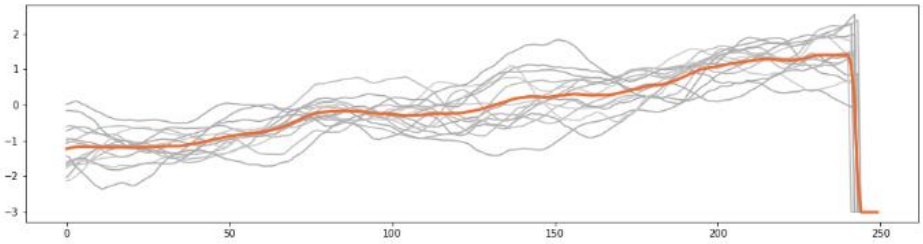

Figura 3. Cluster 1 DBSCAN

2. **Cluster 2:**
    - TimeSeries incluse: 1962,1973,1974,1979,1981,1986, 1991,2002,2013,2017
    - Andamento mediamente decrescente, ovvero con picchi che tendono ad essere concentrati nei primi mesi dell'anno.

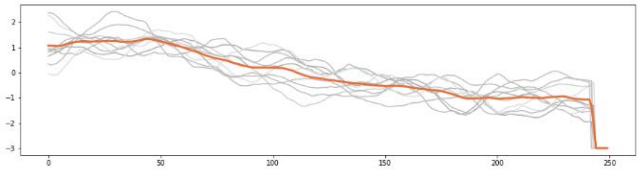

Figura 4. Cluster 2 DBSCAN

3. **Cluster 3:**
    - Times series incluse: 1964,1987,1992,2008,2014,2015
    - andamento "concavo" con un picco massimo nella metà dell'anno.

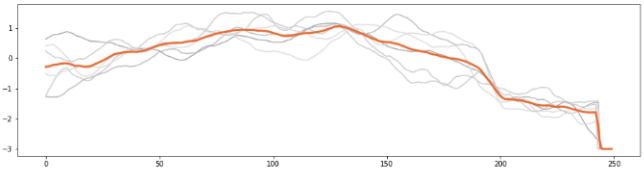

Figura 5. Cluster 3 DBSCAN

Hierchichal Clustering

Con il clustering gerarchico si sono ottenuti risultati migliori rispetto al DBSCAN: i cluster sono più omogenei. Tra questi si è scelto di considerare solamente quelli con composti da molti elementi data la conformazione bilanciata del dataset. Si è scelto di utilizzare il *Dynamic Time Warping* per la distanza dato che altri metodi testati hanno condotto a risultati simili e pertanto non verranno riportati. Inoltre, come metodo di merge, si è utilizzo l'approccio di linkage completo.

Di seguito, vengono riportate le considerazioni sui cluster individuati:

1. **Cluster 0:** raccoglie il periodo con un andamento crescente ripido.
   Sono gli anni in cui si è verificata una maggiore crescita. Ad esempio, considerando l'anno 1963, durante il quale il valore iniziale normalizzato è di circa -1.76 a fine anno raggiunge 2.42. Questo incremento potrebbe essere dovuto al lancio da parte di IMB del 7094 II, ovvero il computer più potente dell'epoca con un sistema di archiviazione elettronica composto da nuovi file di archiviazione su disco, che elabora le informazioni quasi al doppio della velocità del suo predecessore il 1401.

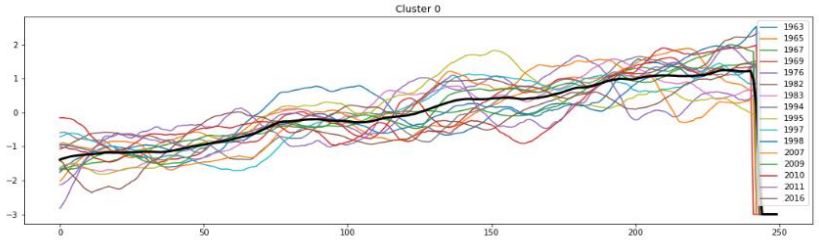

Figura 6. Cluster 0 complete linkage

2. **Cluster 1:** periodo caratterizzato da un andamento decrescente. Il cluster è composto da pochi anni e non è particolarmente omogeneo come si evince dall'andamento del 1977 e del 1992.

Figura 7. Cluster 1 complete linkage

3. **Cluster 2:** costituito da timeseries che hanno il valore minimo del prezzo intorno al 150esimo giorno dell'anno, con valore alla chiusura dell'anno piuttosto alta, come nel caso del 1993. Questo andamento potrebbe essere dovuto alla chiusura di un contratto da 650 milioni di dollari, in cui IBM fornì a Equifax Inc. la gestione dei data center e altri servizi per 10 anni. Inoltre, ci fu un altro accordo tecnologico per 10 anni del valore di 415 milioni di dollari con Southern Pacific Lines per fornire alla ferrovia un nuovo programma per la gestione della sua funzione informatica e una nuova organizzazione per gestire il processo. È necessario specificare che, l'anno 2018 è presente in questo cluster, ma sono presenti solamente i dati relativi al periodo seguente: dal 1/01 al 14/04.

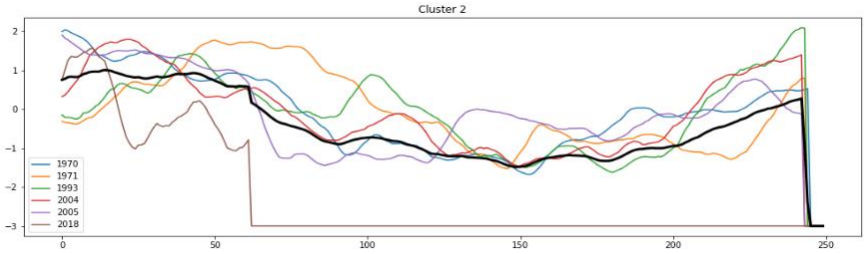

Figura 8. Cluster 2 complete linkage

4. **Cluster 3:** questo cluster non è particolarmente significativo. Se si confronta con il cluster 1, è possibile notare che nonostante vi sia lo stesso andamento la deviazione standard delle time series è molto più elevata rispetto alla media del primo.

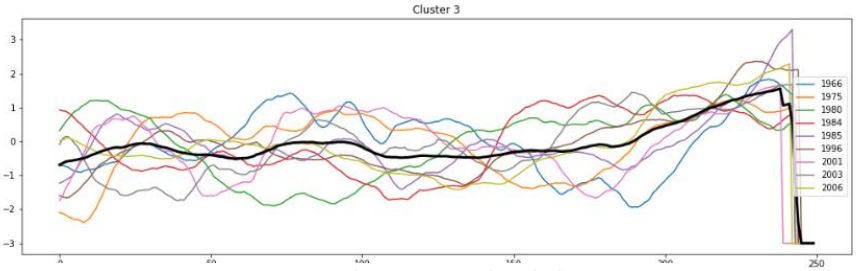

Figura 9. Cluster 3 complete linkage

5. **Cluster 4:** cluster costituito solamente da 6 anni ma molto omogeneo infatti tutte le time series hanno un doppio picco, uno nell'intorno del 50esimo giorno ed uno nel 200esimo.

