Productivity Prediction of Garment Employees

Report per l'Esame di Fondameti di Machine Learning

Olivia Riccomi

135786
Ingegneria Informatica
254439@studenti.unimore.it

Abstract

L'industria di abbigliamento è diventata una delle realtà commerciali più sviluppate in età moderna, portando ad un aumento significativo di domanda e offerta. Per soddisfare la richiesta, sono cresciuti enormemente l'intensità dei ritmi di lavoro e il numero di processi manuali necessari per la produzione.
L'obiettivo di questo progetto sarà valutare, analizzare e prevedere dati inerenti la produttività di un'azienda di abbigliamento attravero l'utilizzo di algoritmi di regressione.

1 Introduzione

1.1 Problem statement

L'industria dell'abbigliamento è, forse, l'industria più importante della storia economica del mondo occidentale che sta ancora attraversando una grande evoluzione.
Il settore della moda si è espanso in maniera significativa soprattutto nella seconda metà del secolo scorso grazie all'avvento della tecnologia, diventando un settore sempre più informato, preparato ed esigente.
Come risultato, un pubblico sempre più ampio presenta numerose richieste che vanno oltre la mera necessità di indumento.
Queste esigenze hanno rapidamente trasformato la produzione di alcune aziende di abbigliamento sulla base della vendita all'ingrosso, aumentando i ritmi produttivi, il numero di processi manuali necessari e il numero di impiegati: oggi circa 75 milioni di persone in tutto il mondo sono direttamente coinvolte nel settore tessile, dell'abbigliamento e delle calzature.
In questo caso, un problema comune è che talvolta la produttività effettiva dei dipendenti non è sufficiente per raggiungere gli obiettivi di produzione nei tempi prestabiliti dalle autorità.
Quando si verifica un ampio divario fra produttività prevista e produttività realizzata, l'azienda deve quindi far fronte a significative perdite economiche.
È dunque necessario analizzare a fondo i fattori che possono influenzare la produttività dell'azienda in modo da prevenire situazioni di perdite economiche critiche.

Gli esperimenti condotti in questo progetto mirano a prevedere la produttività dei dipendenti di un'azienda di abbigliamento attraverso l'analisi di diversi fattori che ne influenzano l'andamento.
Il problema affrontato è di regressione. Verranno utilizzati tre algoritmi, confrontati e analizzati i risultati secondo metriche coerenti con il problema trattato.
Questo report verrà diviso in quattro sezioni principali che tratteranno l'analisi dei dati, i modelli scelti, i dettagli implementativi e la discussione e analisi dei risultati ottenuti.

1.2 Librerie utilizzate

Nello svolgimento del problema di regressione è stato utilizzato come linguaggio di programmazione Python. Sono state inoltre utilizzate alcune librerie open source per eseguire l'analisi dei dati. Le librerie in questione sono:

- Pyplot/Seaborn: Utilizzate per graficare, visualizzare e rappresentare dati;
- Pandas: Libreria flessibile ed intuitiva che fornisce tools per l'analisi di dati e la gestione di file .csv;
- Sklearn: Contiene strumenti efficienti per l'apprendimento automatico e la modellazione statistica;
- Numpy: Aggiunge supporto per la gestione di array multidimensionali. Contiene una vasta collezione di funzioni matematiche in grado di gestire rapidamente grandi quantità di dati.

1.3 Dataset

Il dati analizzati provengono dal dipartimento di ingegneria industriale (IE) di un'unità di produzione di abbigliamento di una rinomata azienda situata in Bangladesh. Il dataset[1] è stato raccolto tra Gennaio 2015 e Marzo 2015 e contiene i dati di produzione dei reparti di cucitura e finitura.

2 Analisi dei dati: l'EDA

"Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods"[2]
Da qua abbreviato in "EDA", Exploratory Data Analysis è un metodo utilizzato per eseguire indagini introduttive su dati in modo da porre ipotesi e formulare teorie. È un'importante prima fase in cui i dati iniziano ad essere intrpretati attraverso rappresentazioni grafiche ed analisi statistiche.

