Progetto di ingegneria della conoscenza 2023
UniBa: Dipartimento di Informatica

IMBALANCE LEARNING WHEN THE MACHINE LEARNS THE HARD WAY

REPOSITORY DEL PROGETTO

https://github.com/kozen88/ProgettoIcon2023

[In caso di dubbi, poco chiarezza della documenta o per maggiori informazioni e presa visione di tutti i risultati dello studio si consiglia la presa visione del repository il quale è strutturato in modo da contenere tutti i passaggi in modo chiaro di quanto è stato fatto durante lo studio, in particolare i notebook sono nella cartella source nella quale vi è un file contenenti informazioni per la visione dei notebook, la cartella data contiene il dataset iniziale e le versioni di ogni tecnica utilizzata che né hanno modificato il contenuto, infine nella cartella documentazione vi sono il suddetto documento, la presentazione in power point e la cartella “plot_and_figure” la quale raccoglie per modello addestrato e tecnica utilizzata tutti i risultati ottenuti dallo studio e che non è stato possibile mostrare in questo documento per questioni di spazio e per non appesantire la lettura.]

Diego Miccoli

Matricola 738735

mailto:d.miccoli13@studenti.uniba.it

ICON 6

Wednesday 13 September 2023Diego Miccoli

Progetto di ingegneria della conoscenza 2023
UniBa: Dipartimento di Informatica
Anno 2022-23

INDICE DEI CONTENUT

Wednesday 13 September 2023Diego Miccoli

INTRODUZIONE

Per il Progetto di ingegneria della conoscenza si è scelto di affrontare come principale tematica dello studio il machine learning è in particolare l'attenzione è ricaduta sull'apprendimento supervisionato in contesti ardui. L'idea scaturisce dal data set utilizzato contenete informazioni sui dipendenti di una azienda operante nel settore tecnologico e dalla sua peculiarità di contenere due classi su cui abbiamo basato l'apprendimento dei modelli di classificazione, che sono la classe dei dipendenti che richiedono le dimissioni e la classe dei dipendenti che continuano a lavorare con l'azienda. Il turn-over dei dipendenti è un problema che inevitabilmente colpisce qualsiasi azienda, indipendentemente dal contesto di lavoro, dal settore specifico e dalle dimensioni e importanza dell'azienda stessa, per cui riuscire ad avere una stima di quanti dipendenti in un dato momento possano lasciare l'azienda è un ottima base di partenza per le aziende che dovranno gestire le loro risorse umane e fare fronte alle carenze di staff, inoltre permetterebbe di organizzare campagne di assunzione nei periodi in cui si prevede una perdita eccessiva del personale tale da compromettere il lavoro e le attività commerciali.

Da questo punto di partenza quindi si è deciso di affrontare un classico problema del machine learning che è di attinenza quotidiana dato che un gran numero di problematiche oggi giorno affrontabili con l'aiuto dell'intelligenza artificiale presentano un problema di squilibrio dei dati tra le classi di appartenenza e questo problema è l'imbalance learning.

IMBALANCE LEARNING

L'apprendimento con sbilanciamento di classi, più conosciuto in ambito informatico con il nome inglese di "imbalance learning" o "class imbalance learning," è un'area di ricerca nell'apprendimento automatico la quale si occupa del problema della gestione dell'apprendimento da parte di un agente intelligente delle classi di un dataset le quali non sono distribuite in modo uniforme, cioè alcune classi hanno un numero molto maggiore di esempi rispetto ad altre. Questo sbilanciamento rende difficile l'addestramento di modelli di machine learning in quanto il modello tende a favorire la classe più numerosa a scapito della classe minoritaria, questo perché durante la fase di training il modello vedrà e imparerà a riconosce meglio e gestire le caratteristiche della classe maggioritaria poiché di essa vede più esempi comparati a quella minoritaria.

Nonostante queste difficoltà il problema è attualissimo in numerosi contesti in cui la capacità da parte di un agente intelligente di riuscire a discernere la classe minoritaria da quella maggioritaria con una buona percentuale di successo permetterebbe di risolvere molte delle sfide con le quali oggi l'intelligenza artificiale si confronta e che affliggono l'uomo e la società. Alcuni esempi di tale problematica sono:

- Frodi delle transazioni bancarie
- Previsione di attacchi hacker
- Diagnosi medica
- Sorveglianza dei processi di produzione aziendale
- Classificazione di mail spam
- Previsione di terremoti
- Rivelamento di anomalie nella rete

Questi sono solo alcuni delle problematiche che sono caratterizzate da uno sbilanciamento dei dati, ovvero per affrontare questi problemi i dati esistono e si raccolgono, ma il problema e che gli eventi per così dire positivi verso cui siamo interessati al riconoscimento sono raccolti in quantità inferiori poiché nella realtà quotidiana questi eventi si verificano sporadicamente rispetto alle situazioni opposte viste come di normalità.

TECNICHE PER AFFRONTARE L'IMBALANCE LEARNING

Per aiutare un agente intelligente ad apprendere nel migliore dei modi evitando di apprendere solo la classificazione della classe maggioritaria sono state sviluppate diverse tecniche tra le quali abbiamo:

- COST-SENSITIVE LEARNING
- UNDER SAMPLING
- OVER SAMPLING
- APPROCCIO MISTO DI CAMPIONAMENTO
- ENSEMBLE LEARNIG

Ogni una delle precedenti tecniche affronta il problema in modo diverso, ma tutte hanno il medesimo obiettivo, ottenere modelli di machine learning più equilibrati e capaci di riconoscere correttamente anche le classi minoritarie.

SCOPO DELLO STUDIO

Il nostro studio si focalizzerà sul mettere in atto tali tecniche e valutare criticamente i risultati ottenuti, essendo conscenti che con gli esperimenti che andremo a condurre non si ha l'obiettivo di andare a creare un nuovo modello commerciale di intelligenza artificiale che sia capace di risolvere il problema citato precedentemente e che quindi possa essere distribuito sul mercato. Il focus dello studio sarà invece capire come applicare nella maniera più corretta possibile in base alle mie personali conoscenze attuali nel campo del machine learning le tecniche con le quali affrontare il problema dell'imbalance learning e valutare in modo critico e ponderato i risultati che si otterranno. Con questo obbiettivo riportiamo brevemente i principali risultati raggiunti per ogniuna delle tecniche utilizzate e le metodologie utilizzate per addestrare i modelli e in seguito valutare il loro addestramento attraverso le curve di apprendimento e le metriche affini alla classificazione quali recall, precision, accuracy, f1-score, e le loro valutazioni calcolate con media e la media ponderata in base al numero degli esempi appartenenti alle classi e infine la matrice di confusione. I modelli utilizzati durante lo studio sono stati sottoposti tutti ad addestramento inizialmente con hold out al quale è seguita la fase di convalida incrociata per la conferma o meno dei risultati ottenuti con hold out ed infine si sono ottimizzati i modelli in base ai migliori parametri ed al miglior fold per la convalida incrociata trovato sperimentalmente. Tali modalità inerenti all'addestramento e all'ottimizzazione sono descritte in un apposito paragrafo di questo elaborato.

IL DATASET UTILIZZATO

Il dataset che abbiamo utilizzato per condurre lo studio è stato preso da un noto sito di intelligenza artificiale https://www.intelligenzaartificialeitalia.net/ ed è caratterizzato dall'avere un totale di dieci features descrittive di cui:

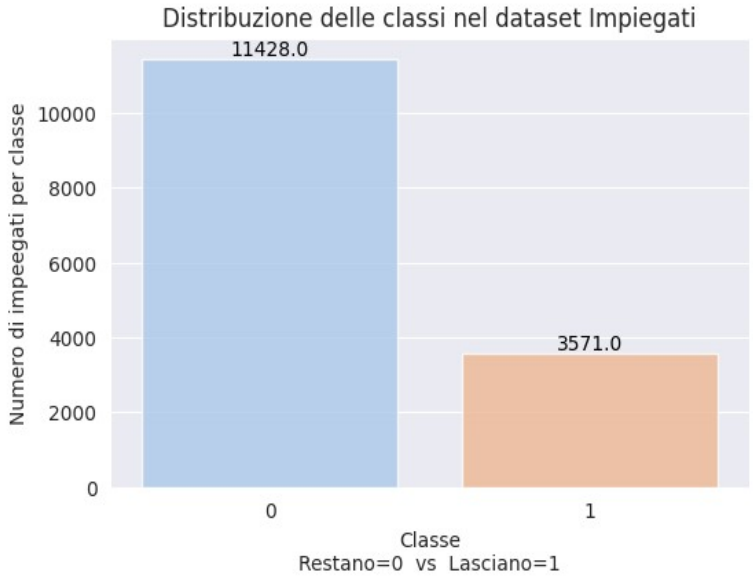
- **Features discrete:** numero progetti, ore medie mensili, tempo speso in azienda, incidenti sul lavoro, promozione negli ultimi 5 anni e richiesta di licenziamento.
- **Features continue:** livello di soddisfacimento e ultima valutazione.
- **Features categoriche:** salario e dipartimento.

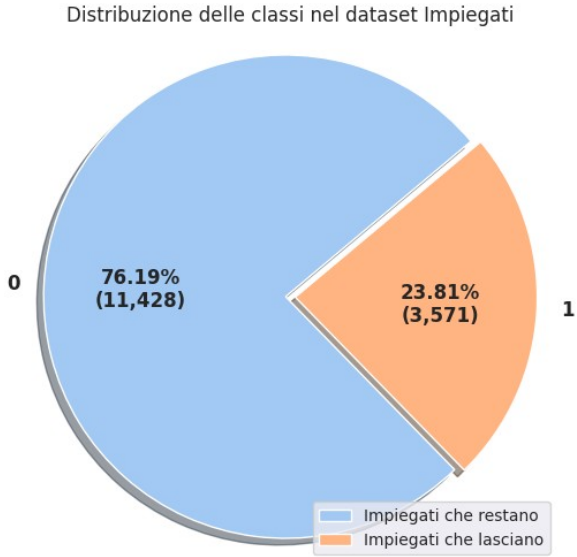
La feature target è richiesta di licenziamento. Una volta analizzato e capito a cosa le variabili si riferissero e le scale di misurazione utilizzate per raccogliere

tali dati si è cercato di andare più nel dettaglio delle caratteristiche del dataset in cerca di problematiche da risolvere prima di affrontare l'apprendimento dei modelli, di modo da ottenere il massimo dai dati e facilitare l'apprendimento dei modelli.

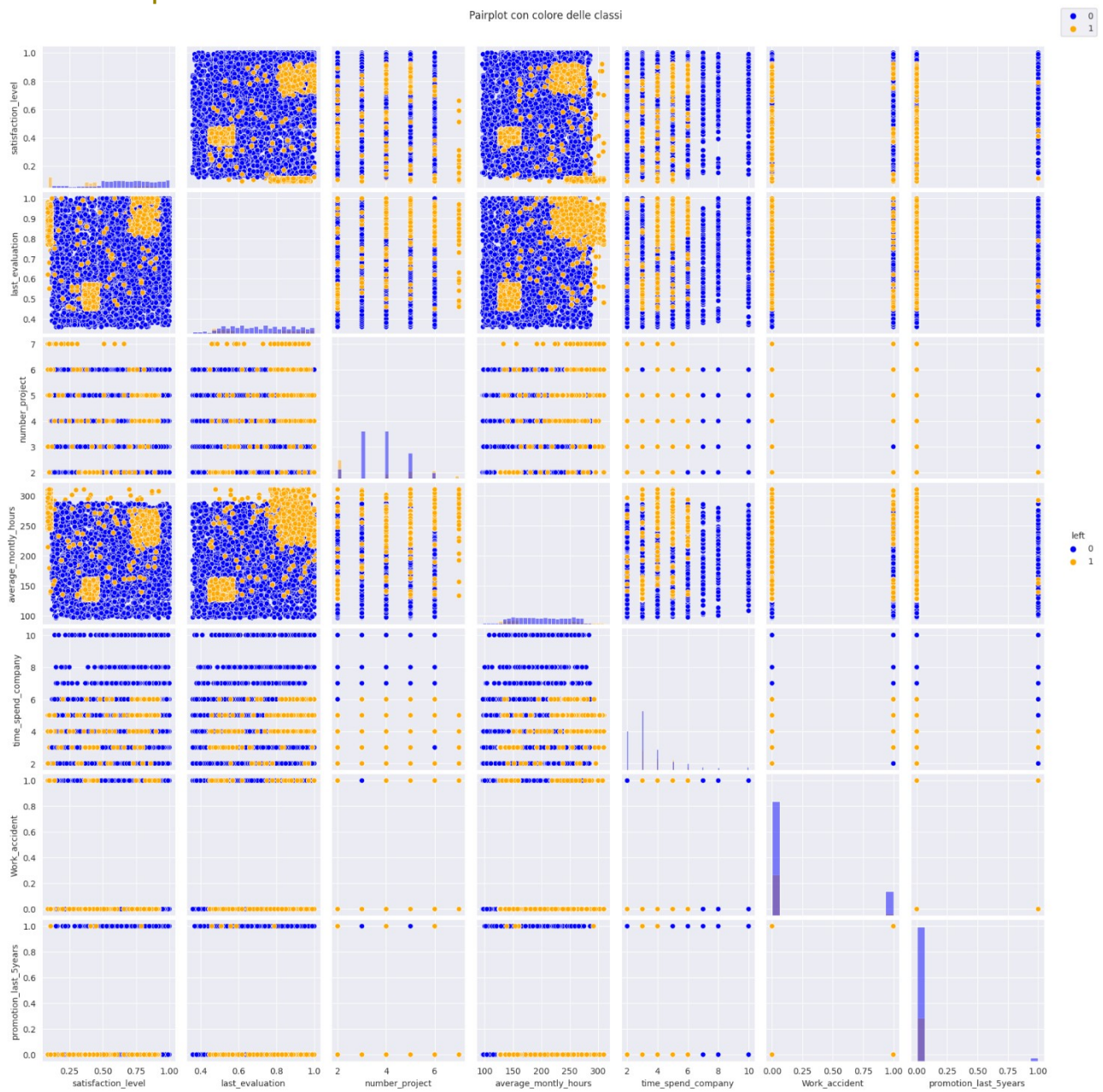
DATA EXPLORING & VISUALIZATION

Le prime operazioni effettuate sono state utilizzate a comprendere meglio la natura e la distribuzione dei dati con i quali stiamo lavorano e hanno portato ai seguenti risultati:

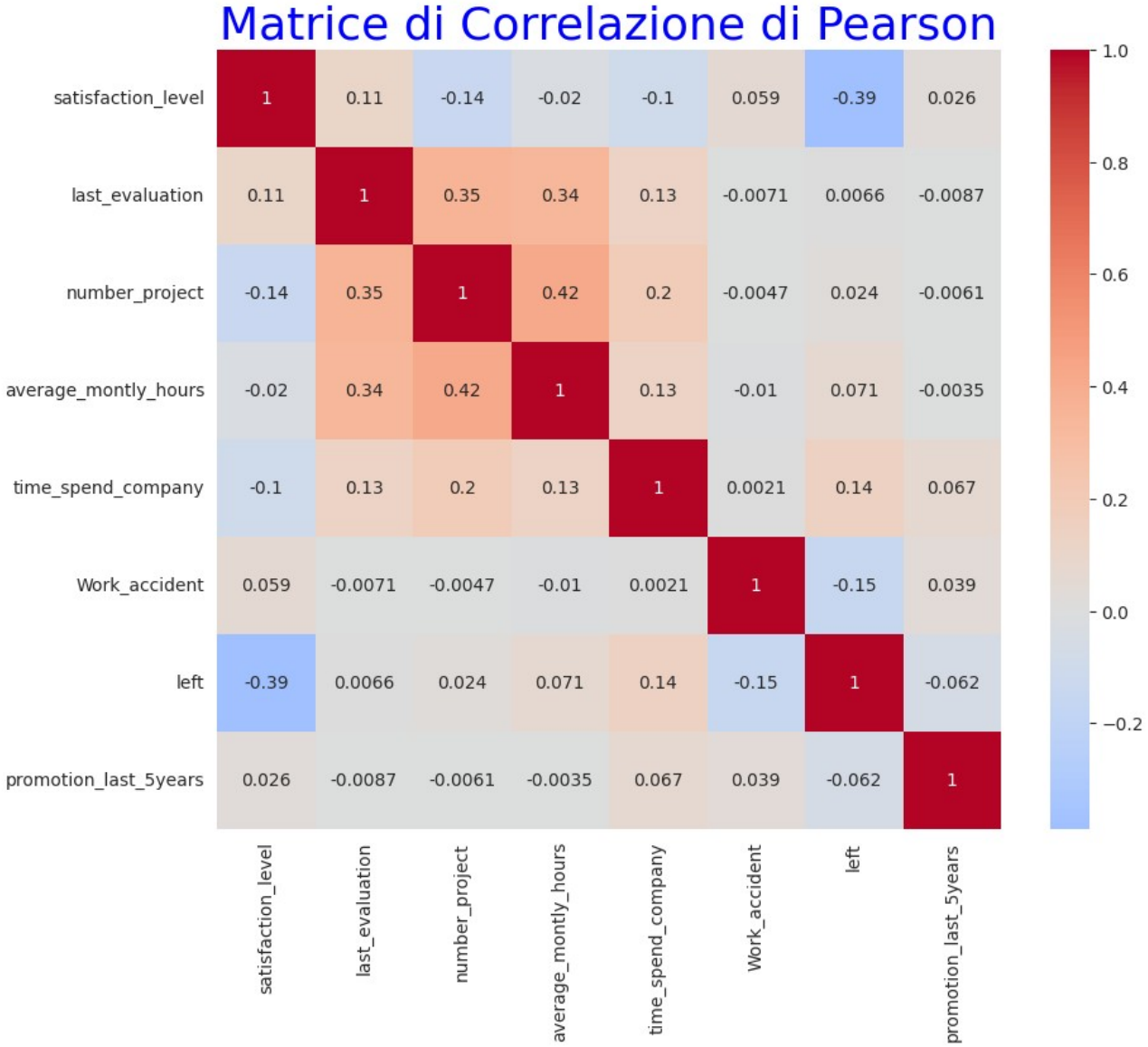




Da questi immagini capiamo che il nostro data set soffre di uno sbilanciamento che ha all'incirca un rapporto 3:1 a sfavore della classe minoritaria ovvero gli impiegati che lasciano il posto di lavoro. Abbiamo in seguito plottato un pair plot per capire e cercare di intravedere possibili separazioni dei dati lungo gli assi delle features prese a coppie per il confronto:

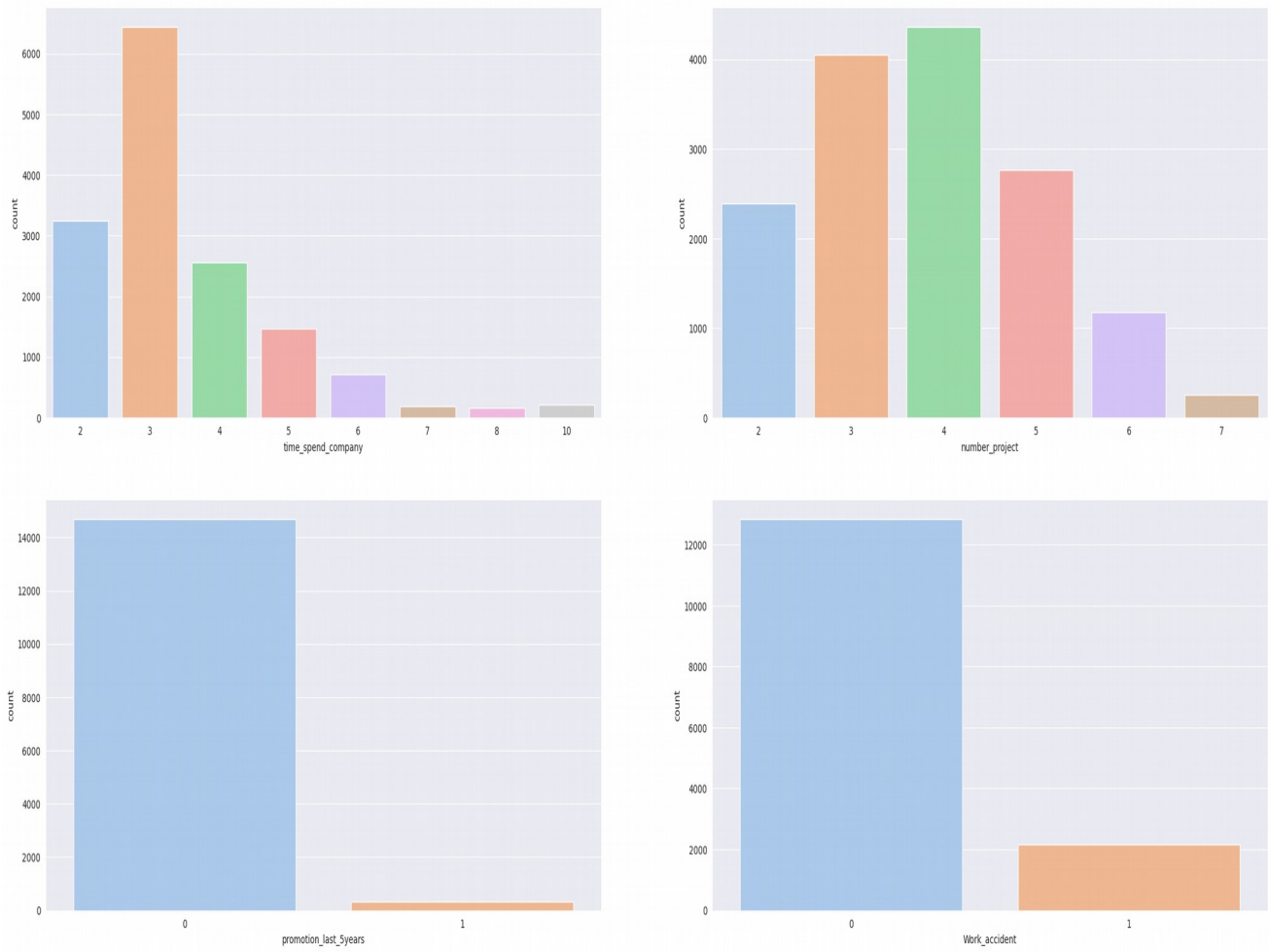


Dallo studio del pairplot ovvero la distribuzione degli esempi visti su di uno spazio cartesiano che prende in considerazione solo due features per volta per visualizzare come in base a tali features gli esempi si distribuiscano nello spazio cartesiano abbiamo appreso che ci sono feature che riescono a distribuire gli esempi in determinate zone dello spazio delle features e per tali ragioni crediamo che ci siano buone possibilità per riuscire a classificare gli esempi in base alle due classi di appartenenza, sfruttando maggiormente tali features del nostro dataset.

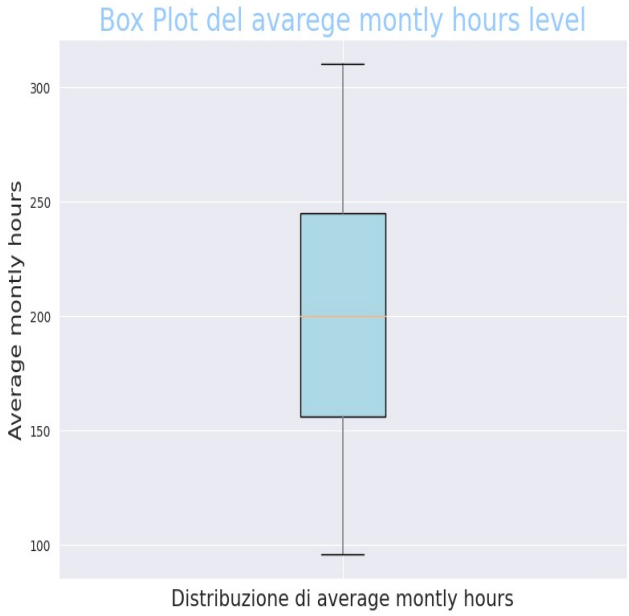


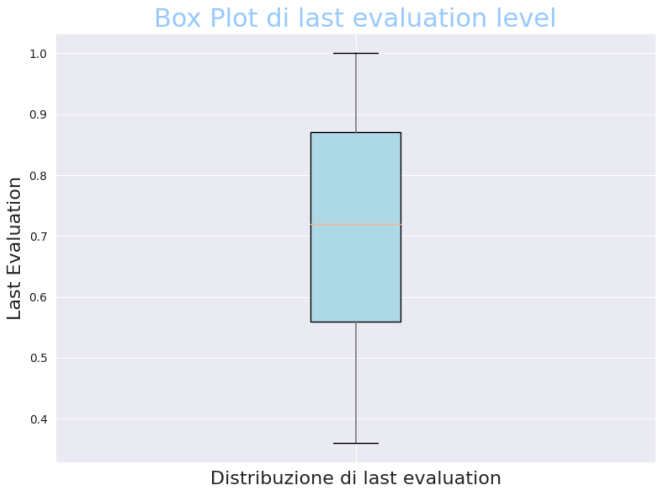
Usando il coefficiente di correlazione lineare di Pearson abbiamo dato conferma di quanto presupposto in precedenza ovvero che il problema di classificazione sia effettivamente affrontabile poiché ci sono features che riescono a clusterizzare gli esempi comuni alla stessa classe. Tali features sono quelle che mostrano una maggiore correlazione lineare nella matrice anche se c'è da dire che la correlazione e lieve-moderata.

Andiamo a dare un occhiata più vicino alle statistiche per ogni una delle features che stiamo utilizzando:



Dalle quali notiamo che le promozioni in azienda sono più uniche che rare e che stranamente c'è un significativo anche se non enorme livello di infortuni sul lavoro. Capiamo anche che la maggior parte dei dipendenti dell'azienda lavora in pianta stabile da tre anni e che passati i cinque anni i dipendenti della azienda fissi diminuiscono drasticamente cosa che fa pensare ad un abbastanza rapido turn over dei dipendenti. Per la gestione del carico lavorativo sembra essere più o meno ripartito equamente dal numero di progetti che vengono assegnati ad ogni dipendente.





I boxplot delle variabili continue ci hanno permesso di capire la distribuzione degli esempi del dataset e le seguenti solo le principali statistiche raccolte:

```
count    14999.000000
mean       201.050337
std         49.943099
min         96.000000
25%        156.000000
50%        200.000000
75%        245.000000
max        310.000000
Name: average_montly_hours, dtype: float64
```

```
count    14999.000000
mean         0.716102
std          0.171169
min          0.360000
25%          0.560000
50%          0.720000
75%          0.870000
max          1.000000
Name: last_evaluation, dtype: float64
```

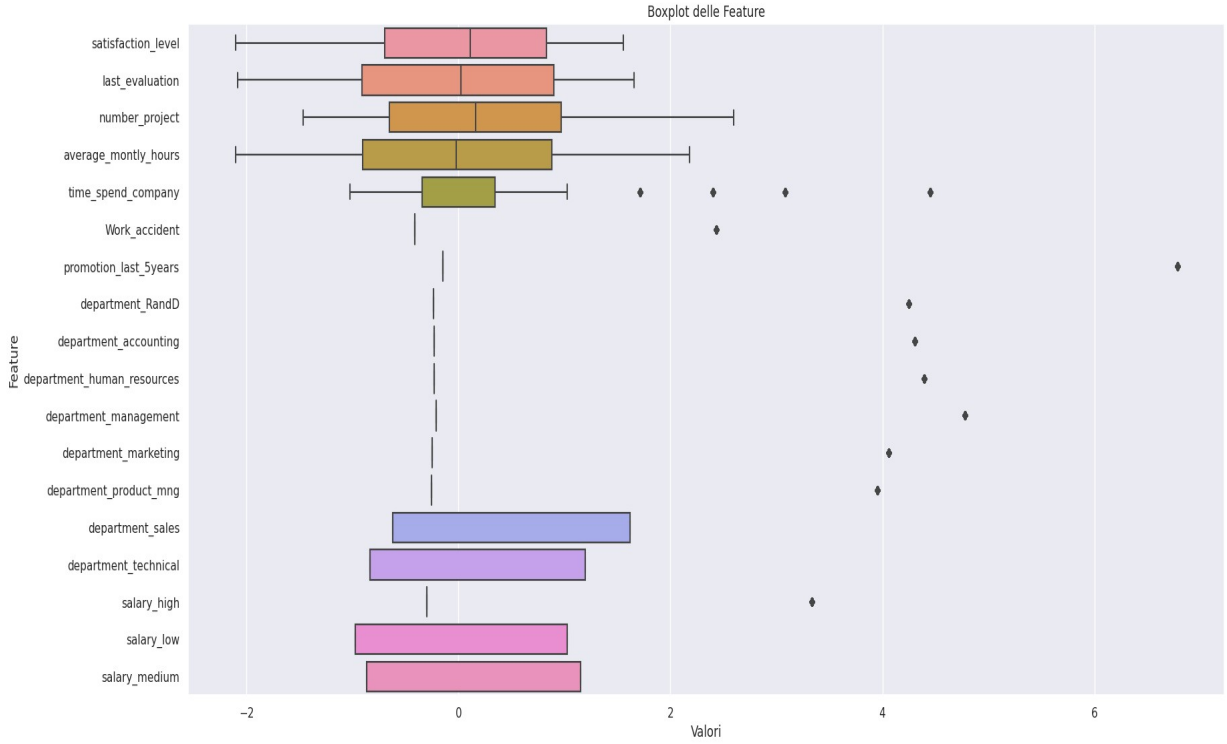
```
count    14999.000000
mean         0.612834
std          0.248631
min          0.090000
25%          0.440000
50%          0.640000
75%          0.820000
max          1.000000
Name: satisfaction_level, dtype: float64
```

Di seguito riportiamo le statistiche generali sulle variabili numeriche presenti nel dataset:

index	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.0	14999.0	14999.0	14999.0	14999.0	14999.0	14999.0	14999.0
mean	0.6128335222348156	0.7161017401160078	3.80305353690246	201.05033668911 26	3.498233215547703	0.1446096406427095	0.2380825388359224	0.021268084538969265
std	0.2486306510611 4257	0.1711691106232 7533	1.2325923553183522	49.94309937128408	1.460136230535 4812	0.3517185523801 7985	0.4259240993802994	0.1442814645785 8232
min	0.09	0.36	2.0	96.0	2.0	0.0	0.0	0.0
25%	0.44	0.56	3.0	156.0	3.0	0.0	0.0	0.0
50%	0.64	0.72	4.0	200.0	3.0	0.0	0.0	0.0
75%	0.82	0.87	5.0	245.0	4.0	0.0	0.0	0.0
max	1.0	1.0	7.0	310.0	10.0	1.0	1.0	1.0

DATA CLEANNING

Dopo aver preso visione delle principali caratteristiche del dataset passiamo al verifica dell'integrità dei dati i quali non presentano eccessivi problemi, gli esempi contenenti troppi campi mancanti sono stati scartati. Un problema riscontrato riguardava l'attributo categorico del dipartimento il quale presentava valori che potevano essere accorpati tutti sotto lo stesso valore categorico per cui sono state apportate queste modifiche. Infine, è stata effettuata un analisi degli outliers ovvero degli elementi che potrebbero influenzare negativamente le fasi dell'apprendimento. Per rimuoverli è stato usato un algoritmo basato su IQR l'Interquartile Range il quale è una misura statistica utilizzata per identificare e gestire gli outlier nei dati. L'IQR è calcolato come la differenza tra il terzo quartile (Q3) e il primo quartile (Q1) di un insieme di dati. Permette di rimuovere dati eccedenti dai range delle distribuzioni di appartenenza.



