Progetto Bosch: implementazione di un sistema di *image classification* per l'analisi automatica di qualità

Lorenzo Lorgna[1] **and Alessandro Maccario**[2]

[1] *829776, l.lorgna@campus.unimib.it*
[2] *865682, a.maccario1@campus.unimib.it*

Compiled July 27, 2022

La Quarta rivoluzione industriale, la *Industry 4.0*, ha avuto una evoluzione dirompente da oltre dieci anni grazie all'enorme quantitativo di risorse finanziarie che sono state iniettate dai diversi governi mondiali ai differenti settori produttivi.
Bosch, operante nell'ambito del settore manifatturiero, ha iniziato ad implementare soluzioni di *Industry 4.0* a partire dal 2011, ottenendo ricavi sostanziali grazie all'applicazione di tecnologie in grado di analizzare e fornire un flusso costante di informazioni ai decisori aziendali.
Sebbene il viaggio di trasformazione di Bosch sia già considerabile maturo, l'azienda ha costantemente bisogno di implementare nuove soluzioni tecnologiche in grado di ottimizzare i processi produttivi, riducendo il tempo che il prodotto necessita per passare dalla fase di ideazione e creazione a quello di produzione e di vendita. Per questo motivo, il progetto in esame vuole sopperire a questo bisogno di miglioramento nel processo finale di controllo qualità di [redacted], tramite un sistema automatizzato di Machine Learning basato sulla classificazione di immagini. Sono state utilizzate diverse tecniche di Machine Learning in grado di classificare in autonomia quelle difettose da quelle non difettose. Nello specifico, sono stati applicati diversi modelli supervisionati fra cui le Support Vector Machine, che presenta performance migliori rispetto a tutti gli altri modelli scelti, tanto in termini di recall che di AUC.
Gli stessi ottimi risultati, in vista di una minimizzazione dei falsi negativi, sono stati ottenuti anche sull'insieme dei dati di testing, fornendo una solida base di certezza nell'implementazione di tale modello in ambito produttivo, ottimizzando il processo e riducendo i tempi e i costi di produzione.

CONTENTS

1 Introduzione 2

2 Struttura del report 2

3 Bosch e l'Industry 4.0 2

4 Descrizione dei dati e analisi esplorativa 3

5 Data preparation 3
5.1 Cropping e ridimensionamento immagini 3
5.2 Divisione in train e test set 3
5.3 Data augmentation 4
5.4 Conversione in forma vettoriale e normalizzazione 4

6 Analisi del problema 4
6.1 Classificazione 4
6.2 Modelli implementati 4

7 Descrizione matematica dei modelli proposti 4
7.1 Dummy Classifier 4
7.2 Logistic Regression 5
7.3 Linear Discriminant Analysis 5
7.4 K-nearest Neighbors 5
7.5 Decision Tree 5
7.6 Random Forest 6
7.7 Naive Bayes 6
7.8 Support Vector Machine 6

8 Performance e valutazione dei modelli 7
8.1 Confusion matrix 7
8.2 Accuracy 7
8.3 Precision 7
8.4 Recall 7
8.5 F1 Score 8
8.6 ROC Curve e AUC score 8

9 Implementazione dei modelli proposti 8

10 Risultati ottenuti e loro interpretazione 8

11 Conclusioni e Sviluppi futuri 9

12 Requirements 9

1. INTRODUZIONE

Focus sull'elemento umano, *lean manifacturing* e connessione essere umano-macchina, sono gli elementi fondanti la cosiddetta *Industry 4.0*. Un modello di produzione che ha preso piede nel corso degli ultimi dieci anni e che ha accompagnato un utilizzo massivo di sensoristica all'interno del ciclo produttivo, permettendo un monitoraggio continuo, senza interruzioni e ottimizzato della catena di produzione. Ma quali sono le fondamentali caratteristiche di questa *quarta rivoluzione industriale*?

- trasparenza dell'informazione;
- interoperabilità dell'informazione;
- capacità di gestire un decision making decentralizzato - il flusso decisionale coinvolge tutta la catena di fornitura, avendo anche la possibilità di avere assistenza da remoto.

Industria 4.0 è il nome che è stato fornito alla nuova rivoluzione industriale, la quarta, dell'ultimo decennio, ovvero il cambio di paradigma chiamato l'*Internet delle cose* e che ha permesso un balzo in avanti sulla strada della digitalizzazione aziendale. Tale termine descrive la profonda e irreversibile trasformazione digitale del settore manifatturiero. La robotica, la sensoristica, l'interconnessione e programmazione sono le basi fondanti del sistema produttivo che caratterizza l'Industria 4.0, in cui i processi di manufacturing sono sempre più automatizzati e si fondano su una continua interazione tra macchinari, sistemi informativi e persone. Tutto ciò conduce al concetto di fabbrica *intelligente* la quale, attraverso l'implementazione di tecnologie digitali, costruisce sistemi interconnessi tra mondo fisico e digitale, che attraversa l'intera catena del valore, apportando innovazione e ottimizzazione della filiera produttiva.
Alla base, i dati: l'enorme ammontare di nuove informazioni prodotte, secondo dopo secondo, dai macchinari interconnessi alla rete aziendale e nuove conoscenze nel campo del trattamento dei Big Data, ha consentito la creazione e lo sviluppo incessante dell'Industria del futuro, già nel presente.