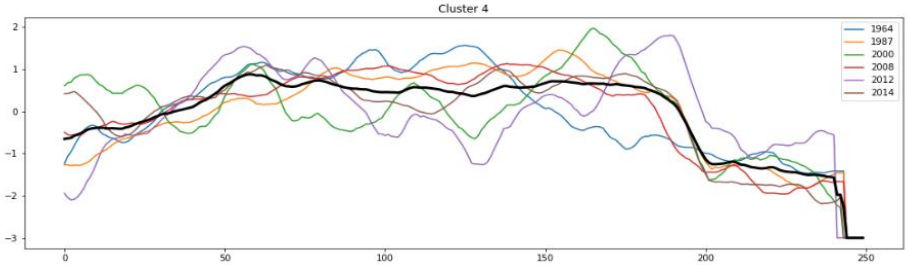

Figura 10. Cluster 4 complete linkage

6. **Cluster 5:** andamento decrescente meno ripido rispetto al primo. Da notare l'andamento dell'anno 1981: all'inizio anno sono presenti valori alti (i più alti del cluster) e poi vi è una discesa repentina. Questo fenomeno è molto probabile che sia stata dovuto al cambio della guida dell'azienda da Frank T. Cary ad John Opel.

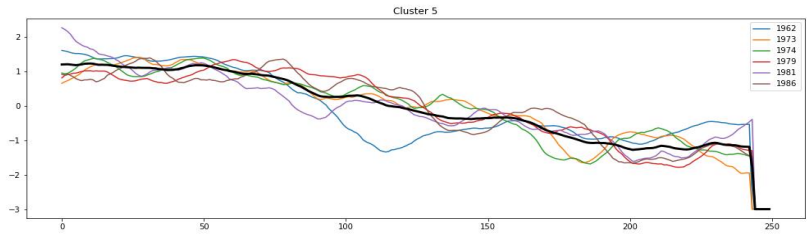

Figura 11. Cluster 5 complete linkage

7. **Cluster 6:** costituito da time series con picco massimo a metà anno. Analizzando il cluster è necessario tenere in considerazione che il 1988 è l'anno in cui La U.S. Federal Aviation Administration assegna a IBM un contratto da 3,55 miliardi di dollari per lo sviluppo, l'installazione e la manutenzione del sistema di automazione avanzata. AAS è una parte importante del piano National Airspace Systems della FAA, una strategia completa per modernizzare il sistema di controllo del traffico aereo statunitense. Inoltre, la U.S. Postal Service assegna a IBM un contratto di quasi 200 milioni di dollari per fornire fino a 900 IBM 9370 e 4381 processori per gestire una varietà di attività amministrative.

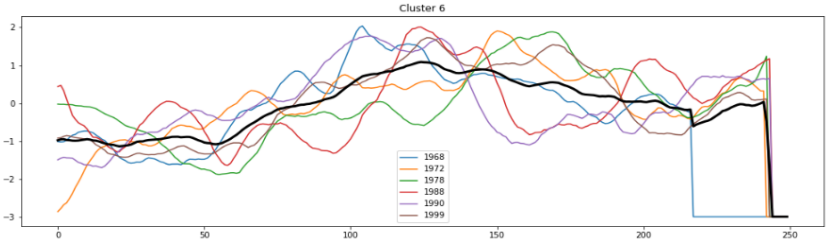

Figura 12. Cluster 6 complete linkage

Sequential Pattern Mining

In questo capitolo, si descrive la ricerca di motifs frequenti all'interno del dataset. L'obiettivo è trovare i pattern costituiti da sequenze contigue e da almeno quattro elementi, come richiesto nelle specifiche del progetto.

L'algortimo utilizzato è il GSP (*Generalized Sequential Pattern*). Si è scelto di dividere la serie temporale totale in frammenti mensili, normalizzando i valori come nella fase di pre processing effettuata durante il clustering. Successivamente, si è applicata la discretizzazione in modo da suddividerle in intervalli di uguale ampiezza associati ad una lettera dell'alfabeto latino.

Ad ogni tentativo sono stati esclusi i frequent pattern che non soddisfacevano il vincolo *maxgap=1*, mentre per quelli che lo soddisfacevano è stato ricalcolato il valore del supporto. Utilizzando il GSP non abbiamo trovato risultati significativi. La motivazione può essere ricondotta a due aspetti: innanzitutto per ottenere motif ad elevato supporto è necessaria una discretizzazione con pochi intervalli che però porta ad avere risultati troppo grezzi, mentre utilizzando una discretizzazione con un numero di intervalli maggiore, si ottengono valori di supporto inferiori al 10%.
In base a queste considerazioni, vengono mostrati due casi, a nostro parere significativi, che sono stati esaminati.

Nel caso di un numero di bins pari a 6, l'ampiezza degli intervalli risulta di 1 (dato che sono normalizzati prendono valori da -3 a +3) e si sono ottenuti motif con un supporto alto. Nonostante ciò, le successioni erano intervalli costanti.

I risultati ottenuti sono i seguenti:

[['c'], ['c'], ['c'], ['c']] sup: 469
[['d'], ['d'], ['d'], ['d']] sup: 450
[['c'], ['d'], ['d'], ['d']] sup: 407
[['d'], ['d'], ['d'], ['c']] sup: 386
[['c'], ['c'], ['d'], ['d']] sup: 377
[['c'], ['c'], ['c'], ['d']] sup:354
[['b'], ['c'], ['c'], ['c']] sup: 354
[['b'], ['b'], ['c'], ['c']] sup: 354
[['b'], ['b'], ['b'], ['c']] sup: 353
[['c'], ['c'], ['b'], ['b']] sup: 352
[['d'], ['c'], ['c'], ['c']] sup: 346
[['d'], ['d'], ['d'], ['d'], ['d']] sup: 344
[['c'], ['c'], ['c'], ['b']] sup: 343
[['c'], ['c'], ['c'], ['c'], ['c']] sup: 331
[['b'], ['b'], ['b'], ['b']] sup: 308

Nel secondo caso abbiamo deciso di usare un numero di bins pari a 16. Utilizzando questo valore gli intervalli ottenuti sono di ampiezza 0.38 e sono stati ottenuti i seguenti risultati, nuovamente ordinati secondo il valore del supporto.

[['j'], ['k'], ['k'], ['k']] sup: 56

[['g'], ['g'], ['f'], ['f']] sup: 53

[['k'], ['k'], ['l'], ['l']] sup: 52

[['g'], ['g'], ['g'], ['g']] sup: 51

[['g'], ['g'], ['g'], ['f']] sup: 49

[['f'], ['f'], ['f'], ['f']] sup: 48

[['f'], ['g'], ['g'], ['g']] sup: 47

[['g'], ['f'], ['f'], ['f']] sup: 47

[['k'], ['k'], ['j'], ['j']] sup: 45

[['f'], ['f'], ['f'], ['g']] sup: 44

Si può notare come la discretizzazione più fitta abbia diminuito drasticamente i valori dei supporti, infatti il motif più frequente si verifica soltanto nell'8% delle 676 Time Series, ottenendo ad ogni modo pattern con valori piuttosto costanti.

