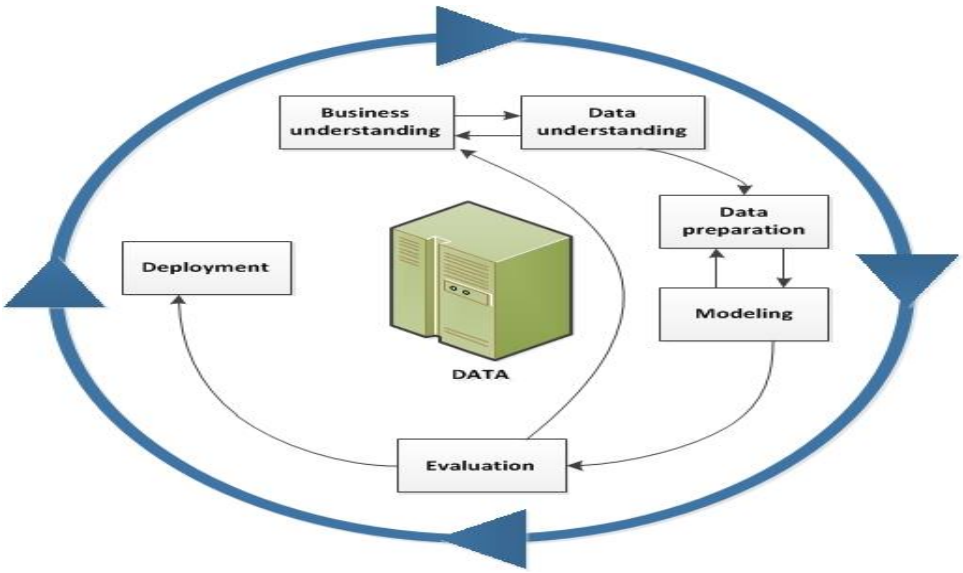
Introduzione

Per la realizzazione di questo progetto ci siamo avvalsi di una metodologia Agile utilizzata nella realizzazione dei progetti di data mining che si basa sul modello CRISP-DM

CRISP-DM che sta per “Cross Industry Standard Process for Data Mining“è un metodo di comprovata efficacia per la costruzione di un modello di Data Mining.

Il presupposto della metodologia risiede nella volontà di rendere il processo di Data Mining affidabile e utilizzabile da persone con poche abilità in materia, ma con elevata conoscenza del business. La metodologia fornisce un framework che prevede sei fasi che possono essere ripetute ciclicamente con l’obiettivo di revisionare e rifinire il modello previsionale:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment



Di seguito si procederà step by step per ogni fase della metodologia in questione.

Business Understanding

Background. Il caso di studio che andremo a trattare ha utilizzato il database Health Facts, un data warehouse nazionale che raccoglie cartelle cliniche complete negli ospedali degli Stati Uniti.

I dati di Health Facts utilizzati sono un estratto che rappresenta 10 anni (1999-2008) di cure cliniche presso 130 ospedali e reti di distribuzione integrate negli Stati Uniti.

Poiché questi dati rappresentano i sistemi sanitari della rete di distribuzione integrata oltre agli ospedali autonomi, i dati contengono sia i dati dei pazienti ricoverati che quelli ambulatoriali, incluso il pronto soccorso, per lo stesso gruppo di pazienti.

Tramite 5 criteri di selezione sono stati presi in considerazione i dati dei pazienti, i criteri sono i seguenti:

1. Il paziente è stato ricoverato.
2. È stato individuato un qualsiasi tipo di diabete.
3. La durata della permanenza in ospedale è stata di almeno 1 giorno e al massimo 14 giorni.
4. Durante la visita sono stati eseguiti test di laboratorio.
5. Sono stati somministrati farmaci al paziente durante l'incontro.

Sono stati mantenuti solo gli attributi potenzialmente associabili alla condizione o alla gestione del diabete.

Sono state estratte 55 caratteristiche tra cui dati demografici, diagnosi, farmaci, numero di visite nell'anno precedente ecc.

In questo studio siamo principalmente interessati ai fattori che portano alla riammissione del paziente, difatti è stato definito un attributo di riammissione avente due valori: "<30", se il paziente è stato riammesso entro 30 giorni dalla sua dimissione, "No" altrimenti.

