Progetto Machine Learning IADA 2024

Cardia Francesco[*], Chiesa Matteo[†], Zedda Giovanni[‡]

9 febbraio 2024

1 Dataset

1.1 Provenienza e storia del dataset

Il dataset "Polish companies bankruptcy data" è una raccolta di dati per task di classificazione, riguardo le predizioni di bancarotta delle aziende polacche. I dati delle aziende in bancarotta sono stati prelevati in un periodo di 12 anni che va dal 2000 al 2012, mentre i dati delle aziende ancora operative sono stati raccolti in un periodo di 6 anni che va dal 2007 al 2013; sono stati raccolti da Emerging Markets Information Service (EMIS), un curatore di ricerca multisettoriale e multinazionale per i mercati in più rapida crescita del mondo.

1.2 Informazioni sul dataset

Il dataset presenta un totale di 64 attributi continui, rappresentanti una serie di rapporti finanziari utilizzati per valutare la performance di un'azienda. Includono misure di redditività, liquidità, efficienza operativa e leva finanziaria, e infine un'etichetta di classe binaria che indica se i dati appartengono ad un azienda che è andata in bancarotta, oppure ad un'azienda operativa.
I dati raccolti sono divisi in 5 parti ("*1year.arff*", "*2year.arff*", "*3year.arff*", "*4year.arff*", "*5year.arff*") con differenti periodi di rilevazioni, da 1 a 5 anni, e dopo alcune analisi effettuate con le funzioni di Scikit-Learn abbiamo deciso di operare su "*5year.arff*". Il dataset da noi analizzato consta di 5910 istanze, per alcune delle quali sono presenti valori mancanti in alcuni attributi.

1.3 Analisi dei dati

Abbiamo fatto un'analisi sulla bontà del dataset grazie ad alcune misure come il bilanciamento tra classe minoritaria e maggioritaria, il numero e la percentuale di valori mancanti, la deviazione standard media degli attributi e il numero e la percentuale di outlier. Si può visualizzare l'analisi dei dati grazie all'apposito

[*]60/79/00112
[†]60/79/00096
[‡]60/79/00110

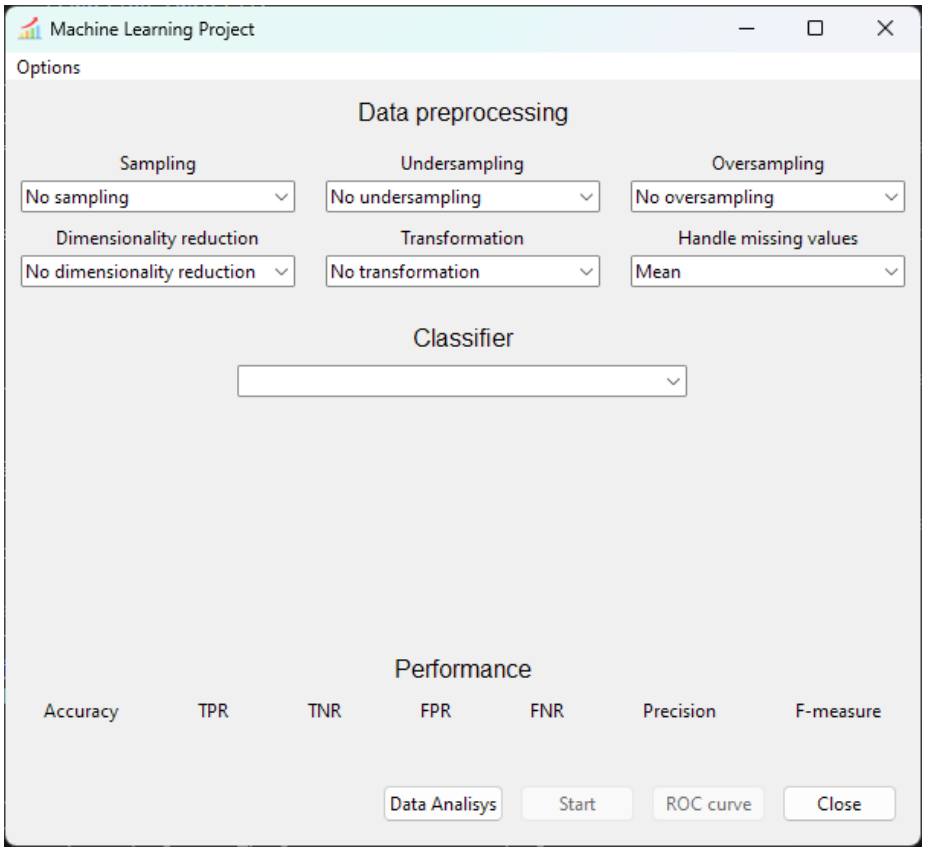

Figura 1: Interfaccia grafica

pulsante nell'interfaccia grafica, dalla quale è poi possibile selezionare un singolo attributo e visualizzarne le statistiche in maniera più dettagliata.

