**Predictions of the survival of Titanic passengers: an ensemble approach based on selected classifiers**

**Progetto della Super Guida “Data Science & Artificial Intelligence”**

*Studente: Tommaso Barbiero*

**1. Introduzione**

La competizione “Titanic: Machine Learning from Disaster” è considerata dalla piattaforma Kaggle una delle opportunità più importanti per l’esercizio e l’apprendimento delle tecniche di Machine Learning [1]. Nello spazio dedicato alla competizione sono disponibili due datasets. Il primo contiene diverse caratteristiche (features) di 891 passeggeri (sesso, età, classe di viaggio, tariffa pagata ecc.) e se questi passeggeri sono sopravvissuti (cioè il valore della variabile target). Il secondo comprende le stesse caratteristiche per altri 418, ma non è dato sapere se questi sono davvero sopravvissuti. Nella letteratura dataset del primo tipo sono noti come train dataset, mentre dataset del secondo tipo sono conosciuti come test dataset. Questo perché nei train dataset, avendo a disposizione sia le features, sia la variabile target, è possibile utilizzare modelli di classificazione e/o di regressione in grado di spiegare la variazione della variabile target, in relazione al comportamento delle features. Individuando e applicando uno o più modelli di classificazione, oppure combinandoli, è possibile prevedere i valori della variabile target dei datapoints (unità statistiche) appartenenti al data set di test.

Lo scopo di questo esercizio consiste nel combinare e mediare le previsioni sul dataset di test, partendo da comuni modelli di classificazione e regressione addestrati sul train dataset. Gli algoritmi utilizzati sono: il Support Vector Machine, il Gaussian Naive Bayes, il Decision Tree, la Random Forest, il K-Nearest Neighbors, e la Regressione Logistica. Si noti che tutti i classificatori (e la regressione) utilizzati producono un output binario: “0” se si prevede che il passeggero sia morto nel naufragio, “1” se invece è riuscito a sopravvivere.

Vengono valutati diversi modelli per ogni tipo di classificatore, in funzione di una griglia dei parametri che sono più in grado di influenzare le prestazioni del modello. Questi modelli vengono individuati tramite una ricerca a griglia [2][3]. Poi verranno selezionati i modelli che superano una certa performance in fase di convalida, tenendo di volta in volta fuori dal campione, su cui i modelli vengono stimati, raggruppamenti di datapoints. Valutando l’abilità previsiva di questi modelli su questi datapoints (di cui comunque si conoscono i valori della target variable), si possono impiegare i modelli che con più probabilità possono dare accurati risultati per i *datapoints* di test (di cui non conosciamo i valori della variabile target). Viene anche selezionato il migliore modello per ogni tipo di classificatore. Nell’ottica del calcolo della media delle previsioni ottenute per i singoli *datapoints*, l’inclusione dei migliori modelli per tipo può attenuare i possibili comportamenti di uno stesso tipo di classificatore che può produrre *bias* nelle previsioni sul dataset di test, nonostante buone prestazioni in fase di convalida. La metrica utilizzata è l’accuratezza (in inglese *accuracy*) che è la somma delle previsioni corrette divisa per il numero totale delle predizioni [2, 4, 5]. Ci sono anche altre metriche di valutazione dei modelli, la *precision* e il *recall* che sono rispettivamente basate sul numero dei falsi positivi e dei falsi negativi. Tuttavia, queste metriche appaiono particolarmente indicate in caso di forte squilibrio tra la dimensione delle classi [2]. Nel nostro caso invece il numero di sopravvissuti (38%) non è drammaticamente inferiore a quello dei deceduti (38%). Per questo utilizzeremo la valutazione dell’*accuracy*

Infine, verrà sottoposto a *Kaggle* un vettore di previsioni (), i cui risultati saranno i valori medi di ogni test *datapoint* ottenuti considerando tutti i classificatori selezionati, e attribuendo un peso maggiore ai classificatori con la migliore accuracy in fase di convalida. Si utilizza infatti un approccio *ensemble* perché si tiene conto di vari classificatori.

Il piano dell’esercizio è il seguente:

* Pulizia dei dati (Sez. 2.1): trasformazione di variabili categoriali in numeriche, per quanto possibile, attribuzione dei valori mancanti, *features engineering* (ossia creazione di nuove variabili che possano fornire un maggiore contributo informativo sul comportamento della *target variable*)
* Analisi esplorativa dei dati (Sez. 2.2): rappresentazione delle frequenze assolute e relative dei sopravvissuti, della suddivisione del set di addestramento per età, sesso, e classe di viaggio. Ci aspettiamo infatti che il contenuto informativo della *target variable* sia legato alle caratteristiche menzionate. Vengono poi costruiti grafici a barre concernenti il numero di sopravvissuti e deceduti per sesso, classe di viaggio, ed età (cioè essere o no adulti)

Si utilizzerà la tecnica della *stepwise backwards regression* per l’omissione delle variabili non significative (anche se i classificatori verranno principalmente impiegati considerando tutte le variabili). In una parte consistente della letteratura scientifica la selezione delle variabili avviene tramite la valutazione dei *p-value* o delle statistiche dei coefficienti di regressione

* Breve presentazione degli aspetti statistici dei modelli di classificazione utilizzati, e discussione sui loro punti positivi e critici (Sez. 3)
* Identificazione dei modelli utilizzati. Selezione dei modelli che contribuiranno al vettore di previsioni ensemble, ottimizzando i parametri più sensibili (Sez. 4)
* Presentazione delle performance del vettore ensemble delle previsioni, e dei migliori classificatori per ciascuna tipologia. Discussione dei risultati e considerazioni conclusive (Sez. 5)

**2. Pulizia e analisi dei dati**

*2.1 Pulizia dei dati*

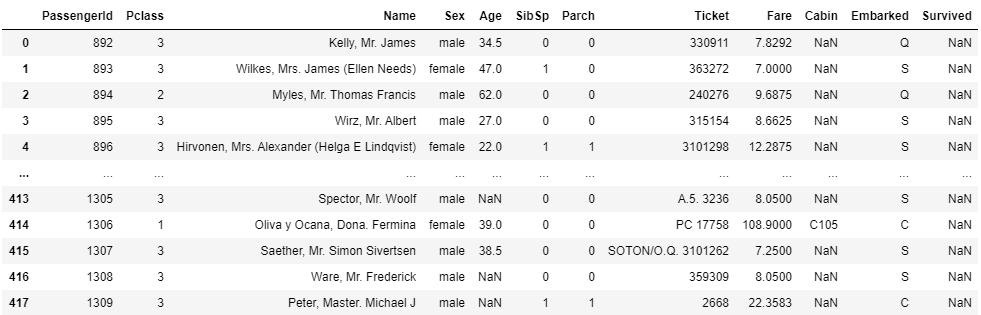
Le colonne dei DataFrame (Pandas) di train (train\_df) sono chiamabili con l’istruzione train\_df.columns . La quale fornisce il seguente risultato

Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',

'Ticket', 'Fare', 'Cabin', 'Embarked', 'Survived'],

dtype='object')

A titolo di esempio si mostra come si presentano i dati, una volta importati



*Fig. 1 Aspetto del DataFrame di test, appena importato*

Le colonne del DataFrame di test (test-df) sono analoghe, eccetto che per la colonna ‘Survived’, che di cui chiaramente non abbiamo informazione. In ogni caso si aggiunge un vettore (418 \* 1), riportante altrettanti valori numpy.nan (dove NaN è acronimo di not a number). Ora che le colonne del train\_df e del test\_df sono le stesse è possibile combinare i due DataFrame in un unico denominato all\_df. In questo modo è possibile pulire e fare operazioni sui dati contemporaneamente su entrambi i DataFrames.

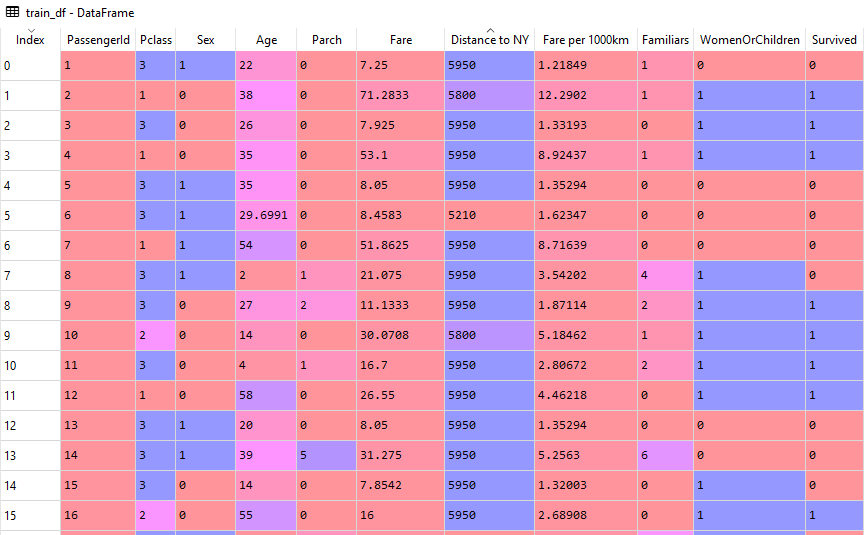
Come si può vedere dalla Fig 1, mi sono parse opportune le seguenti trasformazioni dei DataFrame, rimandando per i dettagli più tecnici al notebook commentato:

* Assegnazione dei valori mancanti della colonna “Fare” al prezzo medio della classe di viaggio (colonna “Pclass”). In questo caso i valori mancanti sono complessivamente 263 (di cui 177 nel train\_df e 86 nel test\_df). È possibile vedere quanti valori mancanti ci sono in un DataFrame, ripartiti per colonne con l’istruzione test\_df.isnull().sum() , per quanto concerne il dataset di test
* Trasformazione in variabile numerica della variabile “Sex”: “0” viene assegnato alle persone di sesso femminile e “1” ai maschi
* Assegno "Southampton" come porto di partenza dataset['Embarked'] dei passeggeri 61 e 829 del train\_set. Sono stati imputati questi valori a "Southampton" sia perché è il porto da cui sono partiti la maggioranza dei viaggiatori, sia perché diversi viaggiatori che hanno pagato una tariffa molto simile ai 80 pounds di queste due persone provenivano da Southampton.
* Si rende quantitativa la variabile 'Embarked' trasformandola nella distanza da New York (destinazione designata del Titanic) (Southampton-NY: 5910 km, Cherbourg-NY: 5800 km, Queenstown: 5210 km). Si presume che il prezzo pagato dipenda anche dal porto di partenza. Quindi viene creata la variabile quantitativa 'Distance to NY'
* Si crea una nuova variabile "Familiars" che somma le variabili 'SibSp' (relativa a Coniugi e Cognati) e 'Parch' (relativa al numero di genitori e figli)
* Si crea una nuova variabile "Fare per 1000km". Questo per normalizzare la tariffa pagata rispetto ai km. È un tentativo per ottenere una variabile economica depurata dall’effetto distanza
* Creo una nuova variabile “WomenOrChildren” che assume valore “1” in presenza di donne o di bambini (persone con età inferiore ai 18 anni), e “0” in caso contrario. Si presume che questi soggetti abbiano avuto maggiori probabilità di sopravvivenza rispetto ai maschi adulti, poiché hanno avuto accesso prioritario alle scialuppe
* Tolgo le colonne che non forniscono, a mio avviso, ulteriori informazioni trasformabili in numero, attraverso l’istruzione

drop\_elements = ['Name', 'Ticket', 'Cabin', 'SibSp','Embarked']

Ad esempio il codice della cabina è disponibile solo per la prima classe, ed eventuali diversità tra le cabine di questa classe si riflettono nel prezzo, e quindi probabilmente sulle possibilità di sopravvivenza. La colonna 'Embarked' è già stata convertita in numero con la colonna "Distance to NY".