2. STRUTTURA DEL REPORT

Il seguente report si struttura in 11 sezioni come descritte di seguito.

La prima sezione, Introduzione, presenta una introduzione all'ambito dell'*Industry 4.0*, le sue caratteristiche e implicazioni economiche e produttive dal punto di vista aziendale.

La seconda sezione, Struttura del report, presenta la suddivisione in sezioni che costituiscono il presente lavoro.

La terza sezione, Bosch e l'Industry 4.0, presenta una breve introduzione al mondo Bosch nell'ambito dell'industria 4.0.

La quarta sezione, Descrizione dei dati e analisi esplorativa, prevede una descrizione del dataset e un'analisi esplorativa dei dati per comprendere il problema della classificazione di immagini nell'ambito delle .

La quinta sezione, Data preparation, consiste nella spiegazione del processo utilizzato per la preparazione delle immagini in vista dell'applicazione dei modelli nelle fasi successive.

Nella sesta sezione, Analisi del problema, viene analizzato il task affrontato mettendo in luce le sfide presentate dalla *image classification*.

Nella settima sezione, Descrizione matematica dei modelli proposti, viene proposta una breve descrizione teorica e matematica dei modelli utilizzati.

Nella ottava sezione, Performance e valutazione dei modelli, la penultima, si valutano gli indicatori di performance ottenuti dai modelli, quali fra questi è il migliore, e quale il suo utilizzo in vista del *deploy* finale.

Nella nona sezione, Implementazione dei modelli proposti, si spiega come sono stati implementati i modelli dal punto di vista del codice in Python.

Nella decima sezione, Risultati ottenuti e loro interpretazione, si valutano gli output ricavati dall'applicazione del miglior modello e si interpretano nell'ambito del task di classificazione di immagini per Bosch.

La undicesima e ultima sezione infine, Conclusioni e sviluppi futuri, definisce le conclusioni ottenute e i possibili sviluppi futuri del progetto in termini di miglioramento delle performance ottenute e dell'applicazione di tale lavoro in ambito produttivo.

3. BOSCH E L'INDUSTRY 4.0

> *We must turn data into knowledge, and knowledge into benefits.*

Così commentò nel 2013 l'allora CEO di Bosch Volkmar Denner durante la ZVEI Annual Convention, a Berlino. L'importanza dell'industria 4.0 è stata chiara fin da subito a Bosch. Infatti, la storia di Bosch nell'implementazione di quelle tecnologie adatte alla trasformazione dell'industria manifatturiera comincia già a partire dal 2011, quando il Governo tedesco diede l'appoggio finanziario a questo progetto pionieristico.
Dopo un primo iniziale momento di implementazione di una *roadmap* per affrontare un percorso di modifica dell'attuale modello di analisi interna della produzione, Bosch ha affrontato questa sfida impiegando tutti i mezzi a sua disposizione arrivando, nel 2014, a presentare il primo robot collaborativo con un operatore umano, denominato *APAS* - automatic production assistant.
Nel 2021, grazie ai primi passi svolti in questa direzione innovativa, Bosch genera quattro miliardi di euro in termini di vendite proprio grazie all'industria 4.0.
Un paradigma non solamente incentrato sul lato tecnologico della produzione ma che abbraccia l'ambito economico, del risparmio energetico e dell'ottimizzazione manifatturiera nel suo complesso.
Quindi, un nuovo tipo di industria in grado di automatizzare la catena di fornitura in modo tale da migliorarla sotto gli aspetti di *continuità*, *ottimizzazione*, e *monitoraggio completo*, tramite la sensoristica implementata all'interno della catena produttiva.

4. DESCRIZIONE DEI DATI E ANALISI ESPLORATIVA

Per avere una maggiore comprensione del dato si è proceduto inizialmente con un'analisi esplorativa. Il dataset fornito da Bosch è composto da 104 immagini rappresentanti non difettose e 19 immagini rappresentanti difettose. Le immagini in questione vengono fornite in due differenti cartelle, *OK* e *NOK*, rispettivamente per le componenti difettose e non difettose. Di seguito, in figura 1 e figura 2, si mostrano un esempio di immagini per ciascuna categoria.

Fig. 1.

Fig. 2.

Come si può già intuire dalla dimensionalità del dataset, si è in presenza di un problema di classi sbilanciate, ovvero il numero di esempi nel set di dati per ciascuna categoria non è bilanciato e quindi la distribuzione non risulta essere uniforme. In tali situazioni il processo di apprendimento dei modelli può risultare distorto, dal momento in cui il modello tende a focalizzarsi sulla classe prevalente e ignorare gli eventi rari. Nel caso preso in esame, come è possibile osservare in figura 3, dove la distinzione in forma numerica prevede la classe 0 per le non difettose, classe più comune, definite come *OK* e 1 per la classe rara, ovvero le , definite come *NOK*, si nota un evidente sbilanciamento.

Le immagini a disposizione hanno singolarmente un peso pari a circa 57.1 MB e hanno una dimensione pari a 5472 x 3648 pixel. Inoltre si presentano in formato bmp, ovvero bitmap. Si tratta di un formato dati utilizzato per la rappresentazione di immagini raster sui sistemi operativi Microsoft Windows.

Le immagini risultano essere memorizzate in scala di grigi, motivo per cui i 3 canali immagine costruiti di default dalla libreria *Opencv* in fase di lettura risultano essere identici, e i valori relativi ai pixel facenti parte di esse hanno valori che variano da 0 a 255.