2 Menu grafico

2.1 introduzione

All'esecuzione del file "*main.py*", viene mostrato un menu grafico realizzato con l'ausilio della libreria Python Tkinter.

Dal menu visualizzato in alto, selezionando la voce "Options", è possibile abilitare il tuning degli iperparametri in real time per i classificatori che ne fanno uso.

2.2 Data pre-processing

Grazie al menu è possibile selezionare le tecniche di pre-processing dei dati che si vogliono applicare ai dati, che comprendono:

1. Sampling
2. Dimensionality reduction
3. Undersampling
4. Oversampling
5. Transformation

Dal menù si può anche scegliere con quale tecnica gestire i valori mancanti nel dataset, come ad esempio usare la media di quell'attributo, il valore più frequente o l'uso di record vicini.

2.3 Classificatori

È possibile scegliere tra 5 differenti classificatori quale utilizzare per la predizione:

1. Albero Decisionale
2. k-Nearest-Neighbor
3. Support Vector Classifier
4. Custom Naive Bayes
5. Custom Ensemble

In base al classificatore scelto compaiono degli elementi aggiuntivi nella GUI, come ad esempio la scelta della misura di purezza per l'albero decisionale, il metodo per misurare la distanza nel k-NN e la scelta se ridurre o no l'influenza del parametro k pesando il contributo del vicino in base alla distanza dal record da classificare, la Kernel Function da utilizzare per la SVM, e infine per il Custom Ensemble è possibile scegliere il metodo di costruzione del classificatore, il metodo di voting, e i pesi da applicare ai singoli classificatori.

2.4 Performance

Per finire nel fondo del menu è presente la sezione Performance, che permette di visualizzare le performance ottenute dal classificatore sul test set con le misure di:

1. Accuracy: frazione delle previsioni che il modello ha ottenuto correttamente.

2. TPR: noto anche come sensibilità o recall, è usato per misurare la percentuale di positivi reali che vengono correttamente identificati.

3. TNR: noto anche come specificità, è il rapporto dei veri negativi rispetto alla somma dei veri negativi e dei falsi positivi.

4. FPR: proporzione di casi negativi erroneamente identificati come casi positivi (cioè la probabilità che vengano sollevati falsi allarmi).

5. FNR: rapporto dei falsi negativi rispetto alla somma dei falsi negativi e dei veri positivi;

6. Precision: proporzione di classificazioni positive corrette (veri positivi) divisa per il numero totale di classificazioni positive predette.

7. F-Measure: nota anche come F1-Score, è una misura dell'accuratezza di un modello di classificazione che tiene conto sia della precisione che del recall. Fornisce un singolo valore che rappresenta la media armonica di queste due metriche, offrendo quindi una visione equilibrata delle prestazioni del modello.

2.5 Menù

Alla fine della schermata sono presenti inoltre 4 bottoni:

1. Data Analysis: permette di visualizzare delle informazioni sul dataset.

2. Start: inizia l'addestramento del classificatore selezionato con le eventuali tecniche di preprocessing dei dati scelte.

3. ROC curve: visualizza la curva ROC del classificatore corrente, è possibile addestrare più classificatori e confrontare le loro curve ROC nello stesso grafico, selezionando dopo l'addestramento di ognuno questo tasto.

4. Close: chiude il programma.

3 Descrizione dei classificatori utilizzati

Come classificatori di libreria abbiamo deciso di utilizzare un albero decisionale, un K-Nearest-Neighbour e un Support Vector Classifier, perché volevamo testare varie tipologie eterogenee di classificatori.

3.1 Albero Decisionale

Un albero decisionale è un algoritmo di apprendimento supervisionato, che divide i dati in sottoinsiemi nell'ottica di massimizzare il guadagno tra nodi. Inizia da un nodo radice, si divide in nodi interni e termina in nodi foglia che rappresentano le classificazioni finali. La potatura viene utilizzata per prevenire l'overfitting.

Per il seguente classificatore abbiamo deciso di effettuare il tuning dei seguenti iperparametri:

- *criterion*: questo parametro determina la funzione utilizzata per misurare l'eterogeneità di un nodo; i valori comuni sono "*gini*" per l'impurità di Gini e "*entropy*" per il guadagno di informazione.
- *max_depth*: questo parametro controlla la profondità massima dell'albero. Se non specificato, l'albero continua a dividere finché tutte le foglie non sono pure o finché tutte le foglie non contengono meno di *min_samples_split* campioni.
- *min_samples_leaf*: questo parametro specifica il numero minimo di campioni richiesti per essere in un nodo foglia.
- *min_samples_split*: questo parametro specifica il numero minimo di campioni richiesti per dividere un nodo interno.