2.1 Analisi preliminare

Il dataset risulta avere una shape di:

- 1197 sample (righe);
- 15 features (colonne);

Facendo un'analisi preliminare si ottengono i seguenti valori dalla funzione *.info()* di pandas:

Feature	**Non-Null Count**	**Dtype**
Date	1197 non-null	object
Quarter	1197 non-null	object
Department	1197 non-null	object
Day	1197 non-null	object
Team	1197 non-null	int64
Targeted Productivity	1197 non-null	float64
SMV	1197 non-null	float64
WIP	691 non-null	float64
Over Time	1197 non-null	int64
Incentive	1197 non-null	int64
Idle Time	1197 non-null	float64
Idle Men	1197 non-null	int64
Number Of Style Change	1197 non-null	int64
Number Of Workers	1197 non-null	float64
Actual Productivity	1197 non-null	float64

Osservazioni:
- Il dataset riporta quattro features categoriche e undici features numeriche. Avendo scelto un problema di regressione, le features categoriche verranno successivamente trasformate in fase di preprocessing in features continue in modo da poter essere correttamente utilizzate negli algoritmi scelti;
- Fin da subito è possibile notare che sono presenti dei valori nulli nella feture "WIP" (infatti non vi sono 1197 non-null values, ma solamente 691). Successivamente spiegherò come verranno trattati.

2.1.1 Descrizione delle features

- Date: *object*. Esprime la data in mesi/giorni/anni;
- Quarter: *object*. Divisione del mese. Possono esserci cinque possibili valori;
- Department: *object*. Dipartimento associato all'istanza. Può assumere due valori: "Sewing" oppure "Finishing";
- Day: *object*. Giorno della settimana. Può assumere cinque valori in quanto "Friday" non è presente nel dataset;
- Team: *int*. Numero del team associato all'istanza;
- Targeted Productivity: *float*. Produttività target assegnata;
- SMV: *float*. Standard Minute Value è il tempo assegnato ad un team per completare un determinato task;
- WIP: *float*. Work In Progress è il numero di items non conclusi per tipologia di prodotto;

- Over Time: *int.* Ammontare di straordinari per ogni team;
- Incentive: *int.* Ammontare di incentivi finanziari (in BDT) che consente o motiva un piano di azione;
- Idle Time: *float.* Ammontare di tempo in cui la produzione è stata interrotta per diverse motivazioni;
- Idle Men: *int.* Numero di impiegati inattivi a causa dell'interruzione della produzione;
- Number Of Style Change: *int.* Numero di cambiamenti effettuati ad un determinato prodotto;
- Number Of Workers: *float.* Numero di impiegati per team;
- Actual Productivity: *float.* Percentuale di produttività effettivamente raggiunta.

2.1.2 Variabile target

La variabile target è quella i cui valori devono essere modellati e predetti a partire dalle altre features presenti nel dataset.
Actual productivity è la variabile target ed è espressa in percentuale che varia da 0 a 1.

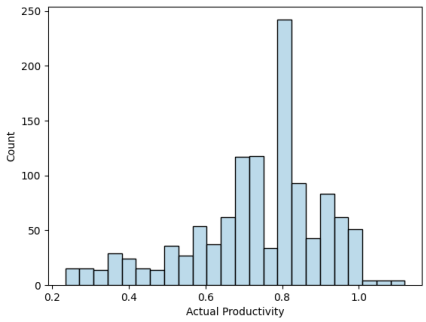


Figure 1: Target Variable Distribution

Come si può notare dalla figura, la variabile non è normal-distributa. È presente un picco di produttività fra 0.8 e 1.0. Il problema principale è capire le ragioni che causano una differenza fra produttività effettiva e produttività target. Quali sono le cause del perché non risulta effettivamente una produttività equivalente a quella programmata? Quanto effettivamente influiscono?
Il grafico sottostante riporta la differenza fra produttività target e produttività effettiva:

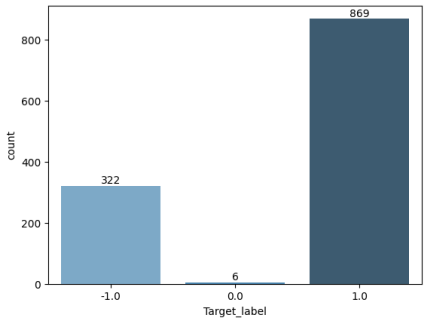


Figure 2: Difference Between Targeted and Actual Productivity

La differenza di produttività è divisa in tre gruppi: 1, 0 e -1: se la differenza fra produttività effettiva e produttività target è positiva significa che la produttività è nell'intervallo di sovra performance. Se la differenza fra produttività effettiva e produttività target è uguale a 0 significa che la produttività soddisfa il target preposto. Infine, se la differenza fra produttività effettiva e produttività target è negativa significa che la produttività è nell'intervallo di under performance. Dal grafico si evince che prevalentemente si è in un intervallo di sovra performance, aspetto positivo.