A questo punto presentiamo come appare il dataset (o più precisamente il DataFrame di train) (Fig. 2). I DataFrame ora hanno tutte le celle piene, e hanno tutti i valori numerici. Si noti che la colonna “PassengerId” non verrà utilizzata come variabile, ma è utile solo come colonna di identificativi necessaria per la sottoposizione del vettore dei risultati a Kaggle



*Fig. 2 Aspetto del DataFrame di train dopo la fase di pulitura e imputazione dei dati mancanti*

L’aspetto del DataFrame di test (test\_df) è analogo, eccetto che per la colonna “Survived” che contiene valori “nan”. Per testare la validità della principale nuova variabile creata, “WomenOrChildren”, si è inserito il vettore (418 \* 1) relativo ai datapoints di test, come previsore della sopravvivenza delle persone del test. Sottoponendo il vettore a Kaggle si è ottenuto 0.74641come *accuracy*, che è già un discreto valore. Ciò conferma quanto meno che questa nuova variabile (feature) “WomenOrChildren” ha un buon contenuto informativo e può essere impiegata nelle successive applicazioni di classificazione e regressione

*2.2 Analisi descrittiva dei dati*

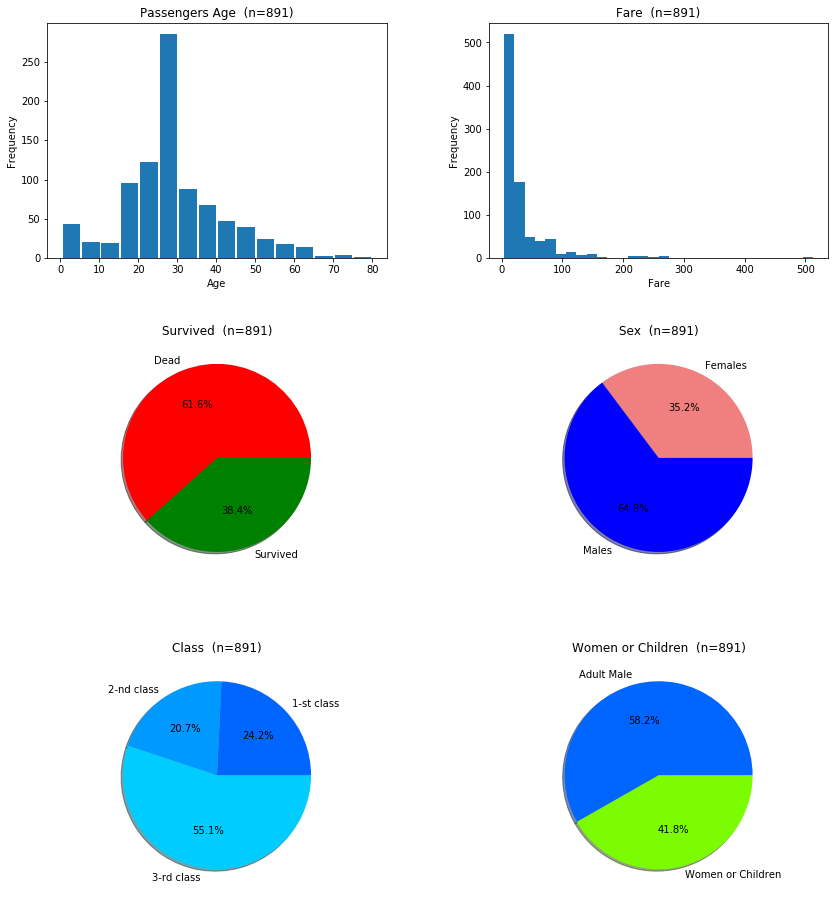
Le figure seguenti mostrano le più basilari statistiche descrittive (Fig. 3-4). Purtroppo il 61% delle persone nel train set perse la vita nel naufragio (Fig. 4 figura medio-sinistra). Altre informazioni rilevanti si possono desumere anche dalle statistiche descrittive richiamate con l’istruzione train\_df[all\_variables].describe()



*Fig. 3 Sintesi delle statistiche descrittive delle variabili utilizzate*

La maggioranza dei passeggeri era di sesso maschile (64.8%) (Fig 3, Fig 4 grafico in mezzo a destra). L’età media era piuttosto bassa, circa 30 anni, e molto pochi erano ultrasessantenni (Fig 4 grafico in alto a sinistra), anzi solo il 25% era aveva un’età superiore a 35 anni. Tutto sommato pochi erano anche i bambini e gli adolescenti, dal momento che il primo quartile della distribuzione dell’età era 22 anni.

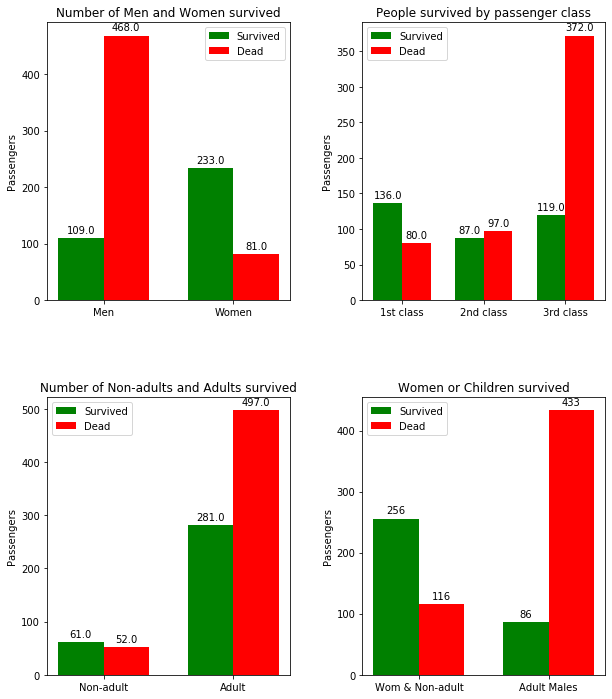
Per quanto riguarda la tariffa pagata la media era circa 33 Dollari, ma ben il 25% ha pagato meno di 8 Dollari. Per contro solo il 25% ha pagato più di 31 Dollari, dunque più della media. In effetti la media è stata innalzata da poche persone che sono state disposte a pagare anche 512 Dollari. In pratica la distribuzione della tariffa è fortemente asimmetrica, ed è per altro concorde con la classe del passeggero: il 55.1% delle persone ha alloggiato in terza classe pagando in media 12.4 Dollari, il 20.7% ha alloggiato in seconda classe con tariffa media 22.2 Dollari, e il restante 24.2% in prima classe ha pagato mediamente 94.3 Dollari, ma con una forte variabilità (per la presenza di suites molto lussuose).



*Fig. 4. Rappresentazioni di base del train DataFrame del Titanic*

Il dato sui parenti (genitori, figli, coniugi, cognati e fratelli) che hanno in media accompagnato una persona è piuttosto basso, 0.90 per ogni passeggero e molte persone (537 su 891) hanno viaggiato da sole. Il grafico in basso a destra della Fig. 4 mostra che il 41.8% era composto da donne e bambini, come vedremo “privilegiati” all’accesso alle scialuppe.

La Figura 5 vuole approfondire tramite grafici a barre multiple informazioni sui gruppi di persone che hanno avuto più chances di salvarsi.



*Fig 5. Numero di sopravvissuti secondo diversi gruppi*

Dalla figura 5 è interessante considerare che:

* Si è verificato uno squilibrio delle chances di sopravvivenza in base al sesso. Mentre il 74.2% delle femmine è riuscito a salvarsi (233 su 314), questa percentuale scende al 18.9% per i maschi (109 su 577). Questa percentuale è un po’ più alta per gli uomini in prima classe, 36.9% (45 su 122), ma è drammaticamente più bassa per i viaggiatori maschi della terza classe, appena il 13.5% (47 su 347)
* Come accennato, le probabilità di sopravvivenza decrescevano man mano che ci si muove verso la terza classe. Infatti, il grafico in alto-sinistra della Fig. 5 mostra che significativamente oltre la metà dei passeggeri in prima classe si è salvata (precisamente il 63%, 136 su 216). La percentuale dei sopravvissuti in seconda classe è di poco inferiore alla metà (47.3%). Invece, poche persone che alloggiavano in terza classe sono riuscite a salvarsi, cioè il 24.1% (119 su 491). Si può quindi affermare che maggiore è stata la tariffa pagata (che riflette la ricchezza personale), maggiore è stata la probabilità di salvarsi)
* Un’altra osservazione riguarda l’età dei passeggeri. Questi sono stati divisi in due gruppi: età minore di 18 anni, e maggiore o uguale a questa soglia. È noto che i bambini erano tra le categorie cui era riservata in via prioritaria l’imbarco nelle scialuppe. Considerando il grafico in basso a sinistra, in effetti, oltre la metà dei bambini o adolescenti si è salvata (53%, pari a 61 su 113). Questa percentuale è però soggetta anche alla classe di viaggio di appartenenza: riguardo alla prima classe, quasi tutti i pochi minori di 18 anni si sono salvati (11 su 12); invece solo 28 “minorenni” su 78 (37.2%) della terza classe hanno evitato la morte. Per quanto riguarda gli adulti, comunque, la percentuale media di sopravvissuti (36.1%, ossia 281 su 778) è molto inferiore a quella media dei “minorenni”
* Come si poteva immaginare, il 68.8% delle donne o minori di 18 anni si è salvato (256 su 372), contro il 16.6% dei maschi adulti (su 519). L’appartenenza alle categorie “prioritarie” ha giocato quindi un ruolo determinante nella sopravvivenza