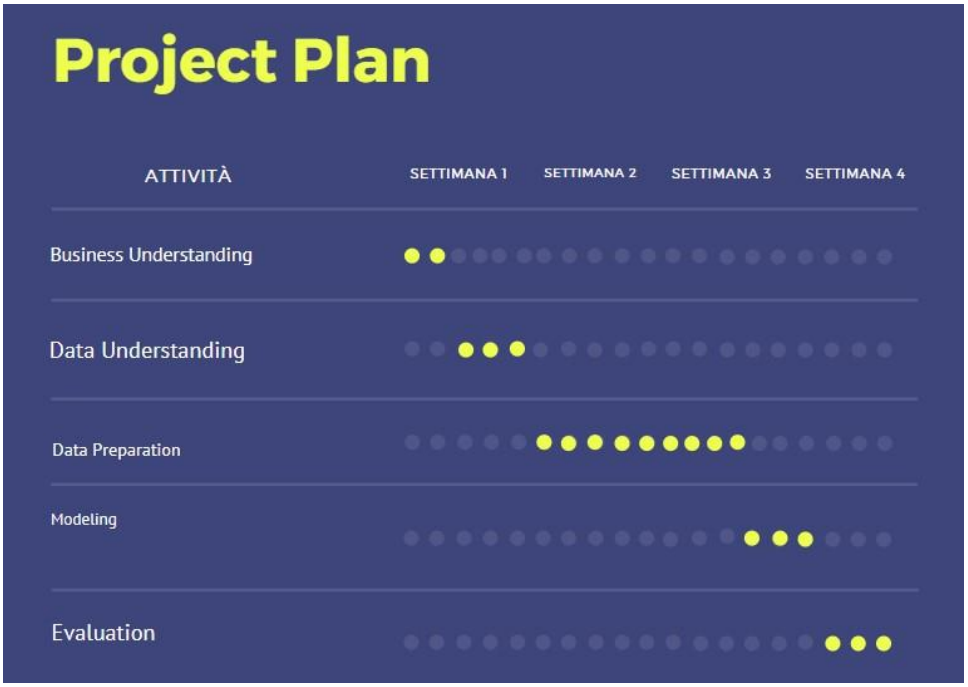
Business Success Criteria. Obbiettivo del caso di studio è capire se un determinato paziente sarà riammesso nella struttura dopo una sua dimissione in un periodo inferiore ai 30 giorni.

Inventory of Resources. Per lo sviluppo dell'analisi, la costruzione del modello e la sua valutazione è stato sfruttato Python (versione 3.7), gli ambienti PyCharm e Jupyter e le seguenti librerie:

1. Matplotlib: libreria utilizzata per la creazione di grafici
2. Pandas: utilizzata per la manipolazione e l'analisi dei dati.
3. Numpy: contiene diverse funzioni e metodo utili per il calcolo scientifico.
4. seaborn: potenzia gli strumenti di data visualization del modulo matplotlib.
5. Re: implementa la gestione efficiente delle espressioni regolari in stile Perl all'interno di Python.
6. Sklearn: è un modulo del linguaggio python con funzioni di scikit-learn utili per il machine learning e l'apprendimento automatico.
7. Imblearn: fornisce strumenti per la classificazione con classi sbilanciate.

Project Plan

I tempi stimati per ogni fase del progetto sono descritti nel grafico sottostante.



Data Understanding

Collect Initial Data Report. Abbiamo parlato nel background delle varie informazioni del dataset, il link per avere accesso al dataset è disponibile al seguente link: https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008 .

Per l'acquisizione dei dati abbiamo usufruito della libreria ***pandas,*** che ci permette di leggere e manipolare i dati presenti nel dataset.

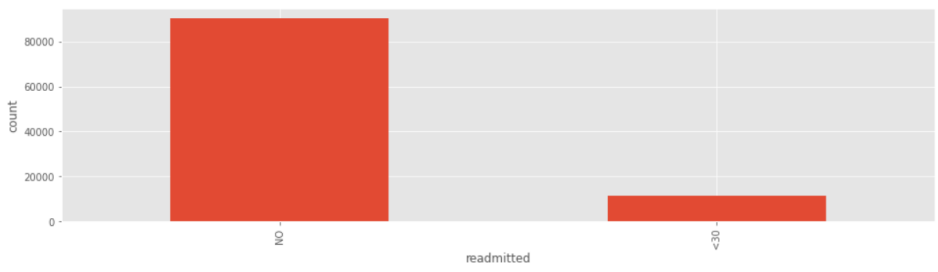
Data Description Report. Il dataset contiene 50 colonne, per un totale di 101.766 record. Il dataset ci fornisce informazioni sul paziente, come l'età, l'etnia e il peso e sulle caratteristiche della sua ospedalizzazione come, ad esempio, in che modo è stato ammesso, dimesso e quanto tempo e quanti test sono stati effettuati. Sono presenti, inoltre, 24 colonne riguardo le medicazioni per il diabete e se ci sono state o no modifiche nei valori prescritti precedentemente. Il dataset contiene anche informazioni riguardo le diagnosi associate al paziente ed alcune colonne "speciali" come *encounter_id* e *patient_nbr* che contengono informazioni perlopiù burocratico-amministrative.