2.2 (A bit of) Preprocessing

Per poter procedere con la rappresentazione dei dati è stato necessario effettuare un controllo sulla presenza di eventuali valori nulli all'interno del dataset. Come emerso nella sezione 2.1 c'è un'alta percentuale di missing values solamente nella feature "WIP": 42,3%. Avendo solamente 15 features per il momento ho deciso di imporre ai valori mancanti la media dei valori presenti nella feature, essendo di tipo continuo e non categorico. Per quanto riguarda la feature "Incentive", come si può notare dal grafico sottostante, sono presenti degli **outliers**, ovvero dei valori anomali chiaramente distanti dalle altre istanze presenti. Gli outliers devono essere analizzati in quanto spesso contengono informazioni sul processo in esame o sul processo di raccolta e registrazione dei dati. Prima di considerare la possibile eliminazione di queste punti, si dovrebbe cercare di capire perché sono comparsi e se è probabile che continueranno a comparire valori simili.

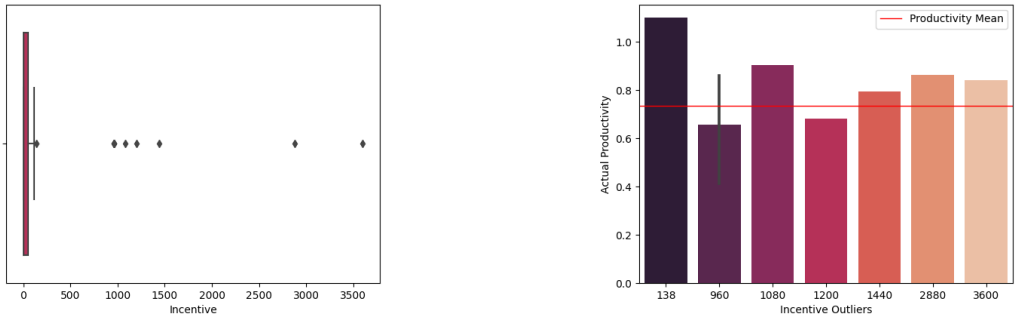


Figure 3: Incentive outliers

Alcune considerazioni: gli outliers presenti nella feature sono un numero limitato (10 valori) rispetto al totale e il discostamento dei valori in questione è significativo rispetto al range dei valori presenti nella feature. Infine, a parte il primo valore, i rimanenti non causano una variazione significativa della produttività, tant'è che l'andamento di quest'ultima non presenta particolari variazioni rispetto alla produttività media. A seguito di questa analisi ho deciso di rimuovere tutti gli outliers a parte il primo valore.
Successivamente ho apportato delle modifiche alle seguenti variabili:

- "Department" ha solamente due valori (sewing e finishing). Inizialmente ne sono presenti tre a causa dell'errore di typing per finishing, quindi sono stati corretti e uniti in due unici valori;
- "Date" è stata trasformata in "day", "month" e "year";
- "Day" è stata rinominata in "week day";

Inoltre, per poter includere le feature categoriche e valutare la correlazione fra le variabili ho applicato alcune tecniche di feature transformation esplicate nel capitolo 2.5.

2.3 Matrice di correlazione

La matrice di correlazione è una matrice quadrata simmetrica che riporta il coefficiente di correlazione fra due varibili. Possono essere utilizzati diversi indici per calcolare la correlazione. In particolare, la funzione *.corr()* di pandas utilizza come standard l'indice di correlazione di Pearson che esprime la relazione di linearità fra due elementi attraverso un range che va da -1 a +1, dove:

- +1: indica una perfetta relazione lineare positiva tra le due variabili;
- 0: indica un'assenza di correlazione;
- -1: indica una perfetta relazione lineare negativa tra le due variabili;

