UNIVERSITÀ DI PISA

IBM Stock Prices and Abalones

Alunni: Atzeni Daniele | Bachini Francesco | Grassi Francesco

Docenti: Pedreschi | Nanni

Anno Accademico: 2017/2018

Data Mining II
2017/2018

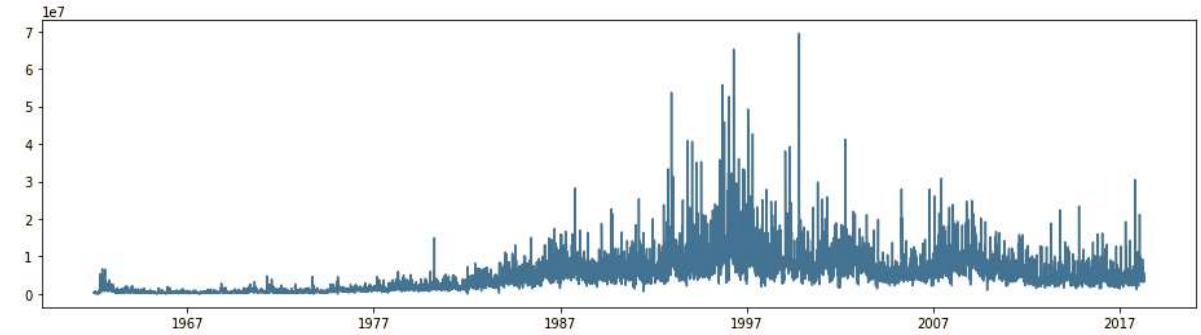
Indice

IBM Stock Prices

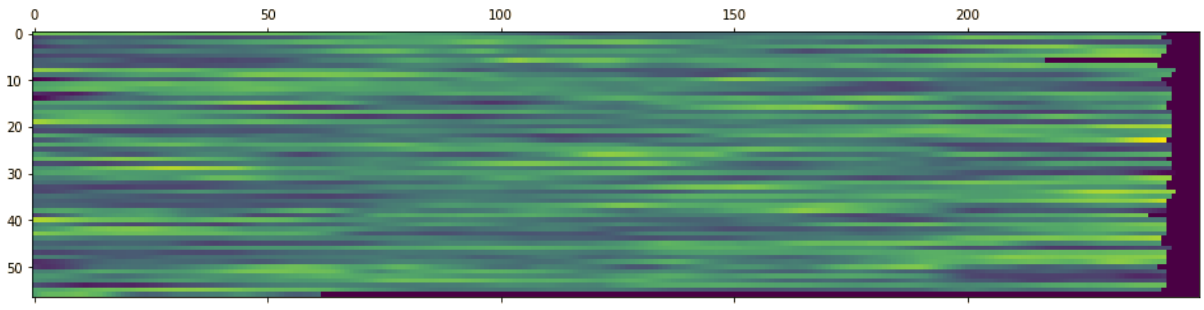
Introduzione

I dati analizzati in questa prima parte della relazione sono costituiti dalla serie storica dei prezzi giornalieri di apertura delle azioni IBM, prezzi collezionati lungo il periodo il cui inizio è datato 2 Gennaio 1962 e la cui fine è datata 13 Aprile 2018. I prezzi analizzati sono in totale 14168, con un prezzo minimo di 4.08$ ed un massimo di 215.38$. La media è di 57.43$ e la deviazione standard è, invece, pari a 56.43$. Come osservabile nella figura sottostante, l'andamento della curva è piuttosto piatto fino al 1995, anno in cui si registra un primo picco di crescita. Infatti, il 75% dei prezzi sono inferiori a 93.76$ e tale picco è poi seguito da una serie di alti e bassi fino al 2010, anno in cui si registra una seconda altrettanto marcata crescita in cui il valore medio dell'azione raggiunge circa i 168$.

A conferma di quanto detto precedentemente, segue il livello del volume delle azioni (figura sottostante), che mostra come tale livello abbia iniziato la sua fase di crescita a cavallo tra il 1977 e il 1982, per poi incontrare i suoi primi picchi a partire dal 1995. Superata poi una prima fase di discesa, dal 2005 in poi si registrano dei livelli di volume più o meno costanti.



Un'ulteriore analisi è stata effettuata attraverso la suddivisione delle Time Series in 56 anni diversi. I valori rappresentati all'interno della matrice (figura sottostante), i quali rappresentano i prezzi delle azioni di ogni anno (asse verticale) e di ogni giorno (asse orizzontale), sono stati normalizzati e si differenziano tra loro attraverso una scala di colori che va dal viola (valore minimo), passando al verde (valore intermedio), fino ad arrivare al giallo (valore massimo).



Il numero dei giorni per ogni anno non è di 365 giorni, come intuitivamente si sarebbe portati a pensare, e questo è dovuto al fatto che di norma i mercati restano chiusi sia il sabato che la domenica, non potendo quindi rilevare alcun prezzo.
Inoltre, nel 1968 e nel 2018 si registra una collezione inferiore di dati. Nel primo caso, la causa è riconducibile a dei problemi sorti nella documentazione della Borsa di New York, la quale decise di rimanere chiusa ogni mercoledì a partire da Giugno. Nel secondo caso, la causa è riconducibile al fattore tempo: i dati sono stati raccolti fino al 13 Aprile 2018.

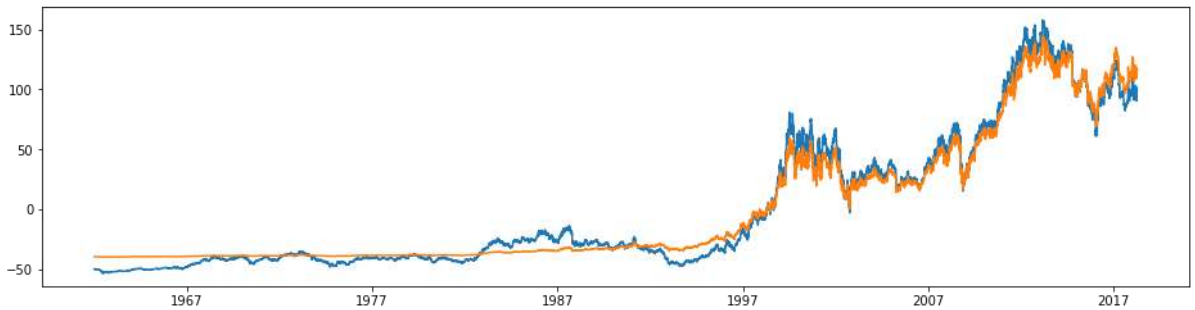
Similarity Analysis

Entrando più nel vivo di questa analisi, il primo passo è studiare se esiste una similarità tra le 56 time series annuali.

Preprocessing

La prima fase è quindi caratterizzata da una fase di pre-processo di queste Time Series, con lo scopo di evitare che i risultati della nostra analisi possano tradursi in risultati poco significativi.
Per ottenere Time Series della stessa lunghezza nei giorni finali di ogni anno è stato inserito il valore -3. In seguito, le Time Series sono state standardizzate, con valori tra -3 e +3, al fine di ottenere un valore medio pari a 0 e una varianza pari ad 1 per ogni anno.
Successivamente abbiamo rimosso il rumore, sostituendo il valore di apertura delle azioni con la media del valore delle azioni nei 10 giorni precedenti. Questa scelta è stata presa in considerazione dopo aver constatato che i risultati ottenuti nel clustering, con e senza rumore, fossero abbastanza simili, dandoci quindi la certezza che l'eliminazione del rumore non avrebbe comportato una cospicua perdita di informazioni. Infine, abbiamo deciso di non rimuovere il linear trend poiché abbiamo notato che, la differenza tra il valore delle azioni ad inizio anno e il valore delle azioni a fine anno è minima, quindi un riallineamento delle Time Series avrebbe esclusivamente comportato la perdita delle informazioni sull'andamento generale delle azioni durante l'anno.