Non tutte le variabili utilizzate nel dataset di addestramento hanno il medesimo potere informativo, cioè hanno la stessa capacità di influenzare variazioni della variabile target. Alcune variabili infatti non sono significative e potrebbero inflazionare la varianza del modello di classificazione, causando *overfitting* nel dataset di addestramento. L’introduzione di variabili non significative potrebbe penalizzare le capacità predittive di un modello di classificazione o di regressione, che risulterebbe erroneamente sensibile a caratteristiche marginali. Un’operazione di *feature selection* potrebbe essere importante per migliorare l’accuratezza delle previsioni. Un approccio tipicamente usato per la selezione delle variabili è la PCA (Principal Component Analysis). Questo approccio si basa sull’ortogonalizzazione delle caratteristiche, partendo dalla direzione (autovettore) che presenta la massima varianza (autovalore). In questo modo è possibile ordinare le caratteristiche ortogonalizzate di un dataset in base alla loro varianza (nel nuovo spazio vettoriale), rimuovendo l’effetto della correlazione tra variabili. In seguito è possibile selezionare il numero di componenti principali desiderato. Questo metodo comunque ignora completamente le variabili target e si presta maggiormente per contesti di apprendimento non-supervisionati. Non è comunque questa la sede per approfondire la tecnica PCA, dal momento che viene utilizzato un approccio che valuta direttamente le relazioni causali delle features sulla variabile target tramite tecniche di regressione . Si tratta dell’analisi delle statistiche e dei *p-value* dei coefficienti nella regressione lineare multipla[[1]](#footnote-1), e l’eliminazione iterativa della variabile meno significativa, finché tutte le variabili rimaste non raggiungono una certa soglia di significatività. Questa tecnica di eliminazione delle variabili è nota come *stepwise backward regression*.

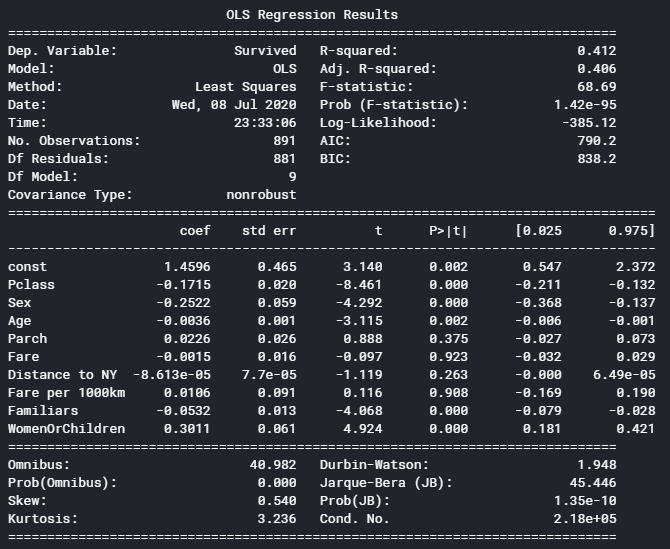
In questo caso la misura della significatività è data dal *p-value*. Maggiore è il *p-valu*e "P>|t|", minore è la statistica (che è la standardizzazione dei coefficienti di regressione) e minore è la significatività della variabile. Assumendo la normalità delle variabili, e assumendo l'ipotesi nulla che le variabili siano indipendenti dalla Y, il *p-value* misura la probabilità di ottenere statistiche uguali o meno probabili di quello osservato durante il test. Dunque, quanto più è grande (il coefficiente di regressione è grande rispetto alla sua deviazione standard), tanto minore sarà il *p-value*, e tanto minore sarà la plausibilità dell'ipotesi nulla di indipendenza

Una tipica soglia massima scelta per il *p-value* è 0.05, che corrisponde circa a una statistica (in caso di perfetta normalità dei coefficienti di regressione corrisponde a 1.96)

L'algoritmo utilizzato scarta di volta in volta la variabile con il *p-value* più alto (ad eccezione dell'intercetta), fintanto che rimangono le sole variabili che presentano un *p-value* inferiore a 0.05. Per fare questo, una volta eliminata una variabile viene ristimato il modello di regressione

Le variabili rimaste sono significative, mentre il coefficiente delle variabili scartate viene considerato pari a 0.

Di seguito presentiamo i principali risultati del modello di regressione completo, tramite la libreria statsmodels:



*Fig. 6. Sommario del modello completo di regressione lineare multipla basato sul train dataset ripulito*

Già da questa prima figura si nota che, a parità di altre variabili, la probabilità di sopravvivenza è influenzata negativamente dalla classe di viaggio: più alta ed economica è la classe minori sono state le probabilità di sopravvivenza. Altre variabili che hanno influito negativamente sulla sopravvivenza sono state il sesso (essendo il sesso “femmina” associato allo “0”, sono state proprio le femmine ad aver avuto più possibilità di salvarsi), l’età, e il numero di parenti al seguito. Infine essere donne o bambini è stato un fattore decisivo per la sopravvivenza.

Nonostante, la pulitura del dataset e operazioni di feature engineering, l’indice è piuttosto basso, 0.412. Ciò vuol dire che i regressori spiegano solo il 41% della varianza della variabile dipendente Y. Ciò indica che nel dataset fornito, probabilmente, non erano disponibili altre variabili che potevano giocare un ruolo significativo anche in fase previsiva della Y. Un esempio potrebbe essere l’etnia dei passeggeri. Oppure anche la presenza di eventuali casi particolari (*outliers*) che il modello non riesce bene a interpolare. Presentiamo più formalmente l’indice (Eq. 2.1 e 2.2), e l’ corretto (Eq. 2.3), che attribuisce una penalità aggiungendo features non significative (Eq. 2.2) [16]:

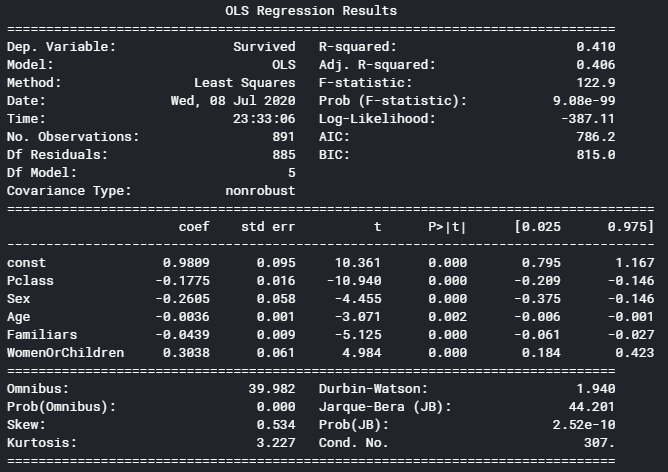
Oppure può essere espresso nel modo seguente:

L’ è invece uguale a:

In cui è il numero di esempi su cui il modello viene stimato, e il numero di regressori impiegati. All’aumentare di si verificano due effetti opposti: diminuisce il denominatore della parte non spiegata dalla regressione, quindi in questo senso dovrebbe decrescere; ma un aumento dei regressori porta anche una diminuzione della somma dei residui e quindi dovrebbe aumentare. Se una variabile è significativa l’effetto di riduzione dei residui, è superiore alla penalità per l’introduzione di una nuova variabile. Comunque .

Per vedere se un modello con meno variabili produce una migliore accuratezza in fase di convalida, proviamo ad estrarre le variabili significative (ossia tutte con il p-value < 0.05, che equivale ad una statistica circa). Vediamo ora come è il modello, una volta omesse le variabili non significative (Fig. 7).

Le variabili significative (Fig- 7) sono: la classe di viaggio, il sesso, l’età, il numero di parenti al seguito, e l’essere donna o bambino. Il segno di queste variabili è appropriato, e loro influenza sulla Y è già stata discussa. Gli indici e sono praticamente uguali a quelli del modello completo, quindi le variabili eliminate non erano in grado di dare alcun contributo alla spiegazione della variabilità della Y. Verrà dato un cenno sull’utilità della riduzione del numero di variabili, verificando l’accuratezza sui dati di test (sottoponendo vettori di previsione su Kaggle)



*Fig. 7. Sommario del modello di regressione con le sole variabili significative*

Si è deciso di modellare i classificatori, a partire dai dati escludendo gli *outliers*. Un modo semplice per individuare ed eliminare automaticamente gli *outliers* è ricorrere ad un algoritmo IsolationForest. Si basa quindi sull’algoritmo delle foreste casuali (vedi Sez. 3,4). Il parametro più sensibile nel modello è "contamination", che viene utilizzato per regolare il numero di *outliers* da escludere [17]. Sia per i dati così come sono, sia per i dati normalizzati, abbiamo scelto come valore di contaminazione 0.025. Ciò vuol dire che è stato tolto il 2.5% dei dati più anomali. La documentazione di IsolationForest si trova qui [18]

**3. Presentazione dei modelli di classificazione utilizzati**

*3.1 Support Vector Machine*

L’idea principale che sta alla base di un algoritmo di Support Vector Machine è la massimizzazione del margine tra la retta di separazione e i punti appartenenti ad una certa classe (label) più vicini a questa retta. In realtà se si prende in considerazione un vettore di 3 o più variabili , si parla di iperpiano di separazione.

Per semplicità si mostra nella figura di seguito una classificazione basata su due *features* , ci può essere un ampio range di rette che separano linearmente le classi ma solo una corre più distante delle altre rispetto ai punti più vicini delle classi. In questo modo si aumenta il potere di separazione tra le labels. Supponiamo inoltre che ci siano 2 labels (“1” e “-1”). Si definiscono due iperpiani paralleli, entrambi abili nel separare efficacemente le due features, costruiti in modo tale che la distanza tra essi sia massima. Nella figura sottostante i due iperpiani di separazione sono presentati come rette.