Classificazione

L'abalone, nome scientifico Haliotis, è un genere di molluschi gasteropodi marini.

Determinare l'età di un abalone è un compito piuttosto complesso che si suddivide in più fasi: tagliare il guscio attraverso il cono, colorarlo ed infine conteggiare il numero di anelli del mollusco al microscopio. Anche l'ubicazione e le condizioni meteorologiche a cui è sottoposto l'abalone sono informazioni necessarie per tale analisi.

Perciò, data la difficoltà di quanto descritto sopra, si è voluto automatizzare tale processo predicendo l'età di un abalone con un classificatore.

Il dataset utilizzato per la classificazione si compone di 4177 istanze descritte da 9 colonne di attributi, illustrati nella seguente tabella (Tabella 1).

Tabella 1. Descrizione e analisi degli attributi

ATTRIBUTO	DESCRIZIONE	TIPOLOGIA	DOMINIO
SEX	Sesso dell'abalone (maschio, femmina o infante)	Categorico nominale	{M,F,I}
LENGTH	Lunghezza del guscio	Numerico continuo	[0.15,0.81]
DIAMETER	Diametro del guscio	Numerico continuo	[0.11,0.65]
HEIGHT	Altezza del guscio	Numerico continuo	[0.01,1.13]
WHOLE WEIGHT	Peso complessivo del guscio e del mollusco al suo interno	Numerico continuo	[0.01,2.82]
SHUCKED WEIGHT	Peso del mollusco (senza il guscio)	Numerico continuo	[0.006,1.48]
VISCERA WEIGHT	Peso delle viscere	Numerico continuo	[0.003,0.76]
SHELL WEIGHT	Peso del guscio (senza il mollusco)	Numerico continuo	[0.005,1.00]
RINGS	Numero di anelli (aggiungendo +1.5 permette di stabilire l'età dell'abalone in anni)	Numerico discreto	[3,29] ∩ N

Nella fase di pre-processing dei dati, seguendo le specifiche del progetto, sono state effettuate le seguenti modifiche:

- Eliminazione delle istanze riguardanti gli infanti (Sex = I). Il Dataset rimanente è costituito da 2835 record.
- Trasformazione da stringhe a valori numerici del dominio dell'attributo *Sex* che è stato mappato in {0, 1}, rispettivamente per il sesso femminile e maschile.
- Discretizzazione dell'attributo *Rings* dividendo il range dei valori dell'attributo in due intervalli.

Al fine di individuare gli intervalli migliori per la discretizzazione si è osservata la distribuzione dell'attributo *Rings* come illustrato di seguito.

Sia il bar chart con curva gaussiana (Figura 13) che il Box Plot (Figura 14) mostrano risultati analoghi: i dati compresi tra il primo e il terzo quartile si trovano nell'intervallo [6-16] circa e includono il 92% dei dati che si concentrano intorno a 10, la mediana. Le istanze rimanenti comprese negli intervalli [3-5] e [17-29] possono essere considerate valori anomali in quanto con una percentuale rispettiva dello 0,84% e 6,2% si discostano nettamente dall'andamento generale.

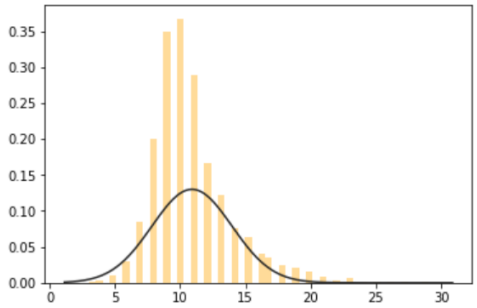

Figura 13: Distribuzione Rings

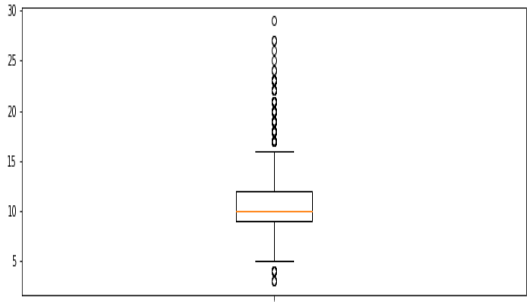

Figura 14: Box Plot Rings

Pertanto, si è optato inizialmente per tre possibili soluzioni:

1. *[6-10] [11-21]* Per la prima soluzione si è pensato di escludere dall'analisi le istanze poco frequenti seguendo i criteri descritti sopra; si è esteso il secondo intervallo da 16 a 21 in quanto, come è possibile notare anche dai grafici (Figura 1 e 2), la concentrazione dei dati diventa quasi nulla dopo i valori 21/22. Al primo intervallo appartengono circa il 55% delle istanze, al secondo il 45%.
2. *[3-10] [11-29]* Per la seconda soluzione si è deciso di includere tutti i record, individuando i due intervalli in base alla mediana (10). Al primo intervallo appartengono circa il 55% delle istanze, al secondo il 45%.
3. *[3-14] [15-29]* Infine anche per la terza soluzione si è deciso di includere tutti i dati dividendo il range dei valori circa a metà. Al primo intervallo appartengono circa l'88% delle istanze, al secondo il 12%.

Le prime due soluzioni, testate con più algoritmi di classificazione, presentano risultati molto simili perciò nelle considerazioni successive si è deciso di indicare solamente i risultati relativi alla discretizzazione *[3-10] [11-29].*
La discretizzazione presentata al punto 3, risulta nettamente sbilanciata in quanto la maggioranza dei record ricade nell'intervallo [3-14]. Come mostrato successivamente, si è osservato che la performance del classificatore costruito su dati sbilanciati emulava quella di un *trivial classifier*, ossia un modello di previsione che restituisce sempre la classe di maggioranza.
Si è deciso perciò di effettuare l'*oversampling* della classe di minoranza [15-29]. A tal fine si è utilizzato l'algoritmo *SMOTE*, tecnica che invece di duplicare semplicemente le istanze, crea nuove istanze che sono interpolazioni della classe di minoranza. Dopo l'oversampling i due intervalli risultano quindi piuttosto bilanciati: 58% dei record per la classe di maggioranza e 42% per quella di minoranza.

I metodi di classificazione scelti per analizzare il dataset sono: *Support Vector Machine, K-nearest neighbors* e alcuni Ensemble Methods, quali *Random Forest, Bagging e Boosting*.

Si è scelto di suddividere il dataset in due insiemi di riferimento (*Holdout validation*) al fine di evitare l'overfitting e l'underfitting dei classificatori:

- *Training Set*: costituito dall'80% dei dati, utilizzato per addestrare il classificatore e per trovare la discretizzazione che massimizzasse la performance dei modelli
- *Test Set*: formato dal 20% dei dati, utilizzato per determinare l'accuratezza del modello misurando la capacità del classificatore di generalizzare i risultati.