3.2 k-Nearest-Neighbors

L'algoritmo k-Nearest-Neighbors (k-NN) è un algoritmo di apprendimento supervisionato che classifica un nuovo punto dato in base alla classe più comune tra i suoi k vicini. Se *k=1* l'oggetto viene semplicemente assegnato alla classe del suo singolo vicino più prossimo.

Per il seguente classificatore abbiamo deciso di effettuare il tuning dei seguenti iperparametri:

- *n_neighbors*: determina il numero di vicini più prossimi da considerare quando si effettuano le predizioni. Un valore troppo piccolo rende il modello sensibile al rumore, mentre un valore troppo grande può portare ad una rappresentatività dei vicini eccessivamente piccola.
- *weights*: specifica come devono essere distribuiti i pesi tra i valori dei vicini. I valori possibili sono "*uniform*" (tutti i punti sono pesati ugualmente) e "*distance*" (i punti sono pesati in base all'inverso della loro distanza, quindi i record più vicini all'oggetto da classificare avranno una maggiore influenza rispetto ai vicini che sono meno simili).
- *metric*: specifica la metrica da utilizzare per il calcolo della distanza. Il valore predefinito è "*minkowski*", che risulta nella distanza euclidea standard quando *p=2*.

3.3 Support Vector Classifier

Il Support Vector Classifier (SVC) è un algoritmo di apprendimento supervisionato che costruisce iperpiani in uno spazio multidimensionale per separare le classi di dati. L'obiettivo è massimizzare la distanza tra l'iperpiano decisionale

e i vettori di supporto più vicini. Può gestire sia problemi di classificazione lineare che non lineare, attraverso l'utilizzo del kernel trick.

Per il seguente classificatore abbiamo deciso di effettuare il tuning dei seguenti iperparametri:

- *kernel*: specifica la funzione kernel da utilizzare nell'algoritmo. I valori comuni sono "*linear*", "*poly*" e "*rbf*". Un kernel lineare produrrà un iperpiano lineare per separare le classi, mentre i kernel "*poly*" e "*rbf*" possono produrre iperpiani non lineari.
- *gamma*: è un coefficiente del kernel per i kernel "*rbf*" "*poly*". Un valore di gamma più alto può portare a un overfitting, poiché il classificatore cercherà di adattarsi perfettamente ai dati di addestramento.
- C: è il parametro di penalità dell'errore; controlla il compromesso tra un confine decisionale liscio e la corretta classificazione dei punti di addestramento. Valori più alti di C possono portare a un overfitting.

3.4 Custom Naive Bayes (primo classificatore custom)

Il classificatore Naive Bayes è un algoritmo di apprendimento automatico basato sul teorema di Bayes. Questo algoritmo calcola la probabilità di appartenenza a una classe per ogni istanza di dati. L'assunzione fondamentale del Naive Bayes è l'indipendenza condizionale delle caratteristiche.

Metodi utilizzati:

- *__init__*: inizializza il classificatore.
- *fit*: addestra il classificatore in base alla tipologia di approccio scelto per la creazione del modello.
- *predict*: predice la classe di appartenenza di un record sulla base del valore assunto dai suoi attributi.
- *predict_proba*: restituisce la probabilità di appartenenza ad ogni classe;
- score: calcola l'accuratezza del classificatore.
- *set_params*: inserito per accomunare questa classe ai classificatori di libreria.

3.5 Custom Ensemble (secondo classificatore custom)

Un classificatore multiplo è un approccio di apprendimento automatico che combina più classificatori per migliorare l'accuratezza delle previsioni. L'idea è di costruire più classificatori di base e predire la classe di appartenenza di un record aggregando le classificazioni ottenute.

Abbiamo implementato il nostro classificatore multiplo dando la possibilità all'utente di scegliere tramite l'interfaccia grafica:

- l'approccio da utilizzare per la creazione del classificatore:

    1. Majority Voting standard: è un metodo di classificazione che prevede l'assegnazione della classe che è stata "votata" dal maggior numero di classificatori.
    2. Bagging: è un metodo di apprendimento d'insieme comunemente utilizzato per ridurre la varianza in un set di dati rumorosi. Nel bagging, un campione casuale di dati in un set di addestramento è selezionato con sostituzione, il che significa che i singoli punti di dati possono essere scelti più di una volta.
    3. Boosting: è un metodo di apprendimento d'insieme che combina una serie di weak learners in un classificatore forte per ridurre al minimo gli errori di addestramento. Nel boosting, il primo classificatore debole viene addestrato con tutto il dataset, i classificatori successivi verrano addestrati con i record del dataset pesati a seconda dell'accuratezza su ogni record dei classificatori precedenti.
    4. Random Forest: è un algoritmo che combina l'output di più alberi decisionali per raggiungere un unico risultato. Oltre a fare uso di bagging, seleziona anche diversi sottoinsiemi casuali di attributi con cui addestrare ciascun albero.