* . (3.1)

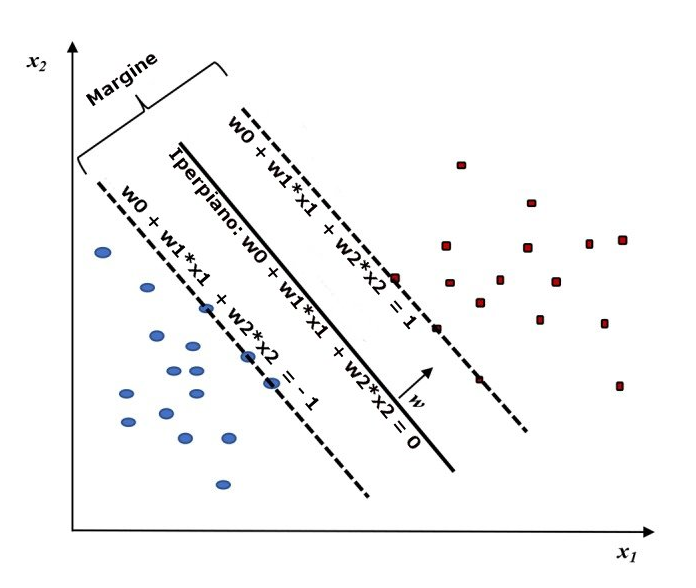
In questo caso i punti giacenti sull’iperpiano o situati sopra di esso prendono come classe “1”

Nel caso di due sole features l’iperpiano diventa

* . (3.2)

In questo caso i punti giacenti sull’iperpiano o situati sotto di esso prendono come classe “-1”. Nel caso di due sole features l’iperpiano diventa

L’iperpiano ottimo di separazione è quello equidistante tra i due iperpiani (3.1) e (3.2) come si può vedere a titolo di esempio nella figura seguente (Fig. 8)



*Fig. 8. Iperpiano ottimo e margine nella classificazione Support Vector Machine. Fonte: [6]*

Dalle Eq. (3.1) e (3.2) si ottiene che per massimizzare il margine si deve massimizzare l’espressione [5,6]:

(3.3)

In cui è la lunghezza del vettore

(3.4)

In cui si considerano tutte le componenti vettori delle variabili da a .

Per trovare il massimo margine, in caso di netta separazione lineare tra labels, si preferisce minimizzare la seguente funzione obiettivo, cioè:

(3.5)

*Support Vector Machine: classificazione a margine soft e funzioni kernel*

Frequentemente è difficile separare linearmente i dati. Infatti, quando gli esempi di addestramento sono tanti, può risultare impossibile costruire una retta in grado di separare tutti i punti in modo che la classificazione corrisponda sempre al reale valore della label . Per consentire comunque una separazione si può:

* Allentare la penalizzazione di classificazioni errate entro una certa distanza dall’iperpiano di margine (permettendo una classificazione a margine soft)
* Utilizzare una funzione detta kernel che combina le *features*, definendo un nuovo spazio vettoriale, che possa consentire una separazione lineare tra le features

Per quanto riguarda la classificazione a margine soft, si introduce una nuova variabile *slack* () che consente l’ottimizzazione del margine anche in presenza di errate classificazioni. Si introduce quindi una nuova funzione obiettivo (3.6) da minimizzare che tiene conto degli errori di classificazione.

(3.6)

Come si può vedere il parametro consente di controllare la penalizzazione dell’errata classificazione. Infatti:

* Se è basso, il modello risulta molto sensibile a errori di classificazione. In particolare se si usano congiuntamente kernel non lineari, il modello in fase di addestramento potrebbe incorrere in problemi di overfitting
* Invece è alto, il classificatore potrebbe essere troppo permissivo nei confronti degli errori, con il rischio di pervenire a previsioni distorte

Di fatto è possibile usare per controllare la larghezza del margine entro cui vengono tollerati errori di classificazione [5].

Come accennato, un altro modo di separare dati altrimenti non separabili linearmente è quello di ricorrere a funzioni kernel. Queste funzioni consistono nel creare opportune combinazioni delle caratteristiche eventualmente non lineari, proiettandole in uno spazio vettoriale diverso [5], tramite una certa funzione di mappatura . Questo spazio, infine, rende separabili linearmente i dati. Un esempio di funzione mappatura (trasformazione di variabili) è il seguente:

(3.7)

Le funzioni di mappatura vengono impiegate all’interno di una funzione kernel, come segue [5]:

(3.8)

Una delle più utilizzate funzioni kernel è il Radial Basis Function (RBF) (comunemente utilizzata in scikit\_learn [7]), detta anche kernel gaussiana

(3.9)

Più semplicemente viene di solito scritta così:

(3.10)

Dove gamma è uguale a (3.11)

Con particolare riferimento al kernel RBF, abbiamo individuato i due parametri verso cui l’algoritmo Support Vector Machine è più sensibile:

* : maggiore è maggiore è la tolleranza nei confronti delle errate classificazioni, aumentando però il rischio di previsioni distorte (biased)
* : maggiore è il valore di (gamma), maggiore è il raggio di influenza degli esempi di addestramento, accentuando tuttavia il rischio di overfitting all’interno del set di apprendimento

*3.2 Gaussian Naive Bayes Classifier*

Il Gaussian Naive Bayes Classifier è un approccio di classificazione basato sulle probabilità condizionate e sul teorema di Bayes.

Siano:

* il valore di una certa classe (o variabile label): nel caso del Titanic “0” oppure “1”
* , in cui è un vettore di variabili indipendenti

Sia inoltre la probabilità che un certo datapoint, avente come valori delle variabili indipendenti certi valori , venga classificato nella classe . è conosciuta anche come posterior probability [2] [4] [5], poiché l’azione di classificazione è logicamente posteriore alle su cui la classificazione si basa. In particolare ci interessa valutare le singole al variare della classe, per determinare quale è la classe più probabile dato un set di valori .

Scomponendo il vettore nelle singole *features* si ottiene (3.12):

In cui la probabilità è la somma delle probabilità delle intersezioni tra il verificarsi dei valori e l’appartenenza ad una certa classe . Quindi:

(3.14)

Quindi la formula (3.13) diventa:

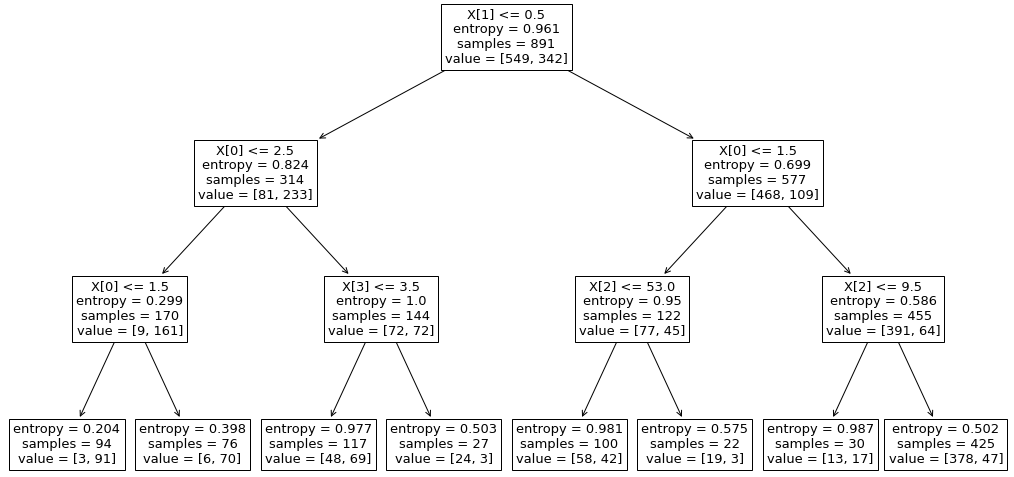
Di conseguenza il classificatore (cioè il previsore) è il valore della classe per cui il numeratore dell’Equazione (3.15) è massimo. Si noti che in questo caso il denominatore di (3.15) è semplicemente un fattore di normalizzazione, e non dipende da una sola specifica classe.

Quindi si può esprimere la classe prevista , come segue:

È possibile implementare un algoritmo Gaussian Naive Bayes consultando la documentazione in scikit\_learn [9]

*3.3 Decision Tree Classifier*

Questo modello di classificazione è un sistema di suddivisione dei dati di addestramento, prendendo decisioni sulla base di una sequenza di domande. Per comprendere più fondo il modello consideriamo la seguente rappresentazione *tree plot*, ottenuta dal *train set* del Titanic. Si precisa che questo decision tree non ha alcuna pretesa di essere considerato particolarmente performante. Si pensi che i migliori Decision Trees testati hanno una profondità compresa tra 7 e 10



*Fig. 9. Decision Tree sul train set del Titanic. Parametri: max\_depth=3, min\_samples\_split=15, min\_samples\_leaf=20*

Si ricorda che: X[0] : Pclass, X[1] : Sex, X[2] : Age, e X[3] :Parch. Vengono divisi i dati di volta in volta in base a queste features, fino a quando l’albero non raggiunge profondità 3. In questa tabella presentiamo più chiaramente i risultati. Questo algoritmo favorisce l’interpretabilità dei risultati. Ad esempio, è molto probabile che maschi adulti che hanno viaggiato in terza classe non siano riusciti a sopravvivere, mentre solo un nodo finale (foglia) relativa alle donne ha dato come esito della classe predetta la non-sopravvivenza (cioè “0”).

Viene fornita di seguito la nomenclatura di base per comprendere un albero decisionale [2] [4] [5] [6] [10] [11] :

* I nodi interni sono i test intermedi che dividono i dati in base a una caratteristica
* La radice è il primo dei nodi interni
* I rami (o archi) sono le risposte ai nodi interni (nel nostro esempio “Sì” o “No”) che rimandano ad altri nodi interni oppure alle foglie
* Le foglie contengono la risposta alla serie di nodi interni che le hanno precedute, e contengono le labels predette. Un *tree plot* di scikit-learn fornisce come risultato il numero di esempi di addestramento ripartiti per la vera classe di appartenenza. Dopodiché viene selezionata come classe predetta, la classe più rappresentata all’interno della foglia

*Tab. 1. Ricostruzione tabulare del decision tree in Fig. 9*

|  |  |  |  |
| --- | --- | --- | --- |
| Prima domanda | Seconda domanda | Terza domanda | Label predetta |
| Sesso = Femmina?  (Sesso < 0.5?) | Se Sì (= Femmina):  Classe = prima o seconda (<2.5)? | Se Sì:  Classe = prima (<1.5) ? | Se Sì:  Label: Sopravvissuto  (3 morte, 91 salve) |
| Se No:  Label: Sopravvissuto  (6 morte, 70 salve) |
| Se No:  Num. Genitori o figli al seguito <= 3 (<=3.5)? | Se Sì:  Label: Sopravvissuto  (48 morte, 69 salve) |
| Se No:  Label: Morto  (24 morte, 3 salve) |
| Se Sì (= Maschio):  Classe = prima (<1.5)? | Se Sì:  Età < 53? | Se Sì:  Label. Morto  (58 morti, 42 salvi) |
| Se No:  Label: Morto  (19 morti, 3 salvi) |
| Se No:  Età <= 9 ? | Se Sì:  Label: Sopravvissuto  (13 morti, 17 salvi) |
| Se No:  Label: Morto  (378 morti, 47 salvi) |

Un importante concetto è che i dati vengono via via suddivisi in base alla caratteristica che produce il miglior guadagno informativo (information gain), e congiuntamente i dati vengono ripartiti in base a un certo livello di questa caratteristica. Ad esempio un nodo interno nel *tree plot* di esempio è stato diviso in base all’età selezionando una soglia di 53 anni

Ci sono diverse metriche per quantificare il guadagno informativo. In questo specifico esempio, si è utilizzato l’Entropia, che misura l’impurità all’interno di un nodo. Esistono altre metriche, come l’impurità di Gini o l’Errore di Classificazione, che tendenzialmente producono risultati di classificazione pressoché analoghi. Quindi il parametro “criterion” del DecisionTreeClassifier di Scikit\_learn [12] è sostanzialmente trascurabile.