Support vector machine

In primo luogo, si sono effettuate delle Grid Search con lo scopo di trovare la combinazione di valori dei parametri che producesse il modello più accurato possibile. I parametri sottoposti sono i seguenti (Tabella 2):

Tabella 2: Parametri SVM

PARAMETRI	VALORI
FUNZIONE KERNEL	*radial basis function* (RBF)
GAMMA	[0.001, 0.01, 0.1, 1]
C	[0.001, 0.01, 0.1, 1, 10]

Estraendo dalla GridSearch (Figura 14) i risultati migliori si nota come il punteggio più alto si ottenga impostando C a 10 e Gamma a 1.

Dopo aver trovato i migliori iperparametri per il training set, la k-fold Cross-Validation con k=5 sul classificatore SVM ha prodotto i seguenti risultati:

```
Model with rank: 1
Mean validation score: 0.731 (std: 0.003)
Parameters: {'C': 10, 'gamma': 1, 'kernel': 'rbf'}

Model with rank: 2
Mean validation score: 0.710 (std: 0.007)
Parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}

Model with rank: 3
Mean validation score: 0.707 (std: 0.011)
Parameters: {'C': 1, 'gamma': 1, 'kernel': 'rbf'}
```

Figura 14: Grid Search SVM

- **Classi sbilanciate [3-14][15-29] :** accuracy 89% e standard deviation +/- 0.01
- **Classi sbilanciate oversampled [3-14][15-29] :** accuracy 81% e standard deviation +/- 0.03
- **Classi bilanciate [3-10][11-29] :** accuracy 73% e standard deviation +/- 0.03

Al fine di valutare le prestazioni del modello, per ogni discretizzazione presa in considerazione, sono state calcolate le relative Confusion Matrix.

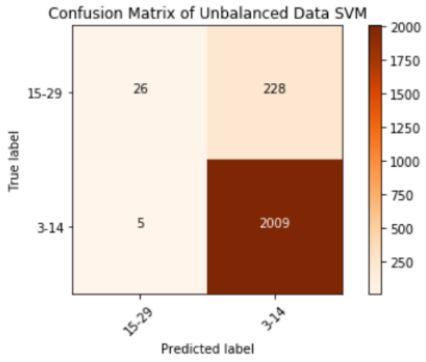

Figura 15: Dati sbilanciati SVM

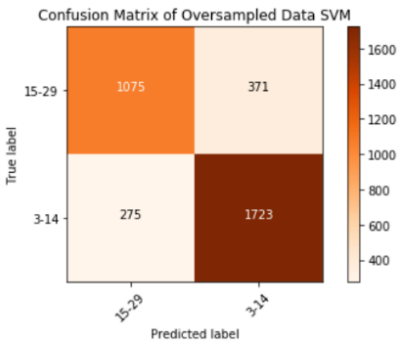

Figura 16: Dati oversampled SVM

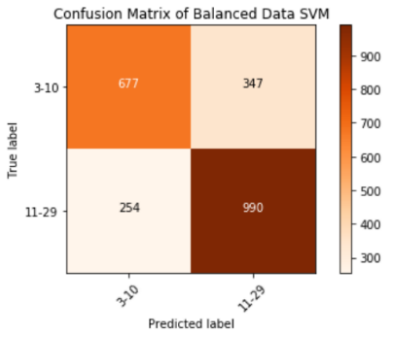

Figura 17: Dati bilanciati SVM

Inizialmente si sono confrontate le performance del classificatore sui dati sbilanciati (Figura 15) con quelle sui dati bilanciati tramite oversampling (Figura 16) per verificare se le assunzioni fatte in precedenza fossero corrette. Osservando la matrice di confusione in Figura 15 si evince che il classificatore, nonostante presenti un livello di accuratezza molto alto (89%) ed una standard deviation bassa (+/- 0.01), non fa altro che restituire semplicemente la classe di maggioranza [3-14], comportandosi quindi come un *trivial classifier*. Il classificatore in Figura 16, nonostante presenti un accuracy più bassa (80%) ed una standard deviation più alta (+/- 0.03) riesce a predire entrambe le classi.

Anche il classificatore costruito sulla discretizzazione bilanciata (Figura 17) riesce a predire entrambe le classi ma con un'accuratezza decisamente minore (73%) rispetto ai dati oversampled.

Si è voluto confrontare ulteriormente le performance del classificatore sulle diverse discretizzazioni attraverso le ROC curves (Figura 18). Osservando l'area sotto la curva, quanto affermato in precedenza viene confermato: il modello con i dati ribilanciati tramite oversampling risulta il migliore mentre quello con i dati non bilanciati il peggiore.

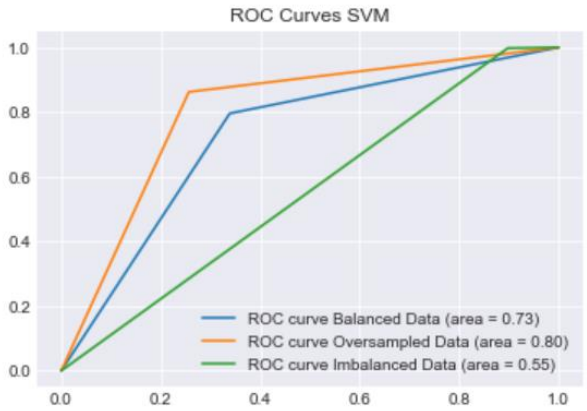

Figura 18: Roc SVM

Dopo aver dimostrato che il classificatore costruito sui dati sbilanciati si comporta come un *trivial classifier,* si è deciso di mostrare in seguito solamente i risultati ottenuti utilizzando la discretizzazione bilanciata e quella ribilanciata tramite oversampling.

K-nearest neighbors

Anche per KNN sono state effettuate Grid Search per le due discretizzazioni prese in considerazione, al fine di trovare, per ognuna, il migliore valore di k.

Come mostrato nelle figure 19 e 20 sono stati testati i valori di k compresi nel range [1-30], scegliendo il valore di k che mostrava l'accuratezza maggiore.

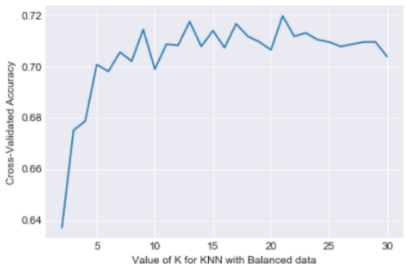

Figura 19: Scelta del miglior K dati bilanciati

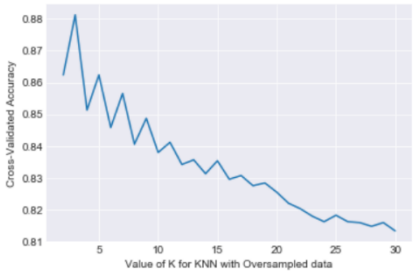

Figura 20: Scelta del miglior K dati oversampled

I parametri utilizzati per costruire il modello sono i seguenti:

Tabella 3: Parametri KNN

	N° DI VICINI	WEIGHTS	METRIC
BALANCED	21	uniform	minkowski
OVERSAMPLED	3	uniform	minkowski

Dopo aver trovato i migliori iperparametri la k-fold Cross-Validation con k=5 sul classificatore KNN ha prodotto i seguenti risultati:

- **Classi bilanciate** [3-10][11-29]: accuracy 72% e standard deviation +/- 0.02
- **Classi sbilanciate oversampled** [3-14] [15-29]: accuracy 88% e standard deviation +/- 0.01

Osservando sia le confusion matrix (Figura 21 e 22) che le ROC curves (Figura 23), si evince chiaramente che, nonostante entrambi i modelli riescono a predire entrambe le classi, il modello costruito con i dati oversampled risulta decisamente più accurato dell'altro (88%).