0	encounter_id	101766 non-null	int64
1	patient_nbr	101766 non-null	int64
2	race	99493 non-null	object
3	gender	101766 non-null	object
4	age	101766 non-null	object
5	weight	3197 non-null	object
6	admission_type_id	101766 non-null	int64
7	discharge_disposition_id	101766 non-null	int64
8	admission_source_id	101766 non-null	int64
9	time_in_hospital	101766 non-null	int64
10	payer_code	61510 non-null	object
11	medical_specialty	51817 non-null	object
12	num_lab_procedures	101766 non-null	int64
13	num_procedures	101766 non-null	int64
14	num_medications	101766 non-null	int64
15	number_outpatient	101766 non-null	int64
16	number_emergency	101766 non-null	int64
17	number_inpatient	101766 non-null	int64
18	diag_1	101745 non-null	object
19	diag_2	101408 non-null	object
20	diag_3	100343 non-null	object
21	number_diagnoses	101766 non-null	int64
22	max_glu_serum	101766 non-null	object
23	A1Cresult	101766 non-null	object
24	metformin	101766 non-null	object
25	repaglinide	101766 non-null	object
26	nateglinide	101766 non-null	object
27	chlorpropamide	101766 non-null	object
28	glimepiride	101766 non-null	object
29	acetohexamide	101766 non-null	object
30	glipizide	101766 non-null	object
31	glyburide	101766 non-null	object
32	tolbutamide	101766 non-null	object
33	pioglitazone	101766 non-null	object
34	rosiglitazone	101766 non-null	object
35	acarbose	101766 non-null	object
36	miglitol	101766 non-null	object
37	troglitazone	101766 non-null	object
38	tolazamide	101766 non-null	object
39	examide	101766 non-null	object
40	citoglipton	101766 non-null	object
41	insulin	101766 non-null	object
42	glyburide-metformin	101766 non-null	object
43	glipizide-metformin	101766 non-null	object
44	glimepiride-pioglitazone	101766 non-null	object
45	metformin-rosiglitazone	101766 non-null	object
46	metformin-pioglitazone	101766 non-null	object
47	change	101766 non-null	object
48	diabetesMed	101766 non-null	object
49	readmitted	101766 non-null	object

Come mostrato dall'output, sono presenti alcuni valori NaN che tratteremo in seguito. Alcune colonne, in particolare quelle contenenti IDs, seppur vengano identificate come variabili numeriche, racchiudo in realtà informazioni categoriche. Importante sottolineare che queste colonne, in riferimento al file *IDs_Mapping.csv,* esprimono spesso lo stesso concetto in maniera differente, nel seguito dell'analisi procederemo all'unificazione delle suddette.

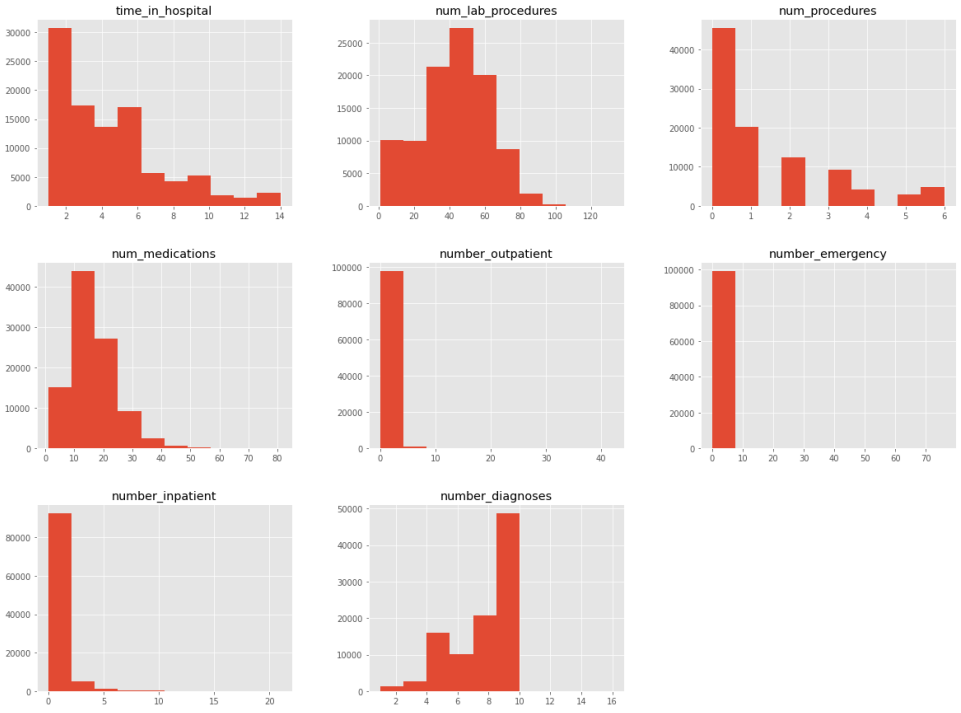
Explore Data

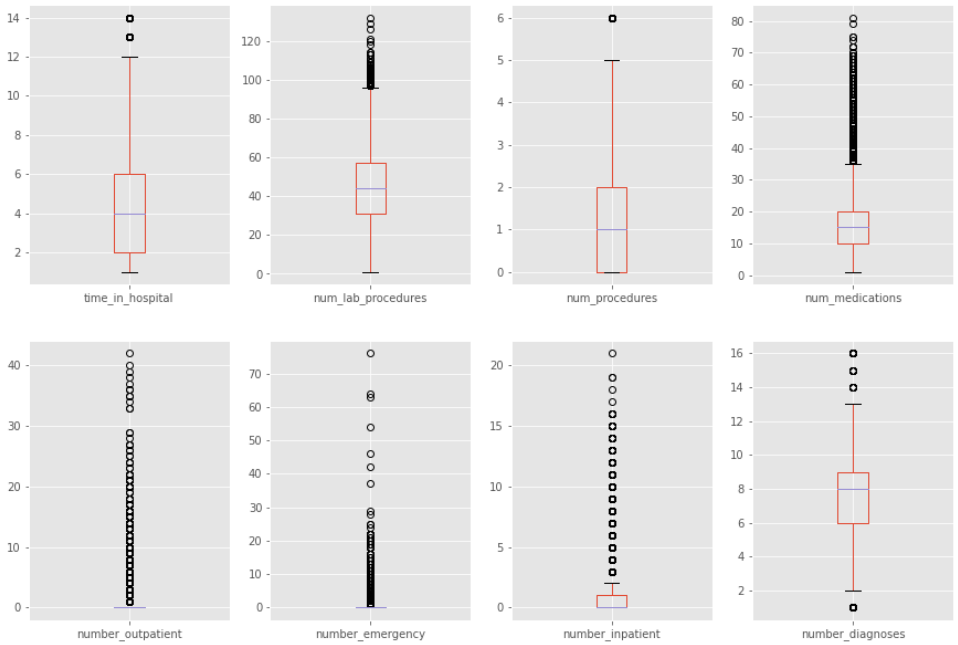
Dal seguente grafico, si evince che il dataset contiene molti più record di pazienti che non sono stati riammessi dopo 30 giorni dall'incontro.