5. DATA PREPARATION

Conclusa la prima fase di esplorazione del dataset si è proseguito con la fase di data preparation che consiste in una serie di operazioni di preprocessing necessarie per avere i dati iniziali nel formato corretto richiesto dalla implementazione dei modelli successivi.

5.1. Cropping e ridimensionamento immagini

Inizialmente, considerando solo uno dei tre canali colori essendo tutti identici, si è deciso di procedere con un'operazione di cropping che ha permesso di ritagliare l'immagine, in modo tale che la componente avesse un maggiore spazio dedicato, come è possibile osservare in figura 4. L'operazione di cropping è stata effettuata ipotizzando che le immagini fornite dal cliente abbiano sempre la stessa posizione. Successivamente, a causa

Fig. 4. Operazione di cropping

delle limitate risorse computazionali, è stato necessario ridimensionare l'immagine di un fattore percentuale pari a 0.6 e come metodo di interpolazione si è optato per *INTER_AREA*, un resampling che si basa sulla relazione delle aree dei pixel. Una volta eseguite tali operazioni, le immagini sono state salvate riscontrando, come previsto, una riduzione della dimensione (132 x 132 pixel) e al tempo stesso anche una riduzione notevole del peso, passando da circa 57.1 MB a 20 KB per singola immagine.

5.2. Divisione in train e test set

Una volta processate le immagini si è proceduto con la suddivisione in train e test set, suddivisione dei dati che risulta utile nelle fasi successive per misurare le performance dei modelli che saranno implementati. Si è deciso di scegliere come proporzioni rispetto alla totalità dei dati per il train e test set rispettivamente i valori 0.65 e 0.35. La funzione utilizzata per eseguire tale suddivisione è *split-folders* che applica una divisione di tipo stratificato.

5.3. Data augmentation

Eseguita la suddivisione del dataset, osservando il training set e tenendo a mente le osservazioni fatte in fase di esplorazione relativamente al problema di class imbalance, è stato necessario procedere nel correggere tale condizione. Per far fronte allo sbilanciamento in termini di numerosità delle due classi, difettose e non difettose, si è deciso di approcciare tale problema tramite un metodo di data augmentation. In particolar modo tale metodo consiste in un insieme di tecniche di manipolazione e trasformazione dei dati che hanno lo scopo di ampliare la dimensione del dataset di partenza, in particolar modo del training set. Nel caso preso in esame , si è deciso di optare per semplici trasformazioni delle immagini di partenza, quali, per esempio, flip orizzontale e verticale e rotazioni di un particolare valore di angolo, come si può osservare in figura 5. Tali operazioni hanno garantito di preservare le caratteristiche principali delle immagini. In un primo momento sono state testate anche trasformazioni più invasive legate alla luminosità dell'immagine ad esempio: tuttavia il rischio principale riscontrato è stato quello di perdita di informazioni dovute allo sbiancamento eccessivo dell'immagine. La seguente operazione di data augmentation ha permesso dunque di aumentare la dimensione del train set, da una situazione iniziale di 67 immagini per la categoria *OK* e 12 per la categoria *NOK* ad uno scenario in cui il numero di immagini per categorie *OK* e *NOK* sono rispettivamente 134 e 132 nel training set.

Fig. 5. Operazione di data augmentation

5.4. Conversione in forma vettoriale e normalizzazione

Come operazioni conclusive di data preparation è stato necessario convertire le immagini processate in forma vettoriale e applicare, in un secondo momento sulle strutture dati ottenute, l'operazione di normalizzazione affinché i modelli che verranno implementati nelle sezioni successive abbiano i dati di input nel formato corretto.

6. ANALISI DEL PROBLEMA

Il problema in esame si inserisce nel contesto del *Image Classification problem*, ovvero della capacità di categorizzare, tramite algoritmi di Machine Learning, diverse immagini divise per categoria, sulla base delle immagini fornite al modello. Questo particolare tipo di task ha una importanza fondamentale per Bosch in quanto in grado di ridurre i tempi di controllo qualità da parte dell'operatore, inserendosi come valido aiuto all'essere umano nei processi decisionali.

6.1. Classificazione

Un problema di classificazione è il processo che prevede di predire una etichetta di classe, ovvero una variabile di tipo categoriale discreta o non ordinata. Il dataset di training e i valori di classe (le etichette di classe) sono utilizzate per addestrare un modello di classificazione in grado di classificare la variabile target a disposizione come anche nuovi dati in input. Il task di classificazione prevede due passi:

- Il primo passo del processo è il cosiddetto *learning step*, rappresentato dalla costruzione del modello stesso;
- Il secondo passo è definibile come *classification step* nel quale il modello precedentemente costruito viene utilizzato nell'opeazione di classificazione vera e propria.