DATA PREPROCESSING

Ai dati ripuliti si è applicato la trasformazione di alcune features tramite one hot encoding, il processo con il quale da una feature qualitativa né otteniamo n quantitative e questo processo è stato apportato sia a dipartimento che a salario provocando l'aumento delle features per espansione. Una volta terminato questo processo poiché abbiamo scelto di lavorare con modelli lineari per supportarne l'apprendimento abbiamo standardizzato tutte le features di modo che fossero tutte tra di loro comparabili e che la magnitudine di una non avrebbe prevalso sul calcolo dei pesi durante le fasi di apprendimento degli algoritmi.

COST SENSITIVE VERSUS COST INSENSITIVE

Siamo giunti all'applicazione della prima tecnica per il trattamento dell'imbalance learning. Abbiamo deciso di prendere due modelli lineari e di confrontarli tra di loro la scelta è stata influenzata dalla libreria di scikit-lerarn

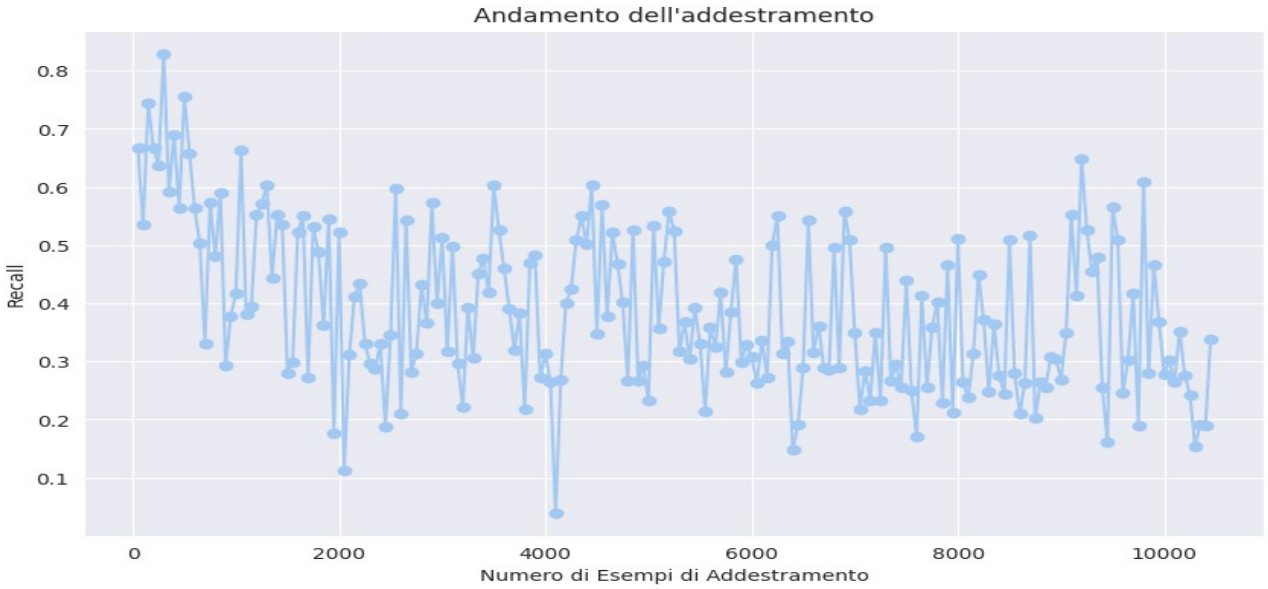
con la quale abbiamo operato poiché non tutti i modelli di apprendimento supervisionati proposti da tale libreria offrono l'opzione dell'apprendimento tramite cost sensitive che consiste nell'associare un peso differente alle due classi per portare a termine l'addestramento. Per tali ragioni la nostra scelta è ricaduta sullo Stochastic Gradient Descent che sarà messo a confronto con il Percettrone per capire chi tra i due riesca meglio ad affrontare le problematiche dello sbilanciamento. Per effettuare lo studio prima si è posta una base di partenza data dall'apprendimento cost insensitive e successivamente si è valutato il cost sensitive.

COST INSENSITIVE LEARNING

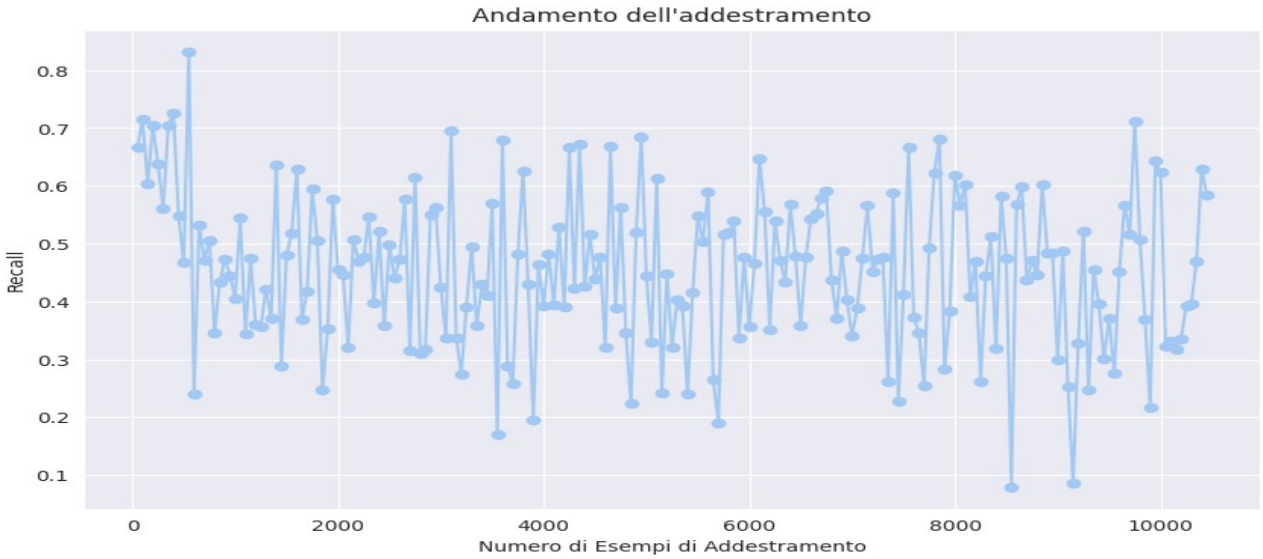
L'obiettivo prevede lo studio dei modelli nelle condizioni peggiori ovvero senza alcun supporto per capire come si comportano e le difficoltà da essi riscontrate. Inizieremo valutando la qualità dell'apprendimento studiato con le curve di apprendimento in base a delle metriche poiché ci troviamo difronte ad un contesto sbilanciato l'accurancy poco si addice per la valutazione e per cui le nostre scelte sono state affidate principalmente al recall dato che ci interessa sapere in modo esaustivo quanti dipendenti decideranno di dare le dimissioni e in seguito abbiamo valutato anche la precision.

Curve di apprendimento a confronto:

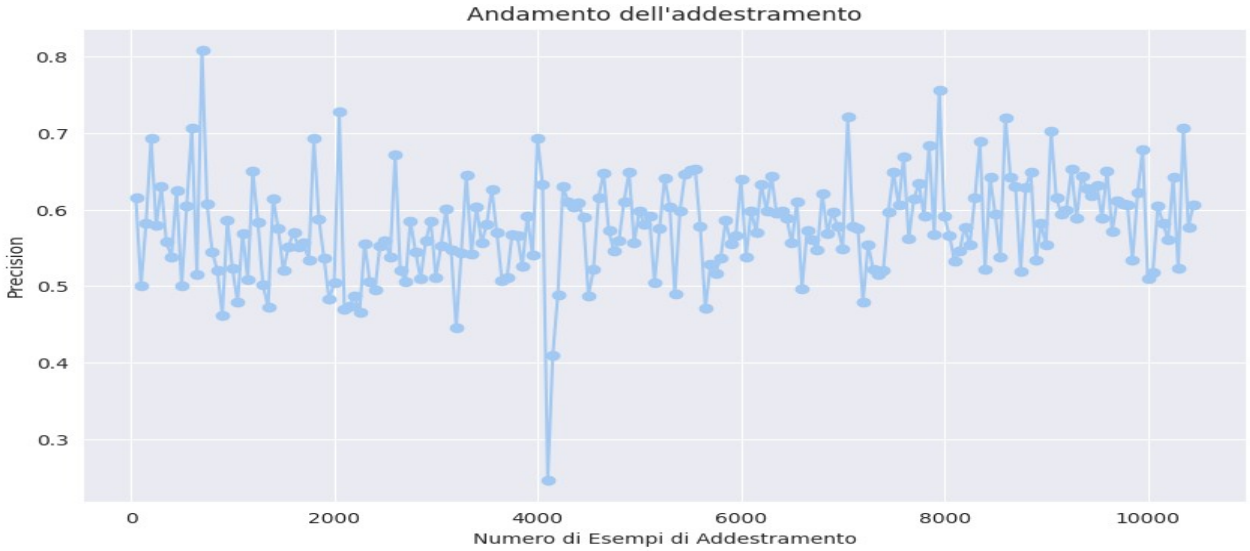
curva di recall dello SGDC:



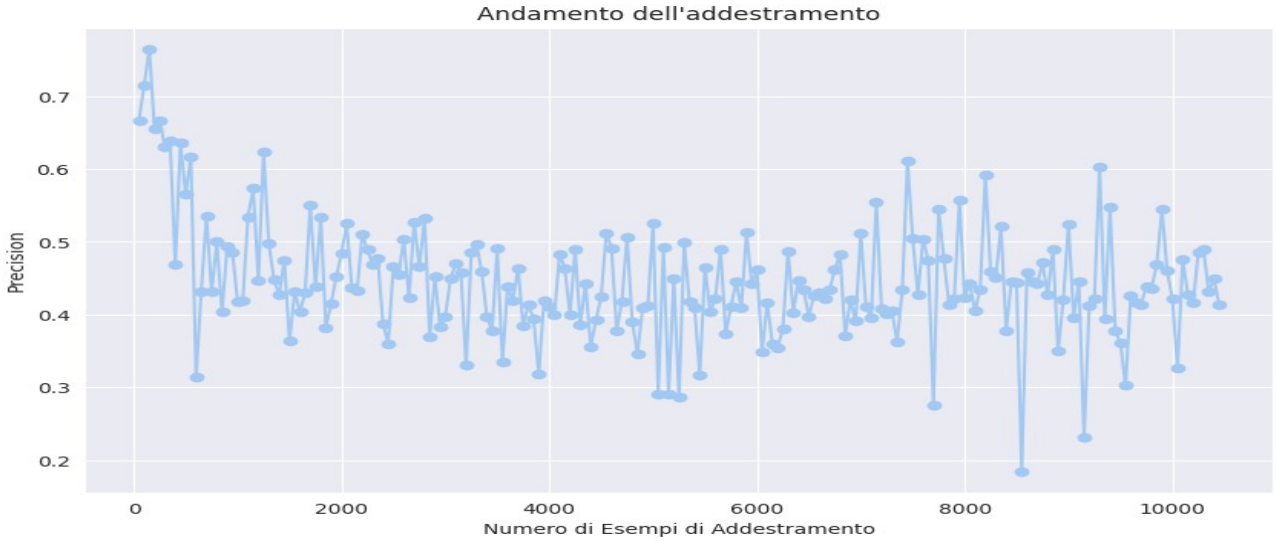
Curva di recall del Percettrone



Curva di precision dello SGDC



Curva di precision del Percettrone



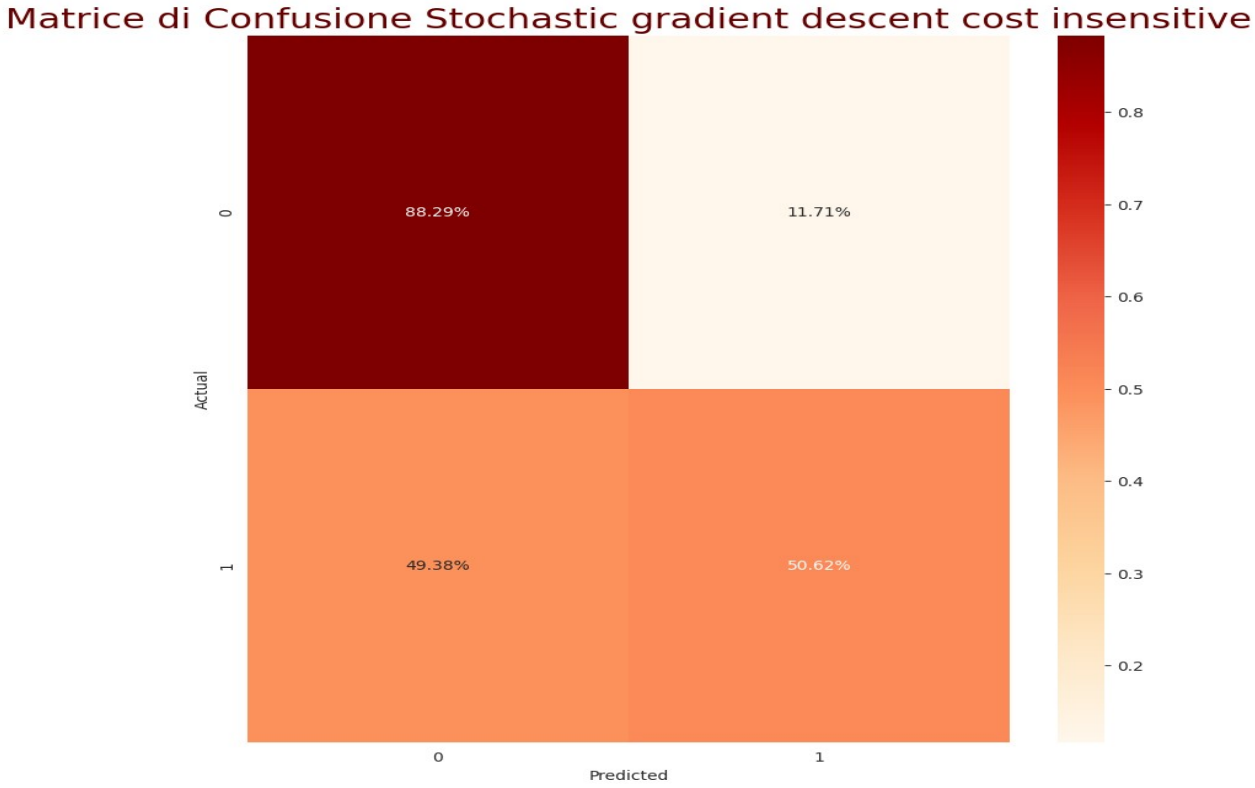
Dalle forti oscillazioni possiamo notare come entrambi i modelli abbiano avuto difficoltà del apprendere i concetti delle due classi e riuscire in seguito a generalizzare sui futuri esempi e di fatto non riescono a stabilizzare ad un valore accurato nessuna delle due metriche con le quali si è valutato l'addestramento.

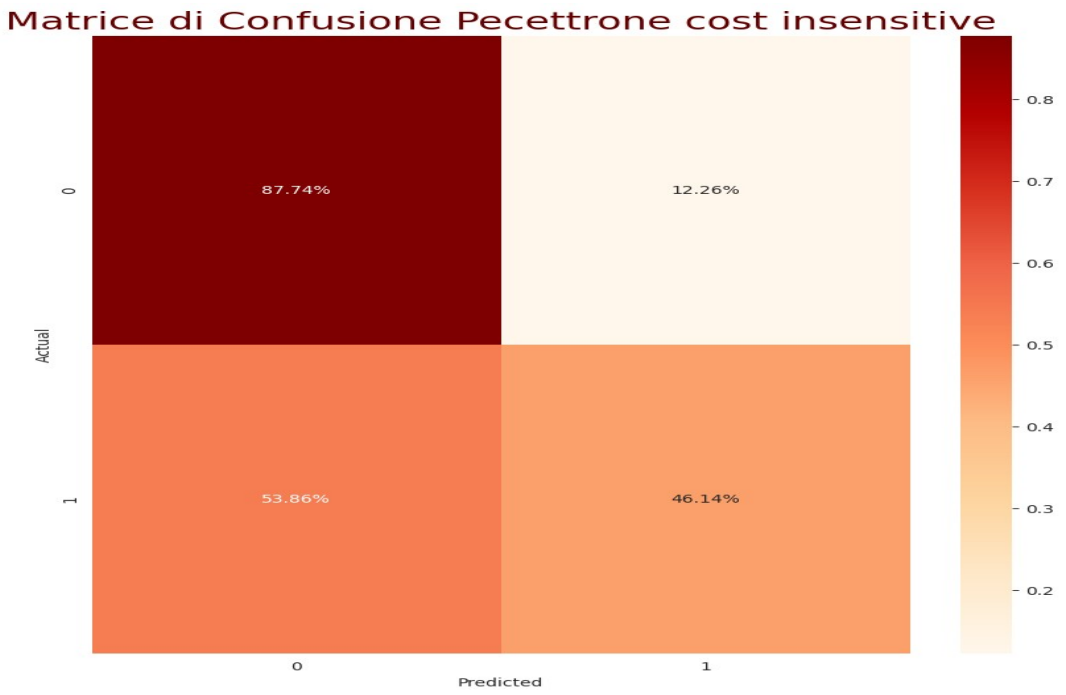
Vediamo adesso le loro performance valutate su di un quadro di classificazione che raccolga le principali metriche con le quali si valutano i classificatori:

```
_____STOCHASTIC GRADIENT DESCENT COST INSENSITIVE______
-----------------------Accuracy-----------------------

0.81

-----------------Classification_Report----------------

              precision    recall  f1-score   support

           0       0.85      0.92      0.88      3451
           1       0.63      0.45      0.53      1049

    accuracy                           0.81      4500
   macro avg       0.74      0.69      0.70      4500
weighted avg       0.80      0.81      0.80      4500
```

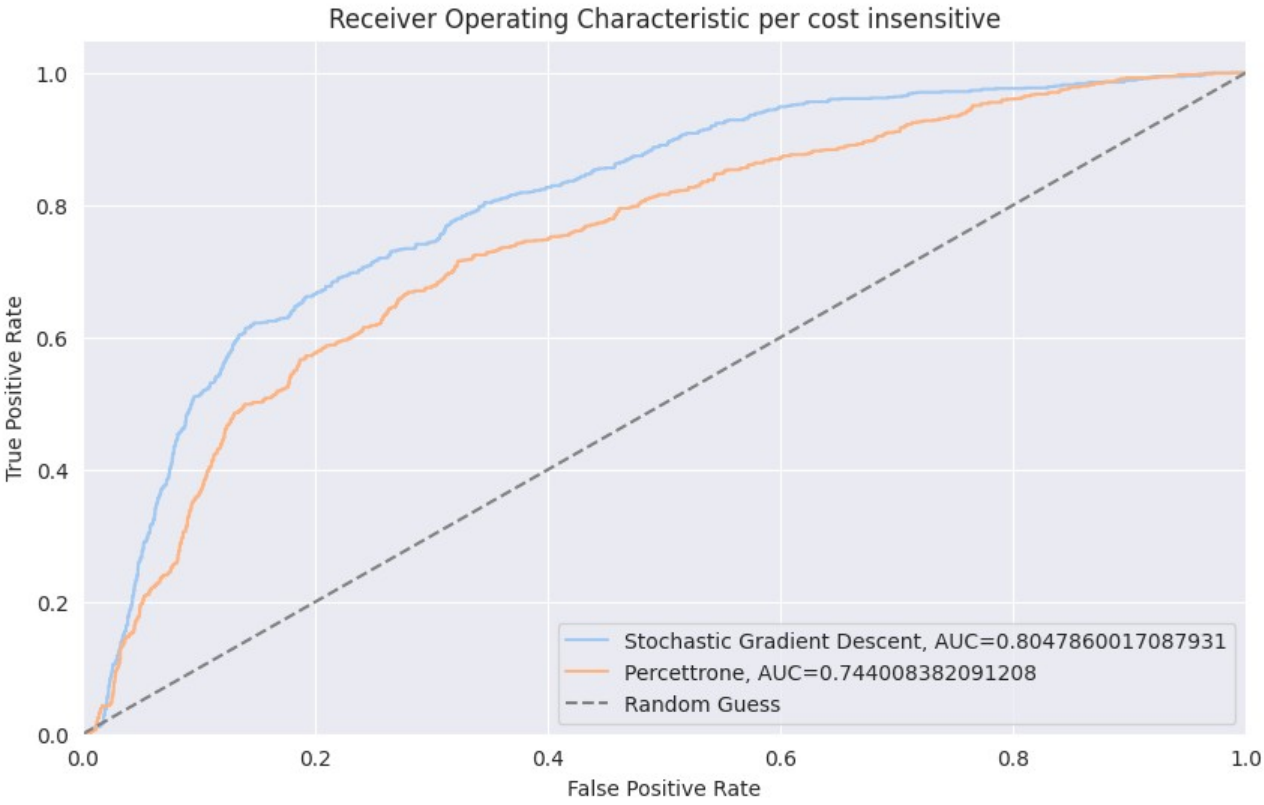
```
_____________PERCETTRONE COST INSENSITIVE_____________
------------------------Accuracy------------------------

0.7804444444444445

-----------------Classification_Report----------------

              precision    recall  f1-score   support

           0       0.84      0.88      0.86      3451
           1       0.53      0.46      0.49      1049

    accuracy                           0.78      4500
   macro avg       0.69      0.67      0.68      4500
weighted avg       0.77      0.78      0.77      4500
```

Le differenze non sono eccessive, ma lo SGDC risulta più performante, cerchiamo di prendere visione in maniera più chiara e facilitata.





Infine, andiamo a metterli a confronto su di un grafico che mostri il Receiver operator characteristic e ne misuri l'area sottostante:



Da gli esperimenti condotti abbiamo capito che in assenza di un supporto esterno dato ai modelli essi hanno difficoltà a trattare lo sbilanciamento, ad ogni modo sono riusciti a completare l'addestramento anche se non nel modo ottimale e possiamo concludere che in tali condizioni lo SGDC è stato più performante del Percettrone.

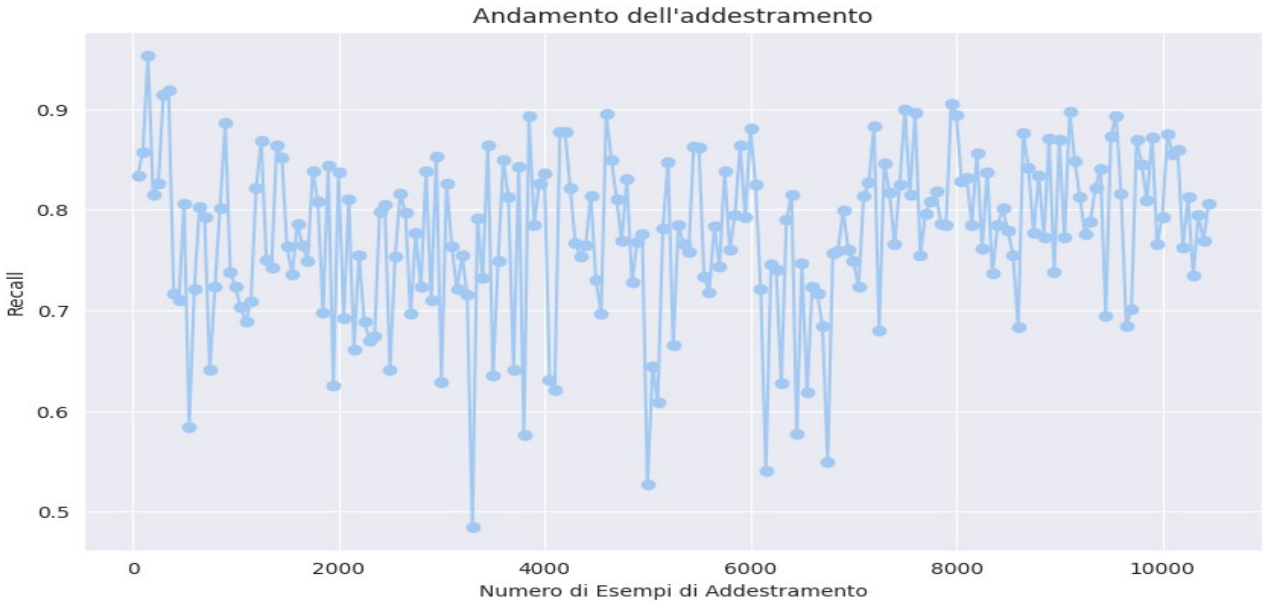
COST SENSITIVE LEARNING

Procediamo ad applicare l'apprendimento con pesi per la gestione dello sbilanciamento. L'idea è assegnare un bonus sugli esempi classificati correttamente delle classe minoritaria e un malus quando la classificazione è fallace, ad ogni modo questo processo è effettuato per entrambe le classi quello che cambia è il peso il quale sarà più importante e quindi porterà a maggiori cambiamenti, per la classe minoritaria, mentre sarà meno incisivo per la classe maggioritaria. Spiegato il funzionamento del cost sensitive learning ci soffermiamo sul fatto che la libreria scikit-learn offre differenti opzioni per la gestione del perso quali la specifica del peso assegnato quando si ha della conoscenza del dominio da parte dell'esperto da sfruttare oppure il calcolo del

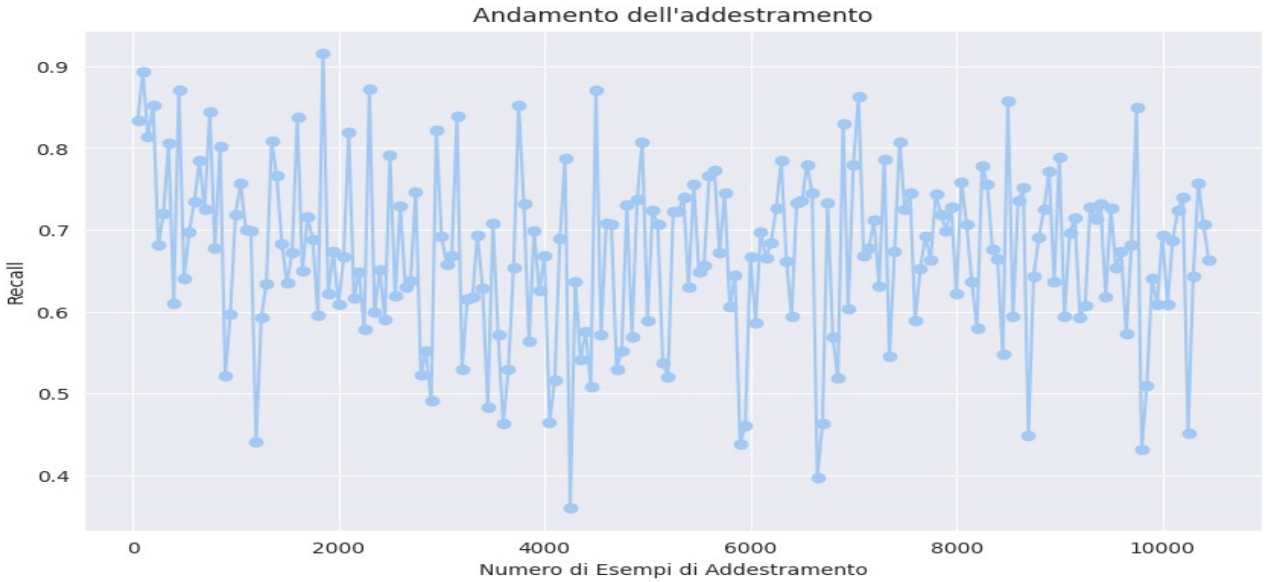
peso commisurato agli esempi appartenenti a ciascuna delle classi ovvero calcolato in base alla proporzione dello sbilanciamento più sarà marcato lo sbilanciamento e più il peso sarà inciso per la classe minoritaria e meno influente per la maggioritaria. Questo approccio è quello che ci è sembrato più sensato ed è stato quello scelto per l'addestramento con cost sensitive.