Ciò implica che il dataset è fortemente sbilanciato e prevedere quando il paziente viene riammesso non sarà semplice.

Analizzando i dati numerici sfruttando grafici di distribuzione e boxblot, notiamo due fattori importanti. Una notevole asimmetria che ci fa intuire come la frequenza assoluta di alcuni valori numeri sia molto inferiore comparata a quella di altri e un importante numero di outlier, in particolare ci riferiamo ai grafici relativi a *number_outpatient, number_emergency, number_inpatient, num_medications e num_lab_procedures.*





Osservano i grafici di correlazione, è possibile notare come alcuni attributi categorici sia più frequenti in una label rispetto ad un'altra. Con particolare riferimento a quelli riportanti l'età del paziente (con maggiore frequenza nelle fasce più avanzante), il tempo trascorso in ospedale (nel quale a diminuire del tempo trascorso decrementa la frequenza delle riammissioni), il luogo di trasferimento dopo il periodo di ospedalizzazione e i livelli di insulina.

Data Quality Report

Come possiamo vedere le colonne weight, player_code e medical_specialty hanno un'alta percentuale di valori NaN.

Oltre alla presenza di valori NaN, che corrispondono ai valori mancanti nel dataset, bisogna tener conto dei dati con valori *Unknown* o simili. Le colonne *admission_type_id, discharge_disposition_id* e *admission_source_id,* nelle descrizioni associate agli id, presentano alcuni valori che si possono accorpare sotto un'unica dicitura.

encounter_id	0.0%
patient_nbr	0.0%
race	2.2%
gender	0.0%
age	0.0%
weight	96.9%
admission_type_id	0.0%
discharge_disposition_id	0.0%
admission_source_id	0.0%
time_in_hospital	0.0%
payer_code	39.6%
medical_specialty	49.1%
num_lab_procedures	0.0%
num_procedures	0.0%
num_medications	0.0%
number_outpatient	0.0%
number_emergency	0.0%
number_inpatient	0.0%
diag_1	0.0%
diag_2	0.4%
diag_3	1.4%
number_diagnoses	0.0%
max_glu_serum	0.0%
A1Cresult	0.0%
metformin	0.0%
repaglinide	0.0%
nateglinide	0.0%
chlorpropamide	0.0%
glimepiride	0.0%
acetohexamide	0.0%
glipizide	0.0%
glyburide	0.0%
tolbutamide	0.0%
pioglitazone	0.0%
rosiglitazone	0.0%
acarbose	0.0%
miglitol	0.0%
troglitazone	0.0%
tolazamide	0.0%
examide	0.0%
citoglipton	0.0%
insulin	0.0%
glyburide-metformin	0.0%
glipizide-metformin	0.0%
glimepiride-pioglitazone	0.0%
metformin-rosiglitazone	0.0%
metformin-pioglitazone	0.0%
change	0.0%
diabetesMed	0.0%
readmitted	0.0%

Data Preparation & Cleaning

Select Data

Come abbiamo visto precedentemente, le colonne weight, player_code e medical_specialty hanno un'alta percentuale di valore NaN, quindi procediamo alla cancellazione di queste ultime.

La colonna *race,* che descrive l'etnia, è probabilmente un'informazione importante per l'analisi da effettuare ed ha una bassa percentuale di valore di NaN, in combinazione con il fatto che la percentuale dei valori NaN relativi alla già rara taget label di interesse è solo dell'1,66%, possiamo scartare questi record senza perdita eccessiva di informazioni.

Una scelta simile a quella riportata sopra si applica alla colonna *gender*, dove possiamo scartare i record con valore *Unknown/invalid*.

Le colonne *examide* e *cytoglipton* hanno un unico valore per l'intera estensione dataset, di conseguenza abbiamo deciso di scartarle.

	examide	citoglipton
count	99492	99492
unique	1	1
top	No	No
freq	99492	99492

Le colonne *encounter_id* e *patient_nbr* rappresentano delle informazioni burocratiche/amministrative, quindi scegliamo di scartarle dal dataset.

Alcuni attributi nel set di dati hanno una varianza estremamente alta o estremamente bassa, rappresentando un basso valore di informazione. Abbiamo deciso di impostare un minimo e un massimo, al di fuori dei quali le colonne vengono scartate.

Analogamente, abbiamo notato che alcuni valori presentano una frequenza relativa alla target label estremamente bassa rispetto ad altre, rappresentando, quindi, potenziali outlier o comunque valori poco significativi da passare al modello abbiamo deciso di scartare le entry che mostrano quella valorizzazione per quelle determinate colonne.