6.2. Modelli implementati

Con l'obiettivo di classificare tra difettose e non difettose, sono stati applicati metodi di *Supervised machine learning* alle immagini a disposizione dopo averle adeguatamente preprocessate. I metodi applicati sono i seguenti:

1. Dummy Classifier Model (DC)
2. Logistic Regression Model (LR)
3. Linear Discriminant Analysis (LDA)
4. K-nearest neighbors algorithm (KNN)
5. Decision Tree Model (DT)
6. Random Forest Classifier (RF)
7. Naive Bayes (NB)
8. Support Vector Machine (SVM)

Per ogni modello è stato implementato l'algoritmo tramite la funzione di *GridSearchCV*, con l'obiettivo di applicare una ricerca esaustiva sui parametri specificati per la ricerca di quelli migliori. Per la valutazione e predizione dei risultati, sono state utilizzate diverse metriche valutative, quali, per esempio, la *confusion matrix*, l'*accuracy*, la *precision*, la *recall* e la *F1-measure*. Nello specifico, la metrica *recall* si è rivelata molto importante in questo task di classificazione in quanto l'obiettivo finale è stato quello di minimizzare i *falsi negativi*. Ciò significa minimizzare il numero di osservazioni classificate come non difettose quando, in realtà, lo sono.

7. DESCRIZIONE MATEMATICA DEI MODELLI PROPOSTI

In questa sezione vengono brevemente presentati matematicamente i modelli utilizzati.

7.1. Dummy Classifier

Un **Dummy Classifier** effettua le previsioni ignorando le variabili di input. Ciò implica che ogni osservazione viene etichettata, di norma, con la classe più frequente di apparizione, senza l'utilizzo di altre informazioni.
Questo classificatore serve come modello di baseline, utile nella comparazione con altri classificatori più complessi utilizzati successivamente.

7.2. Logistic Regression

La *logistic function*, anche chiamata *sigmoide*, la quale accetta come input qualsiasi numero reale t, e ha come output un valore compreso tra zero ed uno, è un modello statistico tipicamente utilizzato per modellare una variabile dipendente di tipo binario, ovvero per classificazioni di tipo binario.
Formalmente, la funzione logistica viene così rappresentata:

$$\sigma(t) = \frac{1}{1+e^{-}t} = \frac{e^t}{e^t+1}$$

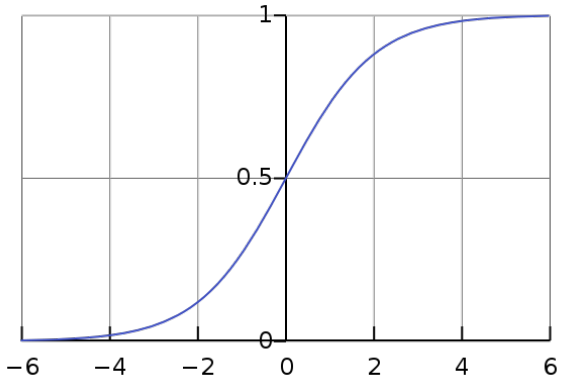


Fig. 6. La funzione logistica standard $\sigma(t)$; da notare che $\sigma(t) \in (0,1) \forall\ t$.

Assumendo che t sia una funzione lineare di una singola variabile indipendente x, è possibile esprimere t come segue:

$$t = \beta_0 + \beta_1(x)$$

Mentre la funzione logistica $p : \mathbb{R} \implies (0, 1)$ può essere espressa come:

$$p(x) = \sigma(t) = \frac{1}{1+e^{-(\beta_0+\beta_1(x))}}$$

Nel modello di regressione logistica, $p(x)$ è la probabilità della variabile dipendente Y considerata come *successo* piuttosto che come *fallimento*.

7.3. Linear Discriminant Analysis

La *Linear Disciminant Analysis* sebbene considerato un metodo più semplice di altri utilizzati anche in questo report, presenta diverse capacità quali, per esempio, quella di classificazione ma anche di riduzione di dimensionalità.

Tale metodologia può essere derivata come algoritmo di classificazione considerando il seguente esempio. Si consideri un generico problema di classificazione: una variabile randomica X è appartenente ad una classe k all'interno di K classi, con densità $f_k x$ in $\mathbb{R}^p$. Una regola predefinita cerca di dividere lo spazio dei dati in K regioni distinte $\mathbb{R}_1, \ldots, \mathbb{R}_k$ che rappresentano tutte le classi. Tramite queste regioni, si alloca x alla classe j se x si trova nella regione j. Ma come sapere in quale regione x viene a trovarsi? Si possono utilizzare le due seguenti regole di allocazione:

- *Maximum likelihood rule*: se si assume che ciascuna classe può presentarsi con uguale probabilità, allora classificare x alla classe j avviene se:

$$j = argmax_i f_i(x)$$

- *Bayesian rule*: se si conoscesse la probabilità a priori, π, allora assegnare x alla classe j se:

$$j = argmax_i \pi_i f_i(x)$$

7.4. K-nearest Neighbors

L'algoritmo KNN è un algoritmo di apprendimento supervisionato. Di norma, nella maggior parte degli algoritmi supervisionati, si allena il modello utilizzando i dati di training con l'intento di creare un modello che generalizzi bene su dati non ancora a disposizione.
Il KNN è anche un algoritmo non parametrico, il che implica che non assume l'esistenza di una distribuzione latente all'interno dei dati.
Il suo funzionamento è alquanto semplice: quando una osservazione è fornita all'algoritmo, con un certo valore di K, l'algoritmo cerca per il vicinato più simile al punto dato. I vicini sono individuati calcolando la distanza tra i punto dato e i punti del dataset iniziale. Possono essere utilizzate diverse funzioni di distanza oltre alla distanza Euclidea, come la distanza Manhattan o la cosiddetta *Cosine Similarity*.
Una volta che i k vicini sono stati individuati, l'algoritmo assegna l'osservazione alla classe di valori più vicina in base al calcolo della funzione di distanza.