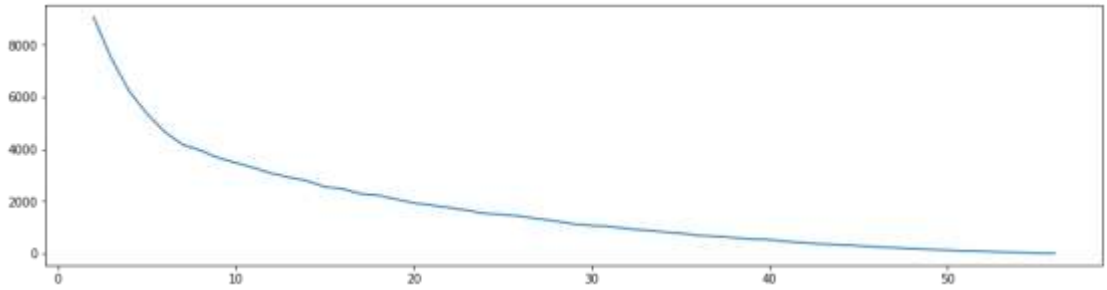
L'analisi fatta si è quindi incentrata tra i prezzi di apertura delle azioni e i prezzi "aggiustati". Tale analisi ha subito messo in risalto la sua diversa interpretazione delle due curve (curva dei prezzi in apertura in blu e dei prezzi di aggiustamento in arancione) (figura sottostante):



Dal 1995 in poi, le due curve mostrano una quasi perfetta sincronia nell'oscillazione dei loro rispettivi valori. Al contrario, in tutto il periodo precedente, la curva dei valori aggiustati è quasi sempre in costante crescita, contando delle ricadute impercettibili. I valori in apertura, invece, mostrano, rispetto a quelli di chiusura, delle lievi oscillazioni, ma sicuramente più marcate e soprattutto più visibili (figura sovrastante).
L'aggiustamento del prezzo di un'azione, per intenderci, può essere determinato nell'eventualità in cui un emittente quotato di un titolo annunciasse uno stacco di dividendi, un consolidamento di azioni, o modificasse la sua struttura di capitale mediante emissione di azioni, bonus o dividendi cash. Di conseguenza, nel primo periodo nessuna di queste cause elencate è stata così marcata da influenzare un grosso aggiustamento del prezzo, spiegando così il perché di tale differenza.

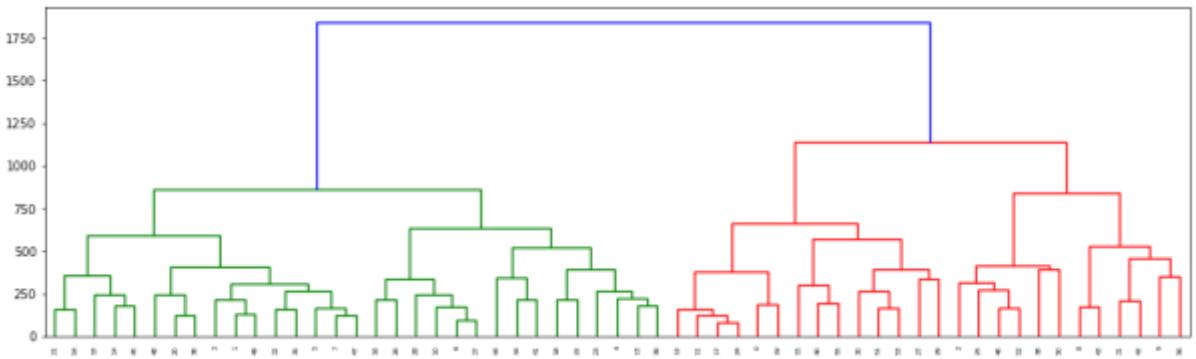
Clustering

In questa analisi delle Time Series abbiamo scelto di adottare tre diverse tipologie di clustering attraverso l'analisi delle similarità di tutte le Time Series annuali.
Gli algoritmi utilizzati per l'analisi sono: DBscan, Hierarchical e K-Means.
Per quanto riguarda il K-Means, la scelta della distanza da utilizzare, come metrica per l'algoritmo, è ricaduta sulla distanza euclidea, plottando quindi l'SSE in funzione del numero di cluster per poter individuare il valore da assegnare al parametro k.



Dalla figura sovrastante si evince che non vi è un vero e proprio "gomito" che ci indichi il numero dei cluster. Tuttavia, è possibile individuare alcuni valori a partire dai quali l'SSE diminuisce meno velocemente. I numeri di cluster presi in considerazione sono stati 6, 8 e 10, i quali hanno mostrato un'alta similarità tra loro, la quale ci ha permesso di scartare le soluzioni con più clusters in modo da rendere meno difficile e lunga l'analisi degli stessi.

Per l'algoritmo del DBscan è stata utilizzata nuovamente la distanza euclidea, dal momento che la DTW tendeva a raggruppare negli stessi cluster anche Time Series piuttosto diverse tra loro. I risultati migliori sono stati ottenuti impostando l'eps a 8,5. Per tale valore sono stati individuati 6 cluster, tutti piuttosto omogenei, ma poiché quasi la metà delle Time Series (28 per la precisione) è stata classificata come noise. Per questo motivo abbiamo deciso di scartare il DBscan.
Nel clustering gerarchico, al contrario degli altri due algoritmi, è stata scelta come distanza la DTW, calcolando la matrice delle distanze, e come algoritmo è stato utilizzato il Complete Linkage (figura sottostante).



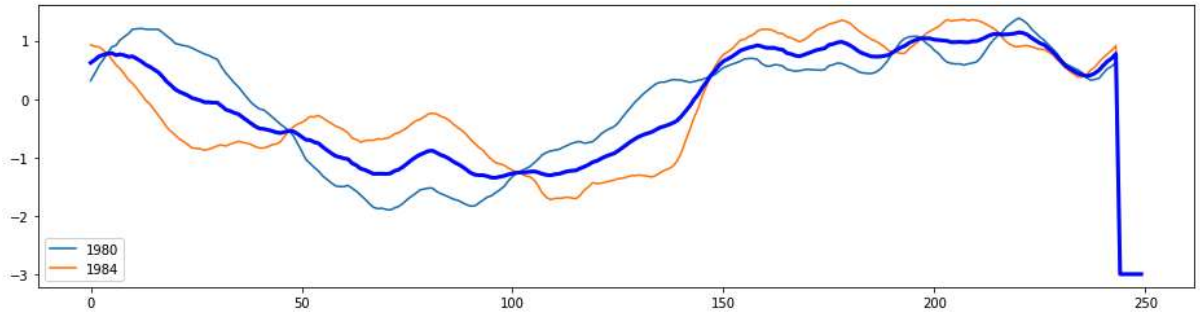
Infine, abbiamo tagliato l'albero in modo da ottenere 7 cluster e abbiamo analizzato i risultati, i quali, nonostante fossero abbastanza soddisfacenti, non abbiamo ritenuto essere all'altezza di quelli ottenuti utilizzando il K-Means.

Analisi Cluster con K-Means

Come detto precedentemente, i valori del numero di cluster settati sia a 6 che a 8 che a 10 sono risultati essere abbastanza simili tra loro. Si è quindi deciso di riportare, per semplicità, l'analisi dei cluster con k=8. Le linee blu delle figure seguenti rappresentano i centroidi dei cluster analizzati. Passiamo quindi all'analisi dei singoli cluster.

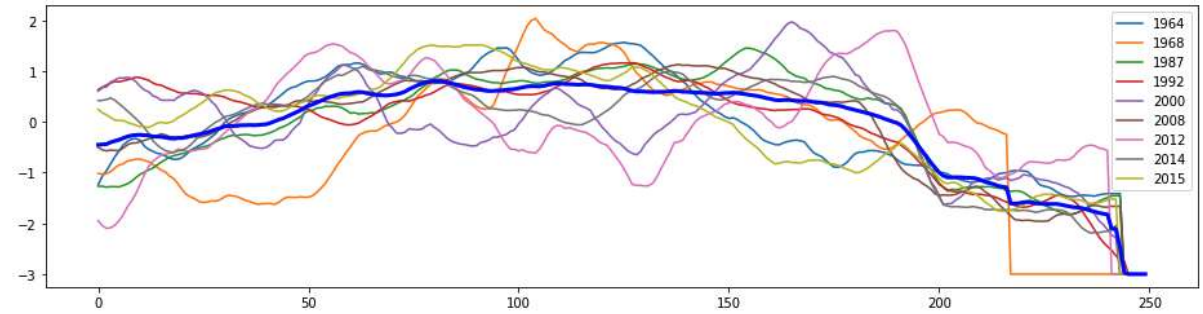
Primo Cluster

Il primo cluster è composto da solo due Time Series, relative agli anni 1980 e 1984, nelle quali il prezzo di apertura per stock IBM è rimasto pressoché invariato rispetto al prezzo di chiusura, con un piccolo rialzo rispetto al prezzo ad inizio anno. Rispetto al centroide si può notare come i due anni siano caratterizzati da un comportamento altalenante del prezzo, carattere che tende poi a stabilizzarsi dopo il centocinquantesimo giorno dei mercati.