Andiamo a prendere visione delle curve di apprendimento dei modelli sottoposti ad apprendimento con cost sensitive:

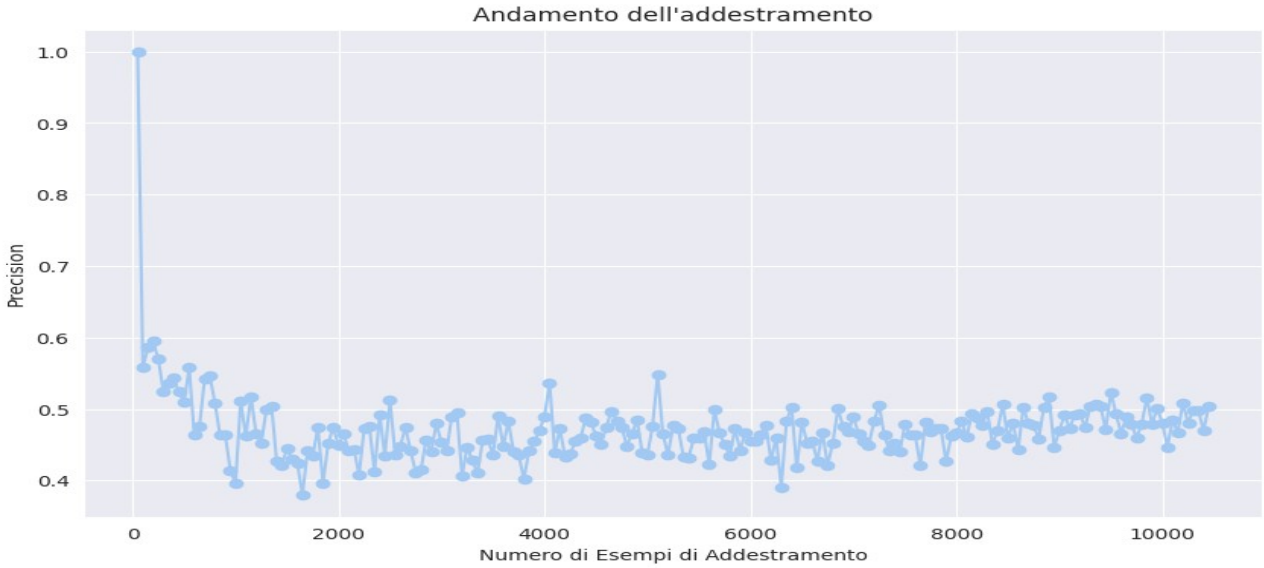
curva di recall per lo SGDC



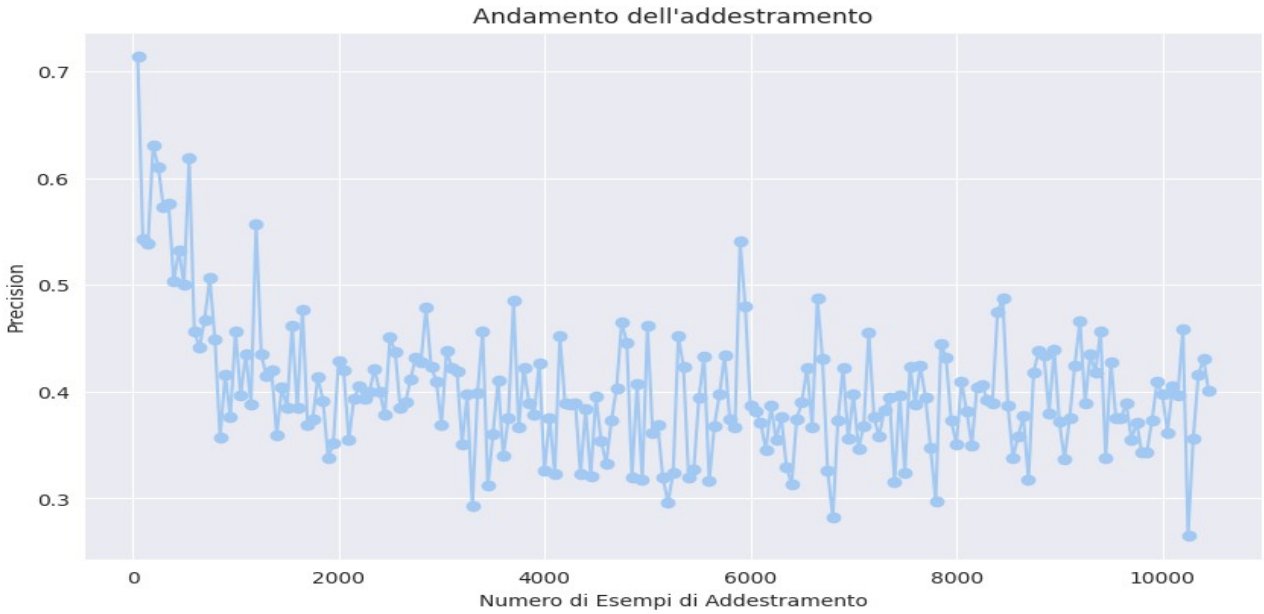
Curva di recall per il percettrone



Curva di precision per lo SGDC:



Curva di precision per il percettrone:



Dalle curve notiamo che anche in questo caso ci sono stati dei problemi di apprendimento per i due modelli, ma le oscillazioni sono diventate meno ampie rispetto alle curve di apprendimento senza i pesi applicati, inoltre in particolare la precision per lo SGDC sembra stabilizzarsi abbastanza si dà subito.

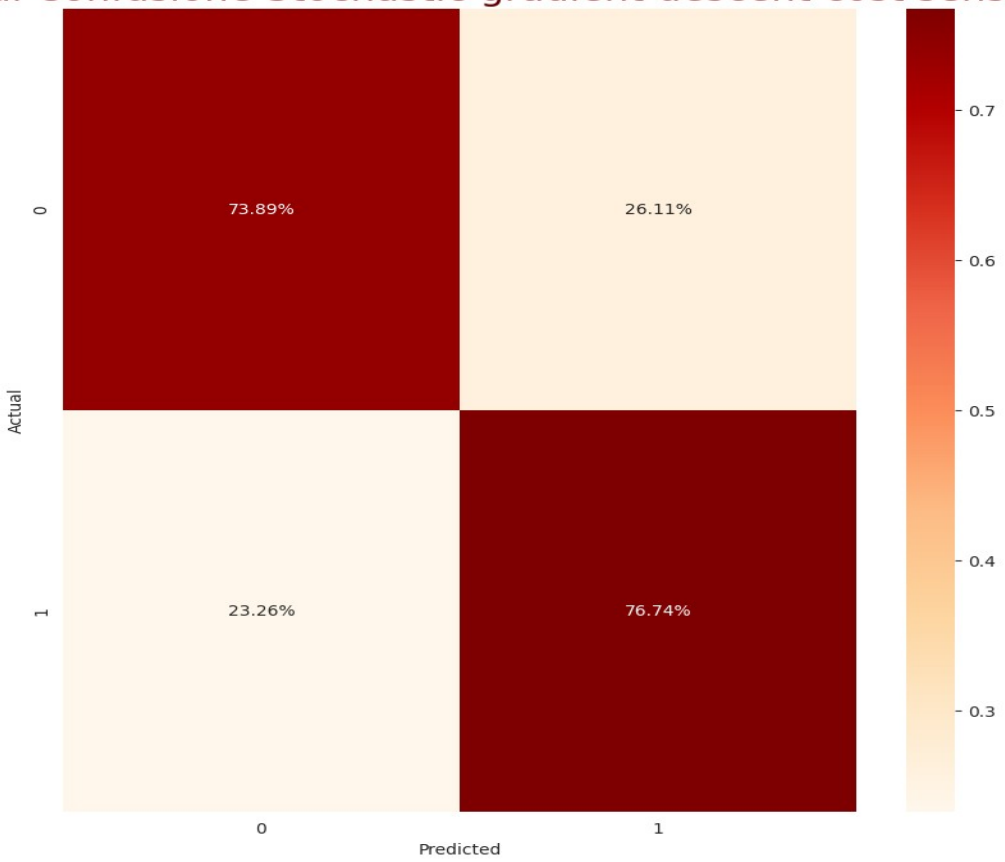
Adesso andiamo a prendere atto del classification report che ci darà maggiori informazione sulle performance di questi due classificatori:

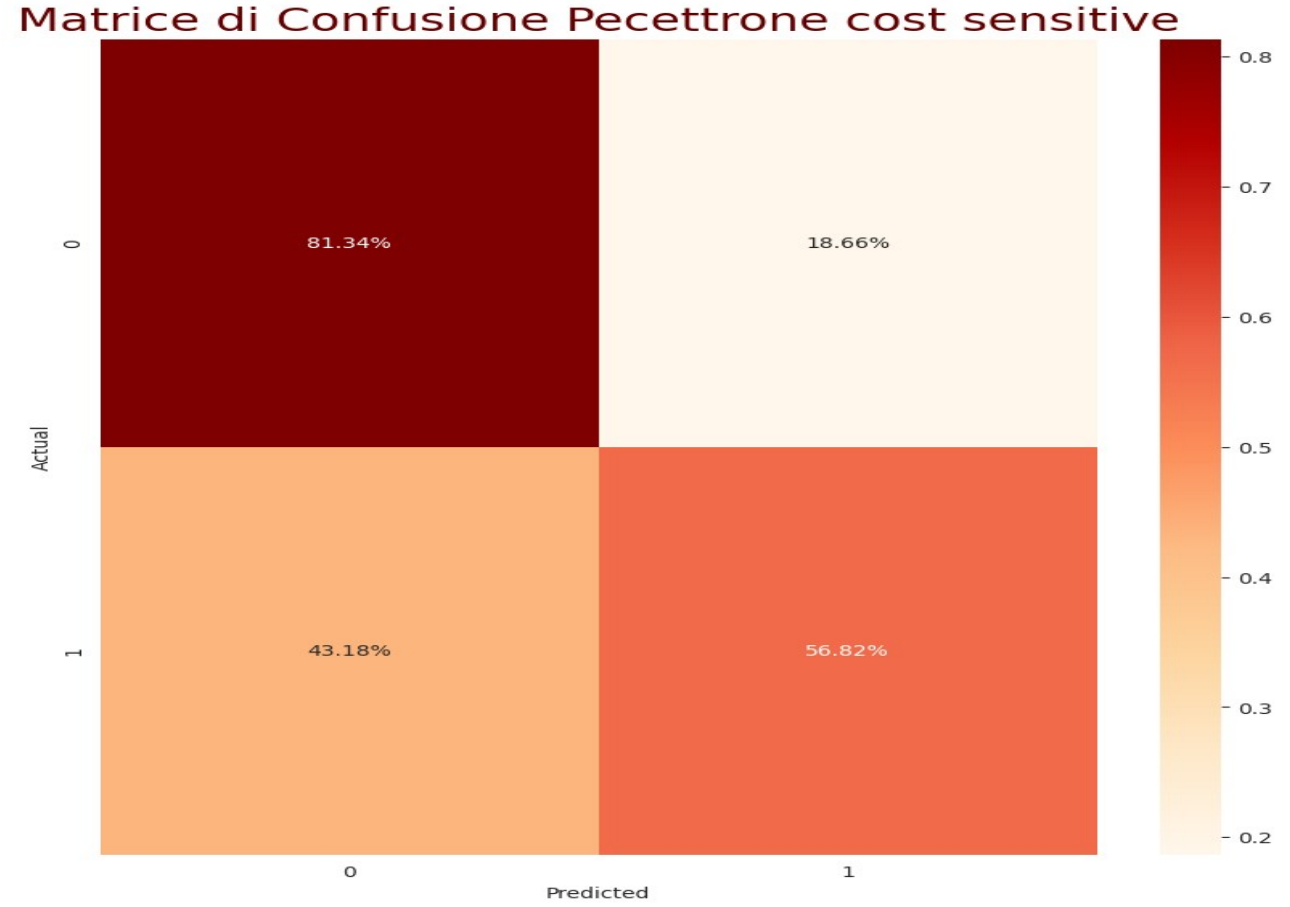
```
_______STOCHASTIC GRADIENT DESCENT COST SENSITIVE_______
------------------------Accuracy------------------------

0.7455555555555555

------------------Classification_Report-----------------

              precision    recall  f1-score   support

           0       0.91      0.74      0.82      3451
           1       0.47      0.77      0.58      1049

    accuracy                           0.75      4500
   macro avg       0.69      0.75      0.70      4500
weighted avg       0.81      0.75      0.76      4500
```

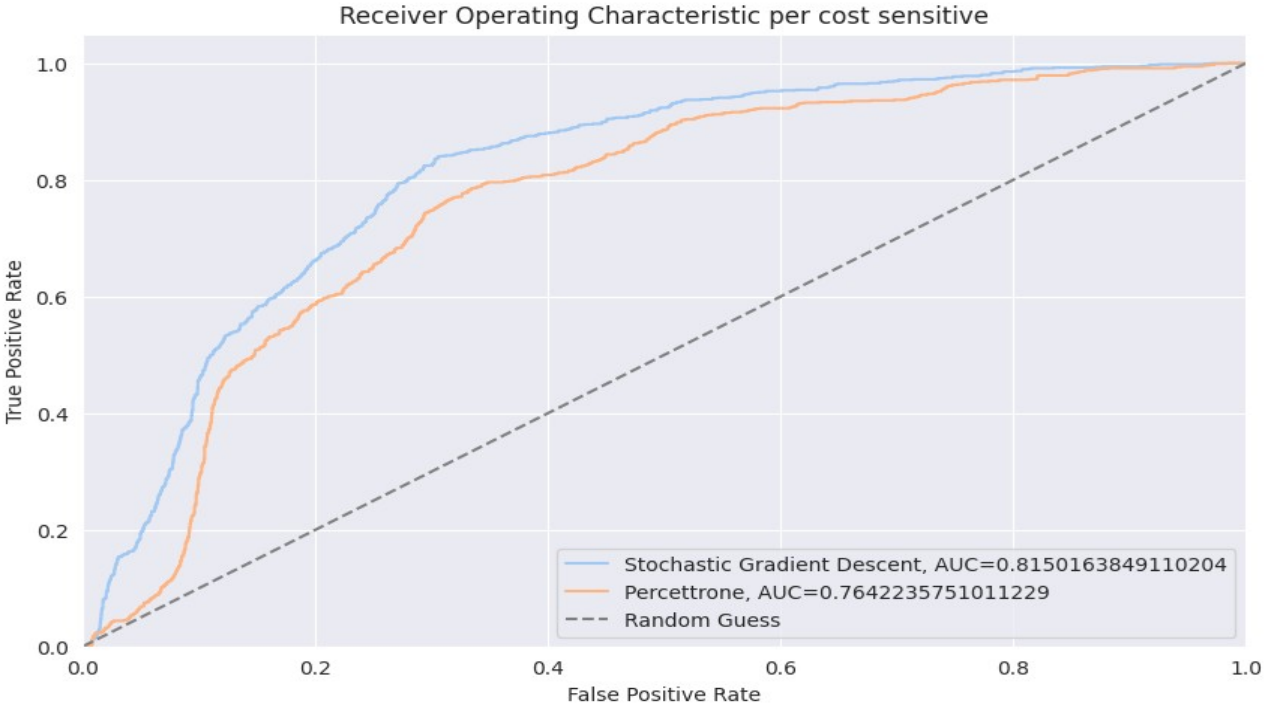
```
_______________PERCETTRONE COST SENSITIVE_______________
------------------------Accuracy------------------------

0.7562222222222222

------------------Classification_Report-----------------

              precision    recall  f1-score   support

           0       0.86      0.81      0.84      3451
           1       0.48      0.57      0.52      1049

    accuracy                           0.76      4500
   macro avg       0.67      0.69      0.68      4500
weighted avg       0.77      0.76      0.76      4500
```

Infine, andiamo a dare una prospettiva in termini di percentuali con le matrici di confusione e poi mostreremo il confronto con il roc score.

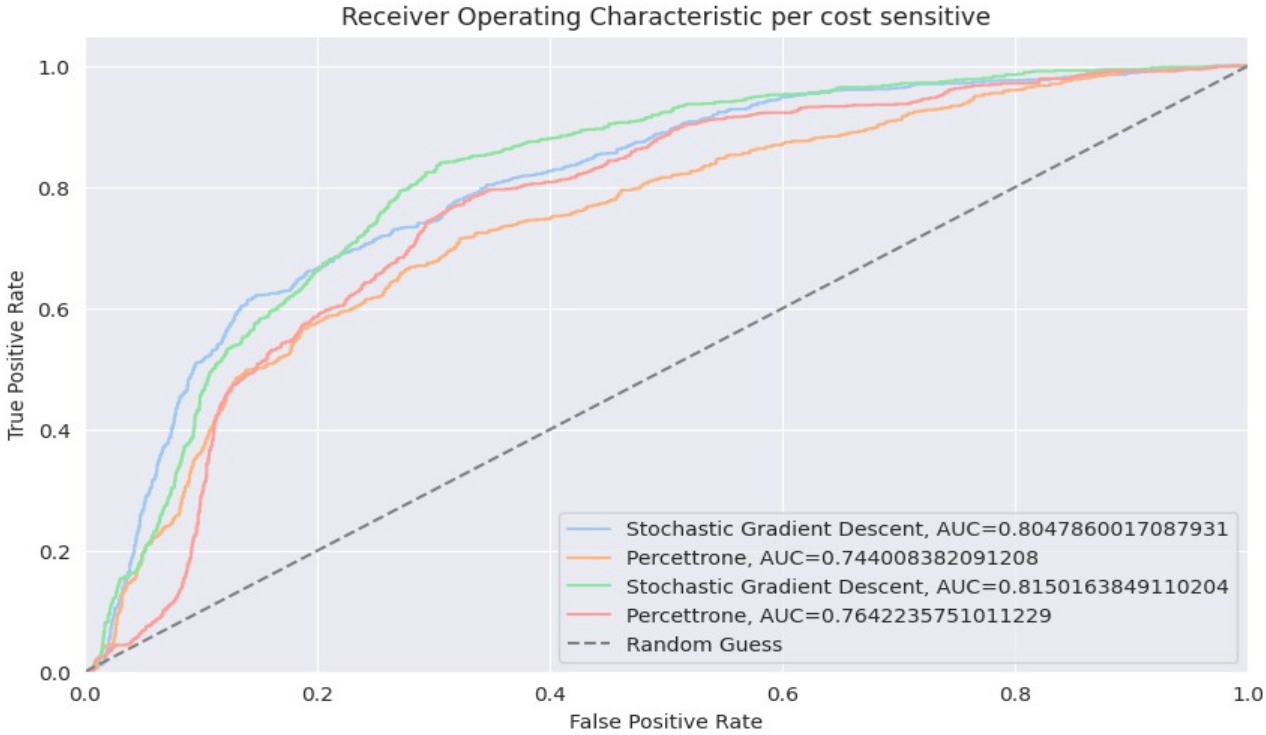
Matrice di Confusione Stochastic gradient descent cost sensitive




Matrice di Confusione Pecettrone cost sensitive
81.34%
18.66%
43.18%
56.82%
Actual
0
1
Predicted
0
1
0.8
0.7
0.6
0.5
0.4
0.3
0.2



Come mostrato dalle metriche prodotte dopo l'addestramento di questi due modelli, anche in questo caso lo SGDC è risultato essere il modello più performante che sovrasta il percettrone su ogni metrica. Quello che abbiamo notato ad ogni modo è un lieve incremento delle performance dovuto al cost sensitive learning per tale ragione anche se ci aspettavano un margine di miglioramento superiore possiamo dire che il cost sensitive ha sicuramente un impatto positivo sull'apprendimento di questi modelli come è possibile vedere nel sottostante grafico:



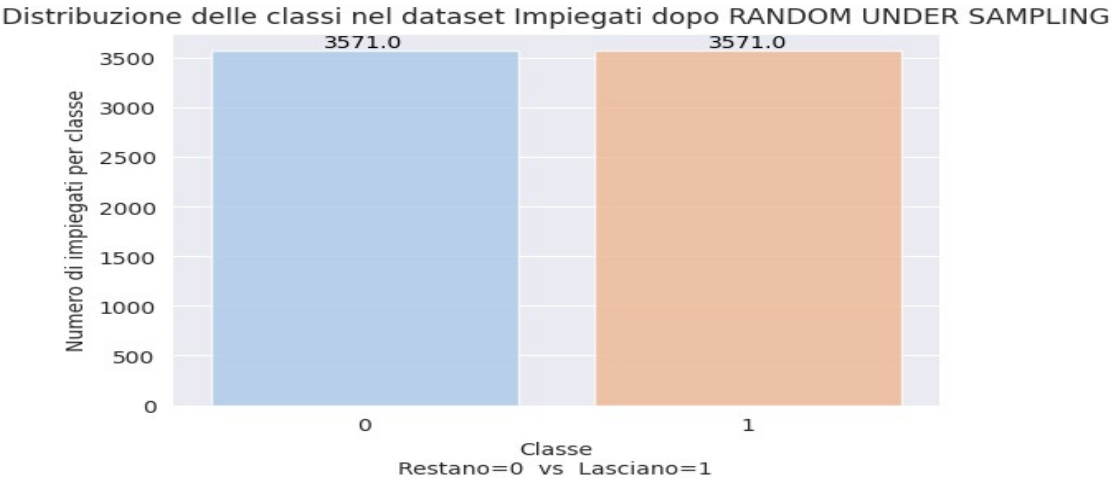
TECNICHE DI CAMPIONAMENTO

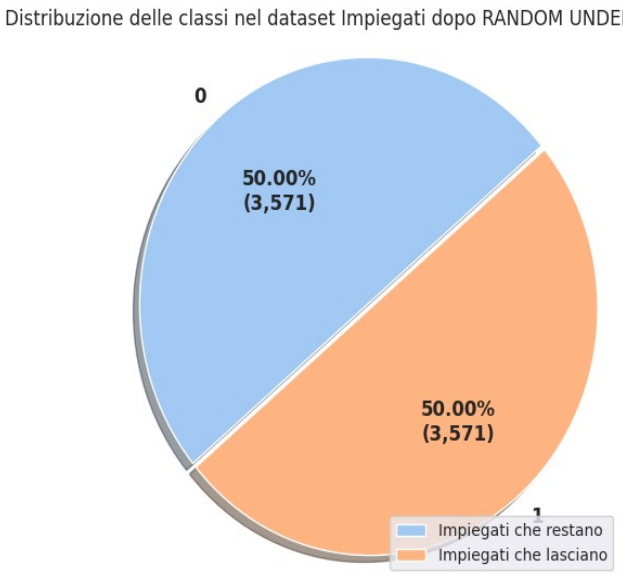
Adesso spostiamo la nostra attenzione sull'applicazione delle tecniche di campionamento esse sfruttano l'idea base della statistica nel campionare opportunamente gli esempi dal dataset sbilanciato con l'obiettivo di ribilanciare il più possibile le classi. Per cui quello che queste tecniche offrono ai nostri modelli è il passaggio da un problema di imbalance learning a un semplice problema di apprendimento per cui il loro obbiettivo è smussare le difficoltà di apprendimento equilibrando lo sbilanciamento e portandolo in condizioni standard. L'idea è allettante, ma questo supporto che offrono a seconda della tecnica che si sceglie ha un prezzo da pagare che chiaramente incide sull'apprendimento dei modelli. Le tecniche di campionamento si dividono in tre gruppi che sono under sampling, over sampling e approccio misto per ogni gruppo troviamo numerosi approcci che affrontano i problemi dello sbilanciamento. Per il nostro studio abbiamo scelto un approccio per ogni uno dei tre gruppi di tecniche e lo abbiamo fatto in modo da diversificare il più possibile i nostri risultati.

Nello studio delle tecniche di campionamento per gestire lo sbilanciamento il nostro interesse si sposta su ulteriori due modelli e questa volta andremo a confrontare le loro performance con le tre tecniche di campionamento e i modelli scelti sono la Regressione Logistica e il Gaussian Naive Bayes.

RANDOM UNDER SAMPLING

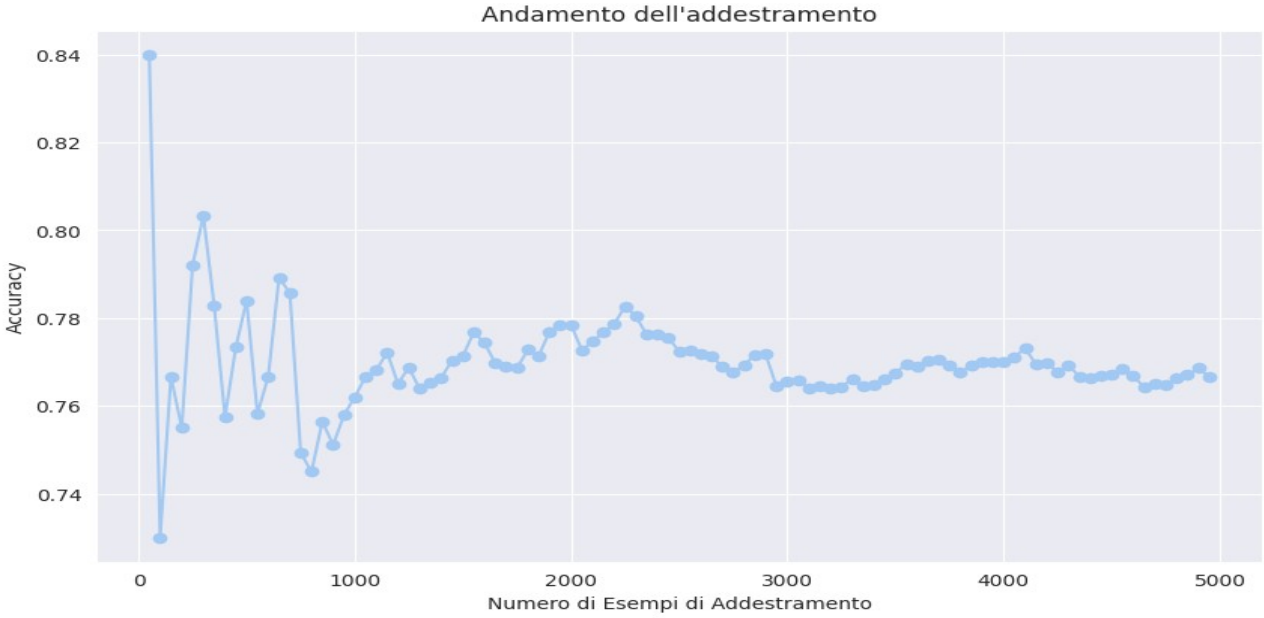
Il random under sampling RUS è una tecnica di under sampling che si basa su un idea semplice e banale, ovvero andare a campionare dalla classe maggioritaria esempi casualmente per scartarli e andare a bilanciare il numero degli esempi tra le due classi. L'aspetto che viene ad essere garantito è il bilanciamento perfetto tra le due classi, ma ad un prezzo alto perché stiamo perdendo informazione utile, cioè, stiamo perdendo l'informazione contenuta nei rappresentanti della classe maggioritaria, inoltre stiamo buttando deliberatamente i dati raccolti con sacrifici e tempo. Per queste ragioni è una tecnica che non viene raccomandata se non in quei casi in cui abbiamo un quantità enorme di dati. Di contro abbiamo scelto questa tecnica anziché le più sofisticate ed evolute ENN edited Nearest Neighbours o il TomeK links, poiché queste tecniche sono usate negli approcci misti o, meglio, negli algoritmi più performanti degli approcci misti e siccome è nostra intensione andare ad usarne uno tra questi ci siamo limitati al RUS. La differenza tra il RUS e gli altri due è che anziché andare a rimuovere esempi casualmente sfruttano qualche inventiva in più per eliminare esempi della classe maggioritaria che non sono fortemente rappresentativi o che risultano essere outliers. Di seguito riportiamo i risultati ottenuti con l'addestramento dopo l'applicazione del RUS.