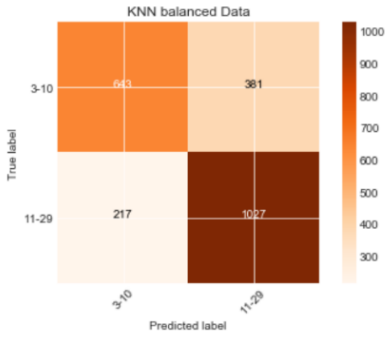

Figura 21: KNN bilanciati

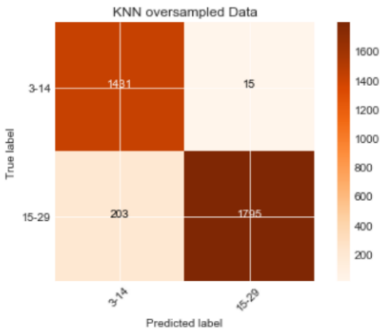

Figura 22: KNN sbilanciati

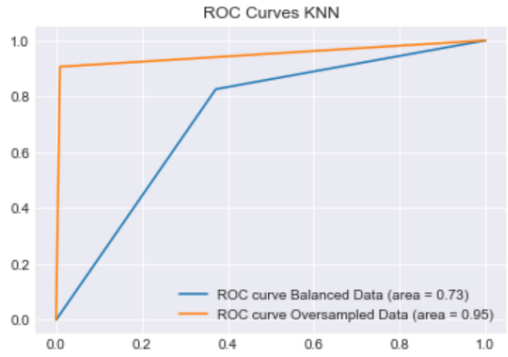

Figura 22: Roc KNN

Ensemble method

In questo capitolo si descrivono le tecniche di classificazione basate sulla combinazione di classificatori deboli (*weak)* con l'obiettivo di ottenerne uno più forte. I metodi di ensemble scelti sono: *Adaptive Boosting, Bagging* e *Random Forest*.

Adaptive boosting

Questa tecnica di classificazione è stata applicata alle classi precedentemente individuate tramite la discretizzazione dell'attributo *Rings* presente nel dataset iniziale, ovvero *Balanced, Unbalanced* e *Oversampled.*

Il classificatore debole eseguito ad ogni ciclo di addestramento è l'albero di decisione. Per ottimizzare i parametri è stata effettuata una Grid Search così da considerare in modo esaustivo tutte le possibili combinazioni.

Di seguito nella tabella 4, vengono indicati i migliori parametri individuati dalla ricerca in base alla classe presa in analisi e, quindi, passati al classificatore:

Tabella 4: Parametri decision tree

	MAX_DEPTH	CRITERION	SPLITTER	MIN_SAMPLES_SPLIT	MIN_SAMPLES_LEAF
BALANCED	10	entropy	random	6	9
UNBALANCED	5	gini	random	4	7
OVERSAMPLED	7	entropy	best	4	4

I Decision *Tree* con i precedenti valori sono stati considerati come "base_estimator" per eseguire l'Adaptive Boosting delle diverse classi. Per la ricerca dei parametri si è effettuato lo stesso procedimento descritto sopra, ovvero tramite la Grid Search, che ha condotto ai risultati seguenti (Tabella 5).

Tabella 5: Parametri adaboost

	N_ESTIMATORS	LEARNING_RATE	ALGORITHM
BALANCED	**220**	**0.5**	**SAMME**
UNBALANCED	**150**	**0.7**	**SAMME**
OVERSAMPLED	**230**	**0.9**	**SAMME.R**

Come indicato precedentemente nel paragrafo "Support Vector Machine", vengono mostrati i risultati più significativi ovvero quelli riguardanti la discretizzazione bilanciata e quella ribilanciata tramite oversampling.

In particolare:

- **Classi bilanciate** [3-10][11-29]: accuracy 70% e una standard deviation +/- 0.02
- **Classi sbilanciate oversampled** *[3-14][15-29]*:accuracy 85% e una standard deviation +/- 0.01

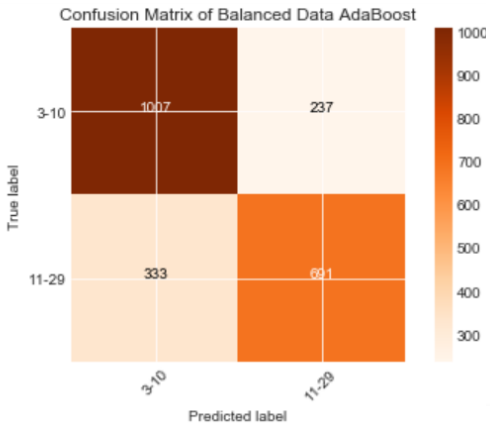

Figura 24: AdaBoost bilanciate

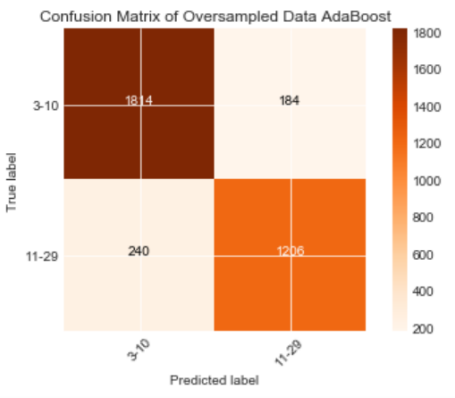

Figura 25: Adaboost oversampled

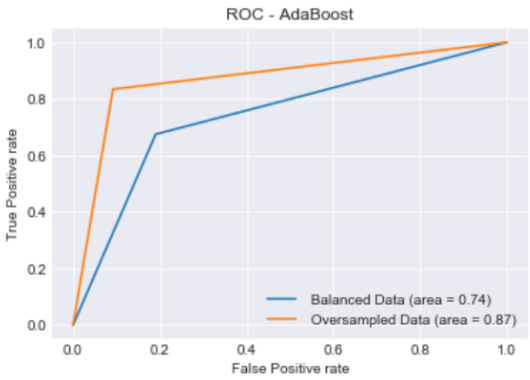

Figura 26: ROC AdaBoost

La classe oversampled (Figura 25) produce un modello migliore rispetto a quella bilanciata con una AUC (Figura 26) di 0.87. Il modello è stato applicato al test set e ha prodotto un'accuracy del 83%.

Bagging

Per l'ottimizzazione dei parametri è stata effettuata una Grid Search. Di seguito, vengono indicati i migliori parametri individuati dalla ricerca in base alla classe presa in analisi e, quindi, passati al classificatore ad albero (Tabella 6).