Construct Data

Derived Attributes. Molte delle patologie riportate nel dataset (afferenti alle colonne diag) fanno parte di macrogruppi di malattie (come indicato dalla documentazione). Di conseguenza, abbiamo deciso di raggrupparli. Particolare è la categoria **Other**, che contiene intervalli di patologie non molto frequenti nel dataset.

Abbiamo poi discretizzato gli attributi numerici presenti nel dataset, con particolare rifermento a *time_in_hospital, num_procedures e num_diagnosis* le quali sono state riarrangiate secondo quanto mostrato nei boxplot.

Come mostrato nel file *IDs_mapping.csv*, alcuni degli ID associati alle colonne *admission_type_id*, *discharge_disposition_id* e *admission_source_id* si riferiscono a concetti simili. Ai fini dell'analisi, infatti, esistono sostanzialmente più modalità per indicare il valore *Not Mapped,* pertanto procediamo ad unificare questi valori.

Merged Data. Un'operazione preliminare che è stata eseguita è quella di unificare le etichette di destinazione.
In particolare, all'interno del dataset, la colonna "readmitted" ha tre valori '<30', '>30' e 'NO'. Poiché lo scopo della nostra analisi è identificare la riammissione in un intervallo temporale inferiore a 30 giorni, 'NO' e '>30' rappresentano lo stesso concetto.

Modeling

Select Modelling Technique

Per la realizzazione del modello abbiamo vagliato differenti tecniche, di seguito descriveremo quello che hanno dato risultati migliori, come sono state misurate le performance e quali criteri abbiamo adoperato per la scelta del migliore.

L'algoritmo **k-nearest neighbors** tratta le entry del training set come punti in uno spazio multidimensionale e

L'algoritmo **k-nearest neighbors** si basa sul fatto che gli oggetti più vicini ad esso descriveranno il suo gruppo di appartenenza, a seconda del numero più alto di vicini con la stessa "label"; verranno dunque controllate le classi dei k elementi più vicini al soggetto da determinare. Dopo aver caricato il dataset ed aver inizializzato il valore k, per ottenere la classe di appartenenza dell'oggetto da predire si itereranno tutti i punti dei dati del training set calcolati; successivamente verrà calcolata la distanza di essi dal valore in input. Dopo aver ordinato queste distanze in ordine crescente, si andranno a prendere le prime k righe dell'array ordinato e sarà presa in considerazione la classe più frequente, la quale verrà restituita come previsione.

Nel **Decision Tree Classifier** si utilizza una struttura ad albero in cui ogni nodo rappresenta un attributo, ogni ramo una regola decisionale, e i nodi foglia rappresentano il risultato, la predizione. Un DecisionTree è costruito ponendo una serie di domande rispetto a un record del set di dati. Ogni volta che viene fornita una risposta, viene posta un'altra domanda finché non si arriva effettivamente ad etichettare l'oggetto con una classe. Questa serie di domande con le loro possibili risposte sono organizzate sotto forma di albero decisionale, ovvero una struttura gerarchica costituita da nodi ed archi orientati. I nodi non terminali possiedono a loro volta delle etichette per separare i diversi set di dati con le loro differenti caratteristiche. Questo algoritmo funziona suddividendo i dati in gruppi separati in base ad una misura di selezione degli attributi. Ognuna di queste divisioni punta a minimizzare il grado di impurità che scaturisce in ogni nodo a seconda dei risultati incorretti.

Il **Random Forest Classifier** è un algoritmo di classificazione costituito da molti alberi decisionali, in modo tale da ottenere una previsione più accurata. Questo modello aggiunge casualità durante la crescita degli alberi: invece di cercare la caratteristica più importante durante la suddivisione di un nodo, cerca la caratteristica migliore tra un sottoinsieme casuale di feature. Difatti solo una parte del sottoinsieme di feature viene preso in considerazione per la suddivisione di un nodo.
Sklearn fornisce uno strumento che misura l'importanza di una feature osservando quanto i nodi dell'albero che usano quella caratteristica riducono l'impurità in tutti gli alberi della foresta; calcola automaticamente questo punteggio per ogni funzione dopo l'allenamento e ridimensiona i risultati.
La differenza con il Decision Tree è che quest'ultimo genera alcune regole che prevedono la classe di appartenenza; nel Random Forest l'algoritmo seleziona casualmente osservazioni e feature per costruire diversi alberi decisionali e calcola la media dei risultati.
Il parametro "n_estimators" rappresenta il numero di alberi che l'algoritmo costruisce prima di fare una previsione; un numero maggiore di alberi aumenta le prestazioni e rende le previsioni più stabili, ma di contro rallenta il tempo di computazione. Nel nostro caso abbiamo notato che un numero ottimale di alberi è 100.