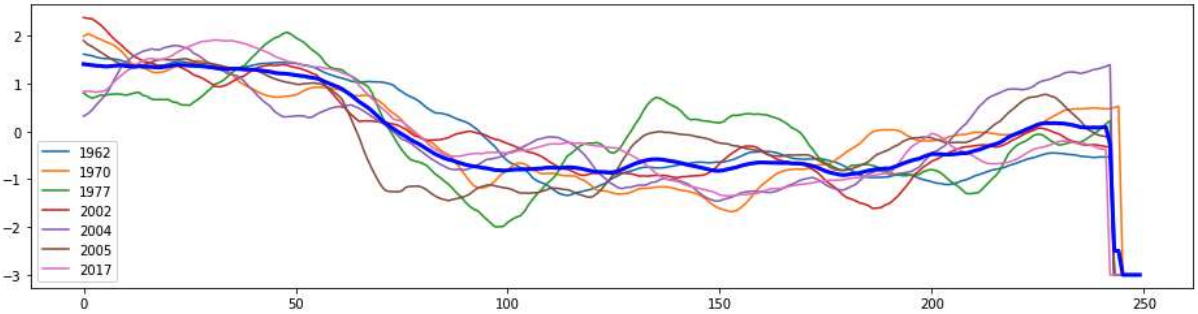
Secondo Cluster

Il secondo cluster è composto dalle Time Series degli anni in cui il prezzo per stock ha subito, nella prima fase, un andamento crescente, per poi subire una svalutazione dei prezzi nel quarto trimestre dell'anno. Fanno eccezione le Time Series rappresentanti gli anni 1968 e 2012, le quali hanno comunque avuto un andamento in linea con il centroide, ma che sono riuscite a terminare l'anno andando in positivo. Ma il 2012 è sicuramente degno di nota, infatti, la causa di questa chiusura positiva potrebbe essere legata al fatto che IBM in quell'anno ha costantemente generato una forte liquidità, un indicatore chiave dei risultati reali dell'impresa. Nel 2012 il free cash flow è stato di 18,2 miliardi di dollari, un record per IBM, concludendo il 2012 con 11,1 miliardi di dollari in contanti e titoli negoziabili.



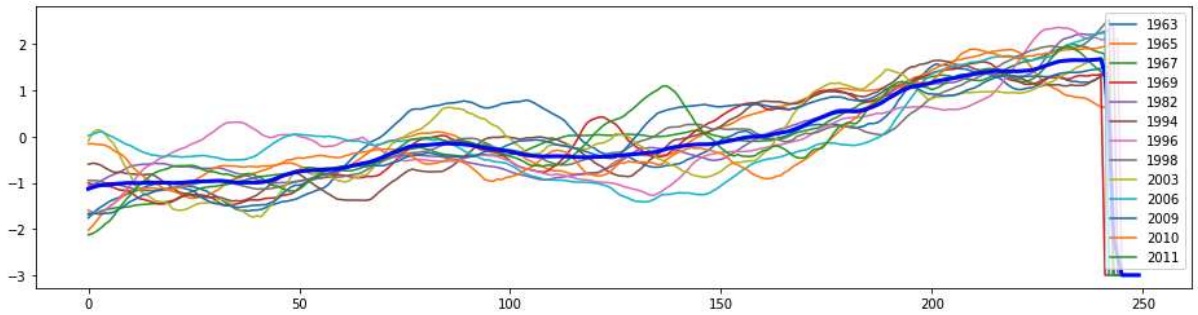
Terzo Cluster

Nel terzo cluster si assiste ad una svalutazione generale e, di conseguenza, ad una curva del centroide che tende al ribasso. Le Time Series, i cui stock hanno perso valore nell'anno, sono rappresentate dalla quasi totalità degli anni, ad eccezione del 2004, il quale è l'unico anno che è riuscito a terminare l'anno andando in positivo.
La causa di questa eccezione potrebbe ricadere sul fatto che Lenovo Group Limited (il marchio leader nel settore dei personal computer in Cina e in tutta l'Asia) e IBM fecero un annuncio nel mese di dicembre su un accordo, in base al quale, Lenovo acquistò la IBM Personal Computing Division per formare il terzo più grande business dei pc del mondo, portando ad una improvvisa impennata nel valore delle azioni.



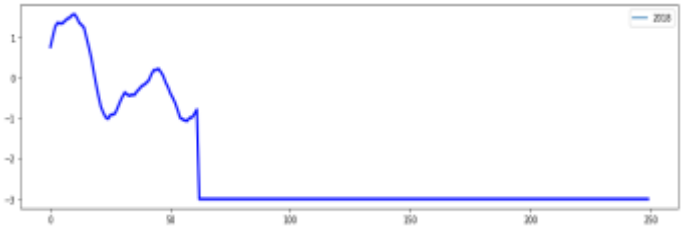
Quarto Cluster

Questo è il cluster che conta al suo interno il più grande numero di anni rispetto agli altri cluster, mostrando abbastanza visibilmente come, nonostante i tanti anni raggruppati, questi siano caratterizzati da una stretta vicinanza con il centroide.
Quest'ultimo è crescente, infatti, in tutti gli anni si registra una chiusura del prezzo delle azioni maggiore rispetto al prezzo di apertura delle stesse.



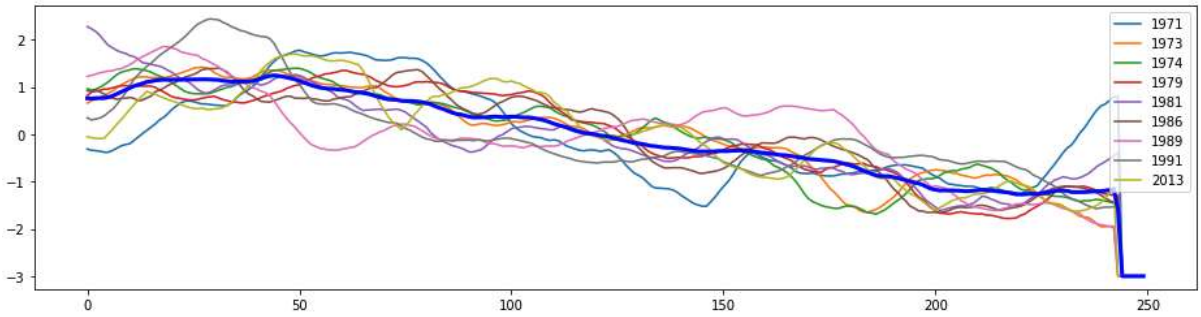
Quinto cluster

In questo cluster abbiamo come unica Time Series quella del 2018, un isolamento il cui motivo è giustificato dalla non completezza dei dati raccolti. La capacità del K-means di generare questo particolare cluster è proprio uno dei motivi per il quale è stato preferito agli altri metodi di clustering.



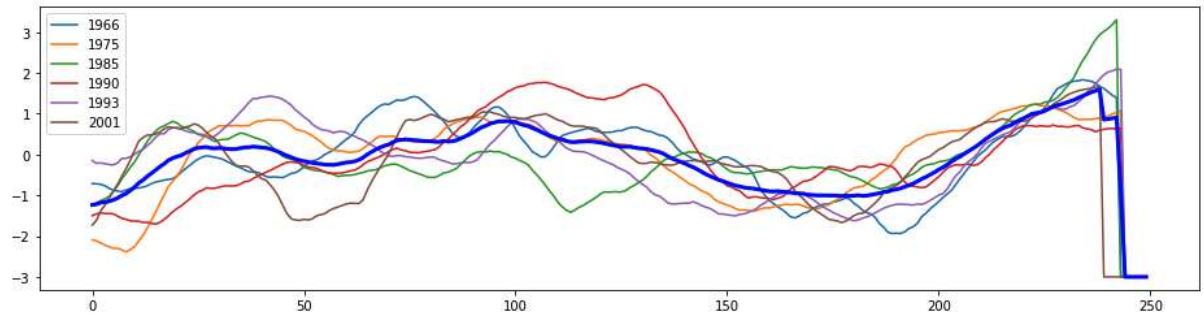
Sesto cluster

A differenza di quanto visto nel quarto cluster, in questo cluster il centroide è in discesa. Le Time Series dei singoli anni contano tutti una perdita del prezzo delle azioni a fine anno, ad eccezione del 1971, il quale conta invece una chiusura in positivo dei prezzi azionari, dopo aver comunque assistito mediamente, tra alti e bassi, ad un anno di svalutazione.
L'impennata finale dei prezzi di chiusura del 1971, potrebbero avere delle giustificazioni più che valide. Basti pensare che questo è infatti l'anno in cui i computer IBM aiutano a guidare gli atterraggi sulla Luna Apollo 14 e 15 e a migliorare le foto scattate da Mariner 9 (la prima navicella spaziale ad orbitare su Marte).