L’entropia di un nodo si misura così [2] [5] [11]:

In cui è la proporzione in un certo nodo degli esempi davvero appartenenti alla classe “1” (cioè a quella dei sopravvissuti), rispetto al numero totale di esempi nel nodo. L’estremità destra della (3.17) si riferisce specificamente al dataset del Titanic.

Un nodo viene ripartito in funzione della massimizzazione dell’information gain, che in questo caso è dato dalla differenza tra l’entropia del nodo genitore e l’entropia media ponderata dei nodi figli ( e ). Sia l’information gain [2] [5]

Solitamente, per ottenere riduzioni significative delle entropie dei figli è necessario aumentare la profondità dell’albero decisionale (ossia il massimo numero di nodi interni). D’altra parte un eccessivo numero di foglie può portare a problemi di *overfitting*, compromettendo l’abilità previsiva su esempi fuori dal campione. Veniamo, dunque, ai parametri di un Decision Tree Classifier che influenzano maggiormente la classificazione [12]:

* Massima profondità (max\_depth): riguarda la massima profondità dell’albero decisionale. Alcuni rami dell’albero decisionale non raggiungono la profondità impostata, se si raggiungono foglie pure (composte solamente da esempi della stessa classe), oppure se le foglie contengono meno elementi rispetto ad una certa soglia “min\_samples\_split” al di sotto della quale non si può più dividere un nodo.
* Il minimo numero di esempi in un nodo necessari per poterlo dividere (“min\_samples\_split”). Ad esempio se in un nodo ci sono 10 elementi, e la soglia “min\_samples\_split” è 15, il nodo non può essere ulteriormente diviso
* La dimensione minima di una foglia (“min\_samples\_leaf”)

In definitiva maggiore è il numero di foglie e maggiore è la profondità, minore è il *bias* sui dati di apprendimento, ma più concreto è anche il rischio di *overfitting.* Per trovare modelli performanti è pertanto necessario un buon bilanciamento dei valori di questi parametri, i cui modelli vanno testati su sets di convalida, come vedremo più avanti

*3.4 Random Forest Classifier*

Il classificatore Random Forest è un esempio di *ensemble method* nel machine learning. Infatti, una foresta casuale può essere considerata come un insieme (ensemble) di alberi decisionali [5]. In particolare il classificatore Random Forest, attraverso una votazione a maggioranza delle previsioni dei singoli alberi decisionali, calcola un vettore di previsioni. In genere i singoli decision trees su cui si basa la foresta sono profondi, e hanno elevata varianza. L’aggregazione e la mediazione delle previsioni, provenienti dai singoli alberi, portano ad una minore variabilità delle previsioni, riducendo il rischio di *overfitting*, e garantendo (“almeno sulla carta”) previsioni più accurate.

Questi sono i passi con cui viene implementato questo algoritmo:

* Estrazione di un campione casuale di bootstrap di dimensione con possibilità di reintroduzione degli esempi [13]
* Sviluppo di un algoritmo ad albero decisionale (Sez. 3.3), impiegando caratteristiche estratte casualmente
* Ripetere i primi due punti per volte, in cui è il numero di alberi decisionali che compongono la foresta
* Aggregare le predizioni di ognuno dei alberi utilizzati, assegnando nel vettore finale di previsioni per ciascun esempio di test l’etichetta della classe corrispondente. Sia quindi il vettore di previsione di un algoritmo di previsione Random Forest, e sono i vettori di previsione risultanti dagli alberi decisionali da 1 a . Considerato che sono solo due le labels, “0” e “1”, il vettore ensemble della Random Forest viene determinato come segue (vedi (3.19) o (3.20):

Oppure:

In quest’ultimo caso si calcola la media dei singoli vettori di previsione. I risultati ottenuti per ognuno dei datapoints di test vengono arrotondati ottenendo valori interi, “0” oppure “1”

Presentiamo, quindi, i principali parametri del Random Forest Classifier in *scikit\_learn* [14], da ottimizzare:

* Il numero di alberi (“n\_estimators”) di cui si compone la foresta. In generale maggiore è il numero di alberi minore risulterà la varianza del classificatore, ma il prezzo da pagare è il maggiore costo computazionale
* Il numero di caratteristiche estratte casualmente (“max\_features”). È opportuno scegliere un numero di caratteristiche relativamente piccolo su cui esercitare il training, ad esempio la radice quadrata del numero totale di features [5] [14]. Un numero relativamente elevato di caratteristiche potrebbe portare al problema dell’*overfitting*
* La dimensione del campione di bootstrap (“'max\_samples'”), la quale però di default produce un buon compromesso tra *bias* e varianza del modello
* Vengono accettati inoltre tutti i parametri del modello Decision Tree. Se non immessi il Random Forest Classifier utilizza valori di default. In presenza di molti alberi la differenza di prestazioni dovuta alla diversità dei parametri degli alberi si attenua. Tuttavia, se si impiegano proprio i parametri ottimizzati dell’algoritmo Decision Tree (invece che quelli di default), si ottengono sul *train set* del Titanic 0.02 – 0.03 punti in più in termini di accuratezza in fase di convalida. Vengono, dunque, utilizzati i parametri ottimizzati degli alberi

*3.5 K-Nearest Neighbors*

Questo metodo di classificazione consiste nell’assegnare ad un qualsiasi punto, in uno spazio vettoriale, avente coordinate , la label prevalente di un certo numero di *train datapoints* vicini. In questo modo è possibile sia confrontare il risultato della classificazione con la vera label degli esempi di addestramento, sia predire le labels di un qualsiasi esempio di test, noti i suoi valori . Si nota che per valutare correttamente le distanze è necessario annullare l’effetto delle differenti scale delle *features* (dovuti ad esempio a diverse unità di misura). Per rendere i ranges delle variabili confrontabili, bisogna normalizzare le caratteristiche [5]. Questa è la formula per la normalizzazione:

Fatta questa necessaria precisazione, si presentano i principali parametri di questo tipo di classificatore in *scikit learn*, che possiamo ottimizzare:

* Il numero dei punti più vicini da considerare (“n\_neighbors”) per la classificazione. Ad esempio se 3 dei 5 punti più vicini ad un certo punto di test hanno come label “1” (invece che “0”), il classificatore assegnerà al punto di test l’etichetta 1. In caso di pareggio l’implementazione di questo algoritmo in *scikit learn* farà prevalere la label dei vicini con la minore distanza del punto da classificare. Un basso valore di “n\_neighbors” comporta un’elevata influenza dei pochi punti vicini, con l’eventualità di generare una classificazione troppo complessa (*overfitting*). Al contrario un alto valore di questo parametro ha la conseguenza di rendere la classificazione di un punto influenzata anche da valori lontani, con il rischio dell’ottenimento di previsioni distorte
* Un altro parametro rilevante è il “weights”. Nel caso in cui il valore di questo parametro sia “uniform” il peso di ogni punto compreso negli “n\_neighbors” più vicini è identico: i punti distanti hanno lo stesso peso dei punti vicini. Al contrario se il valore del parametro è “distance” i punti più vicini avranno un peso maggiore rispetto ai più distanti. Più precisamente tra gli “n\_neighbors” più vicini i pesi vengono assegnati in modo inversamente proporzionale alla distanza del punto da classificare. In questo modo nel caso si scelga un alto valore di “n\_neighbors” si attribuisce maggiore influenza ai punti più vicini attenuando il rischio di *biased predictions*.

*3.6 Logistic Regression*

È possibile utilizzare la regressione logistica per poter prevedere la variabile target di *datapoints* di test. Con questo approccio è anche possibile prevedere con quale probabilità si verifichi un certo valore di . In questo contesto la variabile target è connessa al previsore lineare , ed è compresa tra 0 e 1. Si può esprimere il previsore lineare in forma più esplicita così:

(3.22)

Il legame tra la label Y e le *features* è garantito dalla link function logit (Eq. 3.23) [15][20]. Per semplicità chiamiamo la probabilità di sopravvivenza

La funzione inversa della *logit,* detta anche sigmoide logistica (Eq. 3.24) [5], ci fornisce la probabilità che un certo esempio appartenga alla classe “1”:

A questo punto conosciamo i valori di z e i valori di probabilità che Y=1. La funzione sigmoide ha valore vicino allo 0 per . Aumenta dapprima lentamente, poi sempre più velocemente fino ad arrivare a , in corrispondenza del quale , e cambia concavità. In seguito cresce sempre meno velocemente per tendere asintoticamente a . Oltre , è molto vicino a 1. È possibile così determinare le previsioni della label, sulla base delle *features* :

(3.25)

Ciò equivale a:

(3.26)

I coefficienti del modello di regressione logistica vengono stimati minimizzando la funzione di costo [5], in cui è l’indice degli esempi

Allo scopo una strategia per calcolare i coefficienti è la massimizzazione della funzione , detta log-likelihood [5] [20].

Nell’esercizio si è deciso di testare diverse penalità assegnate per l’introduzione di ulteriori regressori. Questo di fatto si ottiene riducendo i valori dei parametri del modello di un fattore . Come si può vedere dall’Eq. (3.29), all’aumentare di , si riducono i valori dei pesi del modello, specialmente di riduce l’influenza dei pesi meno significativi. In questo modo si aumenta l’intensità della regolarizzazione, cercando di ridurre la complessità del modello [5]. Per stimare i coefficienti si minimizza la seguente funzione di costo

In cui l’indice delle caratteristiche spazia da 1 a , e ne consegue che la regolarizzazione non agisce sull’intercetta. In scikit learn si impiega una funzione di minimizzazione di costo simile alla Eq. (3.29), e si usa un parametro che è l’inverso di [21].