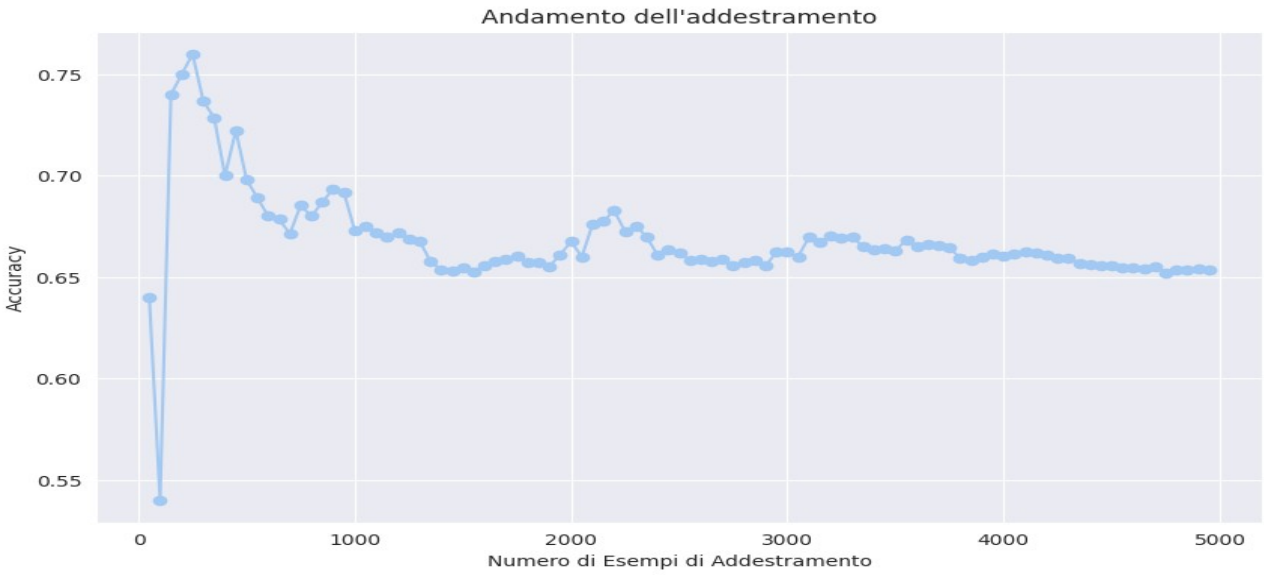


Come precedentemente accennato il RUS è capace di riequilibrare la situazione.

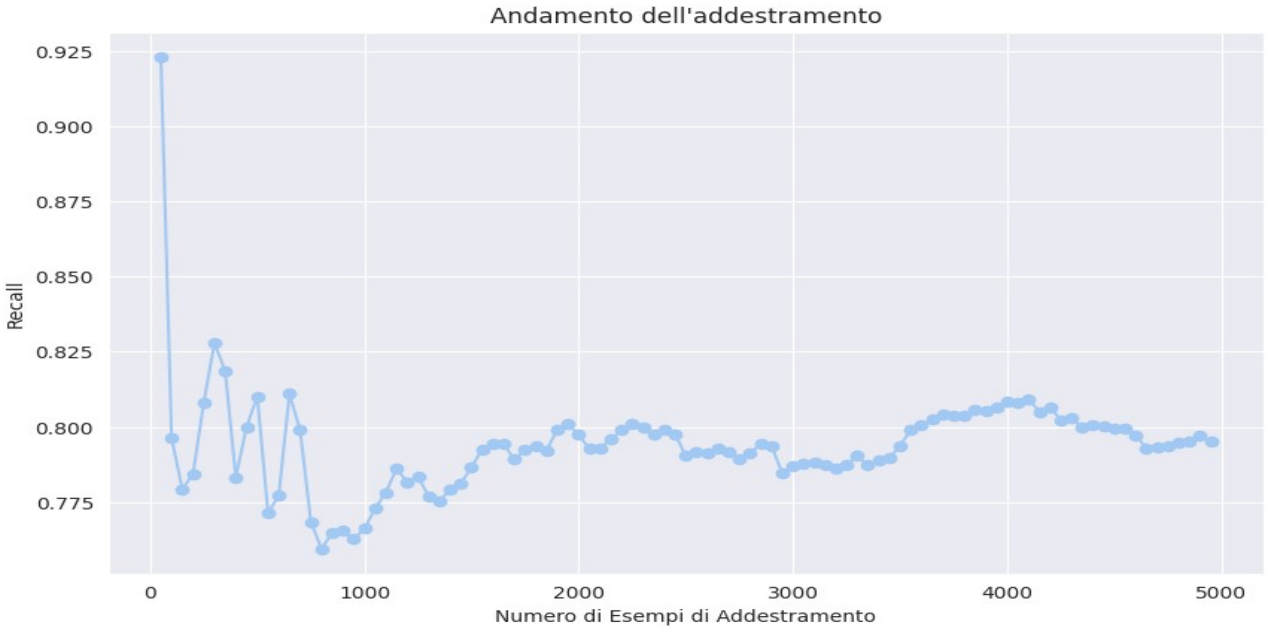
Curva di accuracy della regressione logistica



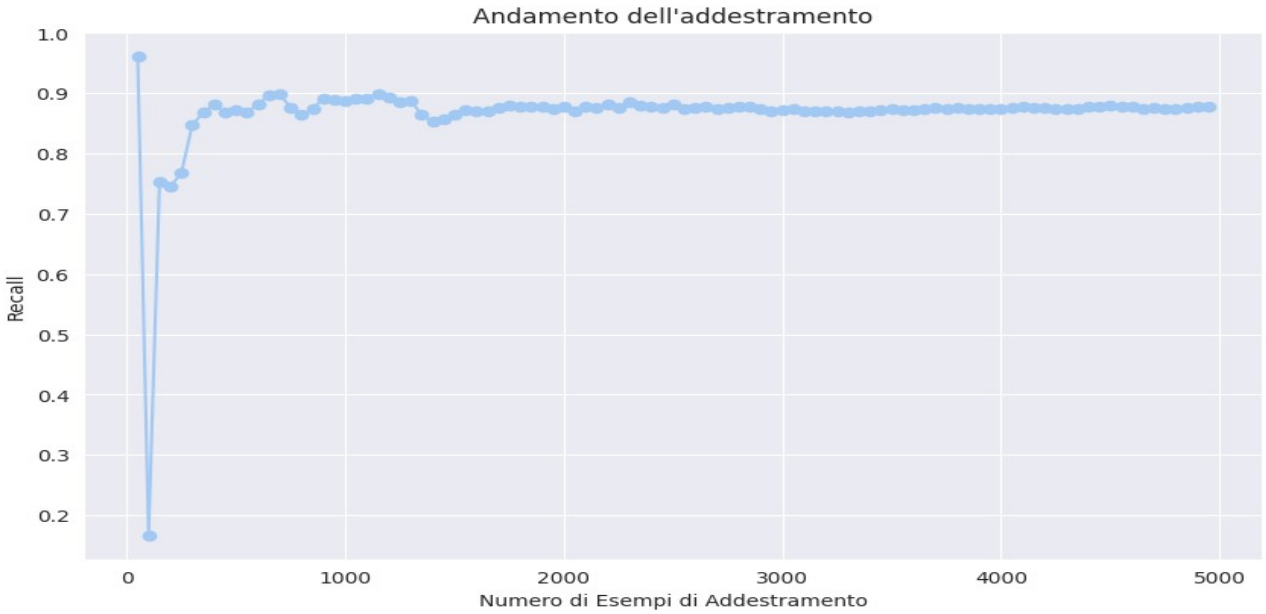
Curva di accuracy del gaussian naive bayes



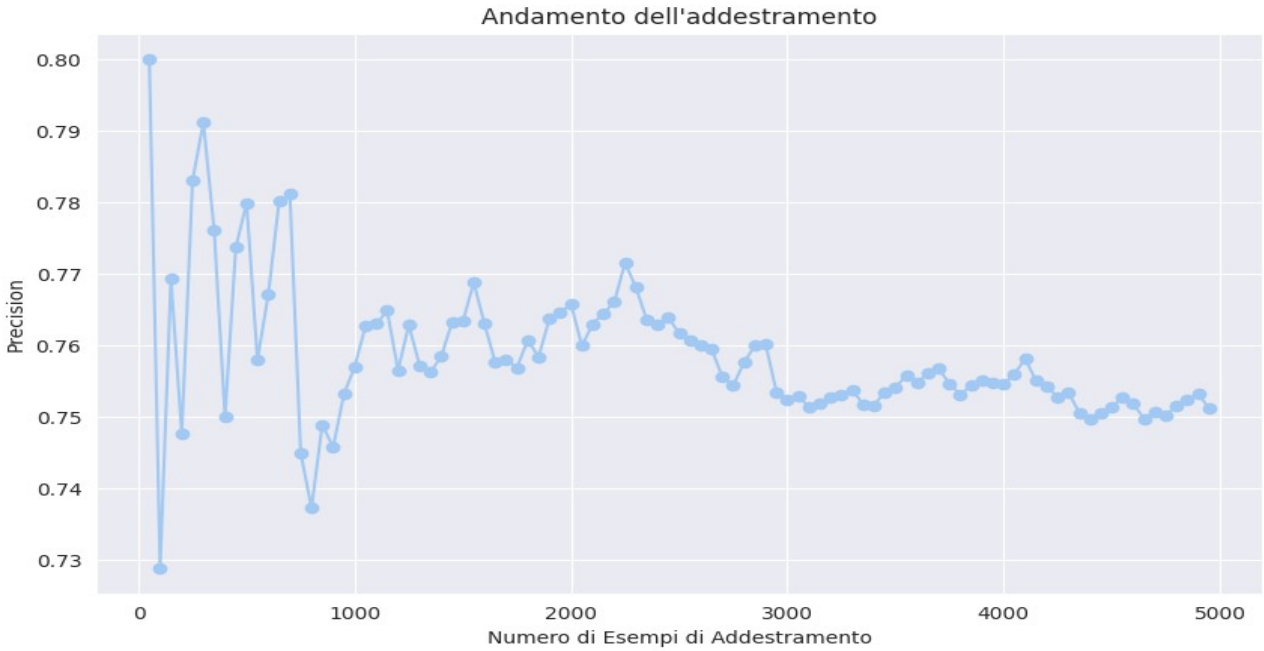
Curva di recall della regressione logistica



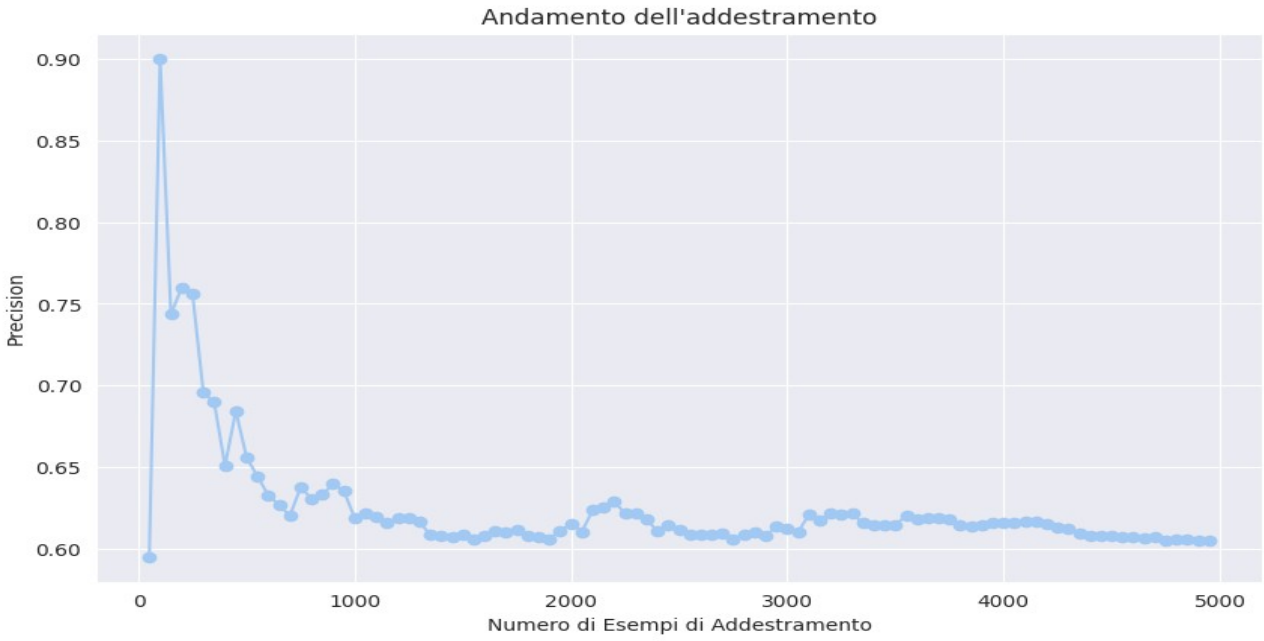
Curva di recall del gaussian naive bayes



Curva di precision della regressione logistica

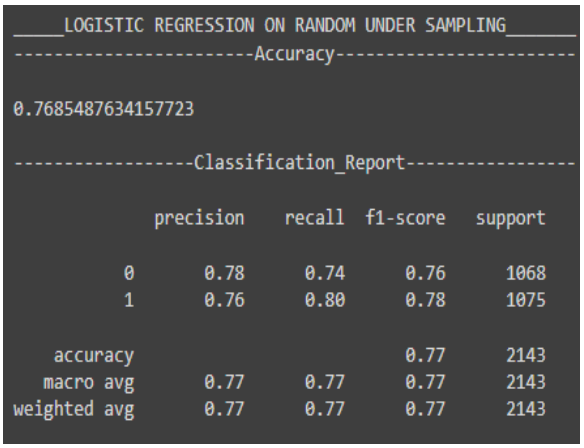


Curva di precision del gaussian naive bayes

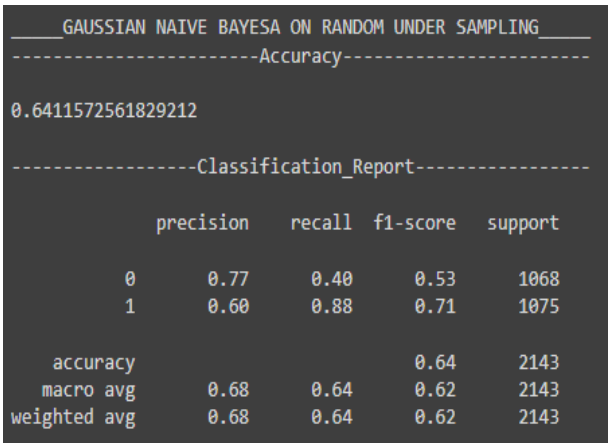


Anche in questo caso l'apprendimento mostra difficoltà le quali sono mostrate dalle oscillazioni che caratterizzano le curve, nonostante questo ci evince come il gaussian naive bayes risenta meno delle problematiche dovute all'apprendimento andando ad ottenere dei risultati di recall interessanti e soddisfacenti.

Andiamo a mettere a confronto i due classificatore sulle metriche generalmente usate per la classificazione.



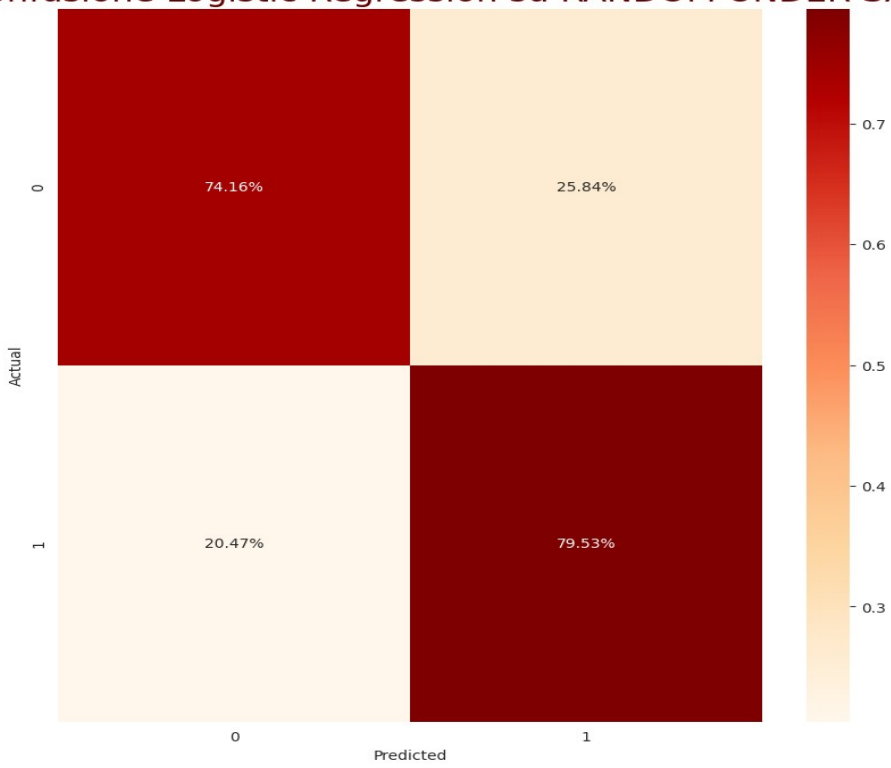
```
_____LOGISTIC REGRESSION ON RANDOM UNDER SAMPLING_____
-----------------------Accuracy-----------------------

0.7685487634157723

------------------Classification_Report-----------------

              precision    recall  f1-score   support

           0       0.78      0.74      0.76      1068
           1       0.76      0.80      0.78      1075

    accuracy                           0.77      2143
   macro avg       0.77      0.77      0.77      2143
weighted avg       0.77      0.77      0.77      2143
```



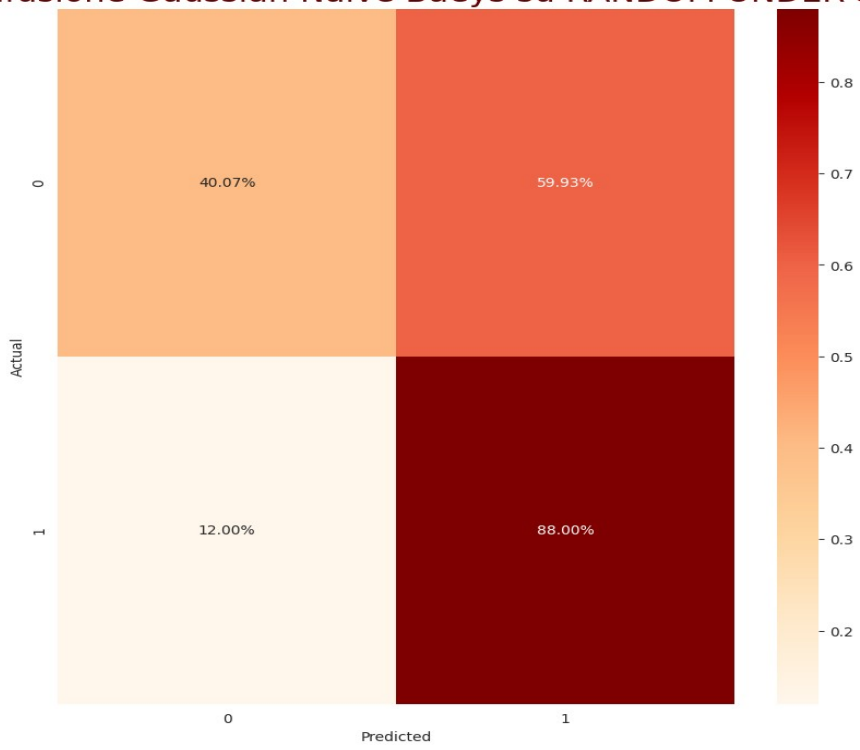
```
_____GAUSSIAN NAIVE BAYESA ON RANDOM UNDER SAMPLING_____
-----------------------Accuracy-----------------------

0.6411572561829212

------------------Classification_Report-----------------

              precision    recall  f1-score   support

           0       0.77      0.40      0.53      1068
           1       0.60      0.88      0.71      1075

    accuracy                           0.64      2143
   macro avg       0.68      0.64      0.62      2143
weighted avg       0.68      0.64      0.62      2143
```

Prendiamo visione delle matrici di confusione associate ai modelli:

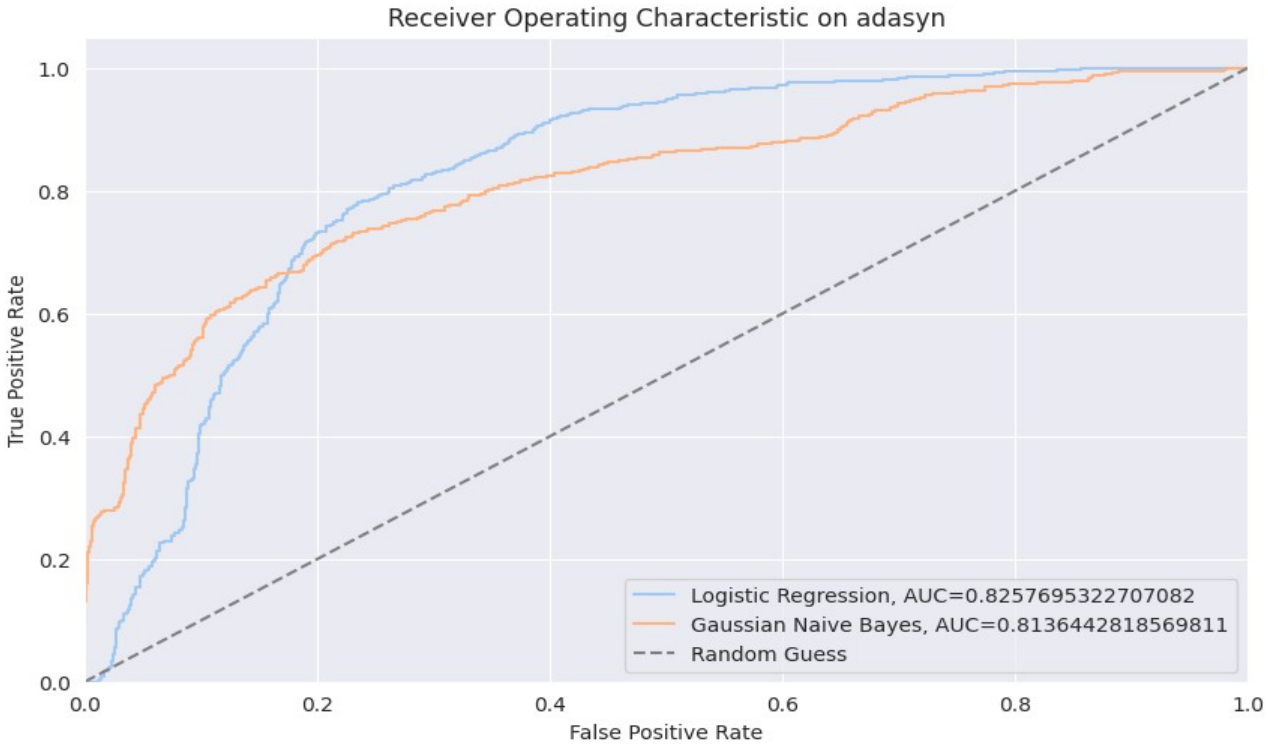
Matrice di Confusione Logistic Regression su RANDOM UNDER SAMPLING



Matrice di Confusione Gaussian Naive Baeys su RANDOM UNDER SAMPLING



Infine, mettiamo i classificatori a confronto su di un grafo con score ROC:



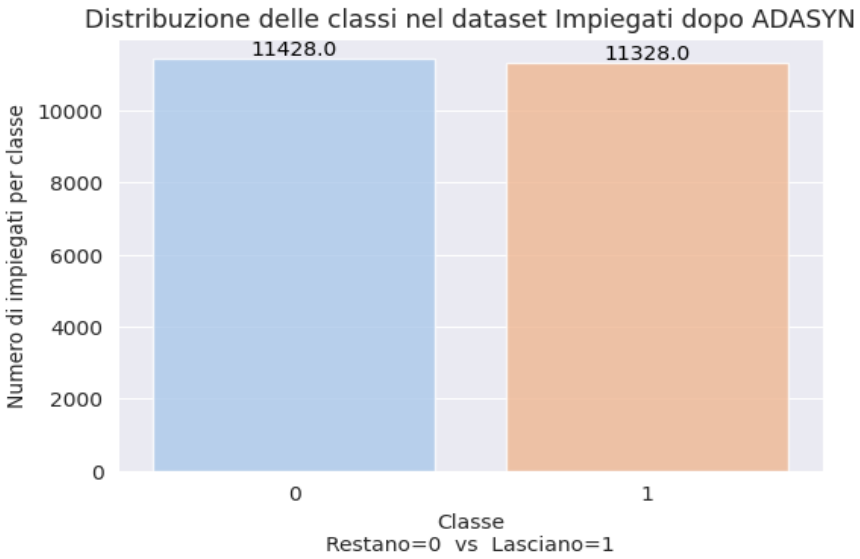
Dal metriche calcolate il confronto non è semplice e lascia margine all'incertezza, tuttavia considerando il problema di partenza e visto che, se fossimo a capo del direttivo delle risorse umane non vorremmo rimanere senza dipendenti si accetta di rischiare falsi allarmismi da parte del gaussian naive bayes sapendo di restare sul lato sicuro del rischio.

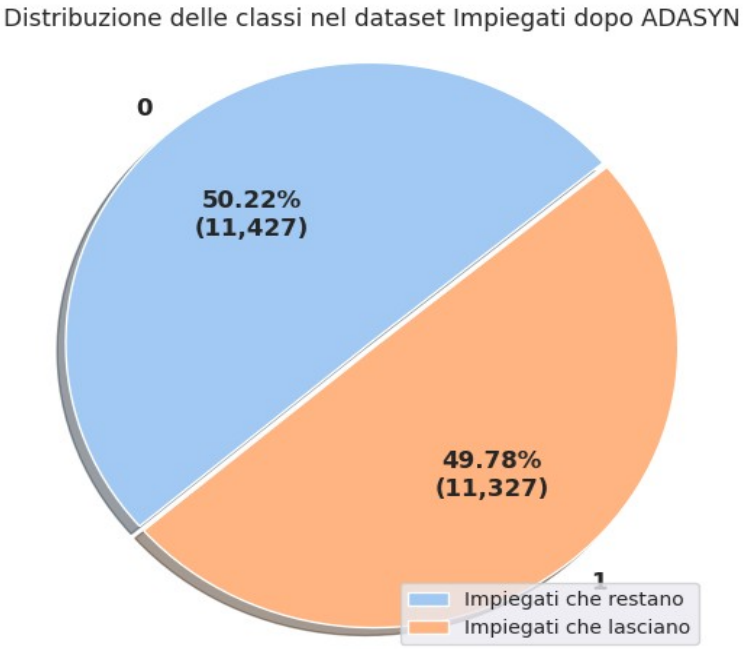
ADAPTIVE SYNTETHIC SAMPLING

Adesso il nostro studio si focalizza su una tecnica di over sampling ovvero l'ADASYN adaptive syntethic sampling. L'idea di questo algoritmo è quella di andare a creare dei dati fittizi ovvero finti che abbiano caratteristiche simili a quelle dei dati degli esempi della classe minoritaria per farlo si basa sul calcolo della densità delle classi nel dataset. La densità di una classe è una misura della distribuzione dei suoi esempi nello spazio delle feature e le classi che presentano una densità molto bassa indicano che sono classi minoritarie e potenzialmente soggette a problemi di sbilanciamento una volta che vengono individuate per ogni esempio nella classe minoritaria, ADASYN calcola il grado di disomogeneità rispetto alle classi circostanti. Il grado di disomogeneità

misura quanto un esempio sia diverso dalle sue vicine, basandosi sulla densità delle classi. Una volta ottenute le misure di densità e di disomogeneità ADASYN si concentra sulla generazione di esempi sintetici per le istanze delle classi minoritarie che hanno un alto grado di disomogeneità ovvero che sono il più possibile differenti dagli esempi della classe maggioritaria per caratteristiche. Questi esempi sintetici vengono generati mediante interpolazione tra gli esempi originali, con un peso maggiore assegnato agli esempi che hanno un grado di disomogeneità più alto. Questa tecnica a differenza della precedente non riesce a garantire il perfetto ribilanciamento del dataset, ma ad ogni modo riesce quasi del tutto a risanarlo tranne nei casi in cui gli esempi della classe minoritaria sono sparsi e immersi in esempi di classe maggioritaria. Ancora una volta questo bilanciamento per riportare l'apprendimento in condizioni normali è pagato con l'immissione di rumore o, meglio, di informazione di fittizia non del tutto veritiera la quale andrà ad influenzare in maniera negativa l'apprendimento, ciononostante in letteratura scientifica risulta essere una tecnica da preferire al sotto campionamento.