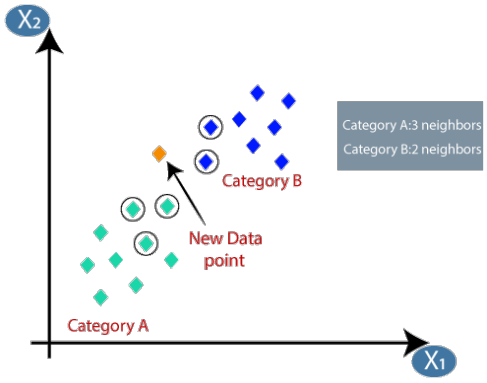


Fig. 7. Esempio dell'algoritmo KNN.

7.5. Decision Tree

Gli alberi di decisione modellano un classificatore binario con l'obiettivo di predire la variabile target usando una o più variabili indipendenti. Questa modalità di classificazione può essere rappresentata tramite un diagramma ad albero che, correttamente, indica se l'input di esempio è uno 0 o un 1, attraverso decisioni sequenziali basate sulle variabili a disposizione, come si può vedere dalla seguente figura.

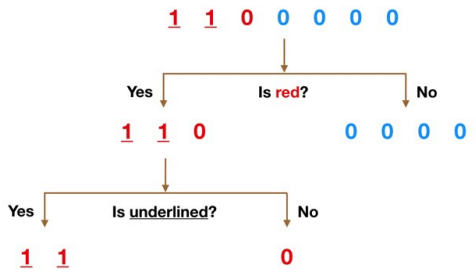


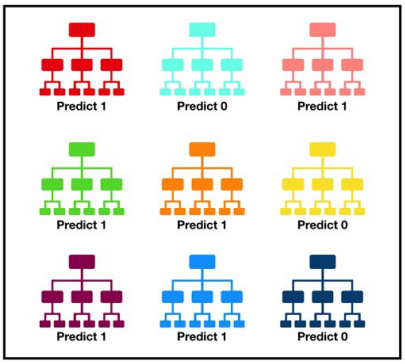
Fig. 8. Un semplice Decision Tree in grado di fare distinzione tra osservazioni appartenenti alla classe definita come 0 e osservazioni appartenenti alla classe 1.

In particolare, un albero di decisione è composto da nodi, contenenti parte dei dati di training e da archi che connettono i nodi tra di loro.
Nella fase di training, il modello considera i dati di training come input e fornisce un albero come output, costruito usando uno specifico algoritmo - come, per esempio, l'ID3 -, il quale è capace di classificare le osservazioni appartenenti ai dati di test in una delle due classi target.
Una volta generato l'albero di decisione, può essere letto nel modo seguente: ciascun nodo - eccetto le foglie contenenti una condizione logica, ad esempio un test riguardante una variabile specifica - e ciascun arco coincide con uno dei possibili output logici derivanti dalla valutazione.

Due delle misure più importanti usate per decidere il tipo di divisione dei nodi sono l'*entropia* e l'*information gain* basato sull'indice di Gini.

7.6. Random Forest

Il classificatore Random Forest consiste in un elevato numero di singoli alberi di decisione che operano in collaborazione. L'idea è quella di utilizzare diversi algoritmi di apprendimento per ottenere una performance predittiva migliore di quella ottenibile utilizzando un singolo albero di decisione.
Considerando il problema di classificazione binaria come presente in questo report, il Random Forest opera nella seguente maniera: data una osservazione che deve essere classificata, ciascun albero appartenente all'insieme predice la classe alla quale l'osservazione appartiene, indipendentemente dagli altri alberi di decisione; dopodiché, la classe con il numero più elevato di voti è quella definita come predizione dal Random Forest. La giustificazione teorica di tale modello è che gli alberi di decisione sono molto sensibili ai dati su quali l'algoritmo viene allenato, facendo sì che piccoli cambiamenti nei dati portino a forti cambiamenti nella struttura degli alberi di decisione.



Inoltre, tale algoritmo permette a ciascun singolo albero decisionale di selezionare randomicamente con rimpiazzamento dal dataset di partenza, ottenendo differenti alberi decisionali.

7.7. Naive Bayes

L'algoritmo Naive Bayes è definito come un classificatore probabilistico che conta la frequenza di apparizione dei valori in un insieme di dati, e ne calcola la relativa probabilità.
L'algoritmo si basa sul Teorema di Bayes e assume che tutti gli attributi siano indipendenti gli uni dagli altri.
Il teorema di Bayes fornisce un modo per calcolare la probabilità a posteriori $P(c|x)$ a partire da $P(c)$, $P(x)$, e $P(x|c)$:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Dove:

- P(c | x): rappresenta la probabilità di c di essere vero, dato che x è vero.
- P(c): anche detta probabilità a priori, è la probabilità di un evento prima che i dati vengano ottenuti.
- P(x | c): questa quantità rappresenta la probabilità di x di essere vero, dato che c è vero.
- P(x): questa è la probabilità osservata rispetto alla variabile predittore tra tutte le osservazioni.