Pertanto, minore è il valore di , più forte è la regolarizzazione, e il modello tende a diventare meno complesso.

**4. Identificazione e valutazione dei modelli selezionati**

Come premessa, si riporta la metrica utilizzata per valutare i modelli ossia l’accuratezza (in inglese *accuracy*) in fase di convalida. Prima verrà spiegato più in dettaglio il concetto di accuratezza e poi verrà approfondita la disamina della convalida K-fold.

Un’espressione dell’accuratezza è la seguente [2][4][5]:

È una metrica di valutazione adatta quando c’è un certo equilibrio nella dimensione delle varie classi. Tutto sommato in questo caso le classi sono equilibrate, 61% deceduti (label “0”) e 39% sopravvissuti (label “1”). Laddove c’è un forte squilibrio tra classi è più importante limitare il numero di falsi negativi o falsi positivi piuttosto che l’accuratezza complessiva. Ad esempio un test medico non è considerato attendibile se non limita al massimo il numero di falsi negativi, nonostante possa avere una buona *accuracy*. Oppure nel caso di una frode bancaria, per avere la maggior possibilità che gli indagati siano davvero colpevoli di trovare i colpevoli, è necessario usare un algoritmo che limiti il numero di falsi positivi. Nel primo caso si ottimizzerà l’algoritmo di previsione in funzione del *recall* (Eq. 4.2), nel secondo in funzione della *precision* (Eq. 4.3)

A differenza dei malati e dei colpevoli che spesso sono una piccola percentuale degli esempi totali, le percentuali di sopravvissuti e di deceduti al naufragio del Titanic sono relativamente equilibrate. Quindi, la metrica dell’*accuracy* è idonea.

Nello spiegare la *cross validation* facciamo ricorso ad un esempio di codice utilizzato dal classificatore Support Vector Machine. Per validare un modello bisogna prima creare un classificatore (es. di tipo SVC)

clf=svm.SVC()

Viene creato inizialmente senza parametri perché vogliamo valutare i diversi modelli che si ottengono attraverso tutte le possibili combinazioni di parametri presenti nel dizionario “parameters”:

parameters = {'kernel':['rbf'],'C':[10000,20000], 'gamma': [0.05,0.1]}

Il dizionario “parameters” è l’argomento di un oggetto GridSearchCV che consente in modo veloce di creare modelli basati su tutte le possibili combinazioni dei valori dei parametri in “parameters”

clf=GridSearchCV(clf,parameters,refit=True,cv=kf)

Il clf ora diventa una griglia di classificatori. Un interessante argomento è cv=kf. In cui kf è un oggetto di tipo KFold (cv sta per cross validation). Si noti che per valutare un modello è possibile scegliere casualmente un set di esempi su cui il modello non viene stimato (noto come out-of-sample), ma su cui viene valutata l’accuratezza (ottenendo le previsioni , e comunque conoscendo i veri valori della label ). Quanto più in media le previsioni si avvicinano a , tanto migliori dovrebbero essere le capacità di generalizzazione del modello che ha prodotto le . Un’evoluzione dell’approccio out-of-sample consiste nella convalida incrociata, nota in ambiente Python come KFold.

Il KFold consente di dividere in k parti il dataset facendo ruotare il set di valutazione (o di convalida) del modello. In questo modo la valutazione delle prestazioni dei singoli modelli diventa meno sensibile rispetto al semplice out-of-sample, i cui dati potrebbero avere patterns particolari.

In questo modo è possibile stimare ogni modello definito dal dizionario dei parametri, su ogni k set di validazione. Si ottengono così le metriche di accuracy per ogni modello per ogni fold. Si arriva così a calcolare l’accuratezza media di ogni modello di classificazione, basata su tutti i fold. Attraverso la seguente istruzione si addestra la griglia dei classificatori sui dati di train

clf.fit(features\_train\_norm,labels\_train\_norm)

E con la seguente istruzione si stampano i parametri corrispondenti al miglior modello in fase di convalida

print("Miglior classificatore Support Vector Machine ",clf.best\_params\_)

Le seguenti righe di codice costruiscono un DataFrame contenenti queste informazioni sui vari modelli ottenuti:

* Il tipo di modello (SVM, GaussianNB ecc.)
* I parametri impiegati
* L’accuratezza (media) di convalida

df\_perf\_SVM=pd.DataFrame()

mean\_accuracies=clf.cv\_results\_['mean\_test\_score']

parameters\_tested=clf.cv\_results\_['params']

type\_clf="svm.SVC"

clf\_type\_list=[]

for i in range(len(parameters\_tested)):

clf\_type\_list.append(type\_clf)

df\_perf\_SVM['type'] = clf\_type\_list

df\_perf\_SVM['parameters']=parameters\_tested

df\_perf\_SVM['performances']=mean\_accuracies

Di seguito presentiamo i modelli prodotti e le relative accuratezze per ogni tipo di classificatore presentato nella Sez. 3. La griglia dei parametri dei classificatori è stata selezionata per tentativi. Si è comunque cercato di limitare il numero dei valori dei parametri in griglia, per non incorrere in eccessivi costi computazionali

Sulla base dei parametri selezionati, viene istanziato il migliore modello:

best\_clf\_SVM=svm.SVC(kernel=best\_par['kernel'],C=best\_par['C'],gamma=best\_par['gamma'])

Poi viene stimato sui dati di addestramento

best\_clf\_SVM.fit(features\_train\_norm,labels\_train\_norm)

Infine viene determinato il vettore delle previsioni, noti i valori delle features del set di test:

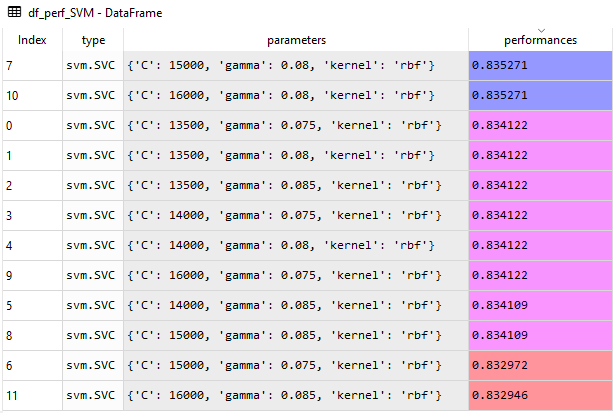
preds\_SVM=best\_clf\_SVM.predict(features\_test\_norm)

**4. Risultati dei classificatori in fase di convalida incrociata**

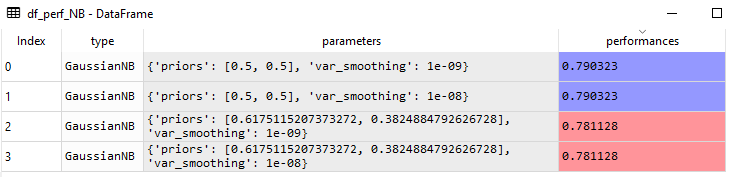
Di seguito presentiamo i risultati in termini di *accuracy* dei modelli stimati, secondo i parametri di volta in volta testati, all’interno del dizionario.

Alcune considerazioni:

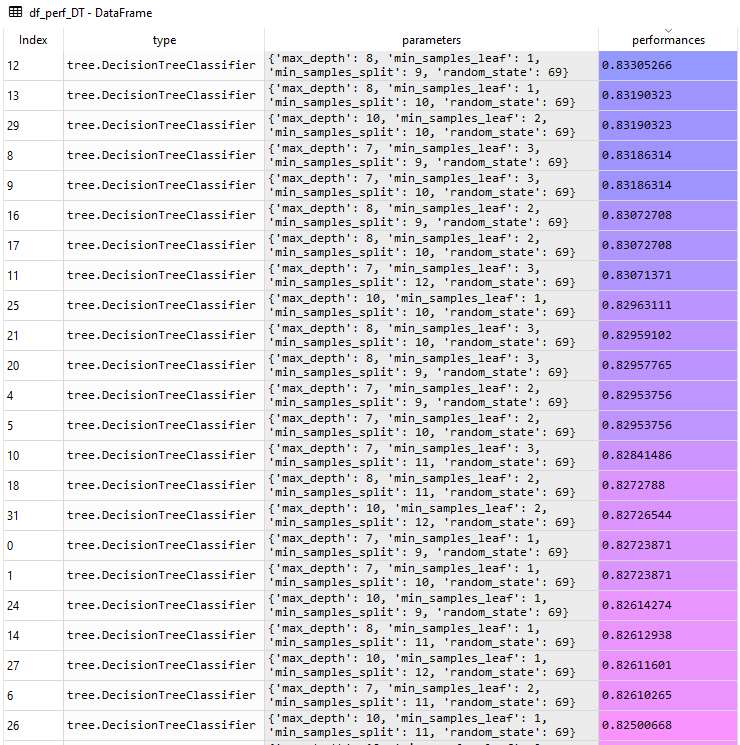
* Nel caso degli algoritmi Support Vector Machine e K-Nearest Neighbors sono state normalizzate (secondo la Eq. 3.21). è stato necessario riportare alla stessa scala variabili con ranges diversi, in questi algoritmi che dipendono fortemente dalla distanza tra i punti nello spazio vettoriali
* Il classificatore GaussianNB è risultato il peggiore in fase di convalida, nonostante si sia cercato di agire sul parametro “priors”. Questo parametro è facoltativo e indica le probabilità di appartenenza alle singole labels. Conoscendo la proporzione dei sopravvissuti al naufragio, per la label “0” si è provato ad assegnare 0.6175, mentre per la “1” si è imputato 0.3825. Purtroppo non si è ottenuto un miglioramento dell’accuratezza di convalida. L’altro parametro “var\_smoothing” concerne la stabilità dei calcoli e non incide sulle performance
* Nel caso del Decision Tree classifier sono stati presi in considerazione un maggior numero di valori dei parametri, poiché gran parte dei valori testati e gran parte delle relative combinazioni producono risultati simili in termini di accuratezza di convalida. Pertanto, nel tentativo di ottenere il miglior modello possibile, a mio avviso non era opportuno ridurre eccessivamente il numero di valori dei parametri
* Il classificatore Random Forest è quello in media più performante. Questo per due motivi: il primo è che si tratta di un *ensemble method*, poiché calcola la media delle previsioni provenienti da vari alberi decisionali, basati su diversi campioni di *bootstrap*. Questo approccio dovrebbe ridurre la varianza relativamente elevata di cui soffrono i singoli alberi decisionali. Il secondo motivo è che gli alberi scelti per comporre la foresta presentano i valori dei parametri ottimizzati. Questa scelta ha permesso di guadagnare 0.01 – 0.02 punti nel risultato di accuratezza di convalida
* Nella Logistic Regression il parametro principale che è stato ottimizzato è stato C. Minore è il parametro , maggiore è la penalità assegnata per l’introduzione di nuove variabili. Si può notare che il valore di selezionato non è “0”. Il solver “lbfgs” è un algoritmo di ottimizzazione.