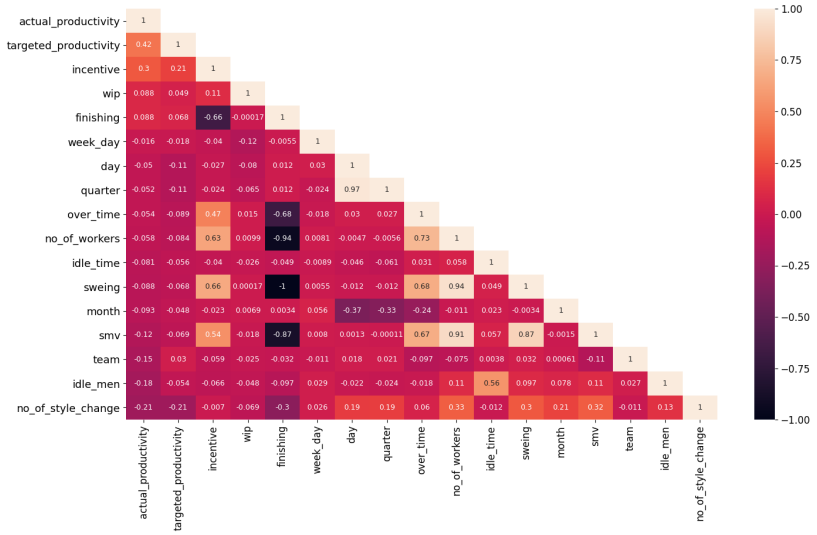


Figure 4: Ordered Correlation Matrix

Osservazioni:

- È presente un'alta correlazione fra Quarter e Day, fra Number of Workers e Sewing e Number of Workers e SMV;
- Nella matrice non è presente il valore "year" perché la feature assume solamente un valore (2015), risultando essere costante;
- C'è una perfetta correlazione lineare negativa fra "Sewing" e "Finishing".

2.4 Comparazione fra features

Analizzando la matrice di correlazione (figura 4) è possibile notare che alcune delle features hanno un indice abbastanza elevato. Lo stretto legame fra "Quarter" e "Day è ragionevole in quanto "Quarter" è riferito alla partizione del mese e "Day" rappresenta il giorno del mese.
È inoltre presente un'alta correlazione fra "Standard Minute Value" e "Number of Workers", infatti dal grafico sottostante è possibile notare che la relazione fra le due variabili sembra essere quasi lineare.

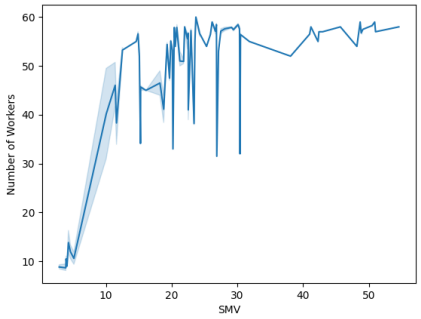
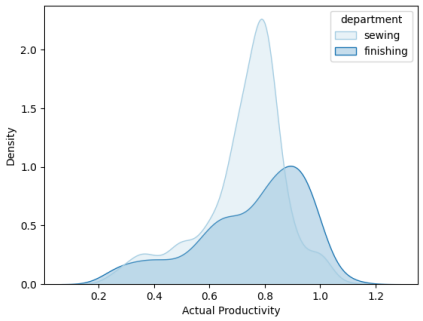
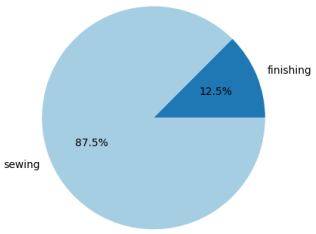


Figure 5: Number of Workers vs SMV

Inoltre ritengo interessante paragonare la produttività fra i due reparti "Sweing" (cucitura) rispetto a "Finishing" (finitura).
Si nota dal grafico che il reparto di cucitura ha performance più alte rispetto alla finitura, e questo potrebbe essere conseguenza di un numero maggiore di impiegati che lavorano nel reparto cucitura (come riportato nel grafico a torta).

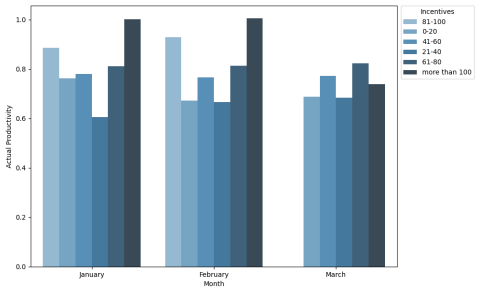


(a) Productivity vs Department

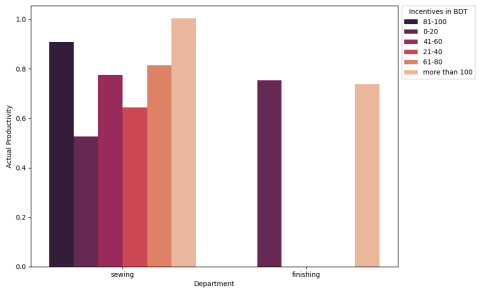


(b) Number of Workers per Department

Dalla matrice di correlazione, si nota che la seconda variabile più correlata con la variabile target è "incentive". Nei seguenti grafici ho riportato gli incentivi raggruppati per mese e per reparto:

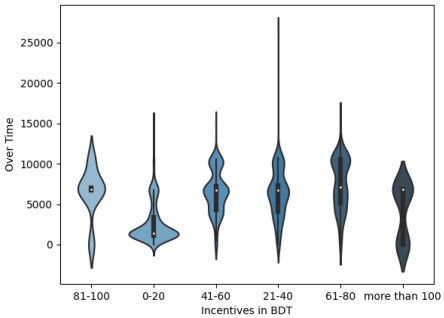


(a) Productivity vs Incentives per Month

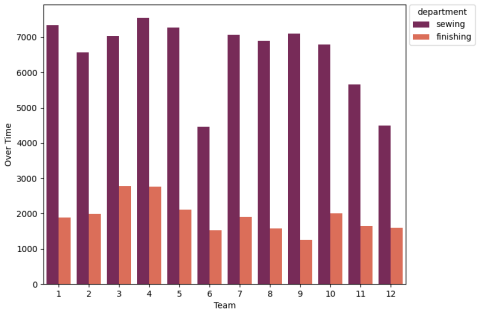


(b) Productivity vs Incentives per Department

Come si può interpretare dal primo grafico, nei mesi di Gennaio e Febbraio sono state erogate alte cifre di incentivi, e dunque la produttività si è alzata. Inoltre, sono stati equamente distributi gli incentivi maggiormente nel reparto sewing rispetto al reparto finishing.
Per quanto riguarda l'overtime, ovvero il numero di straordinari impiegati per completare un task, all'aumentare degli incentivi si nota un leggero miglioramento in termini di diminuzione degli straordinari. Rimane invece un overtime elevato per i team all'interno del reparto sewing e un'overtime basso per i team del reparto finishing.

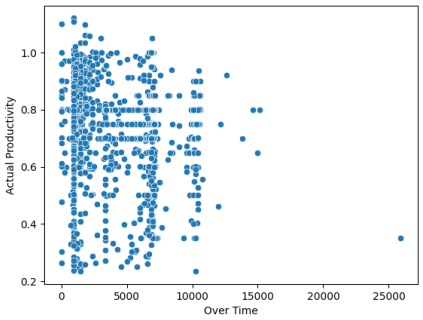


(a) Over Time vs Incentives

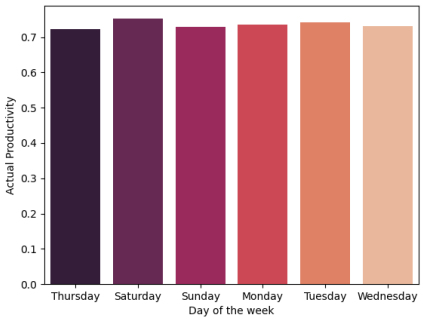


(b) Over Time vs Team per Department

In generale, per bassi valori di overtime ci sono alti valori di produttività. All'aumentare degli straordinari la produttività diminuisce.



(a) Over Time vs Productivity



(b) Day of the week vs Productivity

Infine, la produttività risulta essere abbastanza elevata per tutti e sei i giorni lavorativi.

2.5 Feature transformation

2.5.1 Encoding

Essendo presenti nel dataset features categoriche è stato necessario applicare delle tecniche di feature transformation. Quelle da me utlizzate sono *LabelEncoder()* e *get_dummies()*:

- Label Encoder: tecnica in cui il valore categorico viene sostituito con un valore numerico compreso tra 0 e il numero di classi meno 1. Nel mio caso ho deciso di utilizzare questa tipologia di encoding per le variabili "Quarter" e "Week Day". In particolare sono state così codificate:

Value	Encoding
Quarter1	0
Quarter2	1
Quarter3	2
Quarter4	3
Quarter5	4

Table 1: Quarter Encoding

Value	Encoding
Monday	0
Saturday	1
Sunday	2
Thursday	3
Tuesday	4
Wednesday	5

Table 2: Day Encoding

- Dummizzazione: per ogni valore distinto nella colonna categorica originale, crea una nuova colonna con indicatore 0 o 1. Ho utilizzato questa tipologia di encoding per la variabile "Department", separandola dunque nelle due features "Sewing" e "Finishing";

2.5.2 Normalizzazione

Il dataset è composto da variabli con range numerici diversi, dunque per poter utilizzare gli algoritmi in maniera ottimale è necessario distribuirli in intervalli omogenei. Ci sono tre tecniche che si possono utilizzare: normalizzazione, standardizzazione e scaling.
Ho deciso di utilizzare quest'ultima che prevedere la trasformazione dei valori in un unico range fra 0 e 1. La motivazione è anche data dal fatto che in Support Vector Machine è necessario ridimensionare le features in modo tale che abbiano tutte un'influenza simile nel calcolo della distanza per costruire un iperpiano. Inoltre, in questo modo si evitano le difficoltà legate al calcolo in quanto i valori del kernel di solito dipendono dai prodotti interni dei vettori delle caratteristiche e attributi grandi potrebbero causare problemi numerici.

2.6 Feature selection

Alcune feature sono risultate poco rilevanti sulla base dell'indice di correlazione e sulla base della varianza:

- La feature "Year" è stata rimossa, come precedentemente esposto, a seguito dell'analisi sulla varianza, risultando avere un valore costante. In realtà, anche la feature "targeted_productivity" ha

varianza bassa ma presenta un indice di correlazione abbastanza elevato con la variabile target, dunque ho deciso di mantenerla;

- La feature "Day" è stata rimossa in quanto presenta un'alta correlazione con la feature "Quarter", rischiando dunque multicollinearità.

2.7 Criterio di splitting

Il criterio di splitting utilizzato è stato 80-20, ovvero 80% del dataset dedicato al training set mentre il 20% sarà utilizzato per il test set.

3 Discussione dei modelli

Il problema trattato viene definito di regressione. L'obiettivo è predirre il comportamento continuo della variabile target a partire dalle variabili di input.
Fra i modelli discussi e studiati a lezione, ho deciso di focalizzarmi su tre principali paradigmi di Machine Learning utilizzati nei problemi di regressione:

- Support Vector Machine - Regression;
- K-Nearest Neighbors;
- Ridge Regression.

3.1 Support Vector Machine - Regression

Support Vector Machine (SVM) è un tipo di algoritmo di apprendimento supervisionato utilizzato sia in problemi di regressione che di classificazione. Nei problemi di regressione viene utilizzato l'algoritmo **SVR Support Vector Regressor**.
In generale, l'obiettivo dell'algoritmo SVM è trovare un iperpiano che separi due diverse classi che convivono in uno spazio p-dimensionale, dove p è il numero di features che definisce la dimensione dello spazio. In particolare, si vuole considerare l'iperpiano a massimo margine, ovvero l'iperpiano più distante tra tutti i sample di tutte le classi in modo da avere maggiori garanzie che i nuovi sample vengano correttamente classificati.
Nella ricerca dell'iperpiano, l'algoritmo potenzialmente non utilizza tutti i sample ma solamente i support vectors, ovvero i sample più vicini all'iperpiano. Questo ha grandi vantaggi in termini di risparmio di memoria (si salvano solamente i support vector) e riduzione del costo computazionale.
Il problema di regressione è una generalizzazione del problema di classificazione, in cui il modello restituisce un output a valori continui, anziché un output discreto. In altre parole, un modello di regressione stima una funzione multivariata a valori continui.
Per fare ciò, anziché realizzare un margine che tenga sample separati, SVR realizza un **ϵ-tube**, ovvero un "tubo" che cerca di contenere il maggior numero di sample possibili.