Tabella 6: Parametri decision tree

	CRITERION	SPLITTER	MIN_SAMPLES_SPLIT	MIN_SAMPLES_LEAF
BALANCED	gini	random	3	3
OVERSAMPLED	entropy	best	4	4

I Decision *Tree* con i precedenti valori sono stati considerati come "base_estimator" del classificatore. Per la ricerca dei parametri si è effettuato lo stesso procedimento descritto sopra, ovvero tramite la Grid Search, che ha portato ai risultati seguenti:

Tabella 7: Parametri bagging

	N_ESTIMATORS	MAX_SAMPLES
BALANCED	100	0.2
OVERSAMPLED	250	1.0

I precedenti valori sono stati passati al classificatore per costruire i modelli e ne sono derivati i risultati indicati di seguito:

- **Classi bilanciate** [3-10][11-29]: accuracy 71% e standard deviation +/- 0.02
- **Classi sbilanciate oversampled** *[3-14][15-29]* : accuracy 89% e standard deviation +/- 0.02

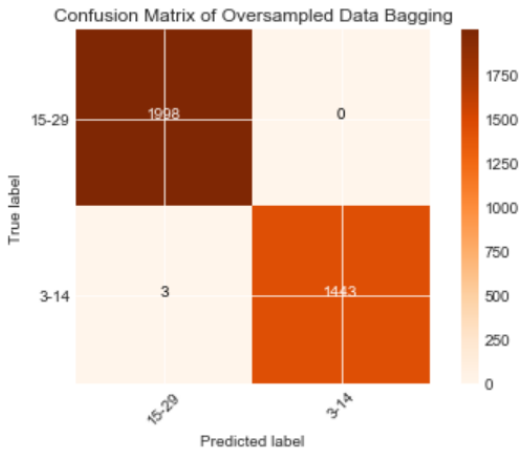

Figura 27: Bagging Oversampled

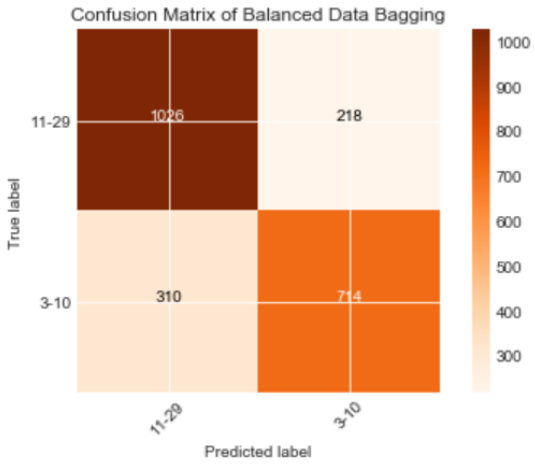

Figura 28: Bagging bilanciate

Anche in questo caso, il modello basato sui dati oversampled (Figura 27) ha una migliore prestazione. Data la precisione con cui distingue le classi, si è valutato anche sul test set e si ha un'accuracy del 87%. Infine, sono state plottate in un unico grafico le curve ROC relative alle classi per valutarne l'andamento. Per stabilire la performance si è presa in considerazione l'*Area Under Curve.* Come nell'AdaBoost, l'AUC migliore risulta essere quella dei dati *oversampled*.

Random forest

Partendo dal data set con la classe già bilanciata sono state effettuate alcune *RandomizedSearch* per cercare di individuare dei buoni parametri da applicare al modello.

Tabella 8: Parametri randomized search

	N_ESTIMATORS	MAX_DEPTH	MIN_SAMPLES_LEAF	MIN_SAMPLES_SPLIT
BALANCED	400	90	4	100
OVERSAMPLED	1400	40	1	2

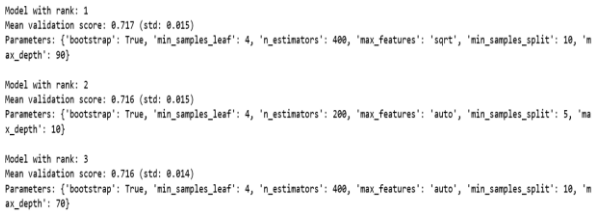

Figura 29: Randomized search bilanciati

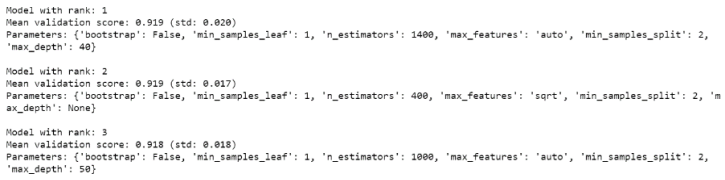

Figura 30: Randomized search sbilanciati

L'applicazione dei parametri ai classificatori ha generato i seguenti risultati:

- **Classi bilanciate** [3-10][11-29]: accuracy 71% e standard deviation +/- 0.02
- **Classi sbilanciate oversampled** *[3-14][15-29]* : accuracy 100% e standard deviation +/- 0.03

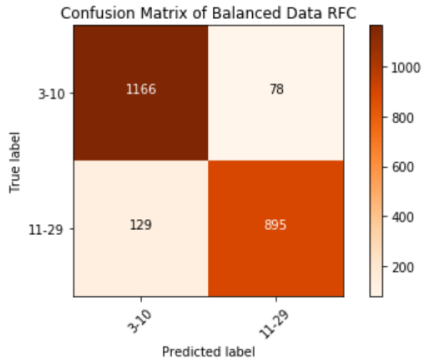

Figura 31: RF bilanciati

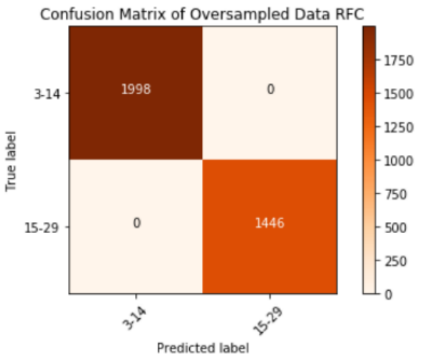

Figura 32: RF oversampled

Come è possibile notare dalla matrice di confusione (Figura 32), i dati oversampled identificano perfettamente le due classi. Il modello è stato provato anche sul test set e ha prodotto un'accuracy del 90%.

Considerazioni

I risultati ottenuti dai modelli testati sul training set mostrano chiaramente che la discretizzazione ottenuta ribilanciando la classe di minoranza tramite oversampling permette di raggiungere livelli di accuratezza maggiore. Pertanto, utilizzando tale discretizzazione, si sono testati i classificatori sul test set per vedere se gli alti valori di accuracy ottenuti sul training sono causati da overfitting del modello o se invece il classificatore è in grado di generalizzare. Nella tabella 9 sono confrontati le performance dei diversi modelli sul training set e sul test.

Anche se l'accuratezza dei modelli diminuisce sul test set, tale variazione non è cosi significativa da poter parlare di overfitting: si osserva, sia dalle confusion matrix che dalle roc curves (Figura 35), che i modelli riescono a predire bene entrambe le classi.