7.8. Support Vector Machine

Nel caso di classificazione binaria, la Support Vector Machine (SVM) è un algoritmo che classifica le osservazioni in due possibili classi basandosi su una tecnica di separazione, cioè una ripartizione dello spazio degli attributi.
Questo avviene considerando i dati di training in $\mathbb{R}^n$, nello specifico ciascun punto dato è un vettore e le sue componenti sono gli n attributi numerici; una volta che i dati sono proiettati in $\mathbb{R}^n$, l'algoritmo ha il compito di trovare un iperpiano se la SVM è di tipo lineare o, altrimenti, una funzione più complessa, che divide i punti dato in diverse classi.
Dopodiché, ciascun dato presente nell'insieme dei dati di test, sarà classificato facendo affidamento su dove il singolo dato si trova nello spazio delle osservazioni.
Considerando il caso lineare, ci sono molte possibilità con le quali si può scegliere la divisione dei punti tramite iperpiani.

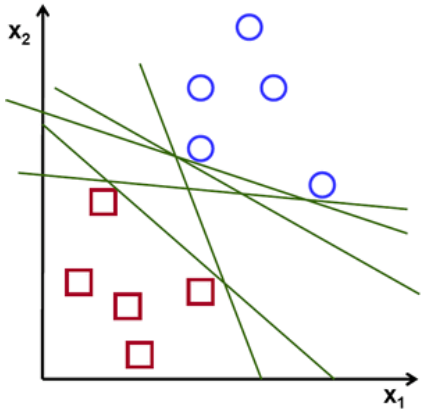


Fig. 9. Ci sono molte possibili rette di separazione (iperpiani in $\mathbb{R}^2$) che la SVM può utilizzare per separare le osservazioni appartenenti alla classe 0 (punti blu) da quelli appartenenti alla classe 1 (punti quadrati rossi).

Si osserva che ci si trova $\mathbb{R}^n$, e si considera la distanza Euclidea. L'algoritmo mira a trovare un iperpiano ottimo, ovvero l'iperpiano che massimizza il margine, il quale rappresenta la distanza tra la più vicina coppia di punti appartenenti a classi differenti. In particolare, solo i punti più vicini all'iperpiano lo influenzano, senza considerare i restanti. Tali valori sono chiamati *support vectors*, come il punto blu e i due quadrati rossi che si trovano sulle linee tratteggiate in verde, come si vede in Figura 10.

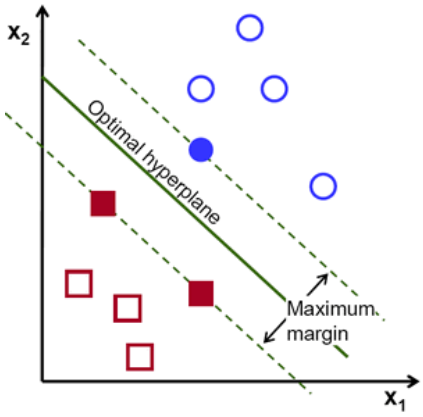


Fig. 10. La retta che l'algoritmo sceglie è la retta che massimizza il margine, ovvero la distanza tra i *support vectors*.

8. PERFORMANCE E VALUTAZIONE DEI MODELLI

Per la validazione dei modelli sono state considerate differenti metriche che vengono presentate di seguito. In particolar modo, considerando l'obiettivo del progetto, maggiore attenzione è stata posta sulla metrica *recall* dal momento in cui risulta più importante ridurre al minimo il numero di falsi negativi. La presenza infatti di falsi positivi non risulta essere un problema quanto lo sono invece i falsi negativi, dal momento in cui si può prevedere l'azione dell'operatore umano nel controllo di tali situazioni limite. Per la validazione dei modelli vengono considerate le metriche di seguito presentate.

8.1. Confusion matrix

E' una matrice di dimensione $2x2$ nel caso di problemi di classificazione binaria con i valori veri sull'asse delle ordinate e i valori predetti sull'asse delle ascisse. Il suo obiettivo è quello di descrivere le performance del modello.

		INDUCER PREDICTION (IP)	
		-1	+1
ACTUAL CLASS (AC)	-1	TN	FP
	+1	FN	TP

Dove:

- TN = True Negatives;
- TP = True Positives;
- FP = False Positives;
- FN = False Negatives.

8.2. Accuracy

E' il rapporto fra il numero di predizioni corrette rispetto al numero totate di campioni di input. E' formalizzata come:

$$Accuracy = \frac{Numero\ di\ Predizioni\ corrette}{Numero\ totale\ di\ predizioni}$$

Per un problema di classificazione binaria, l'accuracy può essere anche calcolata come segue:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

8.3. Precision

La *precision* corrisponde al numero corretto di risultati positivi rapportati al numero di risultati positivi previsti dal classificatore. Tale metrica risponde alla domanda: quale proporzione di valori identificati come positivi sono effettivamente corretti? Essa viene definita come segue:

$$Precision = \frac{TP}{TP + FP}$$

8.4. Recall

La *Recall* rappresenta il numero di risultati corretti definiti come positivi, rapportati al numero di campioni rilevanti. Tale misura cerca di rispondere alla domanda: quale proporzione di veri positivi è identificata correttamente? Matematicamente, la *Recall* è definita come:

$$Recall = \frac{TP}{TP + FN}$$

8.5. F1 Score

Tale misura viene definita come la media armonica tra la *precision* e la *recall*, il cui range di valori è compreso fra $[0, 1]$. Definisce quanto il classificatore è preciso e robusto allo stesso tempo. Matematicamente, viene definito come segue:

$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}}$$

8.6. ROC Curve e AUC score

Una curva ROC - Receiver Operating Characteristic curve - è un tipo di grafico che mostra le performance di diversi modelli di classificazione a differenti thresholds. Nello specifico, la curva ROC mostra due parametri:

- True Positive Rate:
$$\frac{TP}{TP + FN}$$
- False Positive Rate
$$\frac{FP}{FP + TN}$$

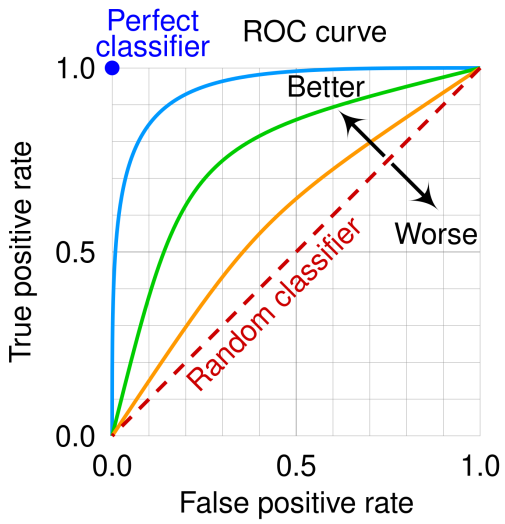


Fig. 11. La curva ROC per il *migliore* e il *peggiore* classificatore.

In maniera complementare, l'AUC - Area Under Curve - misura l'abilità del classificatore di distinguere tra classi ed è utilizzato nella pratica come valore di sintesi della ROC curve. Maggiore il suo valore, migliori le performance del modello nel distinguere tra classe positiva e negativa.

9. IMPLEMENTAZIONE DEI MODELLI PROPOSTI

Per ciascun modello, a partire dal *Dummy Classifier* fino alla *Support Vector Machine*, sono stati instanziati i modelli di base per poi inserirli all'interno di una ricerca esaustiva dei parametri migliori tramite una cross validazione. I modelli proposti infatti sono stati implementati attraverso una GridSearchCV, tramite la libreria Sklearn.
Dopo aver scelto, per ciascuna di essi, i parametri da poter ottimizzare, la cross validazione ha permesso di fare il *fitting* su un numero di modelli pari al numero di combinazioni fra tutti i parametri presenti.
Per tutti i modelli sono stati inoltre calcolati i tempi di esecuzione per avere una chiara indicazione del modello migliore anche in termini di tempistiche.

10. RISULTATI OTTENUTI E LORO INTERPRETAZIONE

Una volta implementati e addestrati i modelli sono dunque stati valutati osservando i risultati forniti tramite *cv_results* di *Sklearn*. In particolare, gli *scoring* presi in considerazione nella *cross-validation* sono stati quelli di *accuracy*, *precision*, *f1*, *recall* e *roc_auc*. Nello specifico, i modelli hanno subito un refitting sulla base della metrica *recall*, di fondamentale importanza nel caso in esame.

Considerando dunque le metriche di cross-validation, come è possibile osservare in tabella 1, è stato individuato il modello SVM come modello dalle migliori performance valutate nel loro complesso.

Classifier	Recall	Precision	F1	Accuracy	AUC
DC	0.00	0.00	0.00	0.00	0.50
LR	1.00	0.72	0.68	0.70	0.77
LDA	0.60	0.72	0.65	0.68	0.66
KNN	0.67	0.73	0.69	0.71	0.80
DT	0.82	0.70	0.72	0.70	0.73
RF	0.69	0.70	0.68	0.69	0.78
NB	0.50	0.70	0.58	0.63	0.67
SVM	0.94	0.69	0.76	0.73	0.78

Table 1. Metriche di cross-validation

Successivamente, si è proceduto con la valutazione delle performance sul test set dei vari modelli, per una maggiore completezza, pur avendo identificato precedentemente come miglior quello della SVM. E' possibile visualizzare i risultati ottenuti tramite le matrici di confusioni riportate in figura 12.

Per il modello SVM, in particolare, si è ottenuto un valore di *Recall* pari a 0.86 e di *Precision*, *F1*, *Accuracy* e *AUC* rispettivamente di 0.50, 0.63, 0.84 e 0.85.

Come si può notare dalla matrice di confusione della SVM, il modello migliore, 31 [redacted] vengono previste correttamente come non difettose e 6 invece correttamente come difettose. Inoltre si nota la presenza di un solo falso negativo e di 6 falsi positivi. Sebbene il numero di questi ultimi sia più elevato rispetto alle altre matrici di confusione, ciò non è un indicatore problematico dal momento in cui si potrebbe ipotizzare il controllo manuale da parte dell'operatore su queste [redacted] predette erroneamente come difettose.

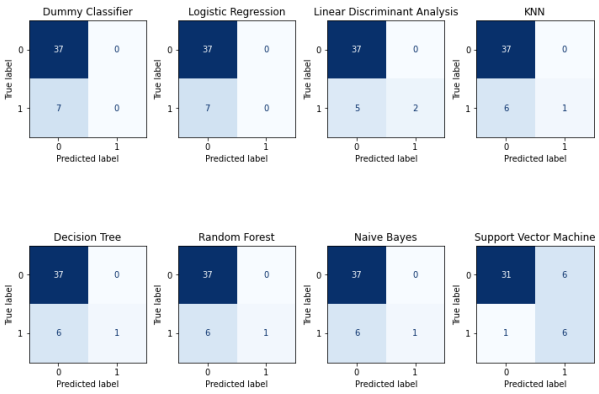


Fig. 12. Matrici di confusione per i modelli proposti.