Prima di passare all'addestramento dei modelli offriremo una visione di come il dataset è mutato dopo l'applicazione di ADASYN:



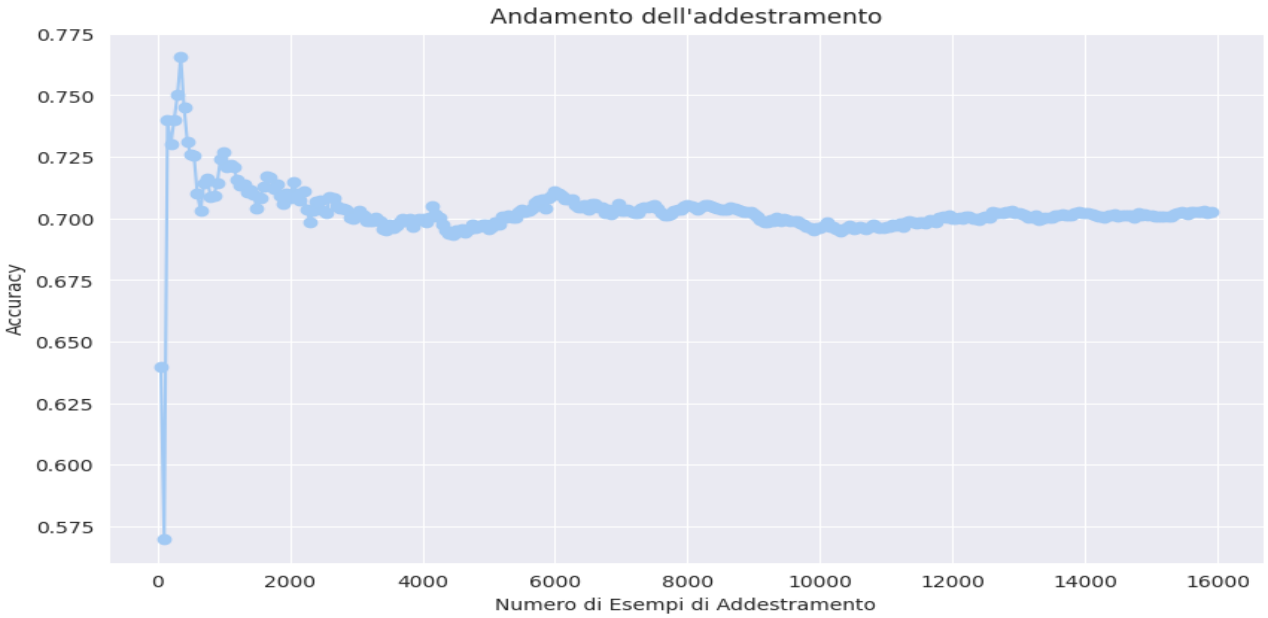


Anche in questo caso lo sbilanciamento è stato risolto la piccola percentuale di differenza è praticamente ininfluente sull'apprendimento dei modelli.

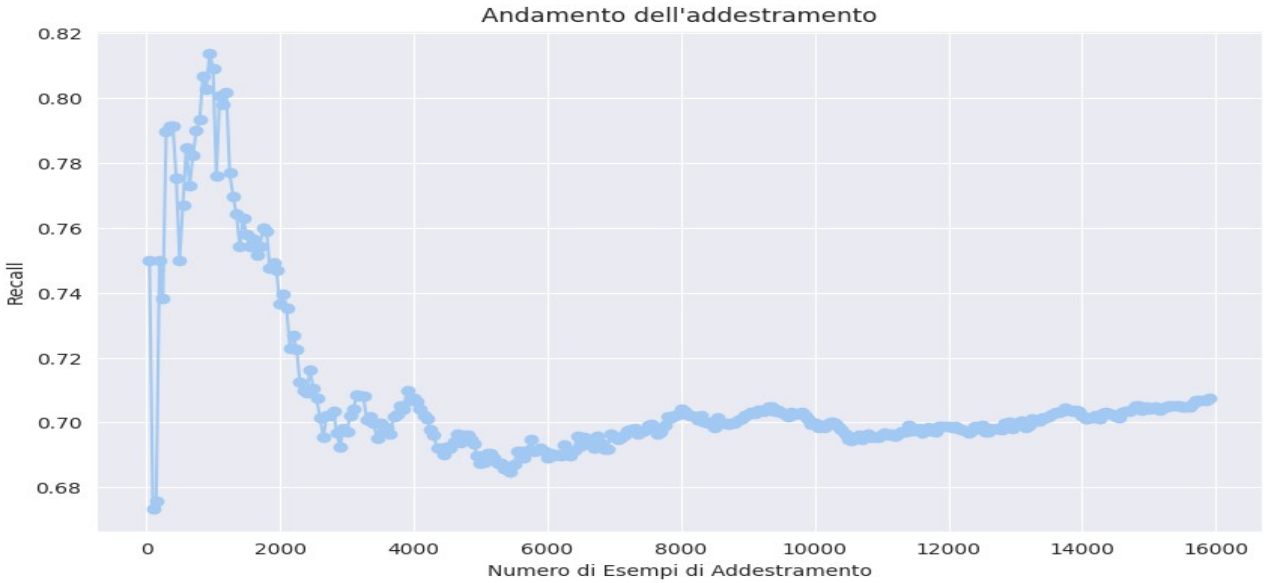
Passiamo a vedere l'apprendimento dei modelli e le loro performance:

Curva di accuracy per regressione logistica

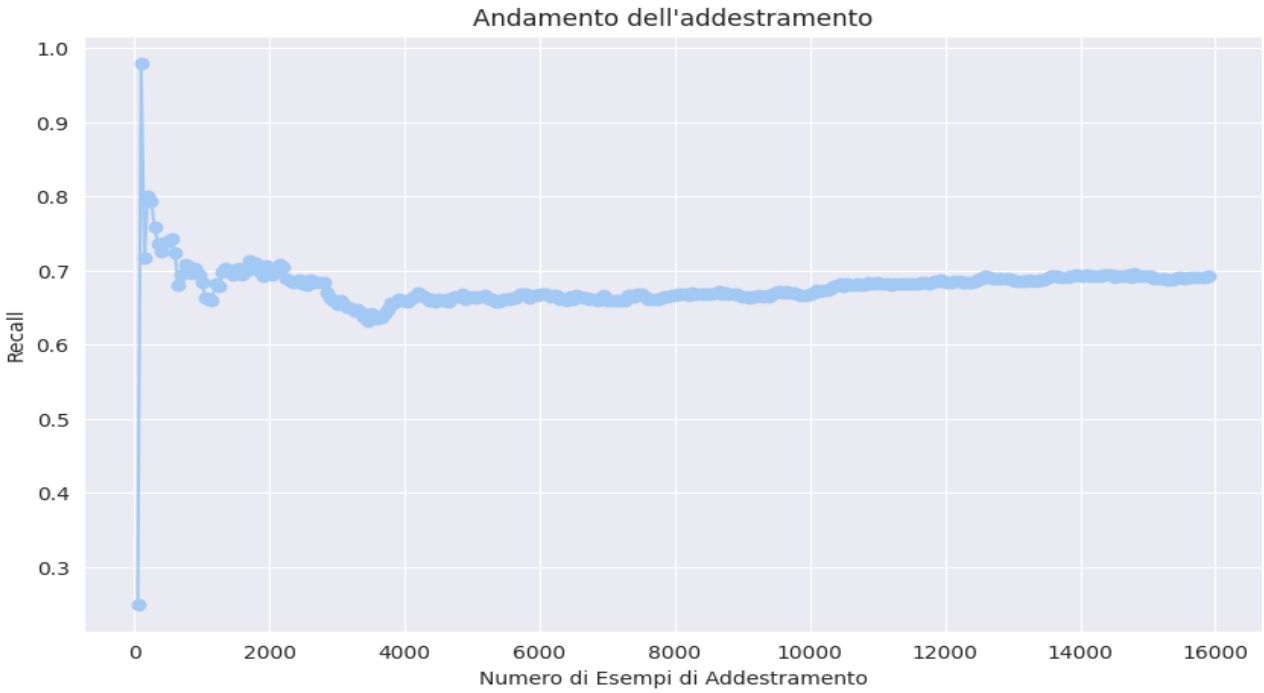
Curva di accuracy per gaussian naive bayes



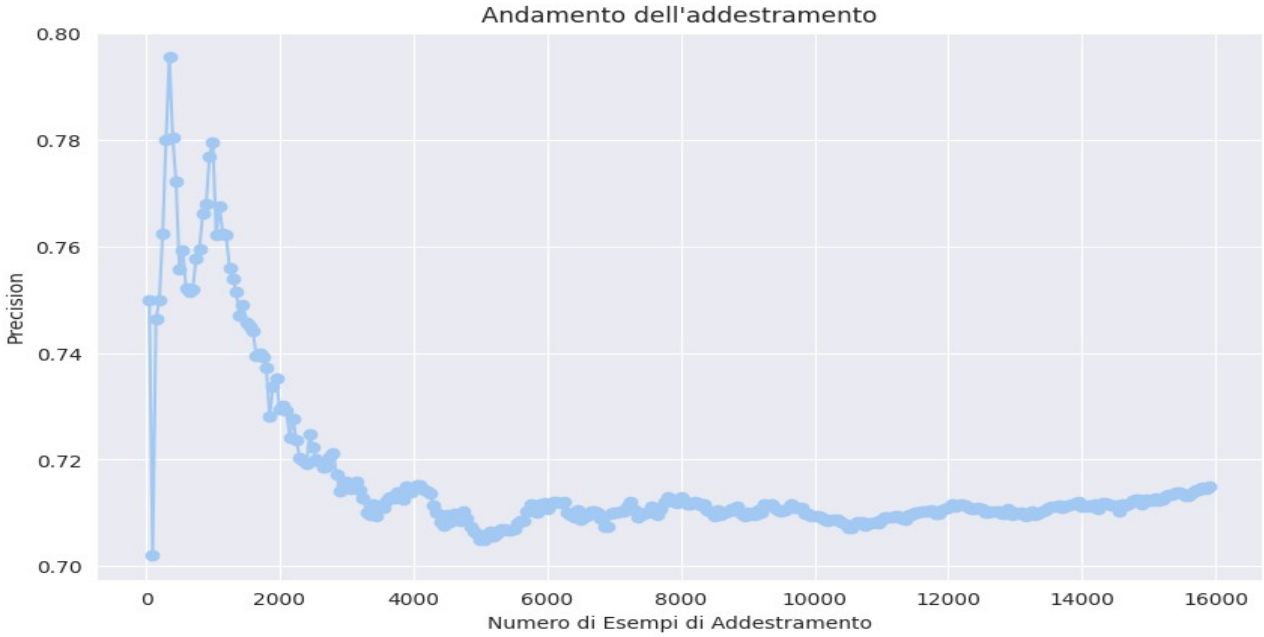
Curva di recall per la regressione logistica



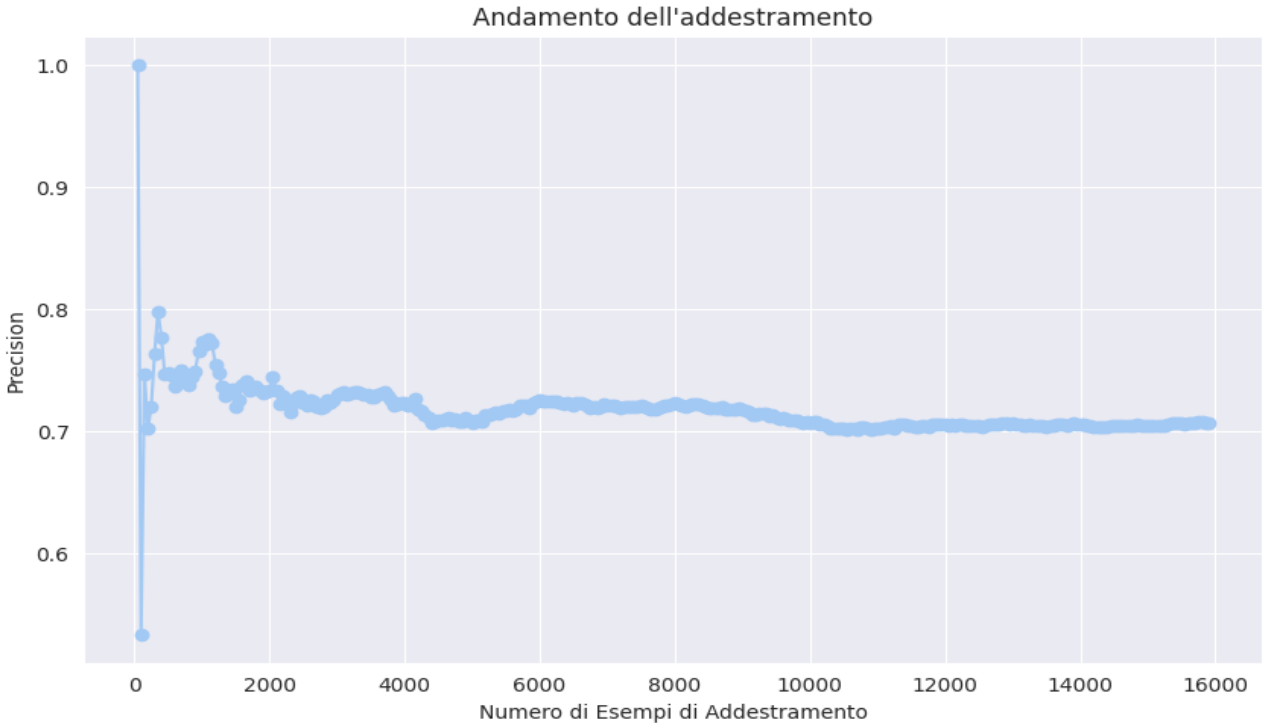
Curva di recall per il gaussian naive bayes



Curva di precision per la regressione logistica



Curva di precisione per il gaussian naive bayes

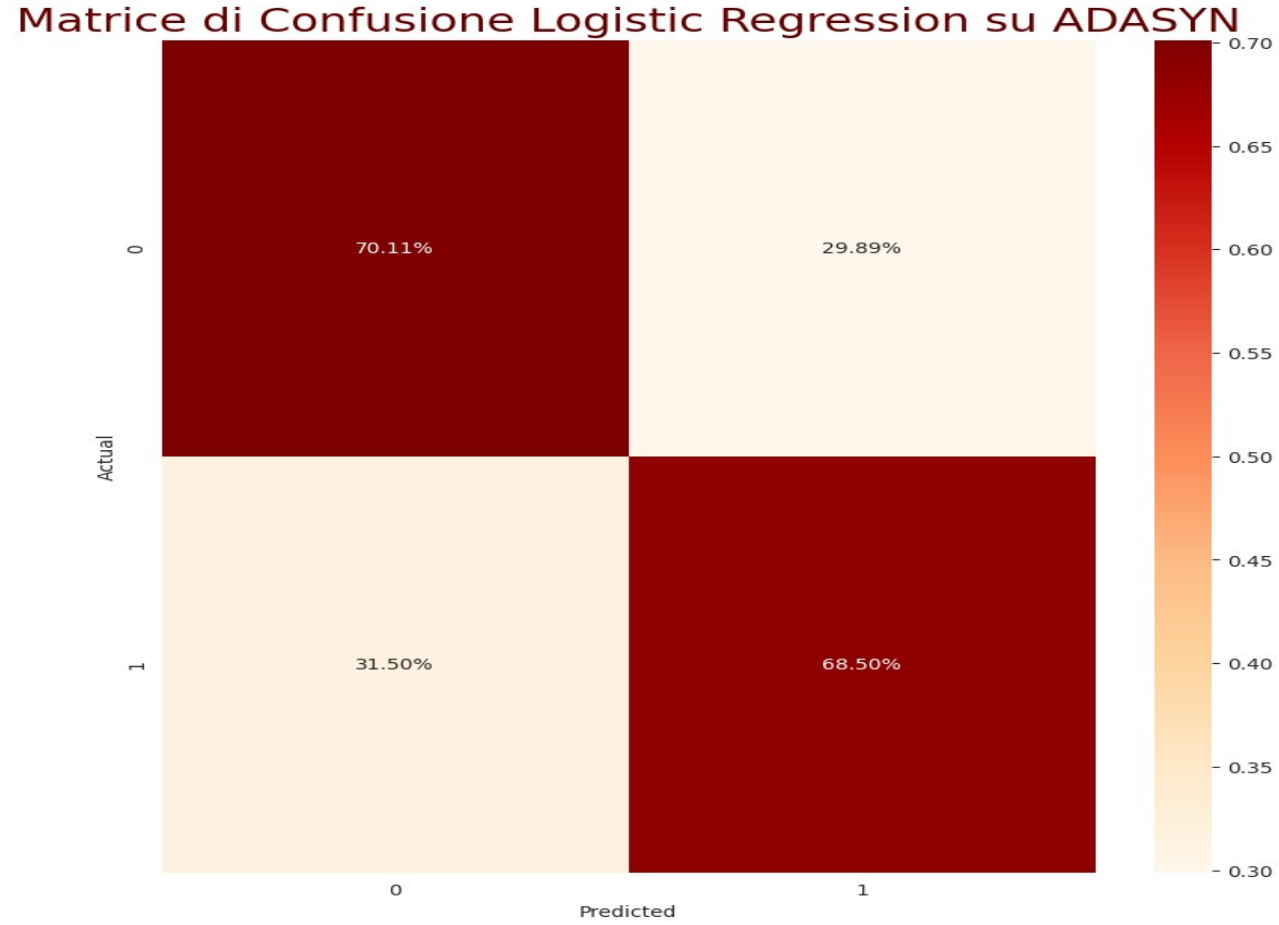


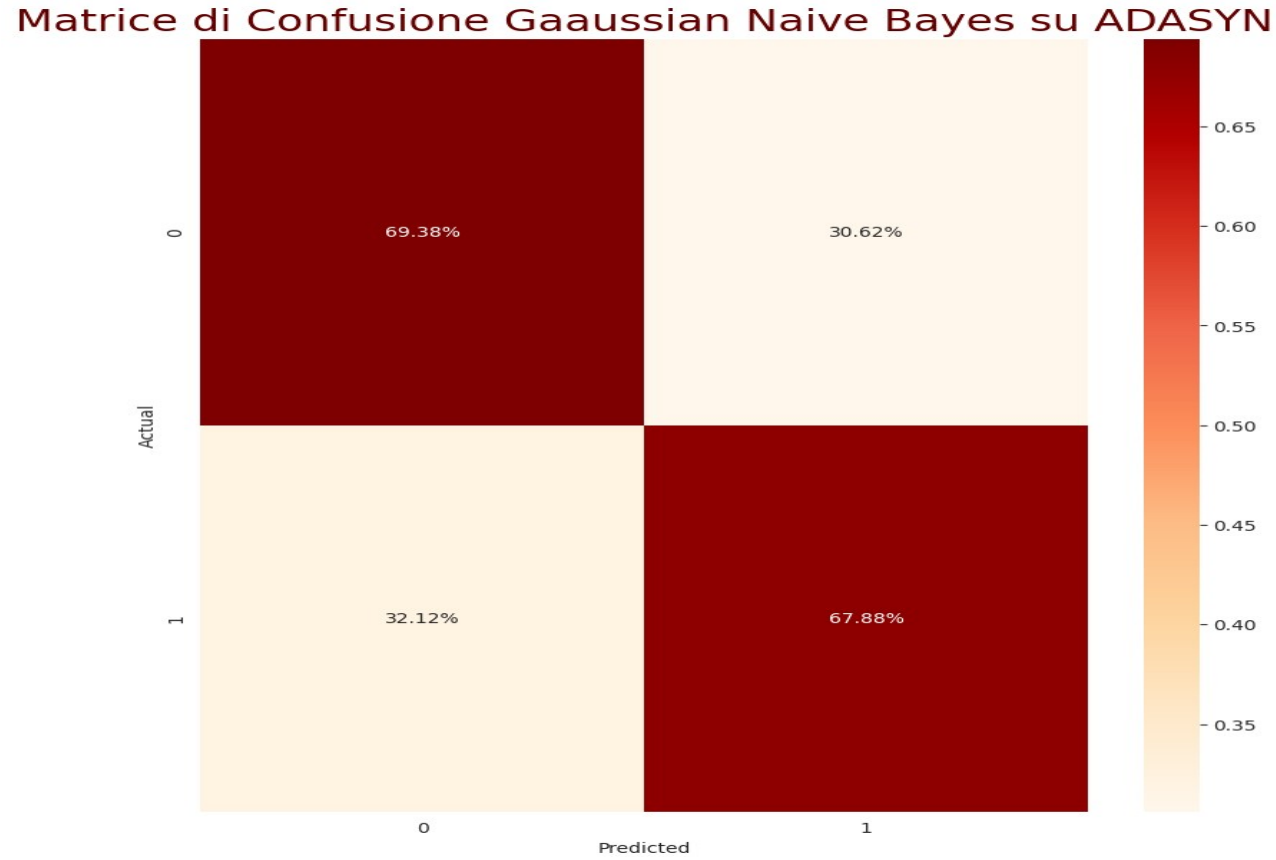
Andiamo a confrontare le restanti metriche e comparare il report di classificazione per i due modelli:

```
_____________LOGISTIC REGRESSION ON ADASYN_____________
-----------------------Accuracy-----------------------

0.6931302182510619

-----------------Classification_Report----------------

              precision    recall  f1-score   support

           0       0.69      0.70      0.70      3446
           1       0.69      0.69      0.69      3381

    accuracy                           0.69      6827
   macro avg       0.69      0.69      0.69      6827
weighted avg       0.69      0.69      0.69      6827
```

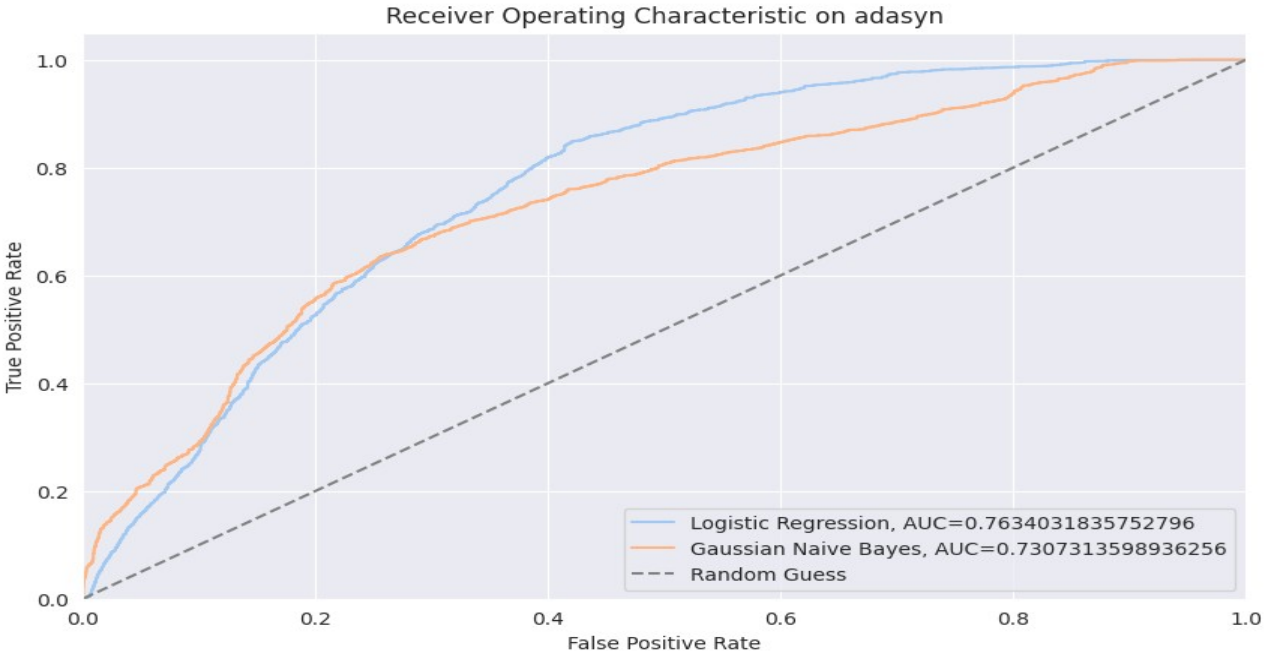
```
_____________GAUSSIAN NAIVE BAYESA ON ADASYN____________
-----------------------Accuracy-----------------------

0.6863922660026366

-----------------Classification_Report----------------

              precision    recall  f1-score   support

           0       0.69      0.69      0.69      3446
           1       0.69      0.68      0.68      3381

    accuracy                           0.69      6827
   macro avg       0.69      0.69      0.69      6827
weighted avg       0.69      0.69      0.69      6827
```

Riportiamo di seguito le matrici di confusione inerenti ai modelli:


Matrice di Confusione Logistic Regression su ADASYN
70.11%
29.89%
31.50%
68.50%
Actual
Predicted
0
1
0
1
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30



Infine, andiamo a confrontare i due classificatore con score ROC su di un grafico:

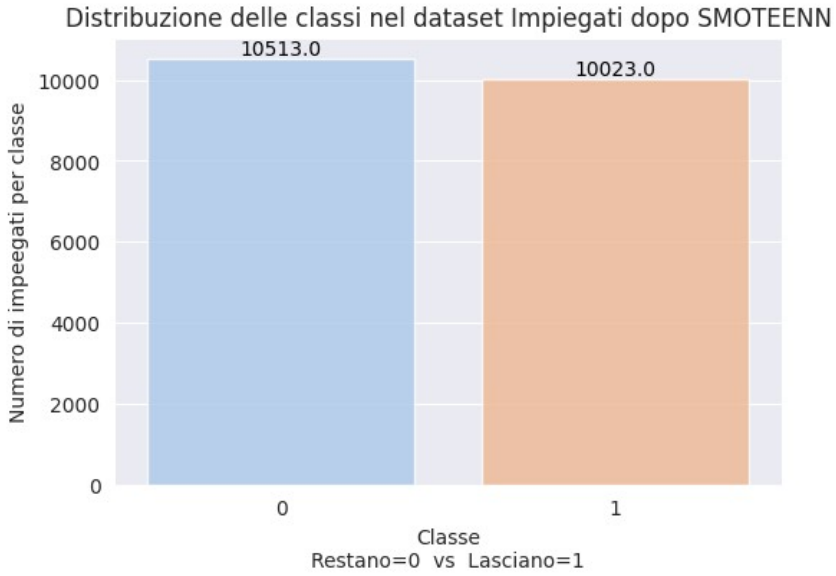


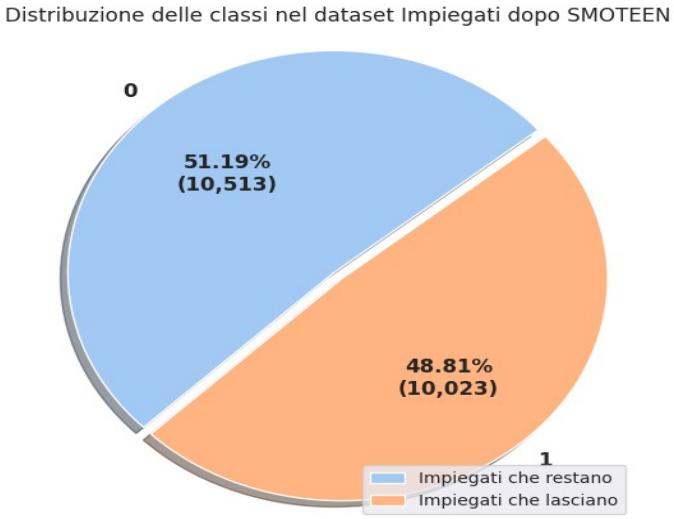
A discapito di quanto notato dalle curve di apprendimento il gaussian naive bayes performa peggio della regressione logistica che risulta essere il modello vincente dopo aver trattato il dataset con ADASYN.

Synthetic Minority Over-sampling Technique Edited Nearest Neighbors

Questa tecnica risulta essere un approccio misto ovvero l'unione di una tecnica di under sampling affiancata ad una di over sampling per cercare di bilanciare gli effetti avversi dell'utilizzo di entrambe le due tecniche. In particolare, lo SMOTE-ENN si basa sull'utilizzo del ENN tecnica di under sampling già citata in precedenza e dello SMOTE tecnica di over sampling la quale va a prendere le osservazioni più vicine in base alla distanza euclidea tra quelle delle classe minoritaria, ne calcola fa la differenza tra i due vettori di features e la moltiplica per un numero casuale tra 0 e 1. In altre parole, si va ad applicare un perturbamento alla distanza tra due punti della classe minoritaria. Così facendo si creano osservazioni artificiali che accrescono il patrimonio di dati, ma non ne modificano troppo il valore.