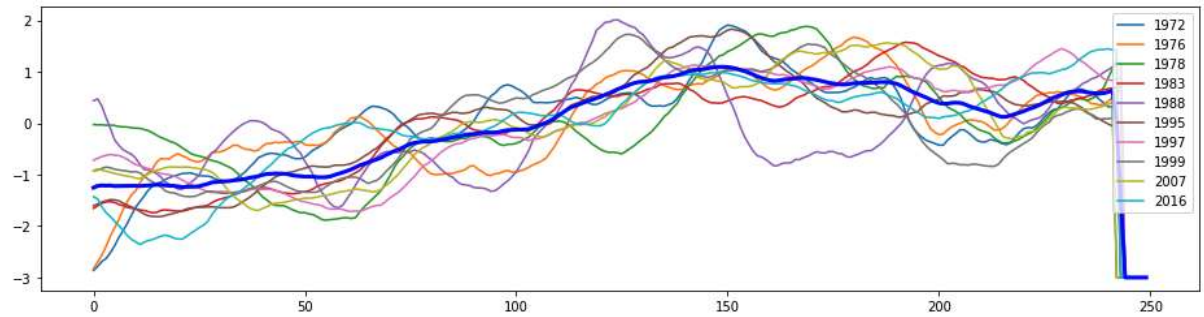
Settimo cluster

La caratteristica del settimo cluster è dettata dal fatto che le Time Series hanno un andamento oscillante e abbastanza discostante tra di loro. Dal centocinquantesimo giorno fino alla chiusura dei mercati, però, la distanza tra i vari anni ed il trend diminuisce drasticamente. Infine, tutti gli anni chiudono in positivo. Particolare attenzione va però riposta nel 1985, l'anno in cui si registra la più alta tra le chiusure dei prezzi di tutte le Time Series. Questo è l'anno in cui l'IBM decise di avviare un programma volto ad aiutare le persone con disabilità ad utilizzare i prodotti IBM, creando un Centro Nazionale di Supporto.



Ottavo cluster

In quest'ultimo cluster abbiamo notato un trend crescente per le Time Series, la maggior parte di esse hanno avuto una crescita costante nei primi tre trimestri dell'anno, per poi assestarsi nell'ultimo trimestre intorno al valore di chiusura di fine anno per stock.
Dopo il 150 giorno di raccolta dei prezzi, nel 1988 si registra un forte calo degli stessi. Il motivo riconducibile a tale grossa svalutazione potrebbe essere che IBM, in quello specifico periodo, decise di eliminare gradualmente le attività di produzione dei siti di Boca Raton in Florida e Tucson in Arizona.



Conclusioni

Riassumendo, il K-Means ha suddiviso le Time Series in gruppi di serie crescenti, decrescenti o altalenanti intorno alla media.
Per poter avere la certezza che il clustering sia stato effettuato con successo, avremmo dovuto esaminare più in dettaglio i risultati ottenuti dall'analisi.
Possiamo però concludere che le Time Series raggruppate nei cluster seguono lo stesso andamento, nonostante ci siano alcuni cluster che mostrano delle oscillazioni dei prezzi abbastanza evidenti.

Sequential Pattern Mining

Attraverso quest'ultima analisi si cerca di scovare, all'interno delle Time Series, dei pattern, ovvero delle sequenze del prezzo delle azioni di IBM. Tali sequenze dovranno poi presentarsi come dei *"motif"*, ovvero come delle sottosequenze di valori contigui la cui lunghezza dovrà essere di almeno 4 giorni.
Per scovare i *motif*, attraverso l'algoritmo *Generalized Sequential Pattern (GSP)*, è stato necessario suddividere la lunga Time Series annuale in Time Series mensili, normalizzando i valori come nel clustering. Successivamente le Time Series mensili sono state discretizzate in intervalli di uguale ampiezza, ognuno associato ad una lettera dell'alfabeto. Per poter scovare il numero ideali di intervalli sono stati effettuati vari tentativi. In ogni tentativo è stato dapprima utilizzato il GSP per individuare i pattern frequenti e in seguito sono stati esclusi quelli che non soddisfacevano il vincolo maxgap=1, mentre per quelli che lo soddisfacevano è stato ricalcolato il valore del supporto.
Il primo numero di intervalli provato è stato 6. Tale valore ha chiaramente permesso di individuare molti *motif* dall'alto supporto, tuttavia i risultati ottenuti sono stati considerati poco significativi, dal momento che i *motif* ottenuti erano quasi tutti una successione di valori costanti e gli intervalli ottenuti dalla discretizzazione erano troppo ampi (di ampiezza circa 1).
Per queste ragioni il secondo tentativo è stato effettuato utilizzando un numero di bins pari a 12, che ha prodotto intervalli di ampiezza circa 0.5. Di seguito sono elencati i primi 10 *motif* ottenuti, ordinati secondo il valore del supporto.

{['h'], ['i'], ['i'], ['i']}, supp=105
{['f'], ['e'], ['e'], ['e']}, supp=97
{['e'], ['e'], ['e'], ['e']}, supp= 94
{['i'], ['i'], ['i'], ['h']}, supp= 91
{['h'], ['h'], ['h'], ['h']} , supp=90
{['h'], ['h'], ['i'], ['i']}, supp=90
{['e'], ['e'], ['e'], ['f']}, supp=86
{['e'], ['f'], ['f'], ['f']}, supp=84
{['i'], ['i'], ['h'], ['h']}, supp=83
{['f'], ['f'], ['e'], ['e']}, supp=82

Si può notare che anche in questo caso i *motif* ottenuti sono principalmente costanti oppure nell'intero *motif* sono presenti due intervalli consecutivi, entrambi sintomi di una discretizzazione troppo grossolana.
Perciò abbiamo deciso di alzare ulteriormente il numero di bins, portandolo 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 supp.

{['j'], ['k'], ['k'], ['k']}, supp=56
{['g'], ['g'], ['f'], ['f']}, supp=53
{['k'], ['k'], ['l'], ['l']}, supp=52
{['g'], ['g'], ['g'], ['g']}, supp=51
{['g'], ['g'], ['g'], ['f']}, supp=49
{['f'], ['f'], ['f'], ['f']}, supp=48
{['f'], ['g'], ['g'], ['g']}, supp=47
{['g'], ['f'], ['f'], ['f']}, supp=47
{['k'], ['k'], ['j'], ['j']}, supp=45
{['f'], ['f'], ['f'], ['g']}, supp=44

Si può notare come la discretizzazione più fitta abbiamo diminuito drasticamente i valori dei supporti, infatti il *motif* più frequente si verifica soltanto nell'8% delle 676 Time Series. Tuttavia, è possibile constatare come il valore delle azioni tenda ad essere stabile quando il valore delle azioni è un valore lontano sia dal massimo sia dal minimo valore mai rilevato (il 'j', ad esempio, corrisponde all'intervallo [0.25, 0.66]). Per individuare un *motif* "meno costante" bisogna abbassare il valore del supporto fino al

valore 40, per cui si ottiene il *motif* {['k'], ['l'], ['k'], ['k']}, che rappresenta un innalzamento nel valore delle azioni seguito da una immediata ricaduta. Per ottenere un *motif* contenente tre intervalli differenti bisogna aspettare supp=33 (5% delle Time Series), per cui si ottengono i *motif* {['j'], ['k'], ['k'], ['l']}, {['l'], ['k'], ['k'], ['j']} e {['l'], ['l'], ['k'], ['j']}, che rappresentano una salita del valore delle azioni nel primo caso e una discesa negli altri due casi.

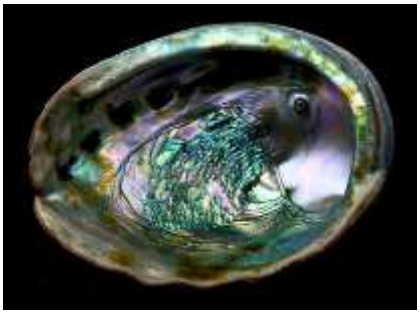
Conclusioni

Riteniamo che il sequential pattern mining non abbia ottenuto risultati molto significativi, dal momento che se si vogliono ottenere *motif* dall'elevato supporto è necessario effettuare una discretizzazione troppo grossolana, tuttavia se si utilizza un maggior numero di intervalli nella discretizzazione si ottengono valori del supporto inferiori al 10%.