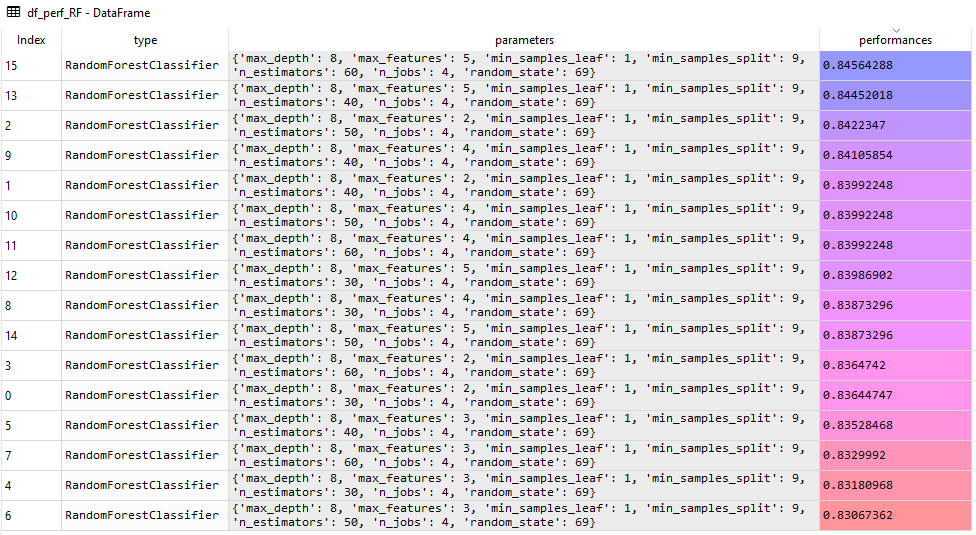
*Fig. 10. Risultati dell’accuracy di convalida del Support Vector Machine Classifier*



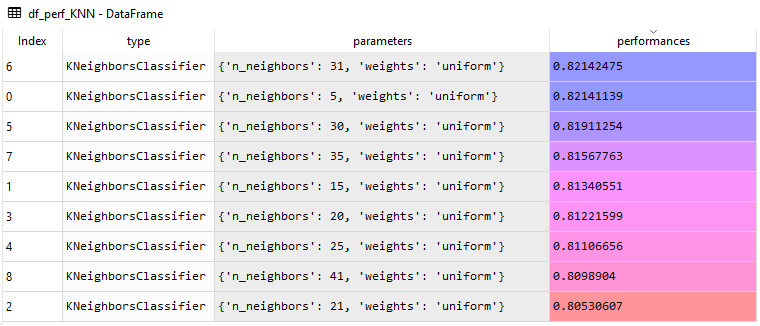
*Fig. 11. Risultati dell’accuracy di convalida del Gaussian Naive Bayes Classifier*



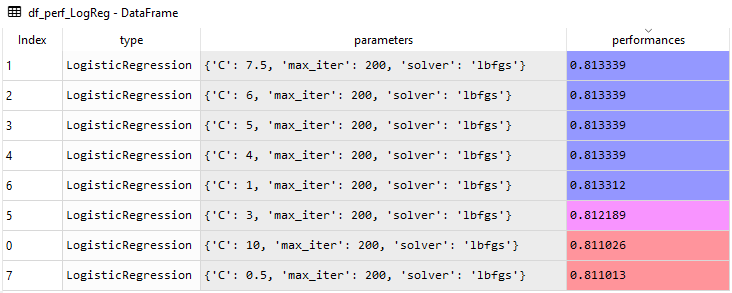
*Fig. 12. Risultati dell’accuracy di convalida del Decision Tree Classifier*



*Fig. 13. Risultati dell’accuracy di convalida del Random Forest Classifier*



*Fig. 14. Risultati dell’accuracy di convalida del K-Nearest Neighbors*



*Fig. 15. Risultati dell’accuracy di convalida del Random Forest Classifier*

*The original ensemble method*

Come detto, l’idea è quella di ottenere un vettore di previsioni (), che sia una media dei migliori algoritmi di classificazione e/o regressione. Si costruisce un originale *ensemble method*, ispirato alla classificazione a maggioranza.

Si è deciso di far partecipare all’”elezione” il classificatore migliore di ogni gruppo, oltreché quelli che hanno superato una certa soglia di *accuracy* di convalida (in questo caso 0.834) in modo tale da:

* Considerare un appropriato numero di modelli (venti-venticinque), scartando quelli che hanno presentato basse accuratezze di convalida, che quindi dovrebbero avere minori capacità di previsioni su dati mai visti. L’adozione di un approccio di *Ensemble learning* dovrebbe produrre previsioni più accurate e affidabili sui dati di test, purché vengano selezionati buoni classificatori
* La considerazione di ogni famiglia di classificatori (attraverso il modello migliore) potrebbe compensare un eventuale effetto di *bias* nelle previsioni, che si potrebbe avere considerando una sola famiglia

Mediante le seguenti righe di codice si presentano i classificatori “votanti” e il loro peso nell’”elezione”, ossia nella determinazione del vettore finale di previsioni (si veda nello script l’oggetto df\_perf\_voting). Si noti che df\_perf\_all rappresenta l’unione di tutti i modelli utilizzati, df\_perf\_group\_best è il DataFrame dei migliori modelli di classificatori per famiglia, e df\_perf\_eligible è il DataFrame che contiene i classificatori con accuratezza superiore alla soglia impostata. Combinando df\_perf\_group\_best e df\_perf\_eligible, si ottiene al DataFrame dei classificatori votanti df\_perf\_voting (eliminando i duplicati): Si veda il codice seguente:

df\_perf\_eligible=df\_perf\_all.loc[(df\_perf\_all['performances'] > 0.834)]

df\_perf\_group\_best=df\_perf\_all.loc[df\_perf\_all.groupby('type').performances.idxmax()]

df\_perf\_voting=pd.concat([df\_perf\_eligible,df\_perf\_group\_best])

df\_perf\_voting = df\_perf\_voting.reset\_index()

df\_perf\_voting=df\_perf\_voting.drop\_duplicates(subset='new index')

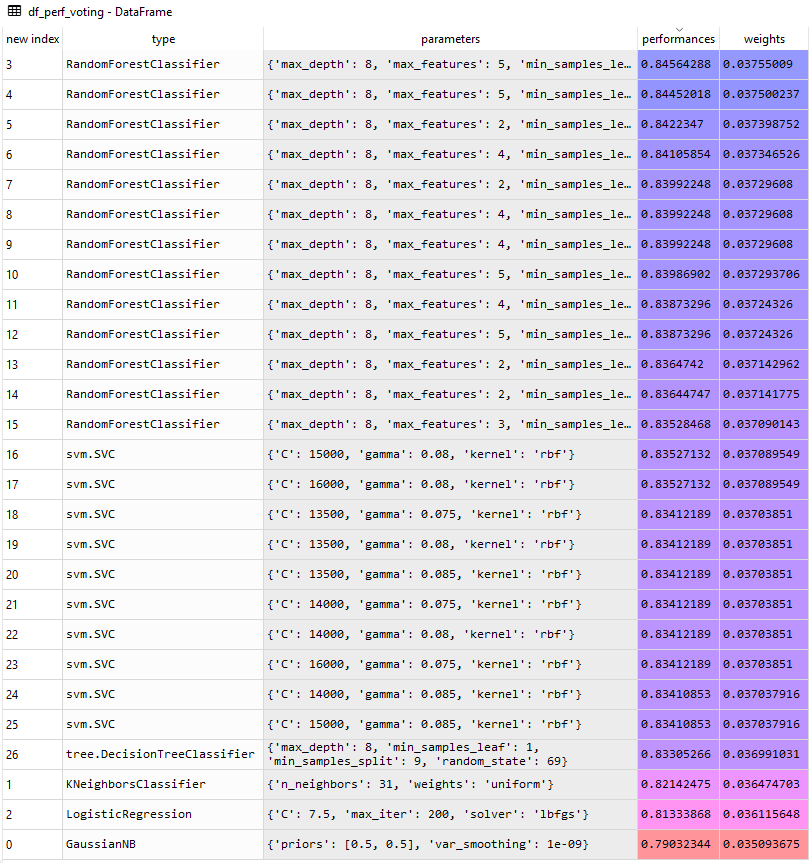
Per la ponderazione dei classificatori, si è scelto di assegnare un’importanza maggiore ai classificatori con maggiore accuratezza di convalida, secondo la formula:

In cui è l’accuratezza di convalida (incrociata) del classificatore . Mentre è la somma delle accuratezze dei classificatori utilizzati, e serve per normalizzare il peso del singolo classificatore.