Prima di passare all'addestramento dei modelli su di un data set trattato con la tecnica dello smote-enn diamo una visualizzazione rapida ai dati per capire quali sono stati gli effetti dell'applicazione dello SMOTE-ENN:



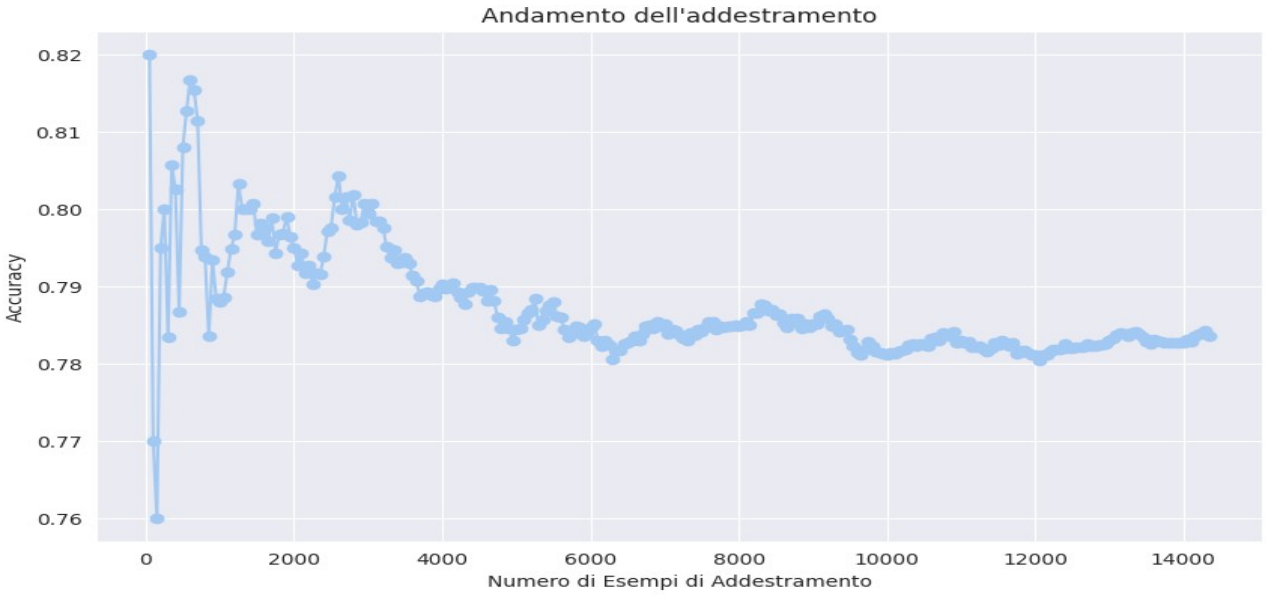


Anche quest'ultima tecnica di campionamento è riuscita a risolvere lo sbilanciamento del dataset rendendo le differenze pressappoco nulle.

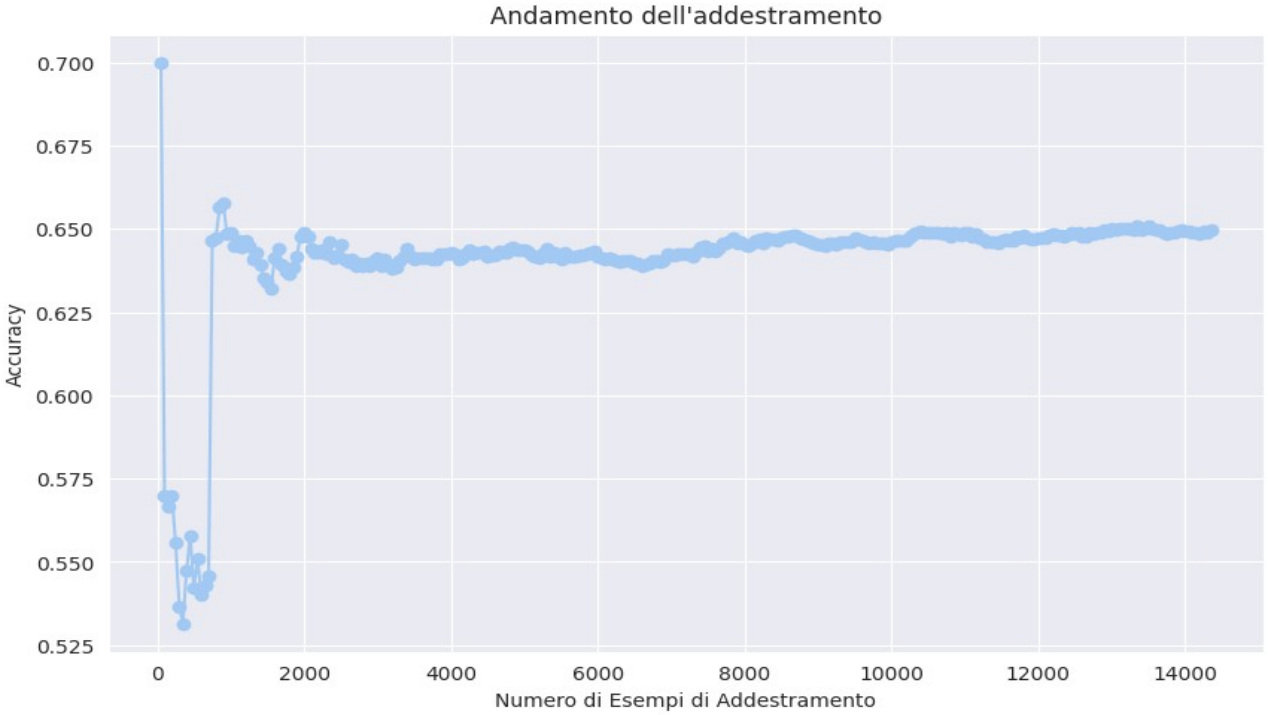
Passiamo a vedere l'apprendimento dei modelli e le loro performance anche in questo caso sono state scelte come mezzo valutativo le tre principali metriche dei classificatori ovvero precision, recall e accurancy e ricordiamo che questa volta utilizziamo anche l'accuracy poiché l'applicazione dello smote-enn ha

bilanciato le classi, in caso contrario in presenza di sbilanciamento non avremmo usato l'accuracy.

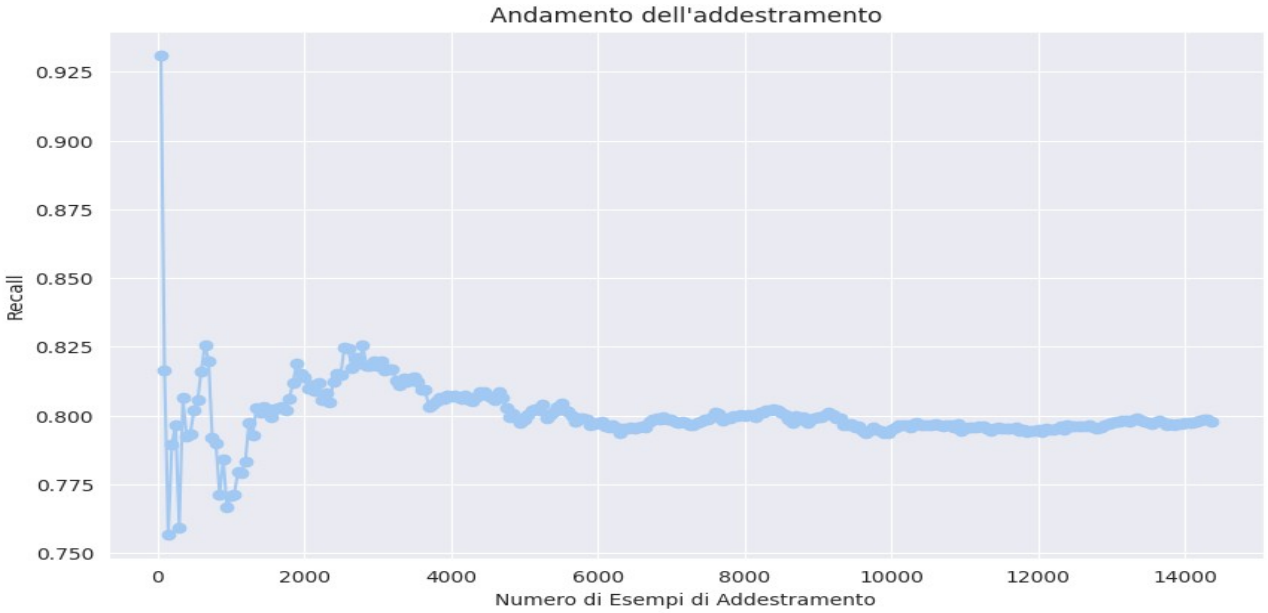
Curva di accuracy della regressione logistica



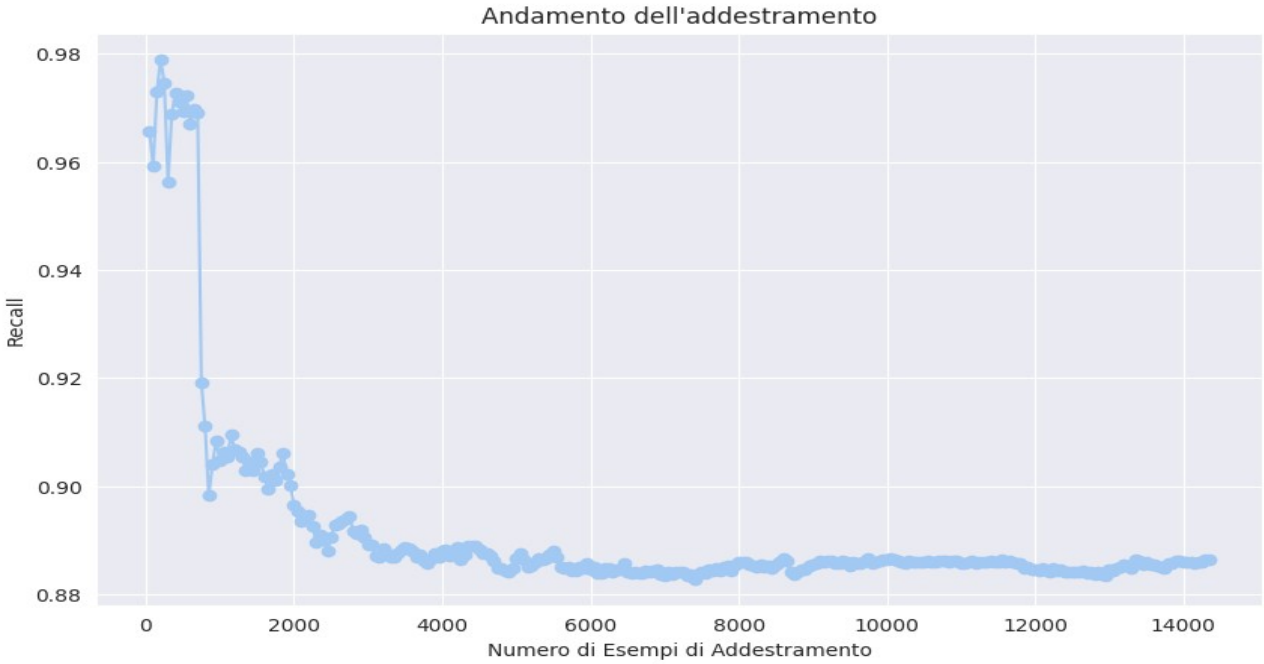
Curva da accuracy del gaussian naive bayes



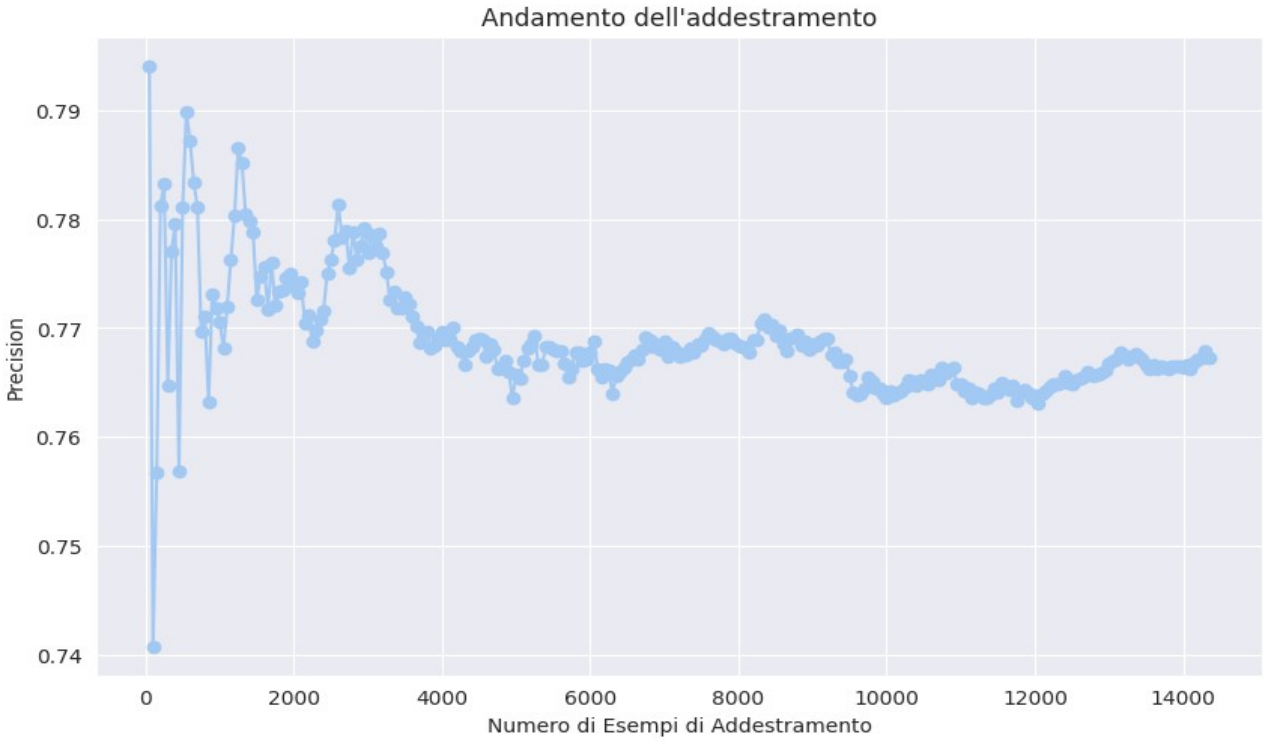
Curva di recall della regressione logistica



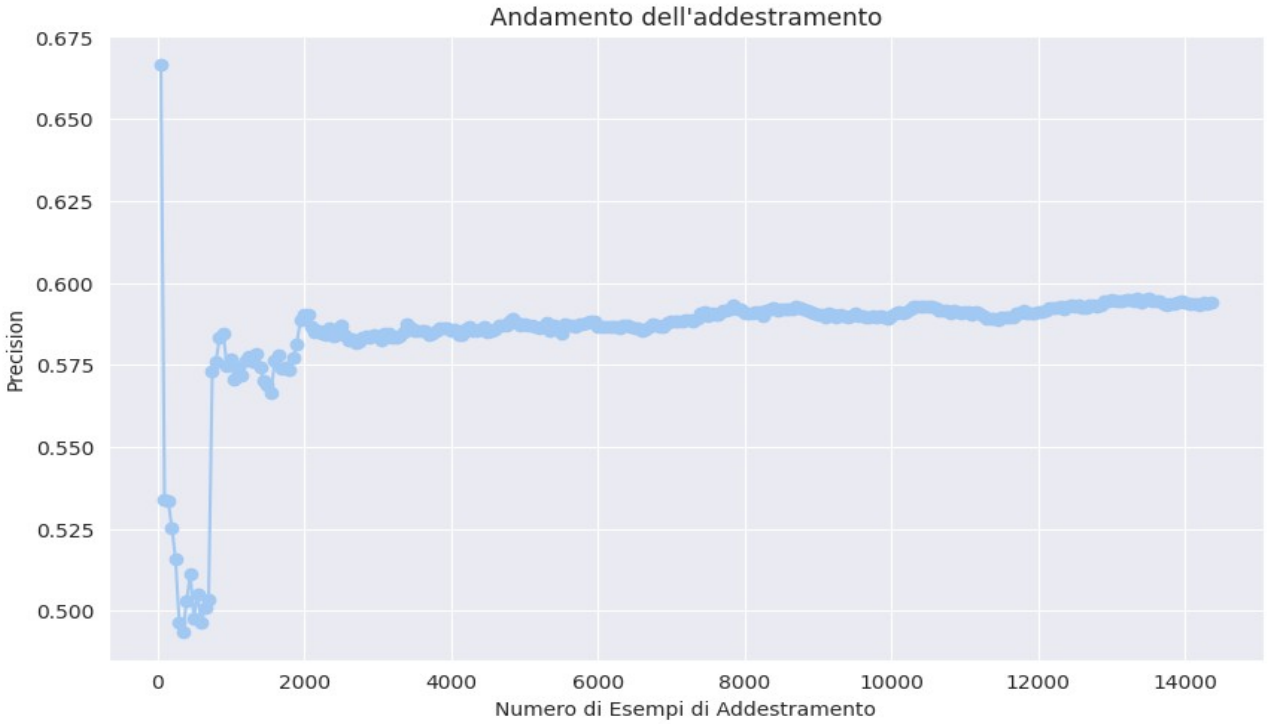
Curva di recall del gaussian naive bayes



Curva di precision della regressione logistica



Curva di precision del gaussian naive bayes



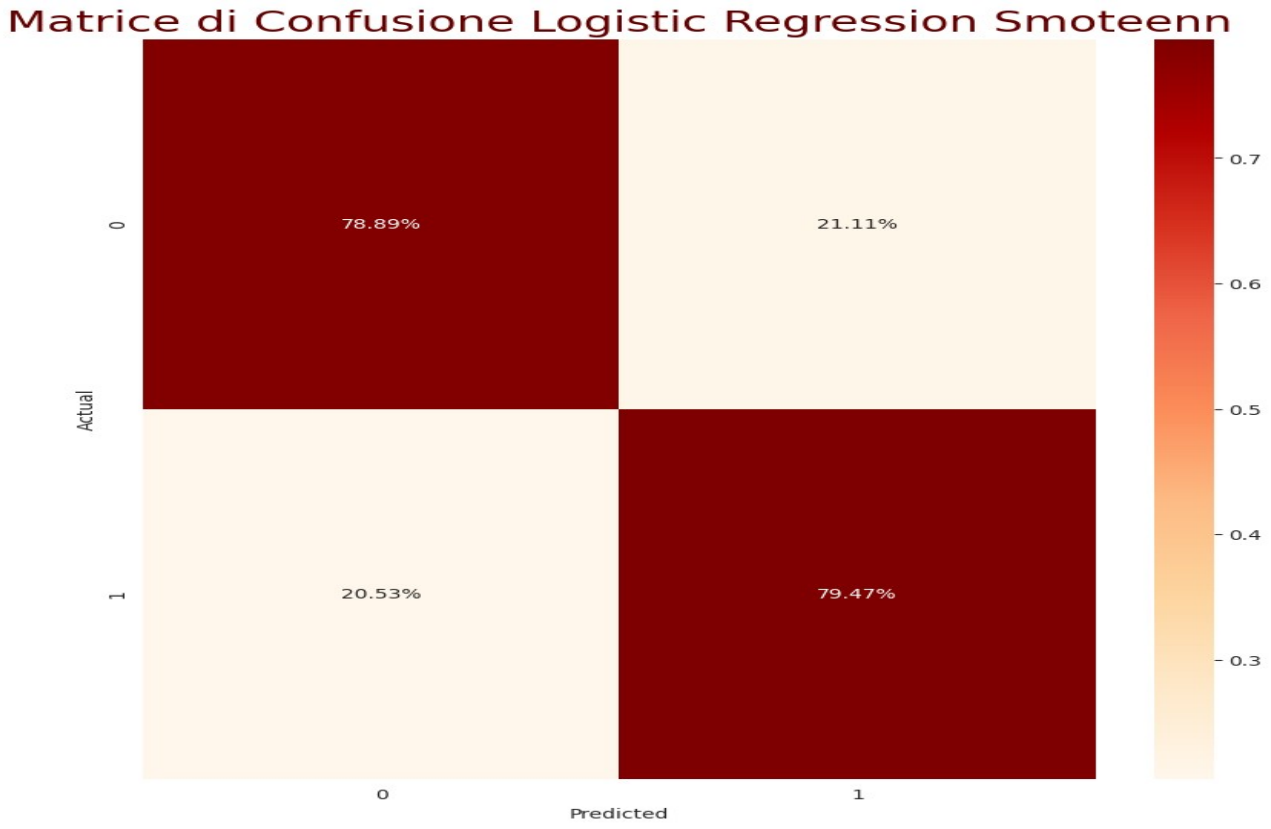
Analizzando i grafici sullo studio dell'addestramento ci rendiamo conto di come i modelli siano stati agevolanti infatti le curve oscillano molto di meno se paragonate alle curve di apprendimento tracciate con l'applicazione delle precedenti tecniche di campionamento. Quello che ci ha colpito però è la velocità della stabilizzazione dell'apprendimento che avviene già con i primi 2000 esempi, il che ci preoccupa perché lascia intendere che tutta l'informazione sia racchiusa in quei pochi esempi e il resto ha scarsa importanza.

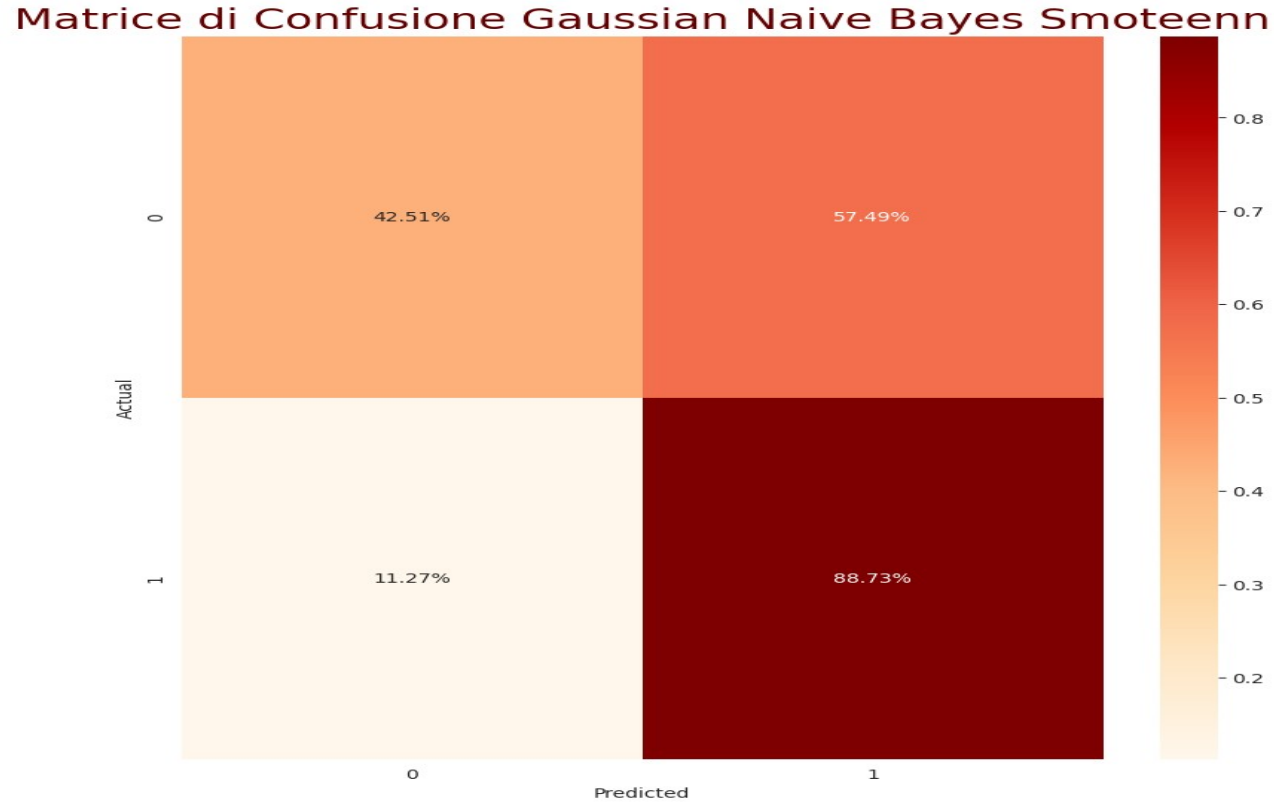
Andiamo a confrontare le restanti metriche e comparare il report di classificazione per i due modelli:

```
__________LOGISTIC REGRESSION ON SMOTEENN_____________
------------------------Accuracy------------------------

0.7917545852945951

------------------Classification_Report-----------------

              precision    recall  f1-score   support

           0       0.80      0.79      0.79      3136
           1       0.78      0.79      0.79      3025

    accuracy                           0.79      6161
   macro avg       0.79      0.79      0.79      6161
weighted avg       0.79      0.79      0.79      6161
```

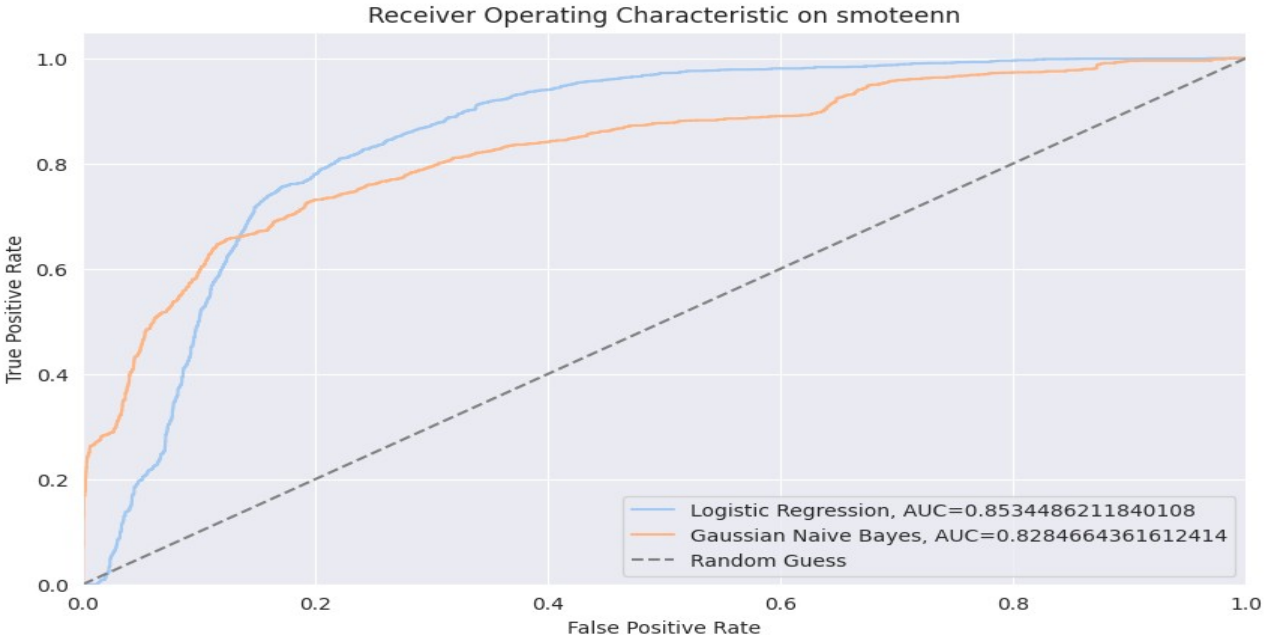
```
___________GAUSSIAN NAIVE BAYES ON SMOTEENN___________
------------------------Accuracy------------------------

0.6520045447167667

------------------Classification_Report-----------------

              precision    recall  f1-score   support

           0       0.80      0.43      0.55      3136
           1       0.60      0.89      0.71      3025

    accuracy                           0.65      6161
   macro avg       0.70      0.66      0.63      6161
weighted avg       0.70      0.65      0.63      6161
```

Riportiamo di seguito le matrici di confusione inerenti ai modelli:



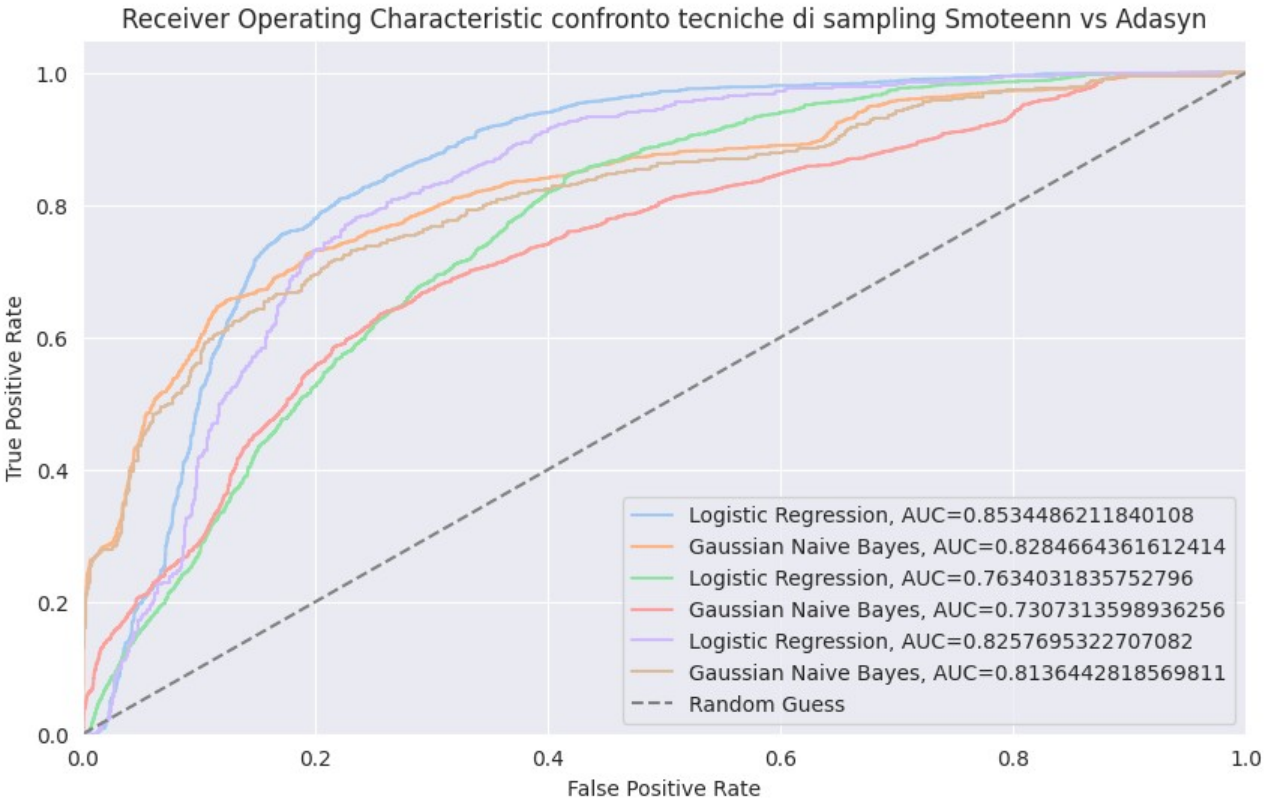


Infine, andiamo a confrontare i due classificatore con score ROC su di un grafico:



Dagli esperimenti di addestramento effettuati sui modelli e dall'accurata analisi effettuata con diverse metriche giungiamo a tirare le conclusioni su quello che sia stato il miglior modello capace di gestire il task. La scelta anche questa volta non ci sembra scontata e si riapre nuovamente il panorama rischio contro non rischio affrontato dopo l'addestramento dei modelli sul random under sampling. Di fatto i due modelli non hanno troppe differenze specialmente se pensiamo che per noi il recall è la metrica che stiamo preferendo per la tipologia del task per cui la scelta risulta ardua tuttavia ci sembra che in questa situazione il Gaussian Naive Bayes stia facendo leggermente peggio e se dovessimo scegliere la scelta ricadrebbe sulla regressione logistica la quale non è una scelta ottimale, ma supera un banale predittore che andrebbe a classificare un esempio con una rapporto 3:1 proprio lo stesso rapporto che caratterizzava lo sbilanciamento dei dati iniziali per cui la Regressione Logistica ci sembra essere una scelta sensata.