Abalones

Classification

Haliotis è un genere di molluschi gasteropodi marini, comunemente noti come aliotidi, orecchie di mare o abaloni ed è l'unico genere della famiglia Haliotidae.
L'età di questi molluschi viene determinata attraverso un taglio della conchiglia effettuato sul cono, il quale la colora e permette di contare il numero di anelli attraverso un microscopio. A causa del lungo tempo richiesto per fare questa analisi, vengono quindi usate altre misure, più semplici e rapide, per predire l'età del mollusco, come la costruzione di un classificatore basato su alcune caratteristiche fisiche della conchiglia.



Il data set iniziale è costituito da 4177 esempi e 9 attributi:

- Sesso: attributo nominale che indica se il sesso del mollusco è un maschio, una femmina o un infante.
- Lunghezza della conchiglia.
- Diametro della conchiglia (perpendicolare alla lunghezza).
- Altezza della conchiglia.
- Peso complessivo della conchiglia con all'interno il mollusco.
- Peso del mollusco (senza la conchiglia).
- Peso delle viscere (dopo aver fatto sanguinare completamente il mollusco).
- Peso della conchiglia (dopo aver levato il mollusco).
- Anelli: aggiungendo 1.5 al numero di anelli si ottiene l'età della conchiglia in anni.

I metodi di classificazione scelti per l'analisi del data set sono: Support Vector Machine, Artificial Neural Network e alcuni Ensemble Methods.
Alcune modifiche sui dati del dataset sono state necessarie per l'applicazione di tali algoritmi:

- Eliminazione di tutti i record associati agli infanti (attributo "sex" = "I", corrispondente a 1342 molluschi)
- Trasformazione dell'attributo *"sex"* da nominale {M, F} in binario {0, 1}
- Discretizzazione dell'attributo *"rings"* in due classi

La discretizzazione dell'attributo *"rings"* ci ha costretti ad analizzare la distribuzione dello stesso per poter definire il range degli intervalli. Si è quindi deciso di utilizzare la mediana come parametro discretizzante, dividendo l'attributo nelle due classi 0-10 e 11-30 (54% dei record la prima classe e 46% la seconda) ed ottenendo così un ottimo bilanciamento.

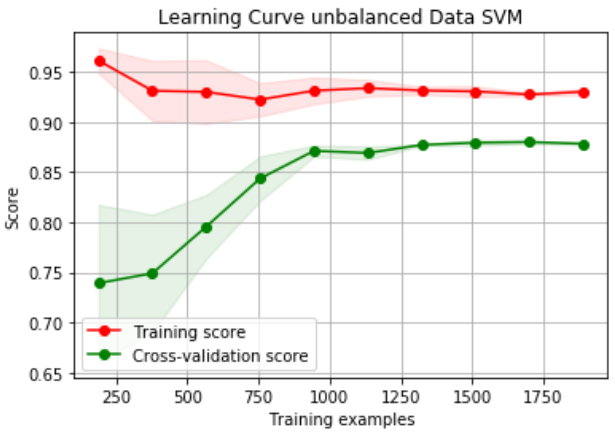
Support Vector Machine

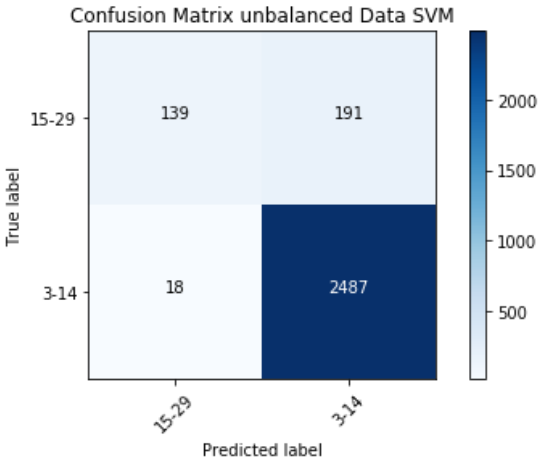
Il primo problema affrontato per questo tipo di classificatore si è incentrato sulla scelta dei parametri da utilizzare. A tal fine sono state effettuate delle *Grid Search* al fine di cercare quali sarebbero stati i migliori iperparametri. Sulla sinistra sono riportati i 3 migliori risultati.

```
Model with rank: 1
Mean validation score: 0.733 (std: 0.015)
Parameters: {'C': 5, 'gamma': 5, 'kernel': 'rbf'}

Model with rank: 2
Mean validation score: 0.729 (std: 0.017)
Parameters: {'C': 10, 'gamma': 5, 'kernel': 'rbf'}

Model with rank: 3
Mean validation score: 0.728 (std: 0.021)
Parameters: {'C': 20, 'gamma': 5, 'kernel': 'rbf'}
```

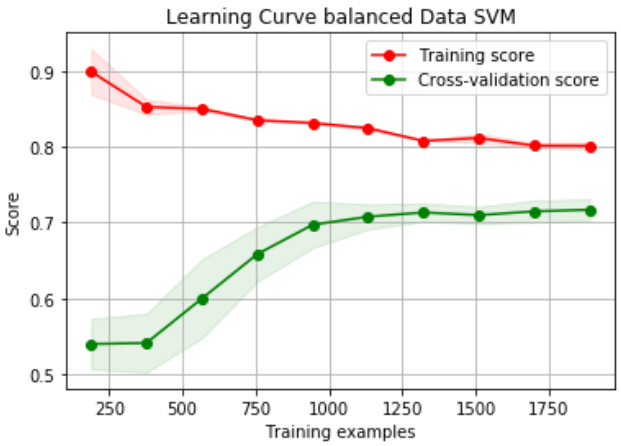
Successivamente abbiamo deciso di effettuare tre diverse prove sul bilanciamento delle classi:

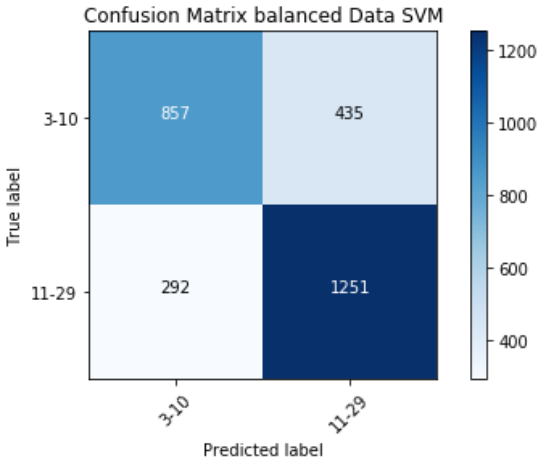
- Non bilanciare le classi ha prodotto l'88% di accuratezza e una deviazione standard di circa 0.02.



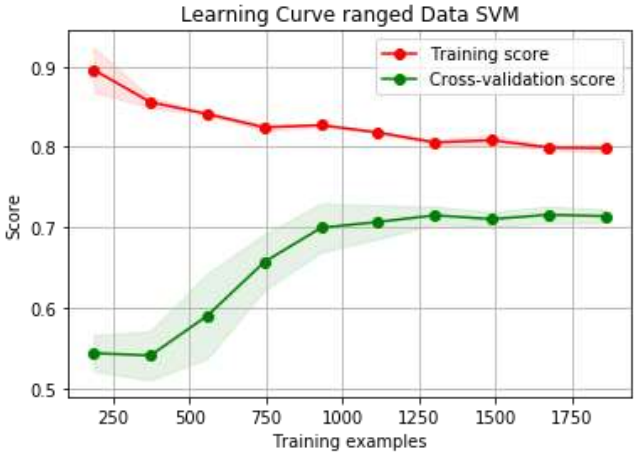


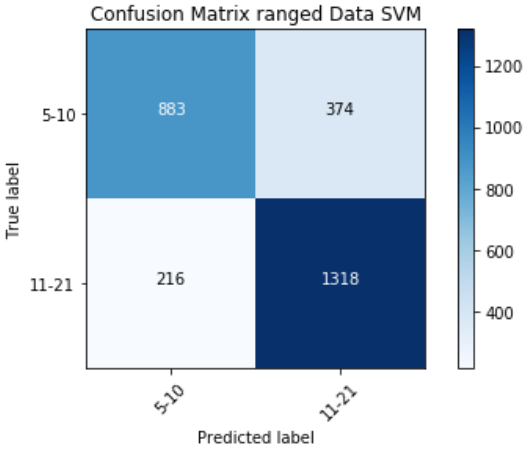
- Bilanciando le classi il livello di accuratezza è sceso a 70%, ma la deviazione standard è salita di 0.02 arrivando a circa 0.04.