L'**AdaBoost** (adaptive booster), come il Random Forest, è un algoritmo di classificatore di insieme, costituito quindi da più algoritmi di classificazione il cui output è il risultato combinato di tali algoritmi di classificazione. L'algoritmo fa riaddestrare il modello in modo iterativo, scegliendo il train set in base alla precisione dell'addestramento precedente; il peso di ogni classificatore allenato dipende

dall'accuratezza raggiunta. Dopo che un classificatore è stato addestrato, ad esso viene assegnato un peso in base alla precisione.
Al classificatore più accurato viene assegnato un peso maggiore in modo che abbia un impatto maggiore sul risultato finale; se un elemento viene classificato in maniera erronea, ad esso viene assegnato un peso maggiore in modo che appaia con una maggiore probabilità nel sottoinsieme di addestramento del classificatore successivo.
L'AdaBoost è stato semplice da implementare, il risultato ottenuto da esso è stato ottimo in quanto corregge iterativamente gli errori del classificatore debole e migliora la sua precisione combinando tutti questi ultimi.

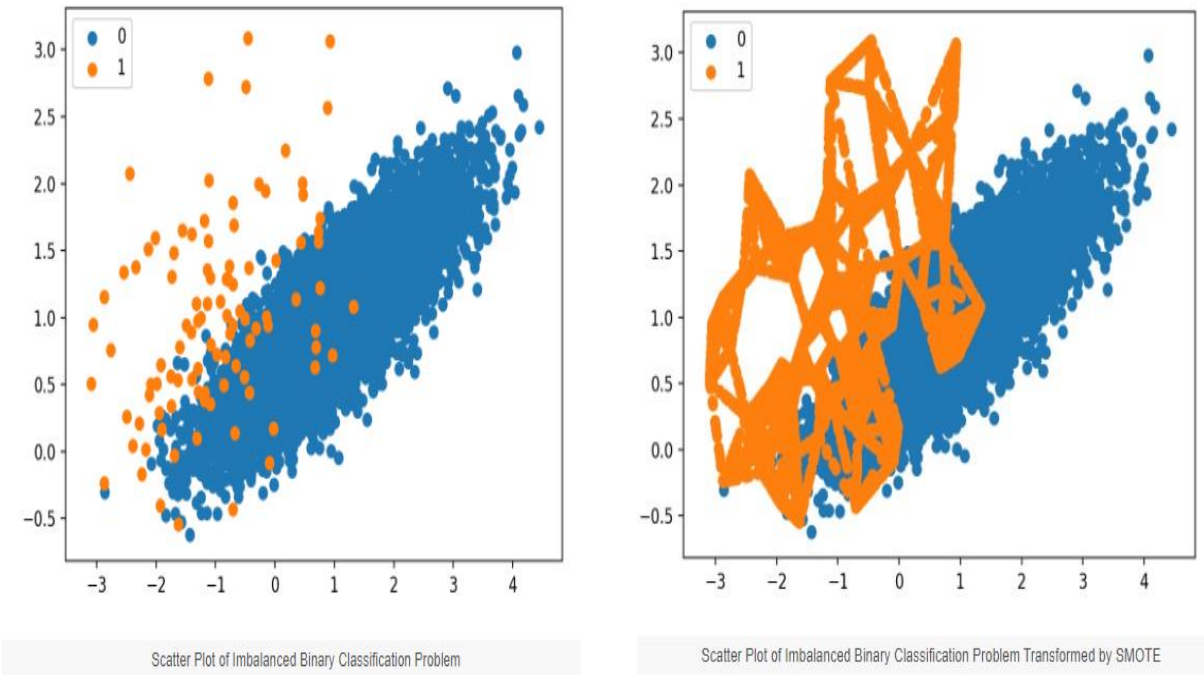
Generate Test Design

Test Design

Prima di eseguire i vari modelli scelti andremo a suddividere il dataset con una proporzione 80% training set e 20 % test set tramite una funzione di sklearn train_test_split.
Tramite una analisi fatta in precedenza abbiamo notato che il dataset è sbilanciato con una proporzione sui risultati negativi del 89 % e sui positivi del 11 %; tramite alcune ricerche abbiamo trovato alcuni algoritmi di bilanciamento, tra cui oversampling e undersampling, e abbiamo cercato di bilanciare questi dati tramite queste due tecniche.

Feauture Scaling. Prima di effettuare previsioni effettive, è sempre buona norma ridimensionare le caratteristiche in modo che tutte possano essere valutate in modo uniforme.
Viene importata la classe StandarScaler, che permette di standardizzare i dati di allenamento e di test per renderli uniformi e confrontabili tra di loro.
Dopo alcune prove abbiamo notato che oversampling strategy si è rivelata la scelta migliore.

Oversampling Strategy. Possiamo utilizzare l'implementazione SMOTE fornita dalla libreria Python di apprendimento sbilanciato nella classe SMOTE.
La classe SMOTE agisce come un oggetto di trasformazione dei dati da scikit-learn in quanto deve essere definita e configurata, adattata a un set di dati, quindi applicata per creare una nuova versione trasformata del set di dati.
Ad esempio, possiamo definire un'istanza SMOTE con parametri predefiniti che bilanceranno la classe di minoranza e quindi la adatteranno e la applicheranno in un unico passaggio per creare una versione trasformata del nostro set di dati.



Immagini in riferimento a documentazione ufficiale https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Build Model

Parameter Settings

Nell'algoritmo **k-nearest neighbors** il parametro *k* rappresenta il numero degli oggetti vicini a quello che viene preso in considerazione; questo parametro ha come valore di default 5 e modificando si va a preferire un numero non troppo grande e dispari, in modo che non ci siano casi di parità tra oggetti differenti.