Confronto finale sulle tre tecniche di campionamento utilizzate:



In ordine nella leggenda è possibile osservare che i modelli sono disposti di due in due e a partire dai primi in alto abbiamo gli score ROC calcolati dopo applicazione dello smote-enn e addestramento in seguito a scendere ritroviamo quelli su adasyn e in fine quelli inerenti al rus con sorpresa notiamo che i modelli si sono comportati peggio sull'adasyn rispetto al rus cosa alquanto insolito, mentre la migliore tecnica capace di dare il migliore incettivo all'apprendimento dei modelli è risultata essere lo smote-enn dove i modelli hanno toccato i migliori punteggi.

ENSEMBLE LEARNING

L'ensemble learning risulta essere un ottimo approccio per affrontare dataset sbilanciati e cercare di apprendere al meglio il concetto che si voglia la macchina apprenda, per tali ragioni andremo ad addestrare alcuni trai più importanti modelli di ensemble. In letteratura scientifica è nota l'efficacia di questi modelli di apprendimento poiché la loro forza sta nella cooperazione, o meglio le loro ottime performance sono basate su principi della statistica e l'idea generale e che avere l'idea di più classificatori anziché di un solo

classificatore, non solo permette di aumentare le performance, visto che dove uno dei classificatori sbaglia gli altri possono sopperire, ma permette anche di riporre maggiore confidenza nel sistema poiché sopportato da più opinioni.

Le tecniche di ensemble si dividono in:

- Baggging
- Boosting
- Stacking

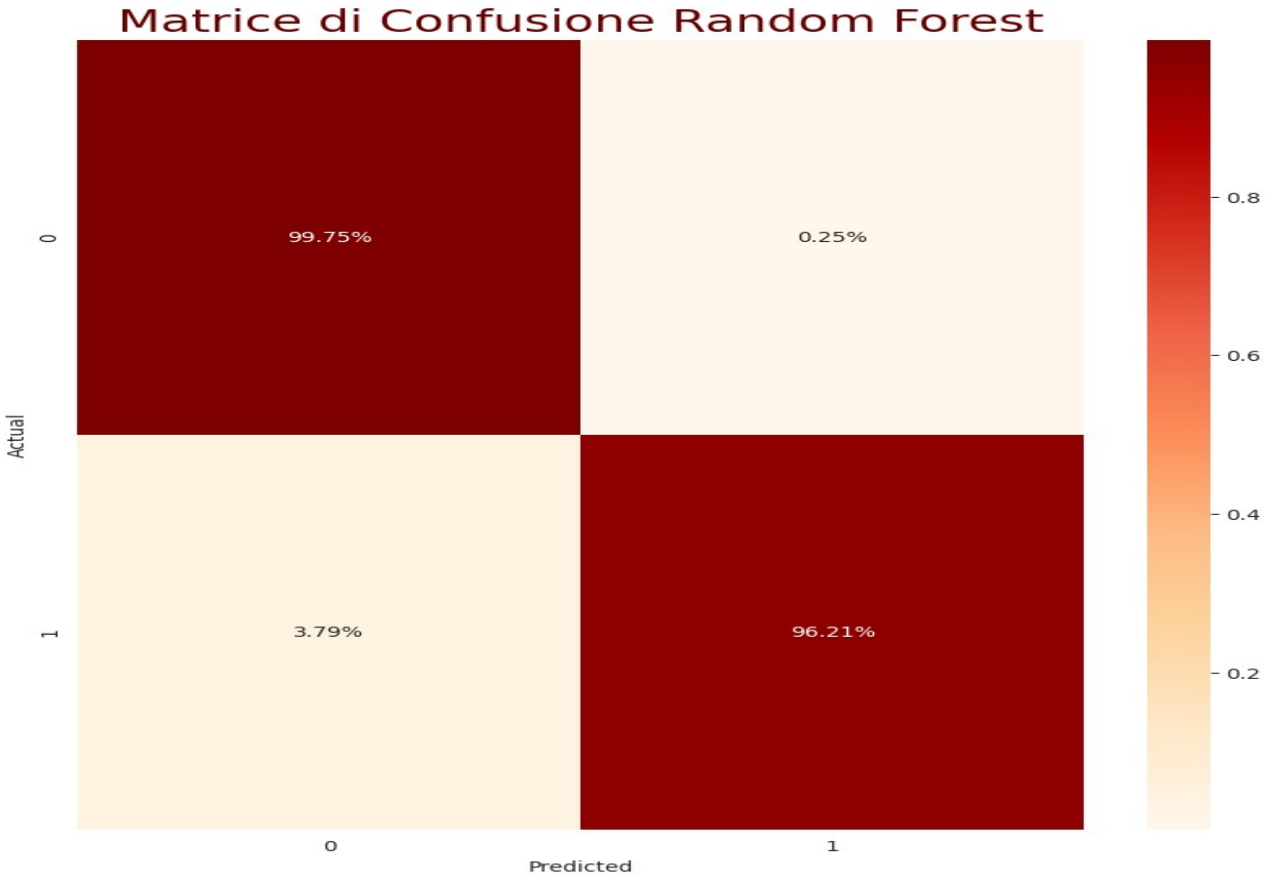
BAGGING: RANDOM FOREST

Ogniuna presenta caratteristiche e peculiarità di apprendimento differenti, per lo studio iniziamo scegliendo la tecnica di Bagging che sta per Boostrap Aggregating l'idea è addestrare più classificatore su differenti training set creati tutti a partire dai nostri dati di partenza per mezzo di un processo di campionamento randomico con reinserimento degli esempi selezionati per un campione. Alla fine, i modelli addestrati produrranno delle predizioni le quali saranno aggregate tramite una funzione di aggregazione nella predizione finale. Il Bagging è la fusione di due tecniche che sono il Boostrap che consiste in effettuare un campionamento con rimpiazzo degli esempi presenti nel dataset utilizzato per il training al fine di produrre n differenti dataset di training i quali serviranno per l'addestramento indipendente di ogniuno dei classificatori base del random forest che altro non sono che alberi di decisione, questo permetterà di avere una notevole varianza tra i modelli dato che questi durante l'addestramento fanno naturalmente un processo di feature selection atto a scegliere le feature con maggiore potere informativo ovvero quelle che permettono di aumentare l'information gain per ridurre l'incertezza sulla classificazione degli esempi. Questo processo porta a diversificazione degli alberi di decisione poiché essi saranno addestrati su dataset differenti che presentano distribuzioni dei dati simili, ma con proporzioni diversificate che portano alla varianza degli alberi ed è in questa varianza che risiede la forza del Bagging. Infine, riportiamo che tra gli algoritmi di Bagging i più noti in letteratura sono il Random Forest e l'Extra Tree di seguito riportiamo gli esperimenti di addestramento condotti sul Random Forest, annotando la peculiarità che i modelli di Bagging sono modelli parallelizzabili durante l'addestramento e questo permette di gestire la complessità di tali algoritmi in tempi utili.

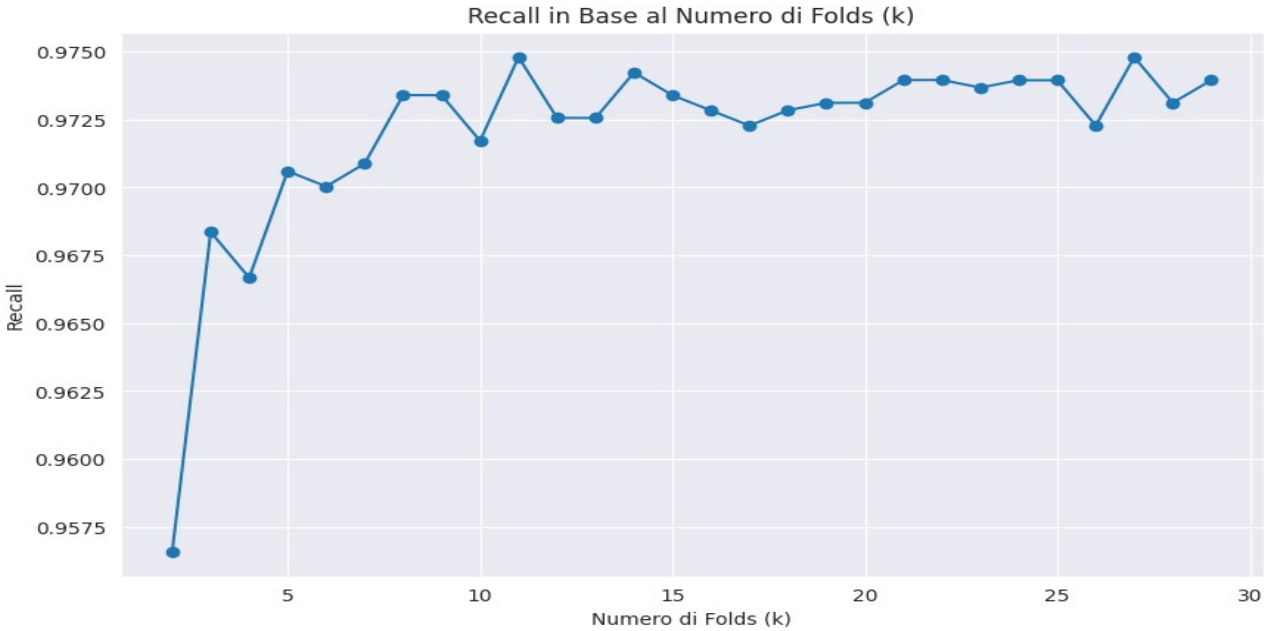
Le performance del Random Forest si sono dimostrate ottime anche fin troppo buone da considerarsi veritiere e questo ha portato ad alcune riflessioni sulla veridicità dei dati raccolti da terzi per questo studio.

```
_________________RANDOM FOREST PERFORMANCE_________________
-----------------------Accuracy-------------------------

0.9890666666666666

------------------Classification_Report------------------

              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2853
           1       0.99      0.96      0.98       897

    accuracy                           0.99      3750
   macro avg       0.99      0.98      0.98      3750
weighted avg       0.99      0.99      0.99      3750
```

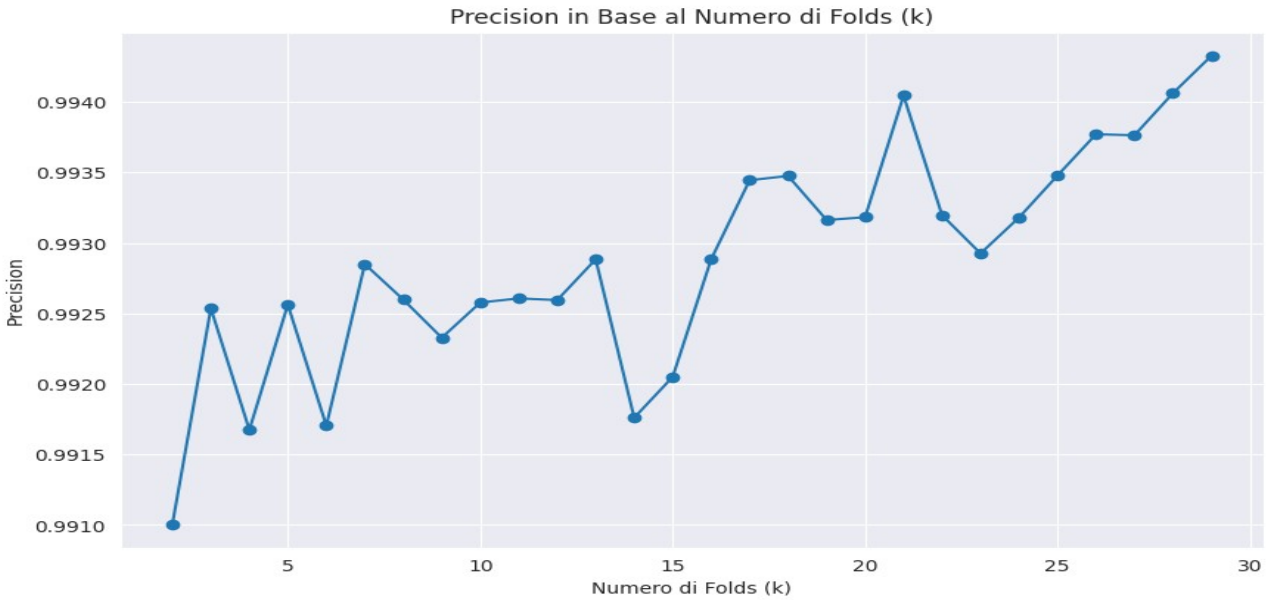
La matrice di correlazione del random forest è la seguente:



Curva di ottimizzazione recall-fold del Random Forest



Curva di ottimizzazione Precision-fold del Random Forest



Le curve di ottimizzazione dei fold della cross validation sono abbastanza in disaccordo dato che il recall raggiunge il massimo stabilizzandosi su un fold k=8, mentre per la precision sembra che più aumentiamo i fold più i risultati migliorino chiaramente per la legge di Strugges sul calcolo teorico del miglior k per la cross validation che nel nostro caso specifico è pari a 15 non ci sentiamo di proseguire con esperimenti per fold maggiori di k=30 per cui essendo che per il recall la stabilizzazione si raggiunge a partire da k=8 e la precision un massimo locale a k=21 scegliamo proprio questo con il k ottimale per la cross validation del Random Forest.

BOOSTING: ADABOOST

Questa tecnica di ensemble learning differisce dalla prima su differenti aspetti in prima istanza sull'idea di apprendimento, infatti mentre per il bagging l'idea è quella di addestrare più modelli base in modo indipendente e in seguito aggregarne le predizioni in questo caso l'idea si basa su di un miglioramento progressivo dei classificatori di base che durante l'esecuzione di tale algoritmo si vanno migliorando di modello in modello. Quindi un modello di Boosting basa il suo apprendimento su di un cosiddetto "weak learner" che altro non è che il

modello di classificazione base utilizzato per l'ensemble, ogni uno di questi modelli viene ad essere addestrato e le sue predizioni sono usate dal seguente modello successivo che sarà addestrato utilizzando dei pesi dati dalle predizioni del suo precedessore, con lo scopo di andare a migliorare le predizioni precedenti specialmente nei casi di mis-classificazione. Questa natura di addestramento sequenziale che va a migliorare di modello in modello l'ensemble è la chiave per capire la seconda differenza sostanziale che si è tra i metodi di Bagging e quelli di Boosting infatti mentre i primi sono algoritmi altamente parallelizzabili ovvero possono andare in esecuzione sui più processori o su più macchine comportando una riduzione del tempo di esecuzione per l'addestramento grazie alla ripartizione del carico con i metodi di Boosting non è possibile ottenere questi benefici poiché l'algoritmo è sequenziale e necessita del precedente classificatore addestrato per migliorare il successivo sfruttando le predizione di quello già addestrato in precedenza.

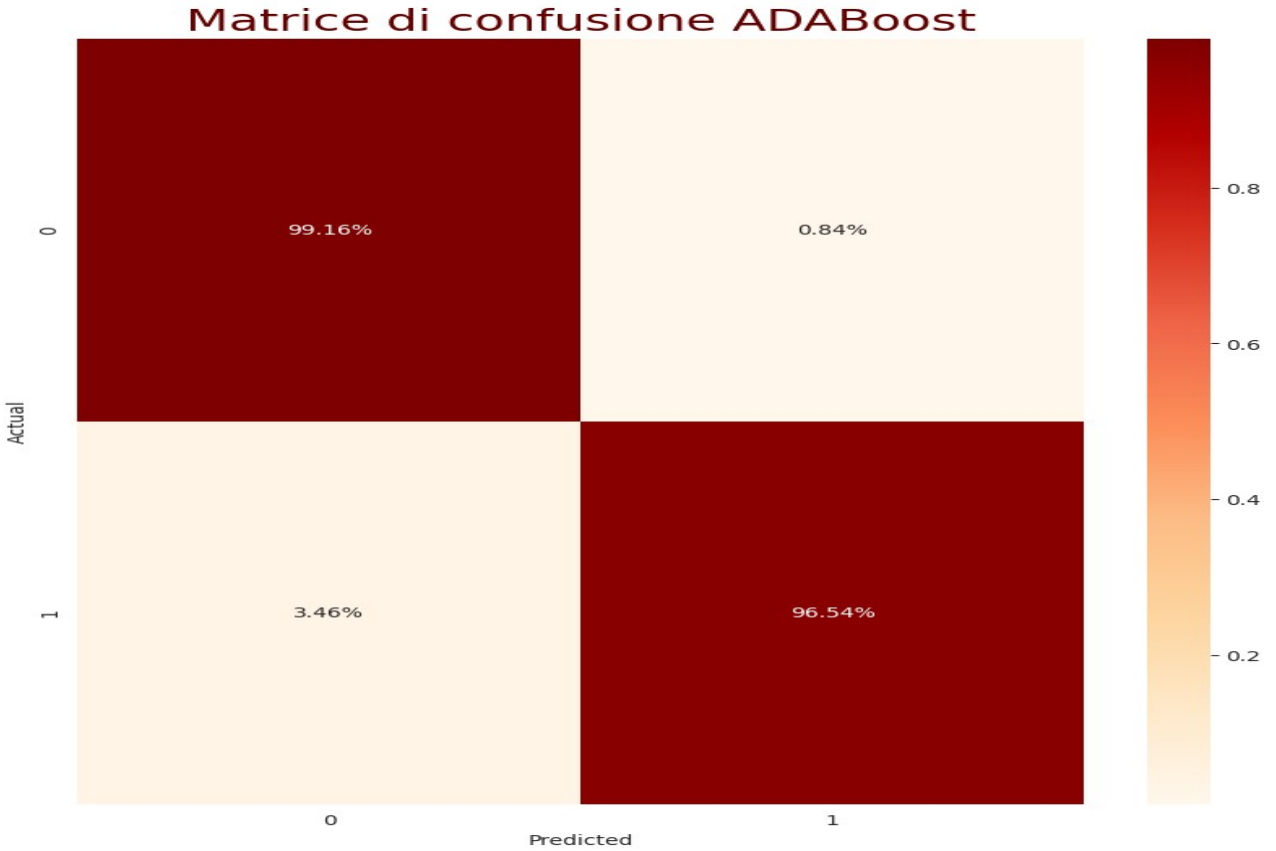
Tra i vari modelli di Boosting i più conosciuti sono lo XGBoost e l'ADABost quest'ultimo è il modello che andremo ad addestrare e valutare per analizzarne i risultati ottenuti.

Di seguito riportiamo i risultati ottenuti con l'ADABoost:

```
____________________ADABOOST PERFORMANCE____________________
------------------------Accuracy------------------------

0.9853333333333333

------------------Classification_Report-----------------

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2853
           1       0.97      0.97      0.97       897

    accuracy                           0.99      3750
   macro avg       0.98      0.98      0.98      3750
weighted avg       0.99      0.99      0.99      3750
```

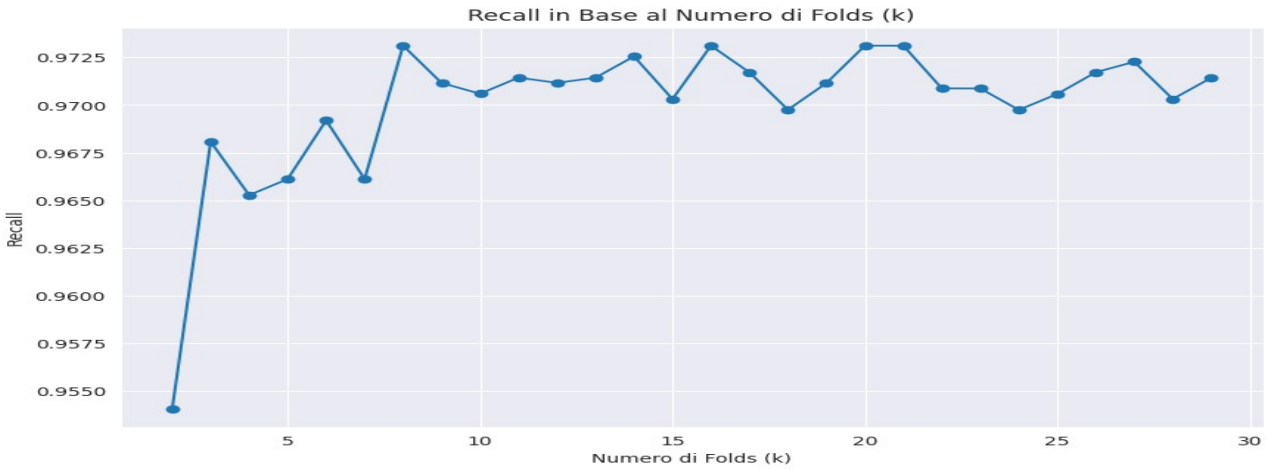
Anche in questo caso i risultati sembrano essere ottimi, difficilmente si può sperare di fare meglio, questo tuttavia ha posto il dubbia e le riflessioni precedentemente affrontate con il random forest sulla natura dei dati.

Riportiamo la matrice di confusione del ADABoost per una comprensione compatta delle prestazioni:

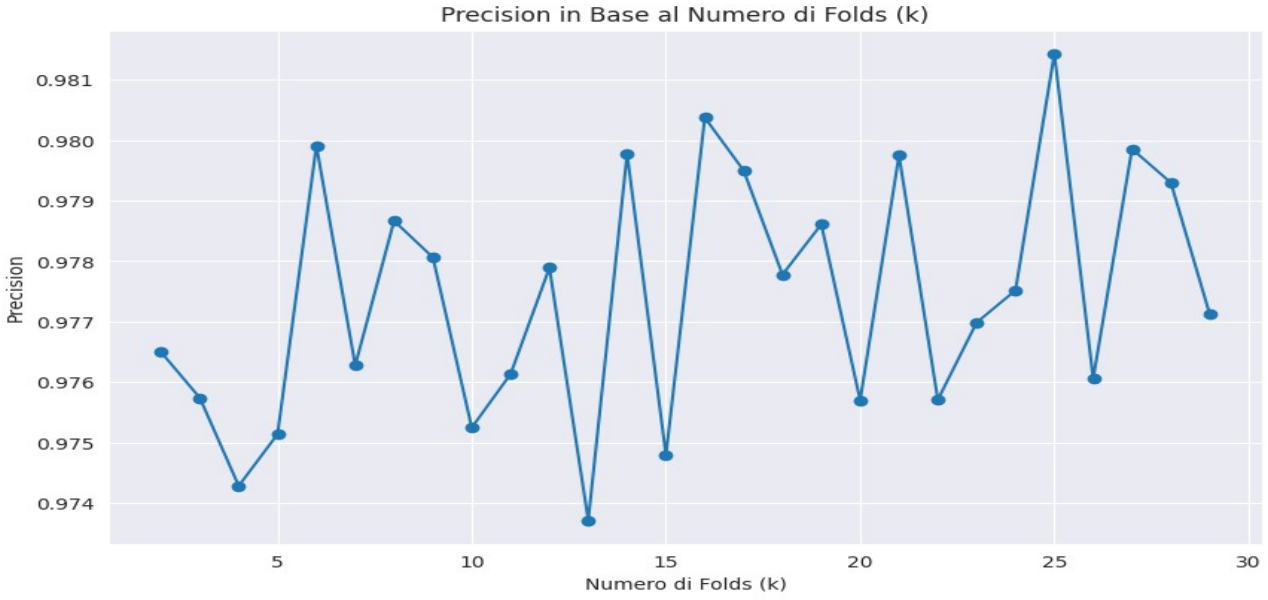


Di seguito si riportano alcune sperimentazioni fatte sull'ADABoost con lo scopo di capire come il modello fosse influenzato nelle performance dal processo di cross validation dove si cerca di trovare il k migliore per il fold da usare per ottimizzare le performance dell'ADABoost. Gli sperimenti di ottimizzazioni sono stati fatti prendendo in considerazione le due metriche principali di recall e precision.

Curve di ottimizzazione recall-fold



Curve di ottimizzazione precision-fold



Questa volta nello studio del miglior k per i fold della cross validation per l'ADABoost le cose si semplificano notevolmente anche se a prima occhiata a

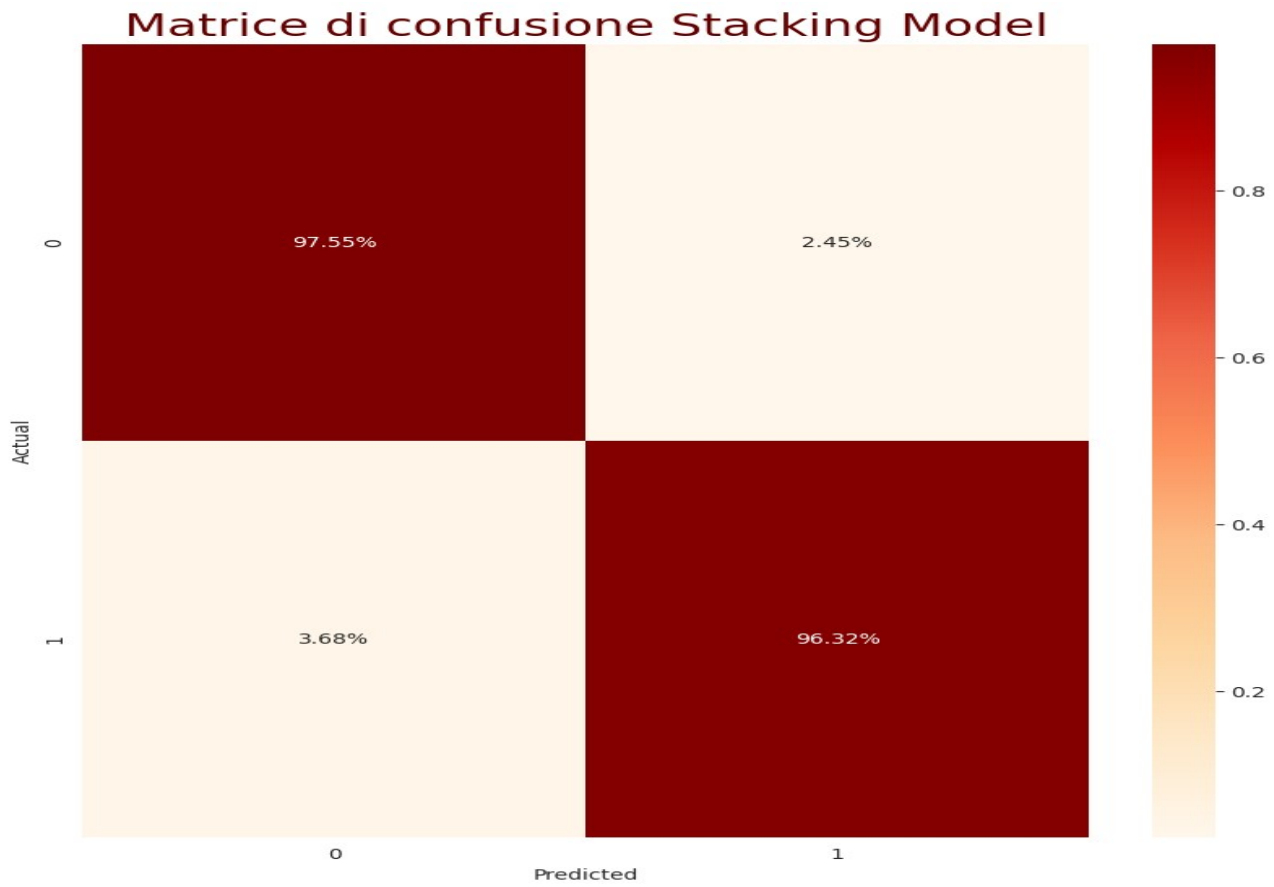
guardare il grafico sembrerebbe un delirio di sali e scendi. In verità guardando attentamente alla scala delle percentuali con le quali viene ad essere calcolata la precision ci si rende conto che il grafo è verro che sta oscillando repentinamente ma su di un intervallo che ha ampiezza 0.005%, ovvero è del tutto insignificante e se aumentassimo la scala vedremo pressappoco una linea retta. In definitiva qui qualsiasi k testato nel range da 2 a 30 andrebbe bene e per tale ragione, considerando anche che il recall come per il Random Forest si stabilizza al toccare di k=8 a valori maggiore del 97%, scegliamo un k che abbassi la complessità di computazione e quindi per l'ADABoost k=8 è il miglior numero di fold utilizzabile per la cross validation.