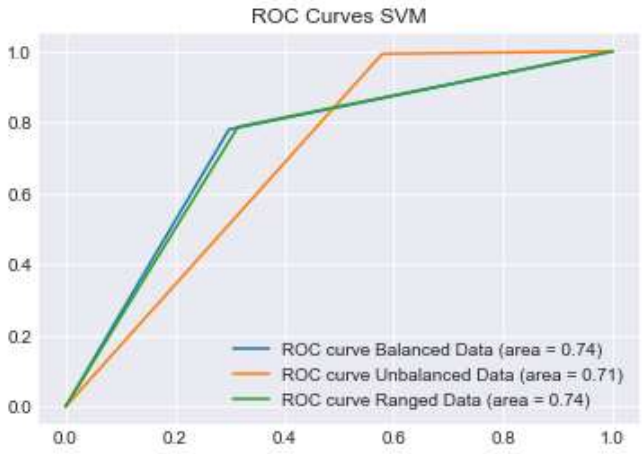


- Infine, si è deciso di spostare il range nella zona dove sono presenti più record e discretizzare, quindi, all'interno delle classi "5-10" e "11-20", ottenendo il 78% di accuratezza e una deviazione standard di circa 0.05.





I risultati sono stati ottenuti utilizzando i parametri ***"C"*** e ***"γ"*** settati a 5, come suggerito dalla *Grid Search*. Infine, nonostante il valore di accuratezza ottenuto dalle classi non bilanciate possa essere fuorviante, questo non può ritenersi soddisfacente per decretare la superiorità di un modello rispetto ad un altro.



Infatti, l'aver diminuito il range della classe rings ha prodotto dei risultati quasi identici, ma non migliori rispetto alla discretizzazione ottenuta attraverso le classi bilanciate, la cui curva è di pochissimo migliore rispetto alla concorrente. Tale risultato è osservabile anche attraverso la figura a sinistra.

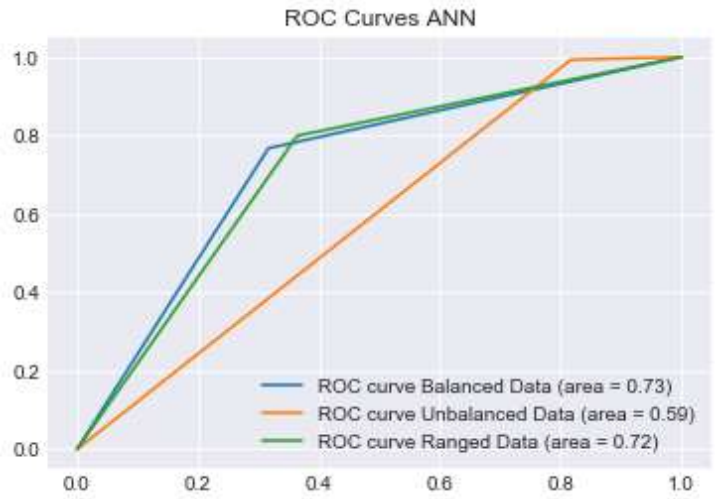
Secondo tale modello, all'aumentare dei parametri ***"C"*** e ***"γ"*** questo stesso peggiorerebbe, ma diminuendo gli stessi, settandoli ad 1, il risultato non cambia di molto, motivo per il quale non sono stati cambiati.

Artificial Neural Network

La rete neurale presa in esame è stata impostata con 3 layer nascosti con 5 nodi ciascuno e un solver "LBFGS". Come fatto precedentemente, anche in questo caso abbiamo deciso di analizzare 3 casi di bilanciamento di classi differenti:

- Classi non bilanciate con l'86% di accuratezza e deviazione standard di circa 0.03.
- Classi bilanciate con il 73% di accuratezza e deviazione standard di circa 0.06.
- Classi con diminuzione di range a "5-10" e "11-20", le quali hanno prodotto il 71% di accuratezza e una deviazione standard di circa 0.05.

Esattamente come nel caso del SVM, anche qui il miglior risultato lo si ottiene attraverso l'applicazione della rete neurale al data set bilanciato, ottenendo un'accuratezza del 20% superiore rispetto al caso del data set non bilanciato. Anche il data set con riduzione del range ha prodotto degli ottimi risultati, ma sempre inferiori rispetto al data set bilanciato (anche se di poco).



Gli iperparametri utilizzati per la rete neurale sono stati trovati attraverso una GridSearch, permettendoci di scegliere i migliori:

- alpha = 0.001
- tasso di apprendimento = 0.01
- momentum = 0.1
- solver = lbfgs

Concludendo, a livello strutturale, possiamo dire che i modelli migliori possiedono un solo layer interno con 2 o 4 nodi ed utilizzano la logistica come funzione di attivazione.

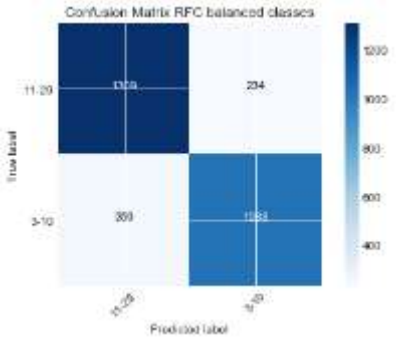
Ensamble methods

I metodi Ensamble utilizzati sono stati: Random Forest, AdaBoost e Balanced Bagging.

Random Forest Classifier

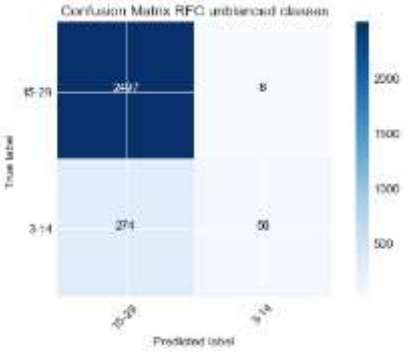
Il primo passo di questa analisi si è incentrato su come ottenere una suddivisione binaria di valori della colonna "*rings*". Si è quindi scelto di tagliare al valore 10, ottenendo una partizione in "3-10 = 0" e "11-29=1", che si è dimostrata essere la più bilanciata tra tutte. La scelta successiva è ricaduta sulla scelta del numero di alberi da far generare al modello del RandForest, una scelta non facile come nei classificatori precedenti, in quanto, in questo caso c'è l'impossibilità da parte del modello di poter utlizzare una GridSearch, a causa dell'alto numero di combinazioni possibili dei valori degli iperparametri da utilizzare. La scelta del numero di questo parametro, dopo vari tentativi, ci ha portati a capire come all'aumentare di questo aumentasse anche l'accuratezza e la varianza media del modello stesso. Il problema, però, è che la scelta di numero troppo grande avrebbe comportato un dispendioso costo computazionale e quindi in tempo. La scelta è quindi ricaduta sulla generazione di 150 alberi. Una volta fissato quest'ultimo parametro, è stata effettuata una *random search* (prima figura del paragrafo) per individuare gli altri

```
Model with rank: 1
Mean validation score: 0.723 (std: 0.023)
Parameters: {'bootstrap': True, 'class_weight': 'balanced', 'criterion':
'entropy', 'max_depth': 11, 'max_features': 6, 'min_samples_leaf': 19, 'm
in_samples_split': 36}
```



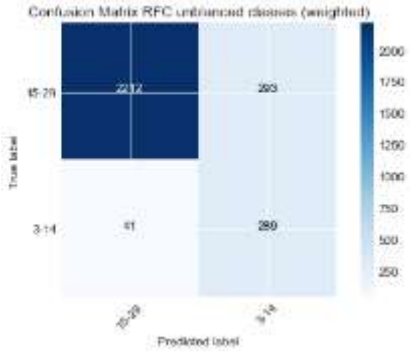
migliori parametri da utilizzare, portandoci al risultato nella figura prima figura del paragrafo.
Dopo aver inserito i parametri mancanti, il risultato ha fatto emergere la capacità del classificatore di riuscire ad imparare su entrambe le classi mantenendo un'accuratezza di circa il 20% rispetto ad un trivial classifier. Tale risultato è confermabile anche attraverso la *confusion matrix* (figura in alto).

Per dimostrare la stretta necessità di avere un dataset bilanciato, abbiamo deciso di fare la stessa identica operazione precedente, con la differenza che, in questo caso, la scelta su dove tagliare la classe *rings_disc* è ricaduta su 14, ottenendo "3-14 = 0" e "15-29 = 1", sbilanciando di molto la classe. Il risultato ha portato ad una elevata accuratezza del modello, ma un'accuratezza fittizia in quanto, in termini di performance, è inferiore a quella di un trivial classifier.