- La tipologia di votazione da utilizzare:

    1. Hard Voting: in questa strategia, ogni classificatore nel gruppo vota per una classe, e la classe che riceve il maggior numero di voti viene scelta come previsione finale.
    2. Soft Voting: invece di classificare il nuovo record con la classe più votata come nel hard voting, ogni classificatore fornisce una probabilità per ciascuna classe. Le probabilità vengono quindi sommate e la classe con la probabilità media più alta viene scelta come previsione finale. Questo metodo è generalmente più accurato rispetto al hard voting, poiché tiene conto della confidenza dei classificatori.

- I pesi da applicare ai singoli classificatori tramite un array di interi, che sono settati di default ad 1 nel caso si decida di effettuare un voting non pesato.

Metodi utilizzati:

- *__init__*: inizializza il classificatore.
- *set_params*: imposta i parametri del classificatore multiplo in base alla scelta effettuata dall'utente sulla GUI.
- *fit*: addestra il classificatore in base alla tipologia di approccio scelto per la creazione del modello.

- *predict*: predice la classe di appartenenza di un record sulla base del valore assunto dai suoi attributi.
- *predict_proba*: restituisce la probabilità di appartenenza ad ogni classe.

4 Tuning

Il tuning è il processo di ottimizzazione degli iperparametri, ovvero i parametri che definiscono la struttura del modello e influenzano la sua performance. Questi parametri non sono appresi dal modello durante l'addestramento, ma devono essere impostati prima di quest'ultimo. Il tuning degli iperparametri può essere fatto manualmente, ma spesso si utilizzano tecniche automatiche come la Grid Search o la Random Search. Queste tecniche provano diverse combinazioni di iperparametri e selezionano quella che produce le migliori prestazioni sul set di validazione. Abbiamo eseguito il processo di tuning tramite il confronto delle prestazioni degli algoritmi con tutte le possibili combinazioni dei parametri, senza pre-processing e con alcune tecniche di pre-processing, selezionando poi i parametri che offrono le migliori prestazioni nella maggior parte di casi, per mezzo della funzione *GridSearchCV* di Scikit-Learn.

5 Confronto senza pre-processing

Abbiamo effettuato un confronto tra i vari classificatori osservando le loro metriche del campo Performance (Tabella 1) e le loro curve ROC (Figura 2) senza applicare alcuna tecnica di pre-processing ai dati, e tenendo le impostazioni di default impostate nel programma per ogni classificatore. Nella legenda delle curve ROC si trova per ognuna anche l'area al di sotto della curva.

Tabella 1: Confronto metriche classificatori senza pre-processing

	Albero	KNN	SVM	Bayes	Ens
Acc.	0.94	0.93	0.93	0.90	0.92
TPR	0.65	0.07	0.00	0.03	0.05
TNR	0.97	0.98	0.99	0.99	0.99
FPR	0.02	0.01	0.00	0.01	0.00
FNR	0.34	0.92	1.00	0.96	0.94
Prec.	0.65	0.29	0.00	0.25	0.85
F-mes.	0.65	0.11	0.00	0.05	0.10

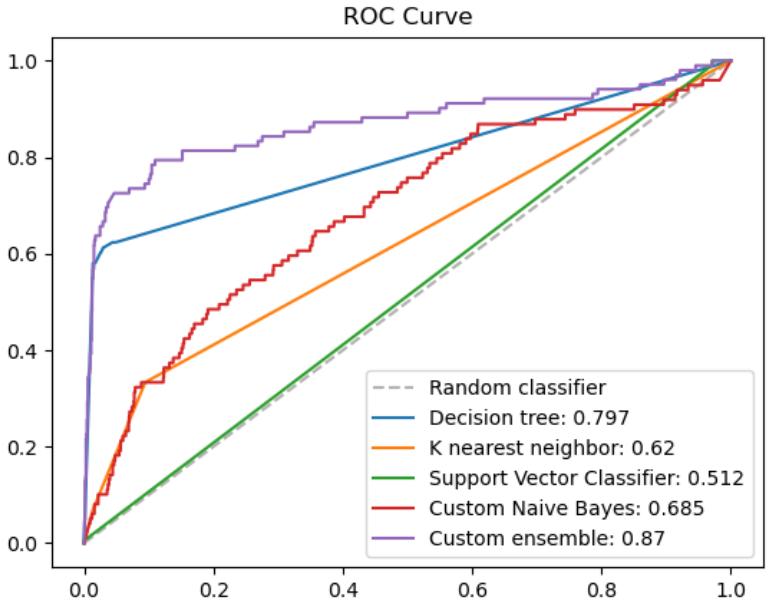

Figura 2: Confronto curve ROC classificatori senza pre-processing

6 Confronto con pre-processing

Abbiamo effettuato un confronto tra i vari classificatori confrontando le loro metriche presentate nel campo Performance, in particolare la loro accuratezza, applicando al dataset una tecnica di pre-processing per volta.