Nell'algoritmo **RandomForest** il parametro *n_estimators* rappresenta il numero di alberi che l'algoritmo costruisce prima di fare una previsione; un numero maggiore di alberi aumenta le prestazioni e rende le previsioni più stabili, ma di contro rallenta il tempo di computazione. Nel nostro caso abbiamo notato che un numero ottimale di alberi è 150. La profondità massima di ogni albero è 2 e feature massime log2 delle feature risultanti dalla binarizzazione (nel nostro caso 142, quindi 7 feature circa).

Il classificatore **AdaBooster** ha riscontrato la sua performance migliore modificando i valori di default come segue: *algorithm = 'SAMME', n_estimators = 25, learning_rate = 0.6*. Il parametro *n_estimators* rappresenta il numero di classificatori da addestrare in maniera iterativa, *learning_rate* equivale al peso applicato a ciascun classificatore ad ogni iterazione di potenziamento. Un tasso di apprendimento più elevato aumenta il contributo di ciascun classificatore; nel caso del nostro classificatore è stato utile abbassare questo valore che di default è di 50. Il parametro *algorithm* è stato pure modificato dal suo valore di default *SAMME.R*; mentre *SAMME* è un algoritmo a valori discreti, il che significa che emette valori uguali a 0 e 1, *SAMME.R* utilizza le probabilità di classe, cioè restituisce la probabilità che un campione appartenga ad una classe, difatti la 'R' dell'algoritmo sta per 'reali'.

Model Description

Riassumendo quanto detto nel paragrafo precedente, i classificatori selezionati sono stati:

1. RandomForestClassifier, con parametri:
    a. lunghezza massima degli alberi, 2
    b. numero di estimatori, 10
    c. feature massime, log2(feature)
2. AdaBoostClassifier con parametri:
    a. algoritmo 'SAMME'
    b. numero di stimatori, 25
    c. peso di apprendimento applicato a ciascuna iterazione, 0.6
3. KneighborsClassifier con parametri:
    a. numero di vicini, 3

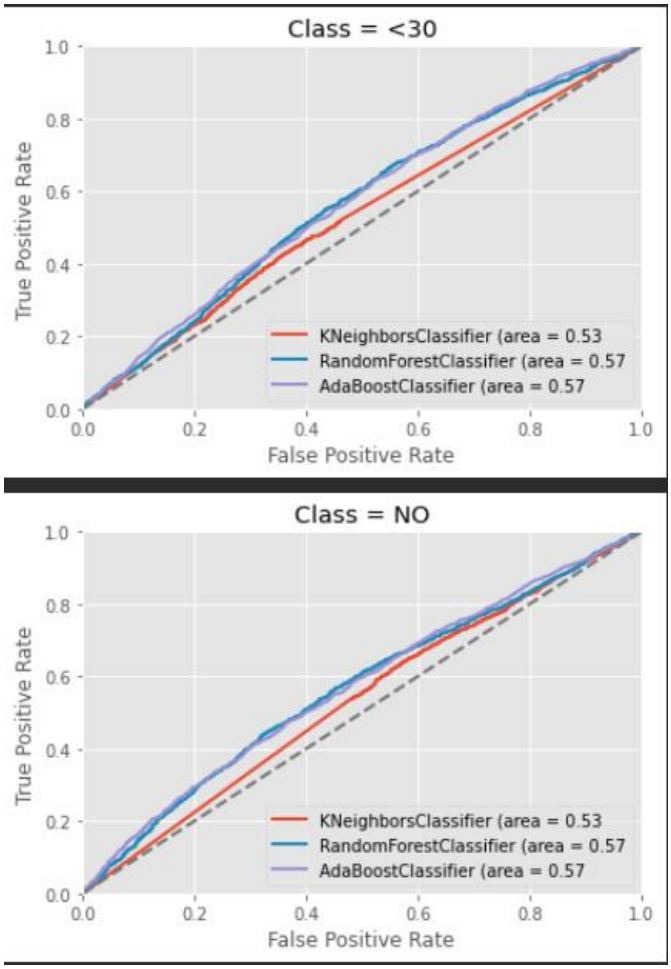
Evaluation

Model evaluation. Per valutare quali modelli fossero i più adatti al nostro contesto abbiamo sfruttato la *cross-validation,* tecnica che divide il traning set in K partizioni della stessa dimensione ed itera il processo di training e test per ognuna di esse. Al termine di questa procedura viene ottenuta una media delle misure. Durante la nostra analisi abbiamo utilizzato la *cross-validation* con K pari a 10, ottenendo le seguenti misure di accuracy e ROC.

```
Accuracy: KNeighborsClassifier: 0.862018 (0.004458)
ROC: KNeighborsClassifier: 0.535825 (0.006523)
Accuracy: RandomForestClassifier: 0.891490 (0.003165)
ROC: RandomForestClassifier: 0.592879 (0.011374)
Accuracy: AdaBoostClassifier: 0.891490 (0.003165)
ROC: AdaBoostClassifier: 0.583988 (0.010329)
```

I modelli con train e test ed applicando le tecniche descritte nella sezione “Test Design” i risultati conseguiti per quanto riguarda accuracy e ROC sono i seguenti:

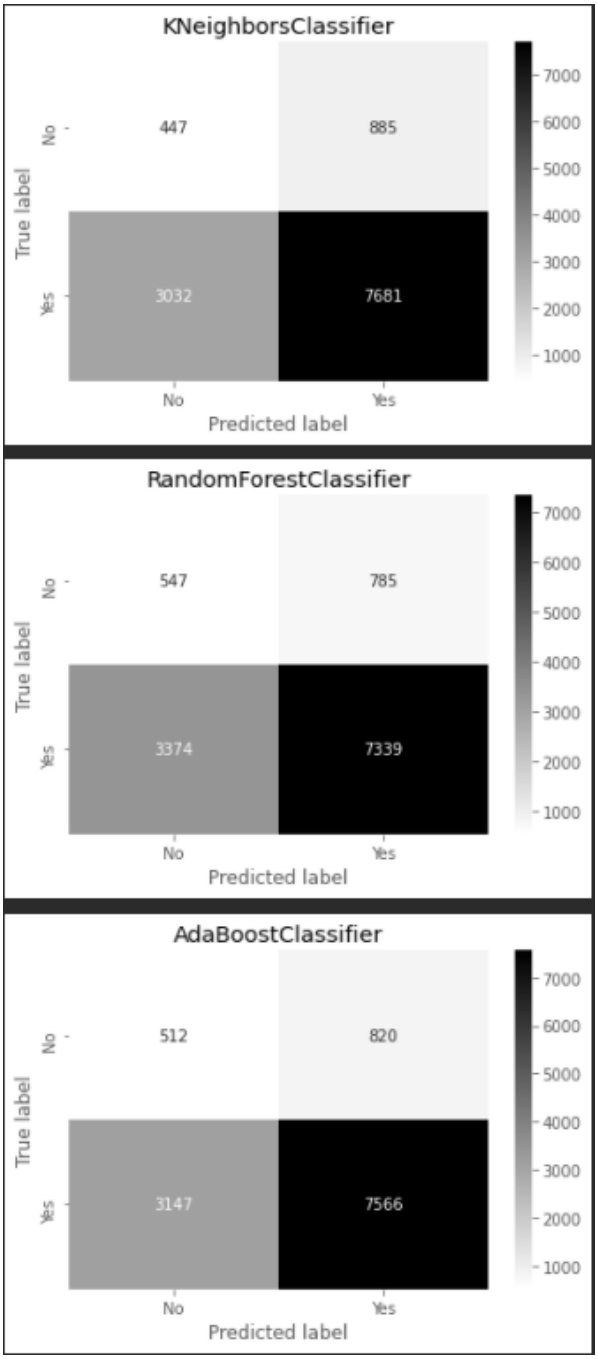
```
Accuracy KNeighborsClassifier: 0.67
Accuracy RandomForestClassifier: 0.65
Accuracy AdaBoostClassifier: 0.67
```


Class = <30
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
KNeighborsClassifier (area = 0.53
RandomForestClassifier (area = 0.57
AdaBoostClassifier (area = 0.57
Class = NO
True Positive Rate
False Positive Rate
KNeighborsClassifier (area = 0.53
RandomForestClassifier (area = 0.57
AdaBoostClassifier (area = 0.57

Evaluate Results

I risultati ottenuti dai modelli sono i seguenti:

```
KNeighborsClassifier Classification Report
              precision    recall  f1-score   support

         <30       0.13      0.34      0.19      1332
          NO       0.90      0.72      0.80     10713

    accuracy                           0.67     12045
   macro avg       0.51      0.53      0.49     12045
weighted avg       0.81      0.67      0.73     12045

RandomForestClassifier Classification Report
              precision    recall  f1-score   support

         <30       0.14      0.40      0.20      1332
          NO       0.90      0.68      0.78     10713

    accuracy                           0.65     12045
   macro avg       0.52      0.54      0.49     12045
weighted avg       0.82      0.65      0.71     12045

AdaBoostClassifier Classification Report
              precision    recall  f1-score   support

         <30       0.14      0.38      0.21      1332
          NO       0.90      0.71      0.79     10713

    accuracy                           0.67     12045
   macro avg       0.52      0.55      0.50     12045
weighted avg       0.82      0.67      0.73     12045
```



Osservando i risultati si evince che i modelli sono molto simili, ma il più recall nella *Random Forest* per quanto riguarda le class label predette positive, lo rende preferibile agli altri, nonostante un valore di recall lievemente inferiore per la class label negativa rispetto agli altri due modelli. Considerando, come scritto in precedenza, il forte sbilanciamento del dataset in favore della class label negativa risulta semplice per i modelli individuare i casi negativi, al contrario di quelli positivi.

Conclusioni

Analizzando i modelli presi in esame, si può osservare quali sono le colonne che hanno aggiunto più peso nel processo decisionale degli algoritmi.
È stato possibile visualizzarle per i modelli Random Forest e AdaBoost e sono di seguito riportate:

- Per il Random Forest le colonne che presentano un peso maggiore sono:
    1. Il numero di diagnosi
    2. Il tempo trascorso in ospedale
    3. L'età
    4. Il luogo in cui è stato dimesso il paziente (discharge_disposition)
- Per l'AdaBoost le colonne che presentano un peso maggiore sono:
    1. Il numero di diagnosi
    2. Il tempo trascorso in ospedale
    3. Il numero di esami effettuati