Tabella 9: Performance modelli

	ACCURACY MODELLO	
	Training set	*Test set*
SVM	81%	80%
KNN	88%	86%
BOOSTING	85%	84%
BAGGING	89%	78%
RANDOM FOREST	100%	92%

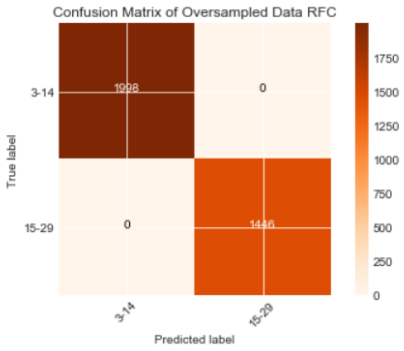

Figura 33: RF oversampled training set

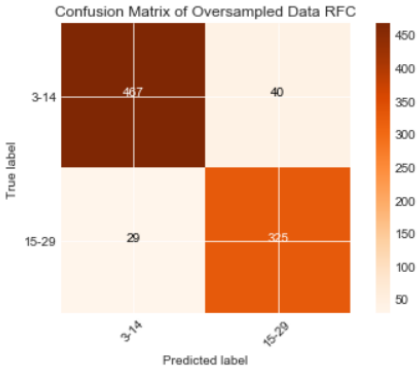

Figura 34: RF oversampled test set

I modelli ottenuti hanno tutti valori molto simili per quanto riguarda accuracy e standard deviation. Il classificatore che performa meglio è il Random Forest (Figura 33 e Figura 34). Probabilmente un'accuracy così alta deriva dall'uso di un numero di alberi molto elevato (1400) su un dataset piuttosto ristretto.

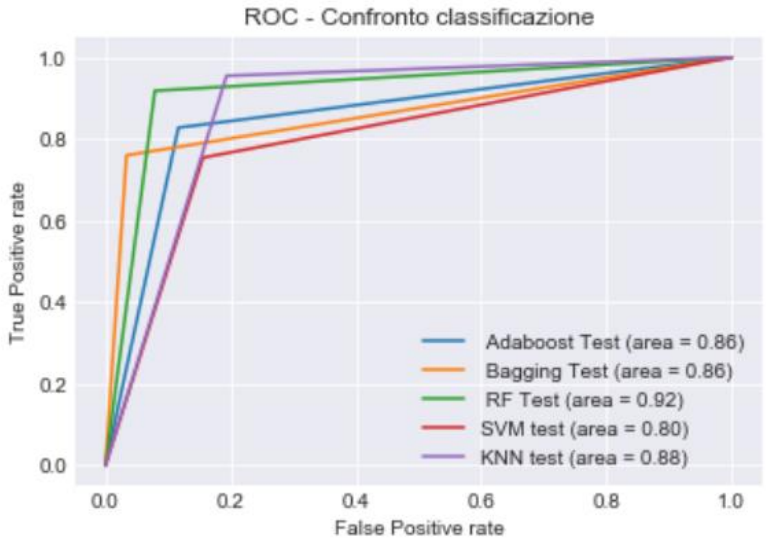

Figura 35: ROC curve confronto performance dei modelli sul test set

Outlier Detection

In questo capitolo, si descrive l'applicazione di alcune tecniche di outlier detection utilizzate nel dataset *Abalone*[2] con l'obiettivo di identificare l'1% dei record con la maggiore probabilità di essere outlier.

Scelta delle dimensioni del dataset

Al fine di individuare il numero ottimale di dimensioni del dataset su cui applicare i vari algoritmi, si sono calcolate le correlazioni tra gli attributi utilizzando il coefficiente di Pearson. Come si evince dalla matrice di correlazione (Figura 36), vi sono valori particolarmente alti che indicano un'elevata correlazione tra gli attributi del dataset.

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings
Length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole_weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked_weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera_weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell_weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

Figura 36: Correlazione tra gli attributi

Pertanto, sono stati scelti i quattro attributi più utili al fine dell'individuazione degli outlier ossia *Lenght, Height, Shucked_weight* e *Rings.*

Scelta delle tecniche

In seguito alla selezione degli attributi, è stata necessaria un'attenta valutazione per decidere quali fossero le tecniche più adatte per l'identificazione degli outlier nel dataset di riferimento.

Si è ritenuto che, i modelli basati sugli approcci *depth-based, deviation-based* e *high-dimensional,* fornissero dei risultati poco significativi. Quindi, si è optato per l'applicazione di 3 algoritmi basati su differenti approcci. In particolare:

- Sulla clusterizzazione: DBSCAN
- Sulla densità: LOF
- Sulla distanza: Mahalanobis

DBSCAN

Per stimare i parametri ottimali dell'algoritmo, l'epsilon e i minpoints, si sono utilizzati i grafici del k-dist[3] usando come funzione di distanza quella euclidea e prendendo in considerazione il range dei valori corrispondenti al gomito della curva. In Figura 37, viene mostrato il grafico che presenta sulle

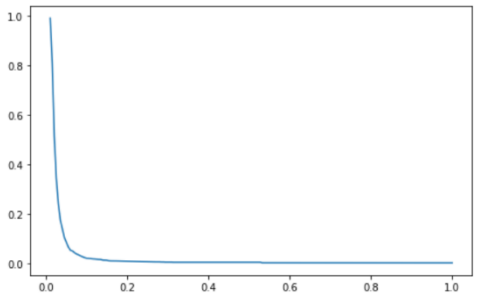

Figura 37: Stima del parametro eps

[2] https://archive.ics.uci.edu/ml/datasets/Abalone
[3] P. Tan, M. Steinbach, V. Kumar, *Introduction to Data Minin*g, Pearson Addison Wesley, 2006, p. 530

ascisse, la percentuale del dataset identificata come outlier e sulle ordinate, l'espilon corrispondente.
I parametri scelti sono ε=0.20 e *minPts* = 200. Questi valori settati nell'algoritmo DBSCAN con *minsample=10* ci hanno permesso l'individuazione 41 outlier, ovvero l'1% del dataset.

Questo procedimento si è applicato a tutti gli attributi selezionati per l'analisi confrontandoli a due a due. Di seguito, i grafici risultanti. In rosso sono rappresentati i punti anomali e in blu i restanti.