6.1 Confronto con tecniche di campionamento

Il sampling è un metodo statistico che permette di ottenere informazioni sulla popolazione basandosi sulle statistiche di un sottoinsieme della popolazione (il campione), senza dover indagare su ogni individuo. Questo è particolarmente utile nei dataset che sono troppo grandi per essere analizzati efficacemente per intero, ad esempio nelle applicazioni di big data analytics o nei sondaggi.

Abbiamo deciso di utilizzare le seguenti tecniche di sampling:

- Campionamento casuale senza reinserimento: in questo metodo, i campioni vengono selezionati casualmente dal dataset originale senza che possano essere nuovamente selezionati durante l'estrazione.
- Campionamento casuale con reinserimento: questo metodo prevede l'estrazione casuale di record, e questi possono essere nuovamente selezionati. Con questa tecnica le estrazioni del campione sono indipendenti

perché l'esito di un'estrazione casuale non è influenzato dall'estrazione precedente.

- Campionamento stratificato fisso: questo metodo è utila quando il dataset è molto sbilanciato. Si esegue selezionando un numero fisso di record da ogni classe.
- Campionamento stratificato proporzionale: questa è una variante del campionamento stratificato in cui la dimensione del campione per ogni strato è proporzionale alla dimensione della classe. Questo garantisce che ogni gruppo all'interno della popolazione riceva la giusta rappresentanza all'interno del campione.

Tabella 2: Confronto tra classificatori con tecniche di campionamento

	Albero	KNN	SVM	Bayes	Ens
No Sampling	0.95	0.93	0.93	0.91	0.93
Random Without Replacement	0.93	0.92	0.92	0.92	0.96
Random With Replacement	0.95	0.93	0.95	0.90	0.94
Fixed Stratified	0.93	0.94	0.91	0.91	0.93
Proportional Stratified	0.94	0.92	0.95	0.91	0.95

6.2 Confronto con tecniche di undersampling

L'undersampling è un insieme di tecniche progettate per rimuovere record dal dataset che appartengono alla classe di maggioranza al fine di bilanciare meglio la distribuzione delle classi.

Abbiamo deciso di utilizzare le seguenti tecniche di undersampling:

- Random undersampling: questa tecnica seleziona casualmente record dalla classe di maggioranza e li elimina dal dataset di addestramento.
- Probabilistic undersampling: questa tecnica seleziona i record della classe maggioritaria che hanno più probabilità di appartenere a tale classe.
- Nearest to nearest: questa tecnica seleziona i record della classe maggioritaria più vicini ai più vicini record della classe minoritaria.
- Nearest to farthest: questa tecnica seleziona i record della classe maggioritaria più vicini ai più lontani record della classe minoritaria.
- Cluster centroid: questa tecnica sottocampiona la classe maggioritaria in N cluster, tanti quanti sono i record della classe minoritaria, attraverso l'algoritmo KMeans, mantenendo poi solo i centroidi di tali cluster.

Tabella 3: Confronto tra classificatori con tecniche di undersampling

	Albero	KNN	SVM	Bayes	Ens
No undersampling	0.95	0.93	0.93	0.91	0.93
Random undersampling	0.74	0.71	0.5	0.54	0.51
Probabilistic undersampling	0.95	0.92	0.82	0.88	0.87
Nearest to nearest	0.83	0.82	0.5	0.66	0.69
Nearest to farthest	0.89	0.67	0.5	0.56	0.54
Cluster centroid	0.78	0.67	0.5	0.70	0.82

6.3 Classificatori a confronto con diverse tecniche di oversampling

L'undersampling è un insieme di tecniche progettate per aggiungere record dal dataset che appartengono alla classe di minoranza al fine di bilanciare meglio la distribuzione delle classi.

Abbiamo deciso di utilizzare le seguenti tecniche di oversampling:

- Random oversampling: questa tecnica seleziona casualmente record dalla classe di minoranza e li duplica nel dataset di addestramento. Questa tecnica non è indicata da usare sul nostro dataset fortemente sbilanciato.
- Oversampling SMOTE (Synthetic Minority Oversampling Technique): tecnica di oversampling che genera esempi sintetici per la classe di minoranza, interpolando dei record vicini.
- Oversampling ADASYN (Adaptive Synthetic): tecnica di oversampling che genera esempi sintetici per la classe di minoranza, interpolando dei record vicini classificati in modo errato.

Tabella 4: Confronto tra classificatori con tecniche di oversampling

	Albero	KNN	SVM	Bayes	Ens
No Oversampling	0.85	0.68	0.93	0.91	0.96
Random Oversampling	0.94	0.93	0.99	0.55	0.97
Oversampling SMOTE	0.89	0.8	0.51	0.62	0.91
Oversampling ADASYN	0.87	0.87	0.65	0.57	0.92

6.4 Confronto con tecniche di riduzione della dimensionalità

La riduzione della dimensionalità e la selezione sono tecniche utilizzate per ridurre il numero di caratteristiche in un dataset, mantenendo il più possibile le informazioni importanti. Questo processo può essere fatto per ridurre la complessità

di un modello, migliorare le prestazioni di un algoritmo di apprendimento, o rendere più facile la visualizzazione dei dati.