Sia il vettore orizzontale () dei pesi (Eq. 4.5):

Nel *jupyter notebook*, a corredo della relazione, si chiama weights

weights=np.array(df\_perf\_voting['weights']).T



*Fig. 16. Classificatori votanti: accuratezza in fase di convalida e peso nell’”elezione”*

In seguito, si è costruito una lista predictions\_vectors contenente i vettori di previsione dei dati di set, per ogni classificatore stimato. Dopodiché questa lista viene trasformata in una matrice con dimensioni (), in cui è il numero di esempi di test (cioè 418). Si può rappresentare la matrice nel modo seguente, in cui ogni , per da 1 a , è un vettore orizzontale di previsione

Nel codice la matrice si chiama prediction\_matrix:

prediction\_matrix=np.array(predictions\_vectors)

Noti i pesi relativi ai k classificatori e la matrice delle previsioni dei k classificatori, è possibile finalmente ottenere il vettore ensemble delle previsioni dei () esempi di test, mediante il prodotto vettoriale tra e

Il risultato è un vettore (), ed è stato ottenuto con l’istruzione

votes\_float=np.dot(weights,prediction\_matrix)

Essendo una media ponderata ha come formato float, quindi per avere un vettore composto solo da “0” e “1”, bisogna arrotondare a intero, e convertirlo in formato intero, perché sia accettato da Kaggle. Sia la versione arrotondata di :

(4.8)

Il risultato è un vettore (), ed è l’obiettivo finale dell’esercizio. Esso è stato ottenuto con la seguente riga di codice in Python:

votes=np.round(votes\_float,0).astype(int)

**5. Risultati sul test set e conclusioni**

Presentiamo i risultati dell’accuratezza delle previsioni sul dataset di test, sottoponendo a Kaggle:

* Il vettore Ensemble delle previsioni
* I vettori del miglior modello in fase di convalida di ogni famiglia di classificatori

*Tab. 2. Risultati dell’accuratezza sul dataset*

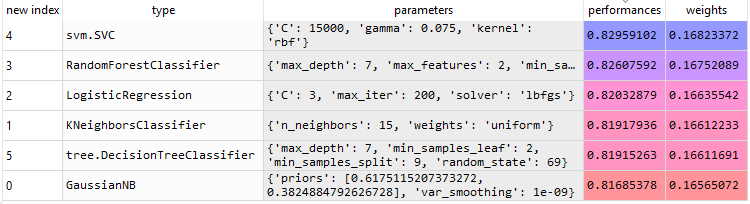
|  |  |  |
| --- | --- | --- |
| Classificatore | Parametri | Accuratezza sul test set |
| Ensemble () | Pesi | 0.78468 |
| Support Vector Machine (best) | 'C': 15000, 'gamma': 0.08, 'kernel': 'rbf' | 0.78229 |
| Gaussian Naive Bayes (best) | 'priors': [0.5, 0.5], 'var\_smoothing': 1e-09 | 0.74401 |
| Decision Tree (best) | 'max\_depth': 8, 'min\_samples\_leaf': 1, 'min\_samples\_split': 9 | 0.75598 |
| Random Forest (best) | Stessi del Decision Tree, 'max\_features': 5, 'n\_estimators': 60 | 0.77033 |
| K-Nearest Neighbors (best) | 'n\_neighbors': 31, 'weights': 'uniform' | 0.76794 |
| Logistic Regression (best) | 'C': 7.5, 'max\_iter': 200, 'solver': 'lbfgs' | 0.77272 |

Da questi risultati emergono alcune interessanti considerazioni conclusive:

* Il miglior classificatore in fase di test è proprio l’Ensemble. Di conseguenza, l’Ensemble si è rivelato capace di fornire previsioni più accurate rispetto ai singoli migliori classificatori di ogni famiglia. I risultati dell’esercitazione mostrano che l’adozione di un approccio Ensemble, purché ottenuta mediando buoni modelli, può fornire risultati più consistenti e affidabili rispetto a singoli modelli. Ciò non significa che il modello Ensemble fornisca sempre prestazioni migliori rispetto a tutti i singoli classificatori su cui si basa, ma è comunque un approccio appropriato se si vuole ottenere buoni punteggi in fase previsiva
* Il punteggio delle *accuracies* di test è inferiore di circa 0.05 – 0.06 punti rispetto a quelle della fase di convalida. A mio avviso il fatto può avere due spiegazioni. La prima è che i dati, su cui i modelli vengono stimati, sono gli stessi sui quali i modelli vengono validati, pur ruotando di volta in volta lo strato train – validazione. La seconda spiegazione, collegata alla prima, è che nei dati di test ci possono essere delle peculiarità per qualche motivo non spiegabili dai dati di addestramento
* Non sempre il miglior classificatore in fase di convalida, è risultato il migliore in fase di test. In fase di convalida i migliori modelli erano tendenzialmente di tipo Random Forest, invece il singolo modello più performante in fase di test è stato un Support Vector Machine, con prestazioni vicine all’Ensemble. Questa differenza può essere dovuta alla casualità con cui sono stati scelti i dataset di train e di set. Comunque la differenza di accuratezza tra i due best models è di circa 0.01 sia in fase di convalida, sia in fase di test
* Tutti i modelli presentano buoni valori di accuracy (>0.75) in fase di test, tranne il GaussianNB che comunque anche in fase di convalida si è rivelato il peggiore.
* Il valore dell’accuracy di test 0.785 ottenuto con il modello Ensemble è sicuramente un buon punteggio. Infatti, questa discussione su Kaggle [19] sostiene che ottenere un’accuratezza di test pari o superiore a 0.80 è un ottimo risultato. Non sorprende che sia piuttosto complicato scollinare questa soglia. Come visto nella Sez. 2.2 gli indici e si aggirano intorno al 40%, quindi le variabili disponibili spiegano meno della metà della varianza della variabile target (la sopravvivenza). Di conseguenza, è molto improbabile ottenere un accuracy score sul test molto vicini a 1, in maniera pulita
* I valori di *accuracy* sul test considerando solo le variabili significative sono simili a quelli dei modelli con tutte le variabili. In particolare l’accuratezza dell’Ensemble è la stessa 0.78468 (per ulteriori dettagli si rimanda all’Appendice A
* Un’ultima nota: è possibile, in fase di ri-sottoposizione dei vettori a Kaggle, ottenere *accuracies* di convalida e di test, che possono variare di pochi millesimi. Infatti, confrontando alcuni vettori ensamble generati dallo stesso file, si è stimato che la deviazione standard dei risultati di accuratezza è di 0.0029. Questo avviene per la casualità che incide sulla stima di alcuni parametri (ad esempio quelli stimati con metodi iterattivi). Comunque, nella cartella “Previsioni definitive variabili complete” vengono presentati i vettori di previsione corrispondenti ai risultati indicati ricorrendo a tutte le variabili. Nella cartella “Previsioni definitive variabili selezionate” si presentano i vettori sottoposti, usando solo le variabili significative.

Appendice A. Selezione delle variabili: risultati di accuracy sulla convalida e sul test

In questa appendice si vedono i risultati dell’accuracy in fase di convalida (Fig. A1) e quelli ottenuti sul data set di test (Tab. 3)



Di seguito si trovano i risultati di accuratezza sul test ottenuti con Kaggle

*Tab. 2. Risultati dell’accuratezza sul dataset (variabili significative)*

|  |  |  |
| --- | --- | --- |
| Classificatore | Parametri | Accuratezza sul test set |
| Ensemble () | Pesi | 0.78468 |
| Support Vector Machine (best) | 'C': 15000, 'gamma': 0.075, 'kernel': 'rbf' | 0.78708 |
| Gaussian Naive Bayes (best) | 'priors': [0.618, 0.382], 'var\_smoothing': 1e-09 | 0.77511 |
| Decision Tree (best) | 'max\_depth': 7, 'min\_samples\_leaf': 2, 'min\_samples\_split': 9 | 0.75837 |
| Random Forest (best) | Stessi del Decision Tree, 'max\_features': 2, 'n\_estimators': 60 | 0.77272 |
| K-Nearest Neighbors (best) | 'n\_neighbors': 15, 'weights': 'uniform' | 0.74880 |
| Logistic Regression (best) | 'C': 3, 'max\_iter': 200, 'solver': 'lbfgs' | 0.77511 |

L’approccio *Ensemble* si conferma buono anche usando modelli con le sole variabili significative: si ottiene lo stesso valore rispetto al caso dell’insieme completo delle *features* (0.78468). In realtà il miglior modello SVM supera di molto poco il valore dell’Ensemble. I risultati appaiono comunque simili al caso dell’insieme completo delle variabili.

**Riferimenti**

[1] Kaggle. Link: <https://www.kaggle.com/c/titanic>

[2] Thrun S., Malone K. Course Introduction to Machine Learning, Udacity <https://www.udacity.com/course/intro-to-machine-learning--ud120>

[3] Scikit Learn. sklearn.model\_selection.GridSearchCV documention <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV>

[4] Ng A. Machine Learning Course, Coursera <https://www.coursera.org/learn/machine-learning>

[5] Raschka S., Mirjalili V (2019), Python Machine Learning – Third Edition, Packt Publishing

[6] Govoni L. Algoritmo Support Vector Machine <https://lorenzogovoni.com/support-vector-machine/>

[7] Scikit Learn. sklearn.svm.SVC¶ documentation <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

[8] Wikipedia, Naive Bayes classifier, <https://en.wikipedia.org/wiki/Naive_Bayes_classifier>

[9] Scikit learn sklearn.naive\_bayes.GaussianNB documentation¶<https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html>

[10] Bagattini F. (2019), Introduzione alla Data Science con Python – slides start2impact

[11] James G. et al. (2013), An Introduction to Statistical Learning with Applications in R, Springer <http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf>

[12] Scikit Learn, sklearn.tree.DecisionTreeClassifier Documentation <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>

[13] Bagattini F. (2019), Introduzione alla Data Science con Python – Concetti Avanzati– slides start2impact, <https://drive.google.com/file/d/1SavLO-CqYxLGfKsnzsnd9b2qjubbljQN/view>

[14] Scikit Learn, KNeighborsClassifier Documentation, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

[15] Smithsons M, Merkle EC. Generalized linear models for categorical and continuous

limited dependent variables. New York: Chapman & Hall, CRC Press; 2014.

[16] Johnston J. (1984), Econometric Methods, third edition, McGraw-Hill, New York, pp. vii + 568, ISBN 0-07-032685

[17] Brownlee J. (2020) 4 Automatic Outlier Detection Algorithms in Python. <https://machinelearningmastery.com/model-based-outlier-detection-and-removal-in-python/>

[18] Scikit learn. sklearn.ensemble.IsolationForest¶ Documentation <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html>

[19] Kaggle. Titanic leaderboard: a score > 0.8 is great! .<https://www.kaggle.com/carlmcbrideellis/titanic-leaderboard-a-score-0-8-is-great>

[20] Pereira M. et al. (2016) The Logistic Lasso and Ridge Regression in Predicting Corporate Failure, Procedia Economics and Finance, Volume 39, Pages 634-641 <https://www.sciencedirect.com/science/article/pii/S2212567116303100>

[21] Scikit learn. sklearn.linear\_model.LogisticRegression Documentation <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression>

[22] Barbiero T, Grillenzoni C. (2019). A statistical analysis of the energy effectiveness of building refurbishment. Renew Sustain Energy Rev 2019;114:109297. https://doi.org/10.1016/j.rser.2019.109297.

1. Nella letteratura di analisi statistica la selezione delle variabili viene in molti casi effettuata tenendo conto delle statistiche T e dei p-value. Seguendo questo approccio questo paper [22] ha valutato l’importanza dei fattori che spiegano le ristrutturazioni edilizie. [↑](#footnote-ref-1)