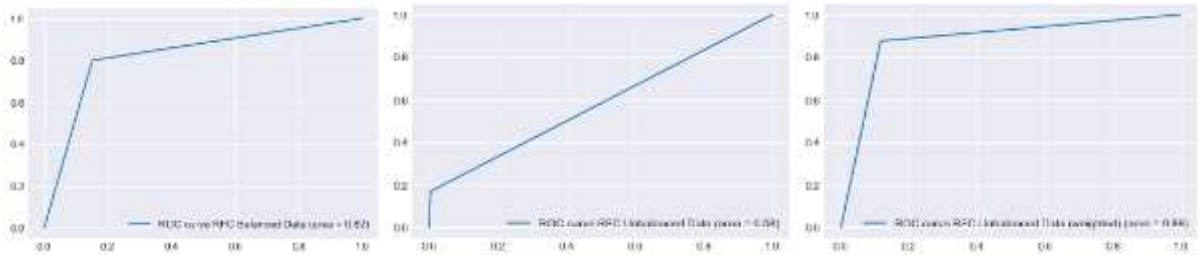


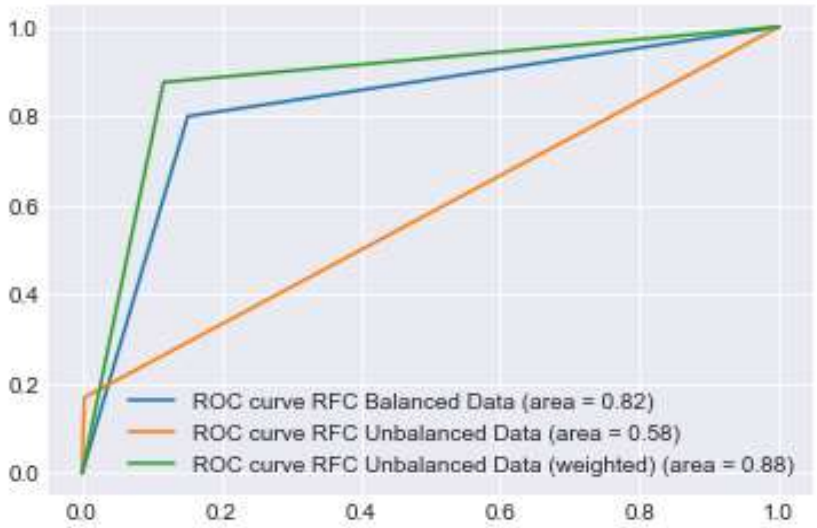

```
Model with rank: 1
Mean validation score: 0.888 (std: 0.001)
Parameters: {'bootstrap': True, 'class_weight': None, 'criterion': 'gin
i', 'max_depth': 12, 'max_features': 4, 'min_samples_leaf': 14, 'min_samp
les_split': 29}
```

Per ovviare a questa brutta soluzione, abbiamo optato per l'aggiunta del parametro *class weight*, il quale ci ha permesso di rimediare al forte sbilanciamento delle classi attribuendo un peso a ciascuna classe sulla base della loro frequenza all'interno del data set. Il risultato ha portato ad un miglioramento netto rispetto al modello con le classi non pesate (figure a sinistra).



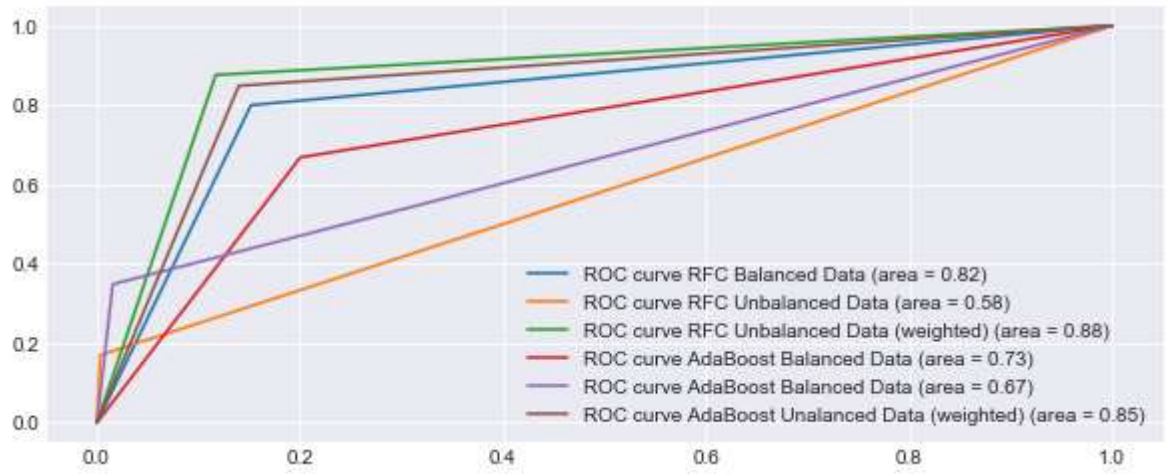
Concludendo questa prima parte, vengono di seguito messe a confronto i 3 modelli analizzati, attraverso la quale emerge come miglior classificatore quello relativo al dataset sbilanciato pesato:





AdaBoost Classifier

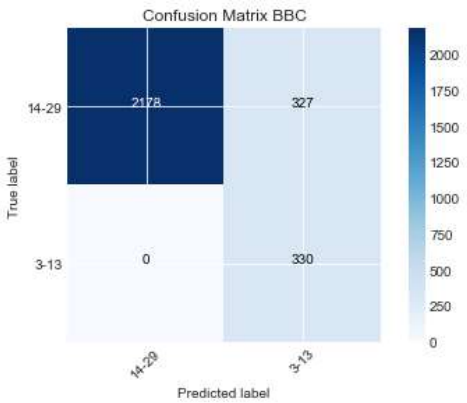
Il secondo modello analizzato è l'AdaBoost, attraverso il quale, utilizzando le stesse identiche operazioni effettuate per il modello RandomForest, siamo giunti alla conclusione che è un modello peggiore. La differenza principale risiede nell'accuratezza delle classi bilanciate dei due modelli, infatti, nel RandomForest è dell'82% mentre nell'AdaBoost è del 73%, come osservabile dalla figura sottostante:



Infine, è possibile osservare come, anche nel caso delle classi sbilanciate pesate, il RandomForest riesca ad ottenere una migliore accuratezza rispetto al concorrente.

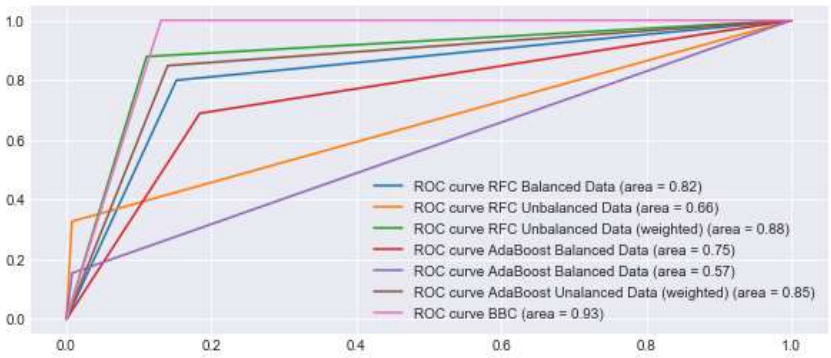
Balanced Bagging

Lo scopo di questo ultimo modello analizzato, è quello di effettuare un bilanciamento interno del data set attraverso un algoritmo chiamato appunto Balanced Bagging. Proprio per via di questa sua funzionalità, il modello è stato applicato al data set con le classi sbilanciate, ottenendo come risultato una moltitudine di classificatori con una accuratezza media superiore rispetto a quella dei due modelli analizzati in precedenza.



Accuracy: 0.79012 (+/- 0.15364)

Attraverso la confusion matrix (figura a sinistra) è possibile osservare come questo modello abbia la tendenza a fare previsioni perfette sulla classe di minoranza.
Concludendo l'analisi sui 3 modelli presi in considerazione, vengono riportate in figura le curve ROC, attraverso la quale è possibile osservare come il classificatore con la performance migliore di tutti è il Balance Bagging, seguito dal RandomForest con le classi bilanciate, per poi terminare con il modello AdaBoost.