Abbiamo deciso di utilizzare le seguenti tecniche di dimensionality reduction:

- Principal Components Analysis (PCA): questa tecnica costruisce nuovi attributi combinando linearmente quelli originali, in modo che catturino la massima variazione dei dati e che siano ortogonali tra di loro.
- Sparse Random Projection: questa tecnica riduce la dimensionalità proiettando gli attributi originali utilizzando una matrice casuale sparsa.
- Gaussian Random Projection: questa tecnica riduce la dimensionalità proiettando i dati in un sottospazio a dimensionalità inferiore utilizzando una matrice casuale gaussiana.
- Feature Agglomeration: questa tecnica produce dei nuovi attributi derivati dall'unione degli attributi originali, in precedenza raggruppati tramite un algoritmo di clustering gerarchico, e poi agglomerati.
- Variance Threshold: questa tecnica rimuove tutte le caratteristiche con una varianza inferiore a una certa soglia dal set di dati.
- Best chi2 score: questa tecnica seleziona le caratteristiche con i valori chi2 più alti. ATTENZIONE: per utilizzare questa tecnica è necessario che gli attributi non contengano dati negativi (si può per esempio attivare la standardizzazione Min-Max).
- Best mutual info score: questa tecnica seleziona le caratteristiche con i punteggi di informazione reciproca più alti.
- Backward Selection: questa tecnica inizia con l'intero set di caratteristiche e rimuove iterativamente la caratteristica che ha il minimo impatto sulle prestazioni del modello predittivo.
- Forward Selection: questa tecnica inizia con un set vuoto di caratteristiche e aggiunge iterativamente la caratteristica che ha il massimo impatto sulle prestazioni del modello predittivo.
- Correlation Selection: è una tecnica di riduzione della dimensionalità che rimuove le caratteristiche altamente correlate tra loro dal set di dati. Questo perché le caratteristiche correlate tendono a portare informazioni simili, quindi la loro rimozione può aiutare a ridurre la dimensionalità senza perdere molte informazioni.

6.5 Confronto con tecniche di trasformazione

Le tecniche di trasformazione sono utilizzate per modificare i dati in un formato che è più adatto per l'addestramento dei modelli. Queste tecniche sono spesso utilizzate per normalizzare o standardizzare i dati.

Abbiamo deciso di utilizzare le seguenti tecniche di trasformazione:

Tabella 5: Confronto tra classificatori con tecniche di riduzione della dimensionalità

	Tre	KNN	SVC	CNB	CEN
No Dimensionality Reduction	0.95	0.93	0.93	0.91	0.95
Principal Components Analysis	0.94	0.93	0.93	0.91	0.93
Sparse Random Projection	0.92	0.93	0.93	0.11	0.92
Gaussian Random Projection	0.90	0.93	0.93	0.92	0.93
Feature agglomeration	0.95	0.93	0.93	0.11	0.94
Variance threshold	0.95	0.93	0.93	0.90	0.95
Best Chi2 score	0.91	0.93	0.93	0.93	0.93
Best Mutual Info score	0.95	0.93	0.93	0.92	0.94
Backward selection	0.95	0.94	0.93	0.92	0.94
Forward selection	0.95	0.94	0.93	0.92	0.94
Correlation selection	0.94	0.93	0.93	0.64	0.95

- Standardizzazione Z-Score: questa tecnica trasforma i valori degli attributi in modo che abbiano una media di 0 e una deviazione standard di 1.
- Standardizzazione Min-Max: questa tecnica ridimensiona tutti i valori di un'entità in modo che si trovino nell'intervallo [0, 1]. Questo viene fatto sottraendo il valore minimo dell'entità da ogni valore, e poi dividendo per l'intervallo dell'entità.
- Normalizzazione L1: questa tecnica scala i dati in modo che la somma dei valori assoluti di ogni caratteristica sia uguale a 1.
- Normalizzazione L2: questa tecnica scala i dati in modo che la somma dei quadrati di ogni caratteristica sia uguale a 1.
- Normalizzazione L-max: questa tecnica misura la "dimensione" di una funzione, ed è definita come il valore assoluto massimo che la funzione assume su un determinato dominio.