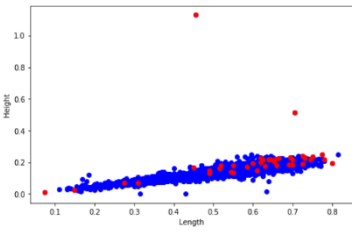

Figura 38: Length/Height

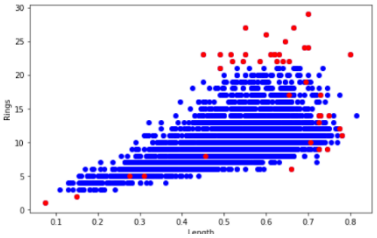

Figura 39: Length/Rings

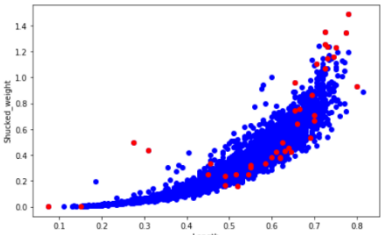

Figura 40: Lenght/Shucked_weight

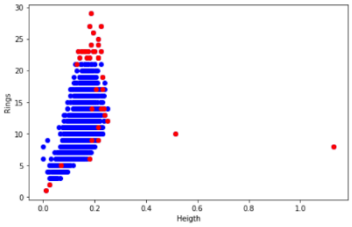

Figura 41: Height/Rings

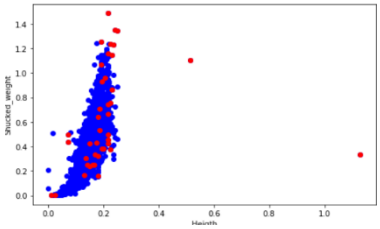

Figura 42: Height/Shucked_weight

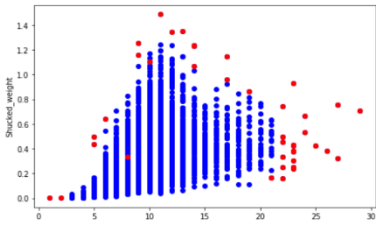

Figura 43: Rings/Shucked_weight

LOF

Il Local Outlier Factor è un algoritmo *density-based* che identifica gli outlier confrontando la densità di un determinato punto rispetto a quella dei suoi punti vicini. Quindi, in base a questo approccio, un punto viene definito outlier se la sua densità è notevolmente inferiore a quella dei propri *kNN*.
Conoscendo a priori il numero di outlier da individuare, ovvero l'1% del dataset, si è settato il paramento *contamination* a 0.01. Inoltre, si sono effettuati diversi test per ricercare i *n_neighbors* e il parametro ottimale è risultato *15*.

Di seguito, vengono mostrati i risultati, come nel DBSCAN, a coppie di due attributi.

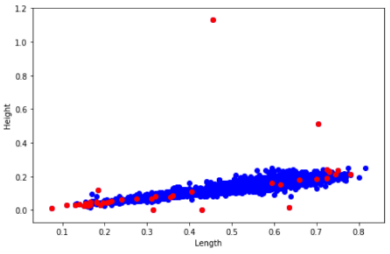

Figura 44: Length/Height

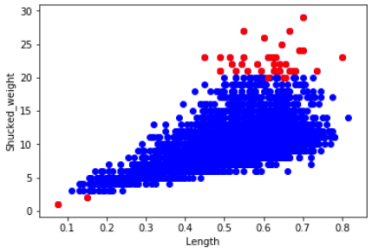

Figura 45: Length/Shucked_weight

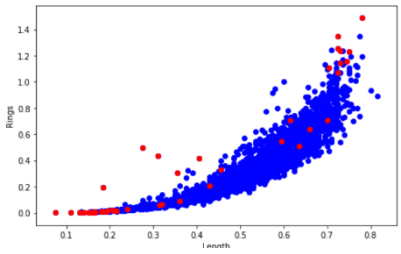

Figura 46: Length/rings

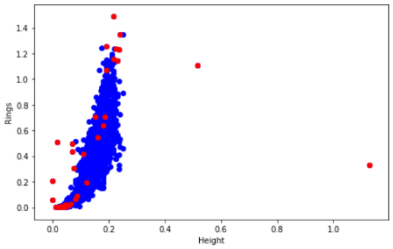

Figura 47: Height/rings

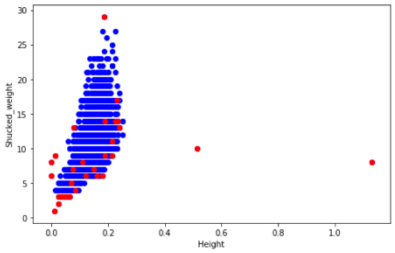

Figura 48: Height/Shucked_weight

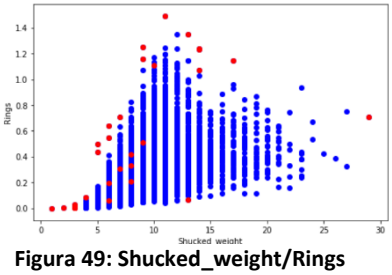

Figura 49: Shucked_weight/Rings

Mahalanobis Distance

La distanza di Mahalanobis tiene conto delle correlazioni tra gli attributi permettendo di valutare quanto i valori si allontanano dalla distribuzione dei dati. Al fine di individuare l'1% degli outlier, si è applicato l'algoritmo impostando una soglia tra il 1% e il 2%. I relativi risultati sono stati plottati a coppie di attributi come viene mostrato nei seguenti grafici.

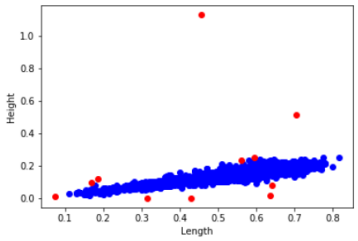

Figura 50: Lenght/Height

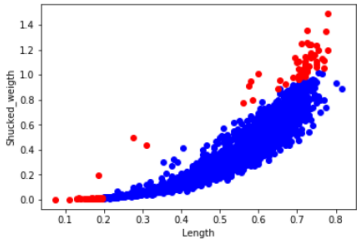

Figura 51: Lenght/Shucked_weight

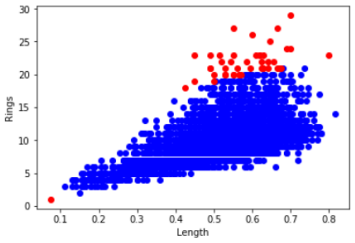

Figura 52: Lenght/Rings

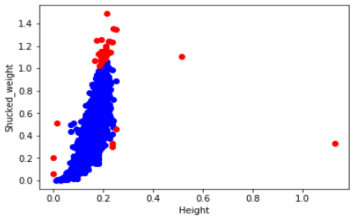

Figura 53: Heigth/Shucked_weight

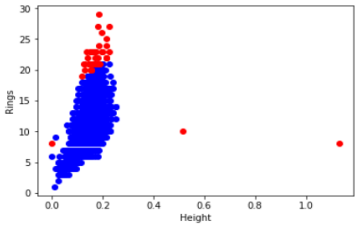

Figura 54: Height/Rings

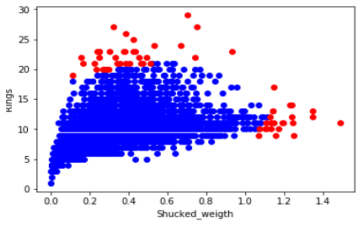

Figura 55: Shucked_weight/Rings

Considerazioni

Per valutare quale, tra i metodi utilizzati, abbia avuto la prestazione migliore sono stati messi a confronto i risultati ottenuti, paragonando tra gli algoritmi i record classificati come outlier.
Il DBSCAN è quello più performante in quanto riesce a determinare gli outlier individuati mediante gli altri due approcci ma anche ulteriori non presenti negli altri. In particolare, confrontando gli outlier del DBSCAN con quelli del LOF e del Mahalanobis, nel primo caso sono stati individuati 24 outlier in comune mentre nel secondo 27.