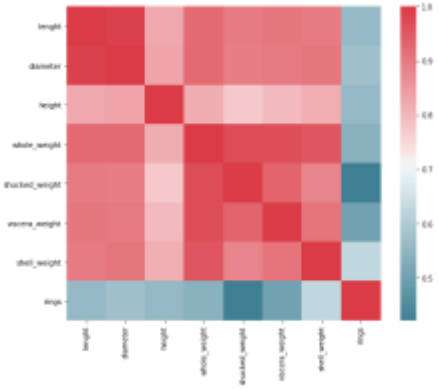
Outlier Detection

Con questo tipo di analisi si è cercato di identificare i record candidati ad essere dei possibili outlier e, per raggiungere tale obiettivo, è stato necessario analizzare le correlazioni tra gli attributi del dataset, al fine di definire il numero ottimale di dimensioni su cui trovare gli outlier.

	lenght	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
lenght	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
hucked_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

Tutti gli attributi della tabella presentano una correlazione altissima tra di loro, tutti ad eccezione dell'attributo "rings". Proprio per questo motivo abbiamo quindi deciso di lavorare su quattro attributi, diminuendo a quattro il numero delle dimensioni. Gli attributi scelti sono: "lenght", "height", "shucked_weight" e "rings". Lo scopo di questa riduzione è stato quello di usare il maggior numero possibile di dimensioni e allo stesso tempo riuscire ad ottenere la massima accuratezza per la definizione degli outlier.

A causa dell'alta correlazione tra gli attributi (figura a destra) abbiamo deciso di non affrontare l'analisi degli outliers con il depth-based, proprio per via del fatto che gli attributi sono molto concentrati nella stessa regione, lasciando nei bordi solo l'1% dei punti.
Anche l'algoritmo *deviation-based* è stato scartato in quanto presenta una complessità troppo elevata per poter effettuare dei calcoli.
Sempre per via dell'alta correlazione tra gli attributi, infine, è stato scartato anche l'algoritmo *high-dimensional*, in quanto non è stato necessario utilizzare tutte e 8 le dimensioni del data-set, ma solo una parte di questo.

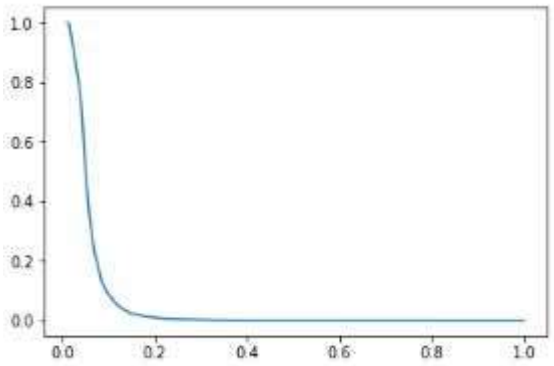


L'obiettivo primario dell'analisi è di individuare l'1% degli outlier e gli algoritmi scelti per identificare i possibili outlier sono stati i seguenti:

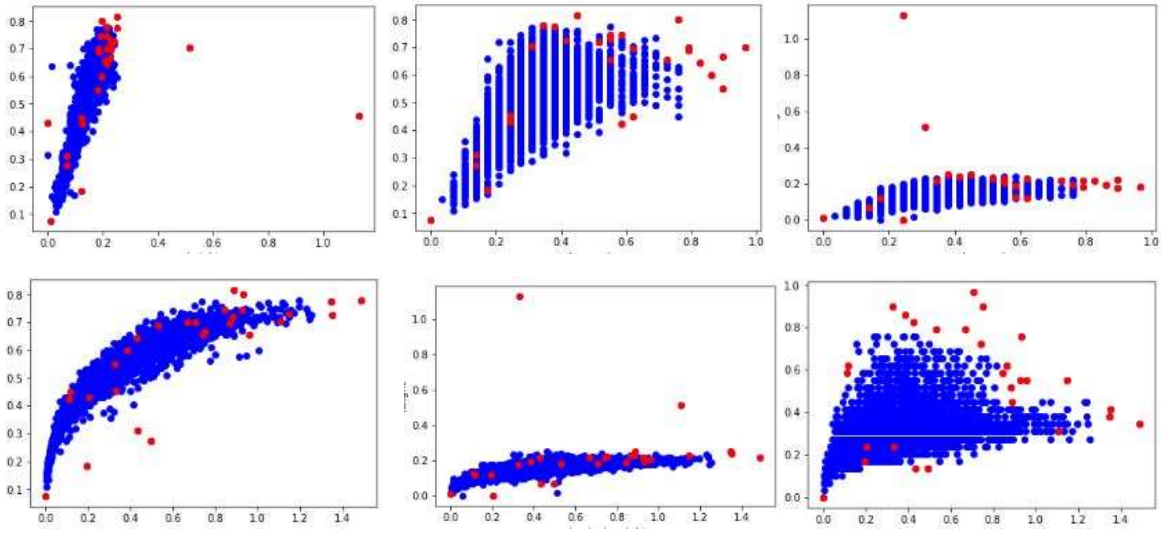
- DB-Scan
- Mahalanobis distance
- LOF

DBscan

L'obiettivo è trovare l'epsilon e il numero minimo di record necessario per l'individuazione dell'1% degli outlier, ovvero di 41 punti. Per la definizione dell'epsilon è stato applicato l'algoritmo con diversi minsample, andando a modificare, per ciascuno, l'epsilon tra 200 valori compresi tra 0.1 e 0.2. Il grafico ottenuto presenta sulle ordinate la percentuale di data set identificata come outlier e sull'ascisse l'espilon corrispondente.

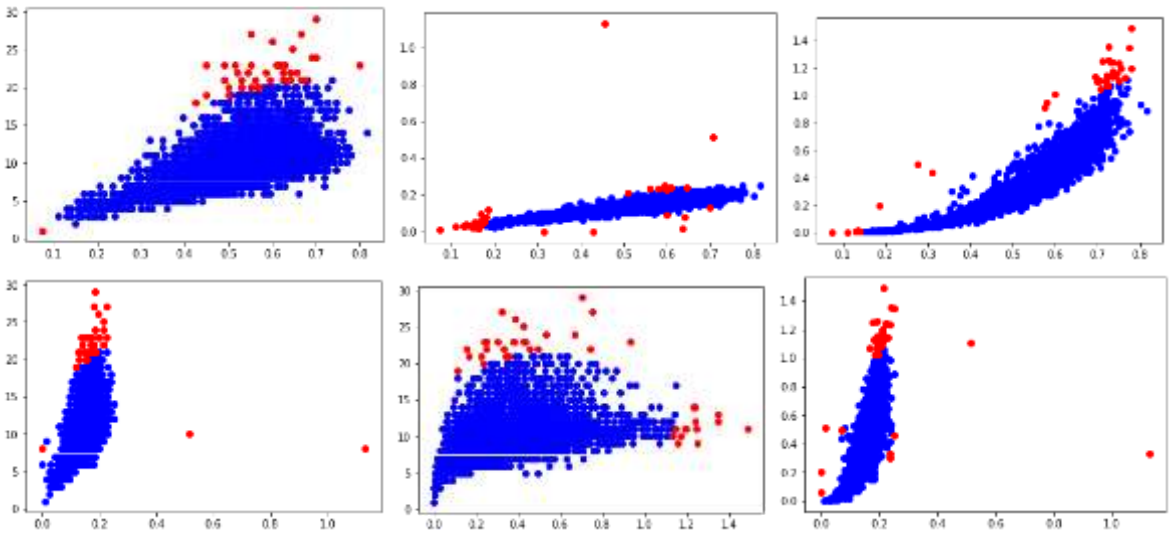


Il valore di epsilon trovato è di 0,21, utilizzando un min_sample di 5. In questo modo siamo riusciti nell'identificazione dell'1% degli outlier, come osservabile nelle immagini sottostanti, nelle quali gli outlier sono rappresentati in rosso.



Mahalanobis distance

L'algoritmo valuta la distribuzione dei dati del dataset per poi identificare gli outliers che si allontanano dalla stessa distribuzione. Anche in questo caso abbiamo cercato di avvicinarci il più possibile alla soglia dei 41 outlier. A tale scopo, abbiamo optato per una modifica della soglia tra l'1% e l'1,5%.



LOF

L'ultimo algoritmo scelto per l'analisi degli outliers è il LOF. Come già preannunciato precedentemente un algoritmo del tipo depth-based è poco efficiente nel nostro caso, a causa dell'alta densità del nostro dataset.
Una volta lanciato l'algortirmo, infatti, il risultato ricavato è stata l'individuazione del 10% degli outlier, un risultato poco soddisfacente ai fini della richiesta sollevataci. Quindi, l'alternativa è stata utilizzare la funzione *negative_outlier_factor* da scikit-learn, per valutare il grado di outlier: minore è il valore e maggiore sarà il grado di oulier.