Tabella 6: Transformation

	Tre	KNN	SVC	CNB	CEN
No Transformation	0.95	0.93	0.93	0.91	0.95
Z-Score Standardization	0.95	0.93	0.93	0.91	0.94
Min-Max Standardization	0.95	0.93	0.93	0.91	0.94
L1 Normalization	0.93	0.93	0.93	0.87	0.93
L2 Normalization	0.93	0.93	0.93	0.88	0.93
LMax Normalization	0.92	0.93	0.93	0.86	0.93

6.6 Confronto con tecniche gestione dei valori mancanti

La gestione dei valori mancanti è un aspetto fondamentale nella preparazione dei dati. I valori mancanti in un dataset possono essere causati da vari fattori, come errori nella raccolta dei dati, problemi di elaborazione dei dati o errori umani. La maggior parte degli algoritmi di machine learning richiede valori numerici e un valore presente per ogni riga e colonna in un dataset, pertanto i valori mancanti possono causare problemi per tali algoritmi.

Abbiamo deciso di utilizzare le seguenti tecniche di gestione dei valori mancanti:

- Media: questa tecnica sostituisce i valori mancanti con la media dei valori disponibili.
- Moda: questa tecnica sostituisce i valori mancanti con il valore più frequente nel dataset.
- Neighbors: Questa tecnica utilizza l'algoritmo K-Nearest-Neighbors per prevedere e assegnare i valori mancanti. Funziona trovando le osservazioni più simili a quella con il valore mancante e assegnando il valore sulla base di esse.

Tabella 7: Confronto tra classificatori con tecniche di gestione dei valori mancanti

	Albero	KNN	SVM	Bayes	Ens
Mean	0.95	0.93	0.93	0.91	0.96
Most frequent	0.93	0.92	0.93	0.90	0.94
Nearest	0.91	0.93	0.93	0.90	0.92

6.7 Confronto con le migliori tecniche di pre-processing

Abbiamo effettuato un confronto tra i vari classificatori osservando le loro metriche del campo Performance e le loro curve ROC applicando le tecniche di pre-processing ai dati ottimali per ogni classificatore.

Tra le opzioni di oversampling abbiamo inserito per completezza anche il metodo di Random Oversampling, che tuttavia non è indicato a causa dello sbilanciamento del nostro dataset. Infatti utilizzando questa tecnica si moltiplicano i record della classe minoritaria, che così avranno un'alta probabilità di essere estratti sia nel training set sia nel test set; in questo modo il classificatore, in fase di valutazione, cercherà di classificare dei record su cui si è addestrato, mostrando delle prestazioni falsate sul suo potere di generalizzazione. Per far notare questa differenza tra l'uso e il non uso del random oversampling, abbiamo incluso le prestazioni migliori sia con questa tecnica (Tabella 8, Figura 3) che senza (Tabella 9, Figura 4). Nella legenda delle curve ROC si trova per ognuna anche l'area al di sotto della curva.

Tabella 8: Confronto metriche classificatori con Random Oversampling

	Albero	KNN	SVM	Bayes	Ens
Acc.	0.96	0.97	0.99	0.71	0.92
TPR	0.99	0.98	1.00	0.43	0.85
TNR	0.94	0.95	0.99	0.99	0.98
FPR	0.05	0.04	0.00	0.00	0.01
FNR	0.00	0.01	0.00	0.56	0.14
Prec.	0.94	0.95	0.99	0.98	0.98
F-mes.	0.97	0.23	0.99	0.60	0.91

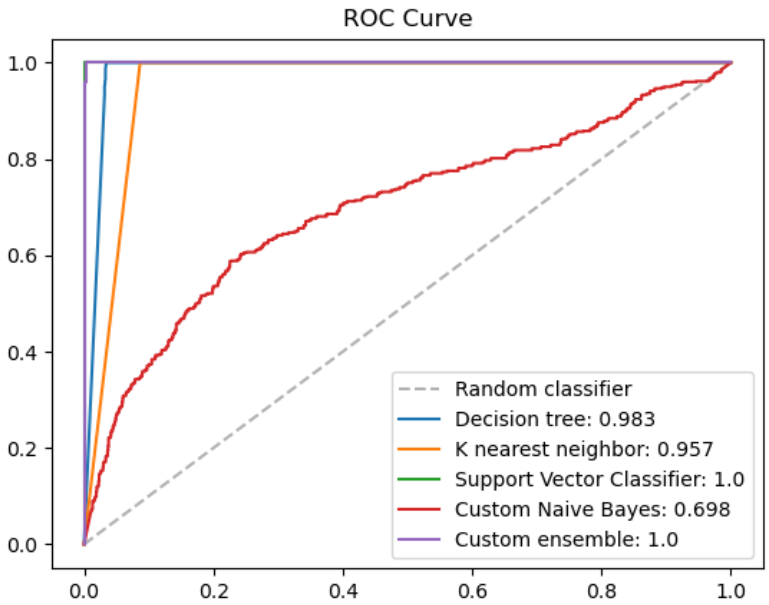

Figura 3: Confronto curve ROC classificatori con Random Oversampling

Per avere una stima più accurata del potere di generalizzazione, abbiamo provato diverse combinazioni di tecniche di undersampling e di trasformazione, ovvero quelle che più sono utili nel caso del dataset su cui abbiamo operato, in quanto campionamento e selezione non offrono un guadagno significativo di tempo, e l'oversampling non è consigliato visto lo sbilanciamento del nostro dataset.