Allo stesso modo, in termini di valore dell'AUC, come mostrato in Figura 13, la SVM presenta il risultato più elevato ($AUC = 0.85$), fornendo una conferma della sua capacità predittiva rispetto al modello di baseline casuale *Dummy*.

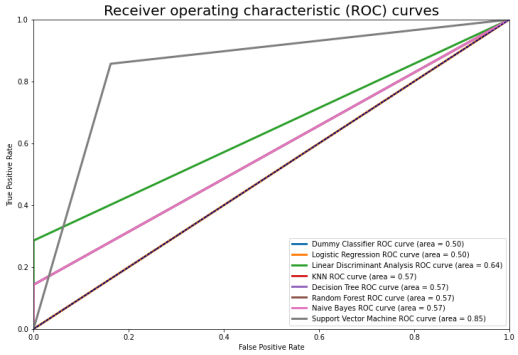


Fig. 13. ROC Curve per i modelli proposti e visualizzazione del modello migliore.

11. CONCLUSIONI E SVILUPPI FUTURI

Nel presente elaborato sono stati utilizzati otto modelli differenti appartenenti alla famiglia Machine Learning con l'obiettivo di risolvere il problema di classificare correttamente difettose da quelle non difettose, a partire dalle loro immagini.
Valutando le performance finali dei modelli, il miglior modello ai fini della classificazione di immagini per Bosch è la Support Vector Machine in quanto fornisce il minor numero di *falsi negativi*, ovvero di immagini classificate come non difettose quando, in realtà, risultano esserlo. Ciò deriva dal fatto che la presenza di falsi negativi potrebbe portare a delle perdite da parte dell'azienda, anche da un punto di vista economico.
Tuttavia, sono presenti dei *falsi positivi*, ma è bene ricordare che l'obiettivo nell'implementazione di un sistema di classificazione ha le sue radici nella capacità di supporto all'operatore umano e non di sostituzione.

Come parte conclusiva del progetto, nella fase di *deploy* ai fini di simulare l'ambito applicativo aziendale, il modello è stato applicato ad un insieme di dati di *inference*, ovvero dati non etichettati. Questi dati sono stati ottenuti facendo una selezione casuale dei dati originali.

In termini di sviluppi futuri con un **numero più elevato di immagini**, con una **più ampia expertise in ambito manifatturiero dell'esperto di Machine Learning** e con **maggiori risorse computazionali**, si è certi che la capacità di **recall** dell'algoritmo possa essere migliorata notevolmente, riducendo la presenza di *falsi negativi*.
Non da ultimo, per il miglioramento delle prestazioni, sarebbe importante implementare modelli basati sulle tecniche di *feature extraction* e, ipotizzando di avere un numero maggiore di immagini di allenamento e maggiori risorse computazionali, sarebbe interessante sviluppare un modello di deep learning, nello specifico una Convolutional neural network (CNN), in quanto rappresenta lo stato dell'arte in termini di Image Classification nell'ambito della Computer Vision. Maggiori risorse computazionali permetterebbero in fase di data preparation di eseguire la sola operazione di cropping senza dover ricorrere ad un ridimensionamento delle immagini, mantenendo dunque una maggiore qualità dei dati.

12. REQUIREMENTS

Le principali librerie utilizzate con le relative versioni sono le seguenti:

- python == 3.8.0
- splitfolders == 0.5.1
- cv2 == 4.5.5
- numpy == 1.22.2
- pandas == 1.4.1
- matplotlib == 3.5.1
- plotly == 5.8.0
- scikit-image == 0.19
- scikit-learn == 1.0.2
- tensorflow == 2.3.0
- pickleshare == 0.7.5
- cloudpickle == 2.0.0

Per le restanti librerie, consultare il file *Requirements.txt* presente nella repository del progetto.

REFERENCES

1. Flavia Profili *La Trasparenza Organizzativa: il Paradosso della Trasparenza nell'Era Digitale*, http://tesi.luiss.it/27782/1/219361_PROFILI_FLAVIA.pdf
2. Yang Xiaozhou. *Linear discriminant analysis, explained*, https://yangxiaozhou.github.io/data/2019/10/02/linear-discriminant-analysis.html , 2019.
3. Bosch Industry Consulting. *From lean to the digital factory - a vision becomes reality*, https://www.bosch-connected-industry.com/de/media/en/whitepaper_1/wp_industry40/whitepaper-i40-studie-de.pdf , 2020.
4. BoschGlobal. *Ten years of Industry 4.0.*, https://www.bosch.com/stories/10-years-industry-4-0-at-bosch/

5. Vegi Shanmukh. *Image Classification Using Machine Learning-Support Vector Machine(SVM)*, https://medium.com/analytics-vidhya/image-classification-using-machine-learning-support-vector-machine-svm-dc7a0ec92e01, 2021
6. Sunil Ray. *6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R*, https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/, 2017
7. Aniruddha Bhandari. *AUC-ROC Curve in Machine Learning Clearly Explained*, https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning. , 2020
8. From Wikipedia. *Convolution*, https://en.wikipedia.org/wiki/Convolution