STACKING

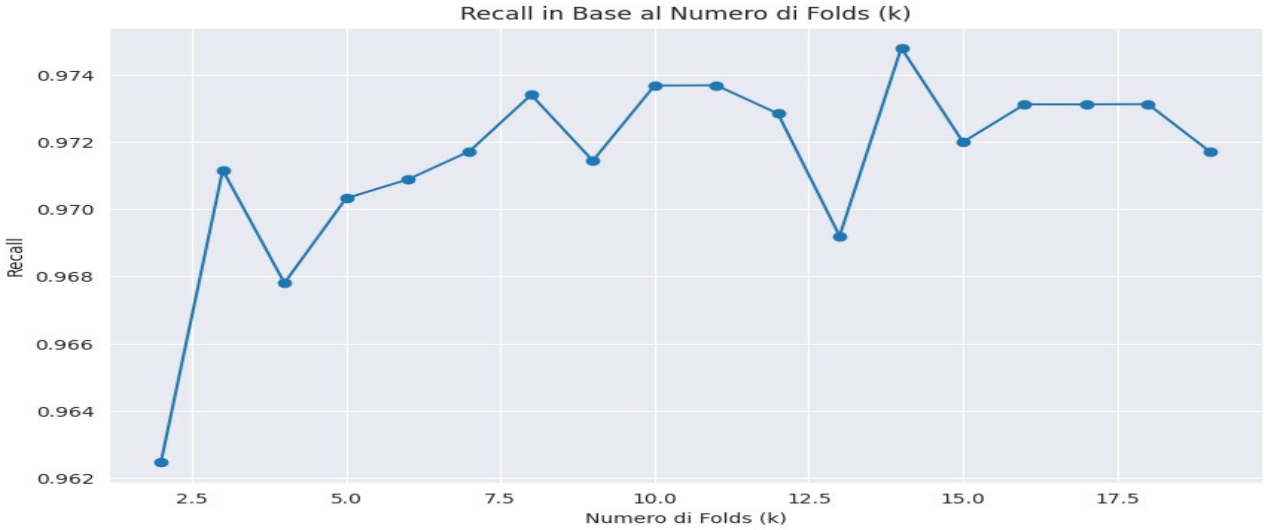
Questa tecnica è molto differente dalle precedenti primo perché utilizza più learner di base creano un mix di classificatori da cui andrà ad estrarre la predizione finale. L'idea alla base è quella di andare ad addestrare indipendentemente un numero di classificatori di base e alla fine del loro addestramento le loro predizioni saranno combinate tramite una funzione aggregate, tale funzione aggregante sarà appresa dal meta-learner un modello finale atto a combinare le predizioni dei modelli di base per fornire la predizione finale. Per il nostro studio abbiamo creato un modello stacking basato su tre principali classificatori di base che risultano essere l'albero di decisione, un classificatore naive bayes e un SVM ovvero un classificatore support vector machine. Una volta addestrati indipendentemente questi tre modelli vengono aggregati tramite un classificatore di regressione logistica che funge da meta-learner con il compito specifico di regolare i pesi affidati alle predizioni dei tre classificatori di base per ottenere in output una predizione che risulta essere una combinazione lineare pesata dei classificatori. Di seguito si riportano i risultati ottenuti dall'addestramento dello stacking model.

```
_________________STACKING MODEL PERFORMANCE_________________
-------------------------Accuracy-------------------------

0.9717333333333333

------------------Classification_Report-----------------

              precision    recall  f1-score   support

           0       0.99      0.97      0.98      2853
           1       0.92      0.96      0.94       897

    accuracy                           0.97      3750
   macro avg       0.96      0.97      0.96      3750
weighted avg       0.97      0.97      0.97      3750
```

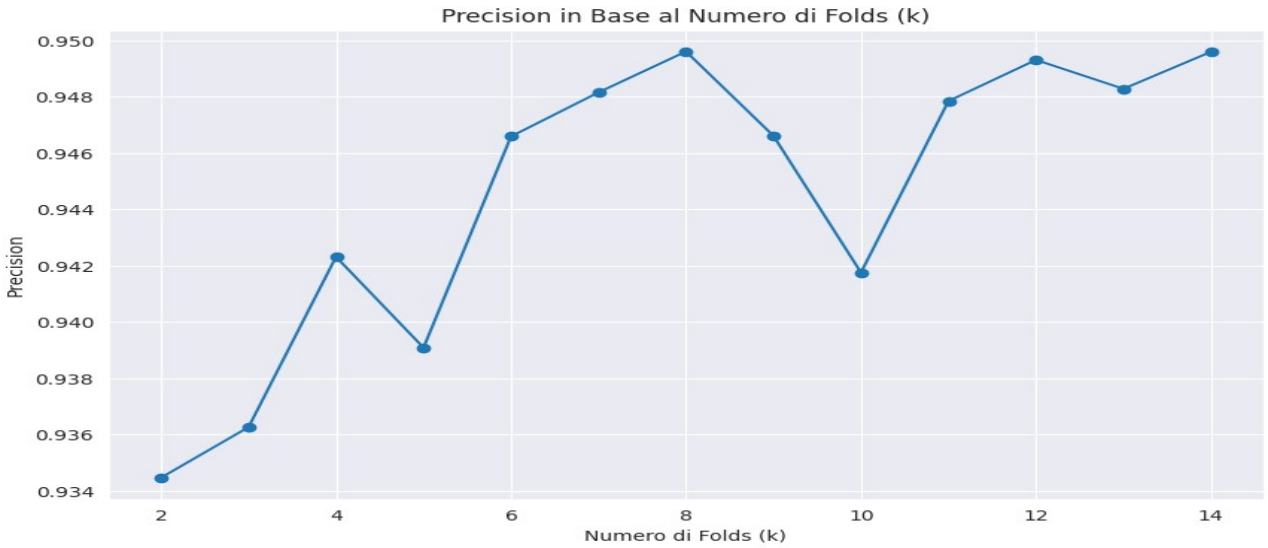
Andiamo a prendere visione della matrice di confusione dello stacking model:



Curva di ottimizzazione recall-fold per lo stacking model



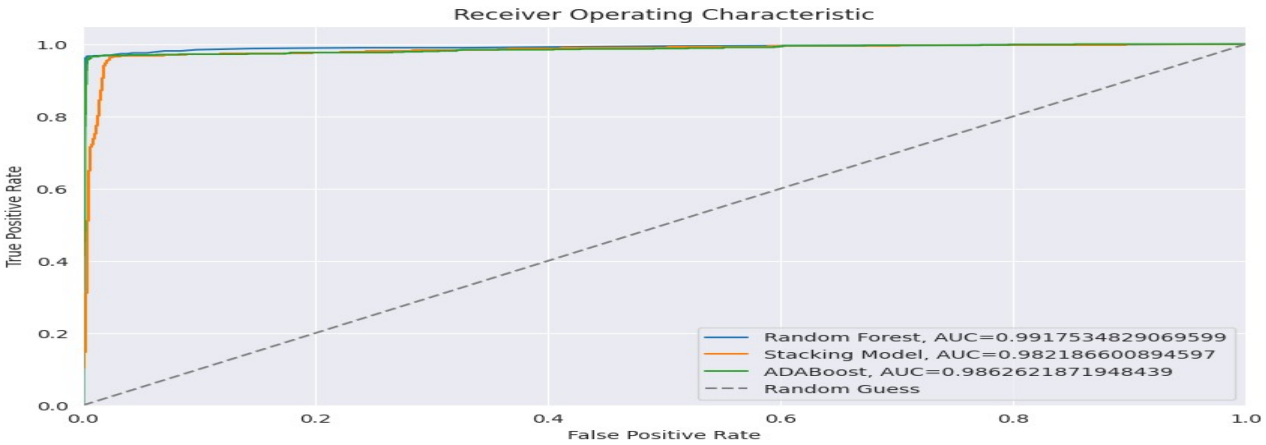
Curva di ottimizzazione precision-fold della cross validation per lo stacking model:



Analizzando le curve di ottimizzazione del k per i fold della cross validation per lo stacking model notiamo che anche in questo caso il recall riesce a stabilizzarsi da k= 8 in su, mentre per la precision abbiamo tre punti di massimo di cui due globali per k =8 e k=14 e uno di massimo locale per k=12 per tali ragioni bilanciando le due scelte il k scelto come ottimo è k= 15 dato che il recall è stabile mentre la precision sembra in crescita all'aumentare dei fold. Per quanto riguardo questo esperimento per trovare sperimentalmente il miglior k abbiamo limitato il range da 2 a 15 visto e considerato che per ottenere i risultati è stato necessario un tempo di calcolo di 3 ore in cloud su

google colab per tali ragioni ci siamo accontentati di una sperimentazione con range più stretto.

Alla fine delle esperimenti effettuati andiamo a tirare le somme sui metodi di ensemble andando prendere visione di un grafico che mostri il ROC score.



Avendo preso visione dei risultati organizzati dichiariamo il Random Forest come il miglior modello tenendo conto delle riflessioni sulla veridicità dei dati ottenuti in base alla struttura dei dati utilizzati.

OTTIMIZZAZIONE DEI MODELLI

Questa sezione riporta la metodologia utilizzata per addestrare e in seguito ottimizzare i modelli sopra visti per cercare di trarne il meglio. In questa sezione raccogliamo anche una breve sintesi di quello che è stata la metodologia di addestramento dei modelli per effettuare gli esperimenti precedentemente discussi.

TROVARE I MIGLIORI IPER PARAMETRI DEI MODELLI

Le ottimizzazioni si sono basate su di una prima ricerca dei migliori iper parametri con cui addestrare il modello la quale è stata condotto tramite un processo di internal cross validation che aveva l'obiettivo di andare a massimizzare il recall dato che il nostro principale obiettivo è riuscire a predire l'abbandono dei dipendenti. L'ottimizzazione è stata fatta utilizzando principalmente la gridsearch una funzione della libreria di scikit-learn che permette di trovare i migliori iper parametri tramite un ciclo di cross validation implementato dalla stessa funzione, questa approccio fornisce una ricerca

esaustiva nello spazio dei parametri che viene ad essere fornito in input. Un altro approccio utilizzato è stato la Randomizedsearch un'altra funzione della stessa libreria che questa volta esegue un ottimizzazione non esaustiva dei parametri di un modello attraverso una ricerca casuale nello spazio degli iper parametri forniti in input. Di seguito si riportano alcune immagini dei risultati ottenuti gli altri sono stati ommessi per non rendere troppo lunga la documenta ad ogni modo è possibile risalire a queste informazioni prendendo visione del codice dai notebook del repository o più semplicemente accedendo alla cartella "plot_and_figure" la quale riporta le immagini di tutti i risultati ottenuti dagli esperimenti fatti.

Migliori parametri per la regressione logistica:

```
__LOGISTIC REGRESSION OTTIMIZATION ON RANDOM UNDER SAMPLING__
Best parameters: {'C': 0.001, 'max_iter': 30, 'penalty': 'l2', 'random_state': 110, 'solver': 'liblinear'}
Best score: 0.8053269076305221
```

Migliori parametri per il gaussian naive bayes:

```
__GAUSSIAN NAIVE BAYES OTTIMIZATION ON RANDOM UNDER SAMPLING__
Best parameters: {'var_smoothing': 0.01}
Best score: 0.8746216867469879
```

Migliori parametri per ADABoost:

```
Best parameters: {'algorithm': 'SAMME.R', 'learning_rate': 1.0, 'n_estimators': 75, 'random_state': 30}
Best score: 0.9083877022075253
```

Migliori parametri per random forest:

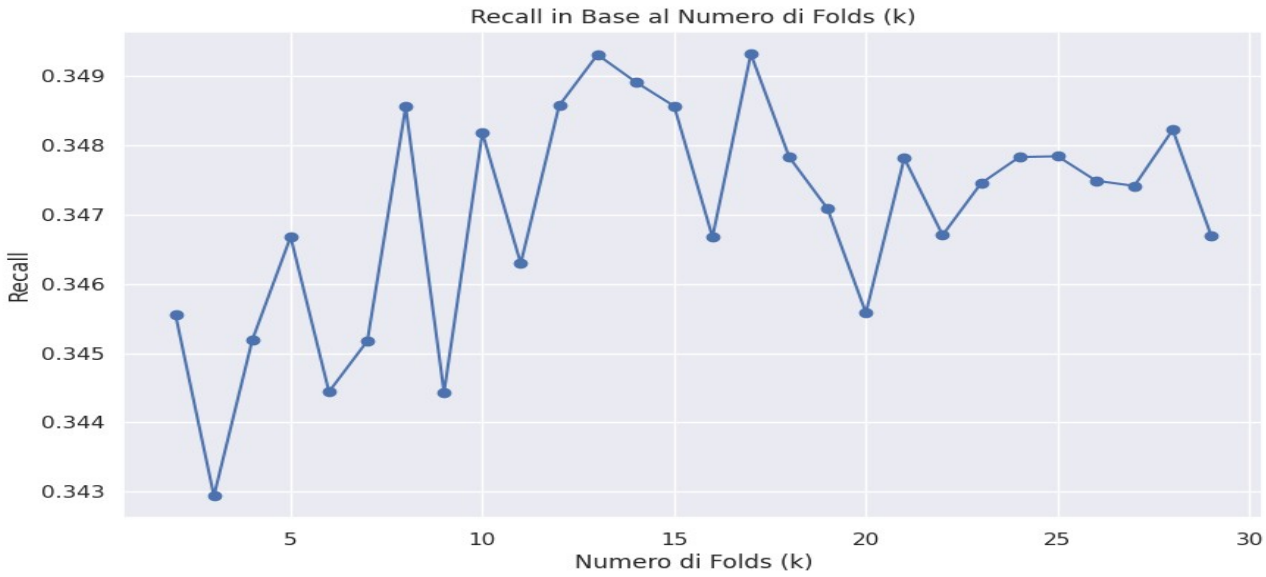
```
Best parameters: {'bootstrap': False, 'criterion': 'entropy', 'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Best score: 0.9648505228349272
```

Una volta trovati i migliori iper parametri per i modelli questi sono stati utilizzati per addestrare il modello finale istanziato con i migliori parametri sul quale si è utilizzato il miglior fold di convalida incrociata per aumentarne il più possibile le performance.

MIGLIORE K PER I FOLD DELLA CROSS VALIDATION

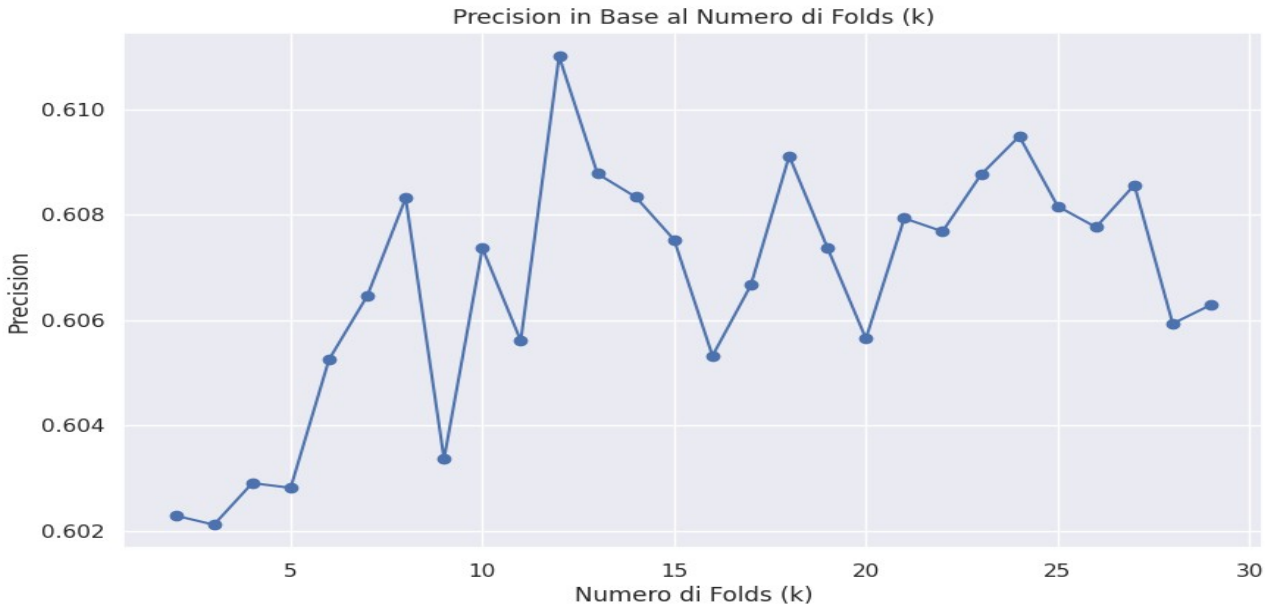
Un altro esperimento condotto sui modelli addestrati è stato quello di trovare il k ovvero il numero di fold ottimale per effettuare il processo di cross validation e questo è stato realizzato ottimizzando una metrica in base al numero dei fold che crescevano progressivamente. Per effettuare questa ottimizzazione chiaramente non abbiamo testato tutti i possibili valori di k cosa che non avrebbe avuto senso dal punto statistico per valori eccessivamente grandi del k poiché avrebbe rarefatto i batch utilizzati, per cui ci siamo affidati alla teoria e alla letteratura scientifica in primo abbiamo calcolato il numero teorico ottimale di k tramite la legge di Strugges che ci dice che il $k = 1 + \log_2(D)$ ove D è il numero di esempi contenuto nel dataset a partire da questo k = 15 arrotondato per difetto abbiamo definito l'upper bound fissato al doppio k=30 e il lower bound con k=2 minimo split che equivale ad effettuare un hold out. Durante la sperimentazione per il ritrovamento del miglior k a livello sperimentale sono stati usati sia il recall che la precision quando abbiamo affrontato il cost insensitive contro il cost sensitive mentre nei casi in cui avevamo il dataset riequilibrato dalle tecniche di campionamento abbiamo anche preso in considerazione l'accuracy visto che il problema non si presentava come sbilanciato ad ogni modo è stato dato più importanza al recall per i motivi descritti nell'introduzione dello studio.

Ottimizzazione del k numero di fold su recall per la regressione logistica



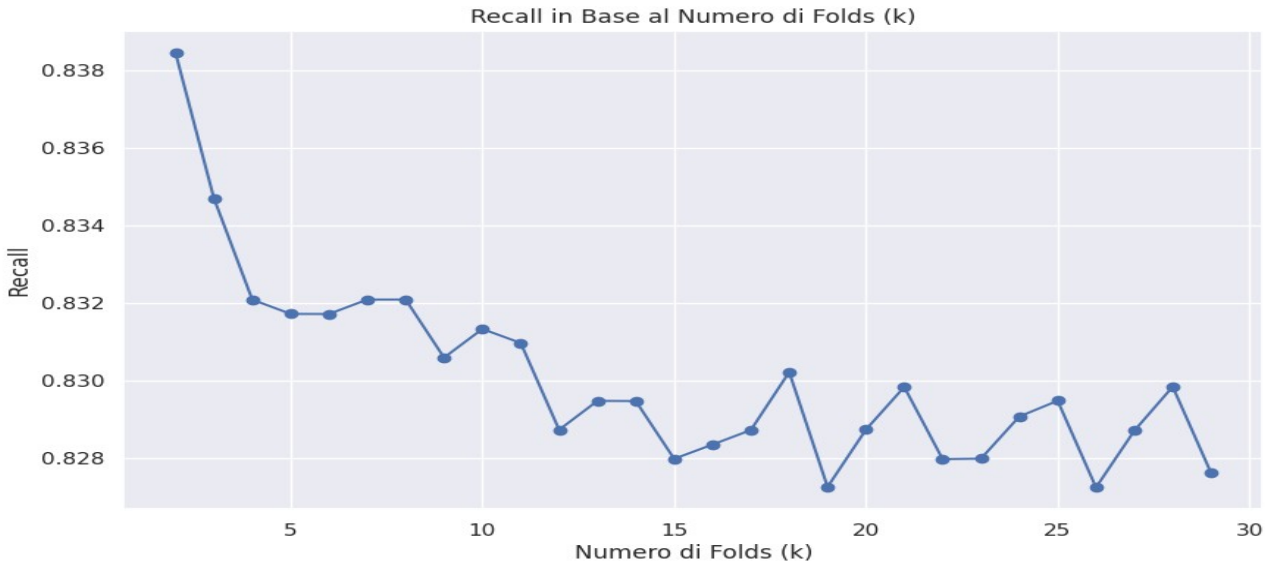
Il miglior k è pari a 13 per il recall

Ottimizzazione del k del numero di fold su precision per la regressione logistica



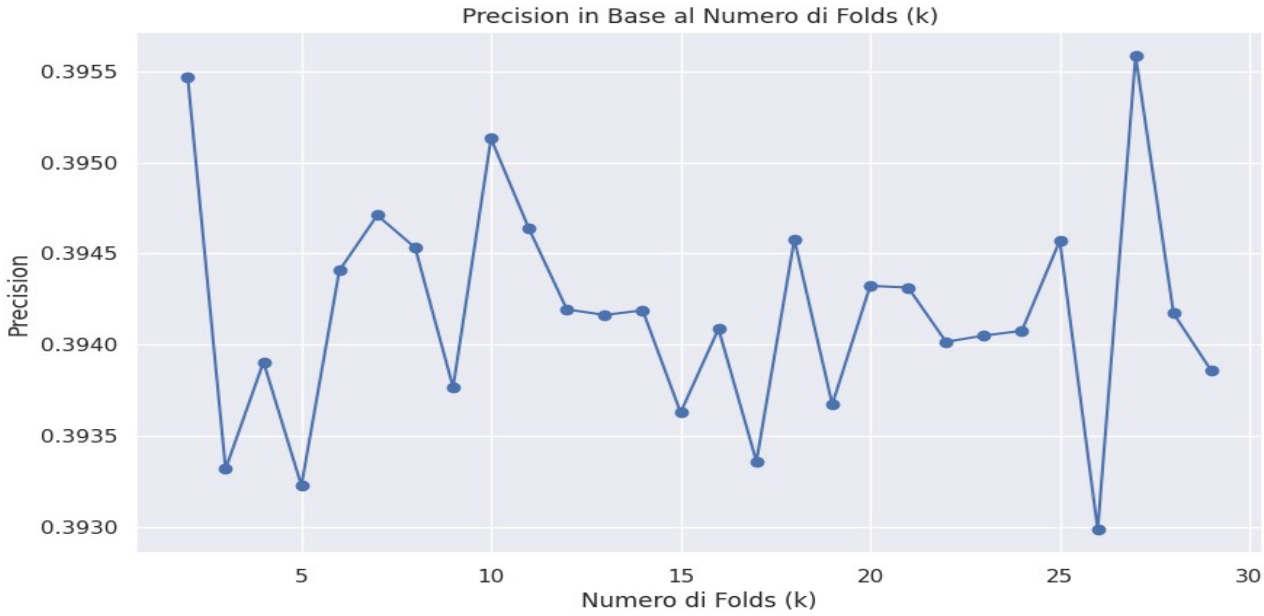
Mentre risulta pari ad 11 per la precision, poiché ad ogni modo abbiamo dato più peso al recall per noi il k ottimizzato per i fold della cross validation è 13 per la regressione logistica.

Ottimizzazione del k fold sul recall per il gaussian naive bayes



In base al recall il k ottimale è dato sembrerebbe da un hold-out, ci è sembrato poco realistico e di fatto è stata ripetuta l'esecuzione più volte, ma il grafico non è variato di molto mostrando un massimo per k piccoli.

Ottimizzazione del k fold sulla precision per il gaussian naive bayes



Per la precision invece i migliori picchi si sono ottenuti per k = 27 e k =10, ma essendo questi punti caratterizzati da minimi locali e globali nelle vicinanze si è preferito scegliere un punto di maggiore equilibrio scegliendo k = 7, il quale è stata la scelta finale per l'ottimizzazione dato che per questo valore anche il recall si comportava con ottimi risultati e massimizzazione costante.

Trovato il miglior fold per la convalida incrociata esso è servito per la verifica dei modelli di hold out di cui è stato messo a confronto la convalida con i principali fold con k = 5, k=10 e k =15 contro il miglior k trovato come precedentemente descritto ed infine solo il miglior k è stato utilizzato per l'addestramento con cross validation del modello finale ottimizzato con i migliori parametri.

```
Stochastic Gradient Descent cost sensitive Cross Validation Recall scores are: [0.74244121 0.78163494 0.76931691 0.83632287]
Stocastic Gradient Descent cost sensitive Average Cross Validation Recall score: 0.7824289817665048
```

```
Percettrone cost sensitive Cross Validation Recall scores are: [0.7745098  0.77030812 0.52941176 0.75524476 0.73389356]
Percettrone cost sensitive Average Cross Validation Recall score: 0.7126736009088951
```

```
Gaussian Naive Bayes Best param Cross Validation Recall scores are: [0.9        0.87428571 0.88       0.86857143 0.89714286 0.88571429
 0.9        0.88714286 0.87410587 0.89270386]
Gaussian Naive Bayes Best param Average Cross Validation Recall score: 0.8859666871040263
```

Ricordiamo che i grafici mostrano solo alcuni modelli ottimizzati per una tecnica utilizzata, ma questo processo è stato fatto per ogni una delle tecniche di campionamento per lo studio sensitive vs insensitive e nell'ensemble learning la metodologia è stata la stessa con leggere variazione e per tali ragioni le immagini inerenti alle ottimizzazioni sono raccolte in apposite cartelle del repository suddivise per modelli e tecniche affrontate.

METODI DI ADDESTRAMENTO DEI MODELLI

L'addestramento dei modelli e come esso è stato svolto nelle fasi dello studio ha seguito una strada precisa in base alle sperimentazioni effettuate, i modelli infatti prima sono stati addestrati con un semplice hold out 70:30 ovvero 70% percento di esempi per il train e 30% di esempi per il test, con tali partizioni sono state affrontate le curve di apprendimento per valutare come i modelli apprendessero all'aumentare degli esempi visti. Le curve di apprendimento sono state studiate tenendo conto delle metriche di recall e precision nei casi in cui il dataset era in sbilanciamento e si è aggiunta l'accuracy quando abbiamo usate le tecniche capaci di ribilanciare il dataset. Dopo aver valutato le curve di addestramento per capire se ci fossero dei problemi con i dati o con le tecniche utilizzate per bilanciare i dati abbiamo terminato l'addestramento e raccolto le predizioni dei modelli sulla partizione di train, le quali in seguito sono state poste a confronto con le predizioni ottenute sulla partizione di test per paragonare i risultati ottenuti e verificare che non si verificassero fenomeni di over fitting sui dati di training o di under fitting rispetto ai dati di training soprattutto durante la sperimentazione dopo l'applicazione del random under sampling. Una volta verificato che non fossero insorte tali problematiche abbiamo completato la valutazione del modello con il report sulle metri di classificazione il quale prevedeva recall, precision, accuracy, f1-score e le loro valutazione calcolate come media e media ponderate ed infine è stata plottata la matrice di confusione. A fine della raccolta delle metriche per validare le performance è stata utilizzata la convalida incrociata con i principali fold k=5, k=10, k=15 e il miglior k trovato tramite un processo di ottimizzazione, questi esperimenti ci hanno permesso di capire se i risultati ottenuti dal modello fossero corretti o dovuti ad un caso fortuito dello split eseguito con hold out. A termine della convalida si è passati all'ottimizzazione degli iper parametri del

modello tramite internal cross validation, una volta ottenute queste informazioni le abbiamo usate per effettuare l'addestramento del modello finale ottimizzato sui fold per la cross validation basandoci sul recall da massimizzare. Questa metodologia di addestramento e ottimizzazione è stata quella che ha caratterizzato gli studi effettuati con le varie tecniche che ci hanno permesso di affrontare lo sbilanciamento del dataset.

Si nota che nella documentazione finale alcune immagini e informazioni inerenti a questi procedimenti sono state omesse per non sovraccaricare troppo questo documento, ad ogni modo è possibile prenderne visione nei notebook o nelle cartelle apposite .\documentation\ plot_and_figure\"nome_del_modello".

CONCLUSIONI

Dallo studio effettuato possiamo concludere che ogniuna delle tecniche utilizzate per affrontare l'imbalance learning ha portato a risultati di miglioramento dell'apprendimento da parte dei modelli e per tali ragioni risultano tutte opzioni valide con le dovute considerazioni che abbiamo affrontato per ogni tecnica. Gli ensemble model sono risultati essere i modelli più performanti e di fatto in letteratura scientifica sono tra le prime scelte per trattare i dati che presentano grossi sbilanciamenti.