Le migliori prestazioni le abbiamo ottenute trasformando i dati con la norma L2, e facendo un undersampling basato sulla distanza "nearest to nearest". Nell'albero abbiamo impostato l'impurezza misurata con l'entropia, per il K-NN abbiamo scelto la distanza di Manhattan senza pesare i vicini, per l'SVM abbiamo impostato una kernel function polinomiale, e nel classificatore multiplo abbiamo usato il boosting con soft voting.

Tabella 9: Confronto metriche classificatori con normalizzazione L2 e Nearest to nearest

	Albero	KNN	SVM	Bayes	Ens
Acc.	0.96	0.95	0.93	0.92	0.95
TPR	0.93	0.90	0.95	0.86	0.95
TNR	0.99	0.99	0.90	0.98	0.95
FPR	0.00	0.00	0.09	0.01	0.04
FNR	0.06	0.09	0.04	0.13	0.04
Prec.	0.98	0.98	0.91	0.97	0.96
F-mes.	0.96	0.94	0.93	0.91	0.95

7 Considerazioni finali sul progetto

Ogni classificatore ha mostrato punti di forza e aree di miglioramento, evidenziando l'importanza di esplorare diverse tecniche di modellazione, e l'importanza della pre-elaborazione dei dati. Modifiche come la riduzione della dimensionalità non fornivano molti vantaggi, se non quelli in termini di tempo di addestramento del modello, ma il nostro dataset non aveva abbastanza attributi da rendere percepibile tale vantaggio. Invece la trasformazione dei dati, in particolar modo la Z-Score e la Min-Max, ha avuto un impatto significativo sulle prestazioni del modello.

Indubbiamente però, la tecnica di pre-processing che più ci è servita è stata il bilanciamento, soprattutto a causa del nostro dataset, a priori molto sbilanciato. Intervenendo su questo campo i classificatori hanno un notevole miglioramento delle prestazioni.

Abbiamo osservato che alcuni classificatori hanno avuto prestazioni superiori in termini di accuratezza, precisione e recall. Tuttavia, nessun modello è stato il migliore in tutte le metriche, sottolineando l'importanza di considerare diverse metriche di valutazione nel contesto specifico del problema. Indubbiamente abbiamo notato come il classificatore Naive Bayes ha le sue migliori prestazioni

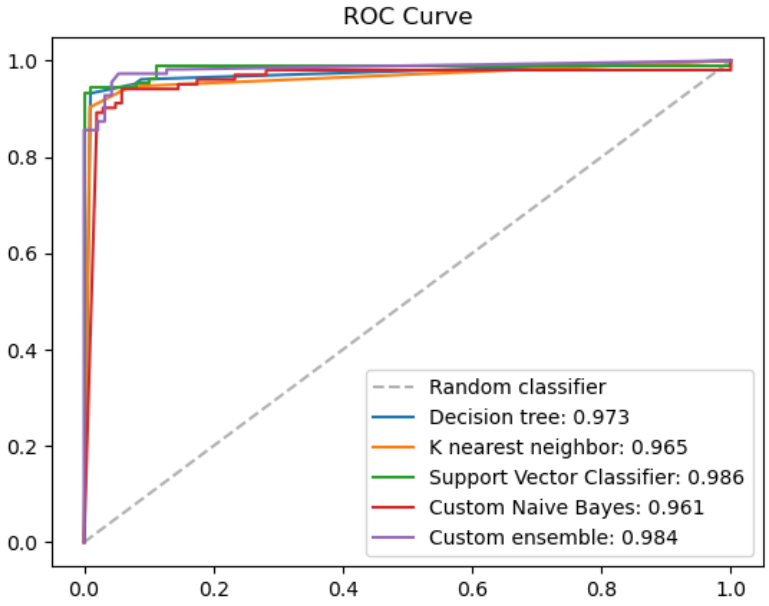

Figura 4: Confronto curve ROC classificatori con normalizzazione L2 e Nearest to nearest

molto distanti dagli altri, a causa dell'assunzione dell'algoritmo stesso riguardo all'indipendenza degli attributi, che è molto limitante.

Il tuning degli iperparametri ha giocato un ruolo cruciale nel migliorare le prestazioni dei nostri modelli. Questo processo, sebbene sia computazionalmente costoso, ha portato a ad alcuni miglioramenti nelle prestazioni del modello.

In conclusione, questo progetto ha dimostrato l'importanza di un approccio metodico e iterativo all'apprendimento automatico. Continueremo a esplorare nuove tecniche e approcci per migliorare ulteriormente le prestazioni dei nostri modelli.