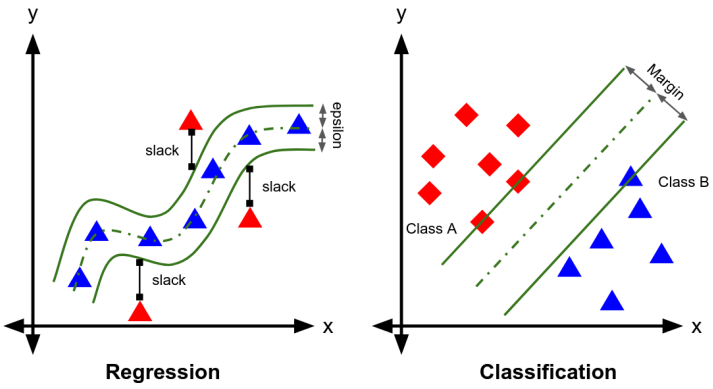


Figure 10: SVM for Classification vs Regression

Il valore di epsilon determina la larghezza del tubo attorno all'iperpiano. I punti che cadono all'interno di questo tubo sono considerati previsioni corrette e non sono penalizzati dall'algoritmo.
Il vantaggio di utilizzare questo tipo di algorimo è che si possono utilizzare anche kernel non lineari per problemi che non possono essere risolti linearmente. Questo avviene attraverso il "kernel trick", ovvero una trasformazione non lineare dell'iperpiano in modo da renderlo uno spazio a piu' dimensioni.

3.2 K-Nearest Neighbors

K-Nearest Neighbors (KNN) è un tipo di algoritmo di apprendimento supervisionato utilizzabile sia in problemi di regressione che di classificazione.
Nei problemi di regressione, **K-Nearest Neighbors** rappresenta un modello di regressione non-lineare, ovvero un metodo per trovare un modello non lineare della relazione tra la variabile dipendente e un insieme di variabili indipendenti.

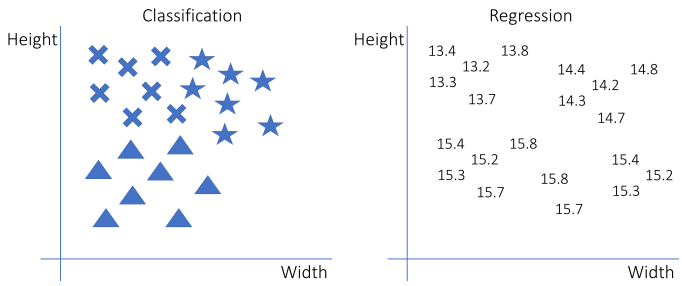


Figure 11: Example of KNN for Classification vs Regression

Il principio di funzionamento dell'algoritmo si basa sulla similarità fra sample calcolandone la distanza relativa. La misura di similarità più frequentemente utilizzata è la distanza Euclidea data dalla seguente formula:

$$d(q^i, q^l) = \sqrt{\sum_{j=1}^{p} (q_j^{(i)} - q_j^{(l)})^2} \tag{1}$$

Vengono inoltre utilizzate la distanza di Chebyschev, la distanza di Manhattan e la distanza di Mikowski. Nei problemi di regressione l'algoritmo approssima l'associazione tra variabili indipendenti e il risultato continuo facendo la media dei k-neighbors più vicini:

1. Viene calcolata la distanza fra l'i-esimo sample e il sample query;
2. Si aggiungono tutte le distanze in un vettore;
3. Si ordina il vettore;
4. Si selezionano i primi K sample;
5. Dei K sample viene restituita la media;

3.3 Ridge Regression

La regressione Ridge, è un tipo di regressione regolarizzata, ovvero un modello che impone un vincolo ai pesi delle features per ridurre la funzione di costo.

3.4 Concetti preliminari

Prima di proseguire con la spiegazione della regressione regolarizzata, è importante introdurre i concetti di **bias**, **varianza**, **under-fitting** e **over-fitting**.

- Bias: è la differenza tra la previsione media del nostro modello e il valore corretto che stiamo cercando di prevedere. Un modello con alto bias presta pochissima attenzione ai dati di training e semplifica eccessivamente il pattern;
- Varianza: è la variabilità della previsione del modello per un dato sample. Il modello con varianza elevata presta molta attenzione ai dati di addestramento (training set) e non generalizza sui dati che non ha mai visto prima (test set). Di conseguenza, tali modelli si comportano molto bene sui dati di addestramento ma hanno alti tassi di errore sui dati di test.

Nei modelli di apprendimento supervisionato, si verifica una condizione di **underfitting** quando un modello non è in grado di acquisire il pattern sottostante dei dati, non riuscendo dunque a rappresentare correttamente nè i dati di test nè i dati di training. Questo solitamente avviene con bias elevato e bassa varianza. Si verifica invece una condizione di **overfitting** quando il modello cattura il rumore insieme al modello sottostante ai dati, causando un eccesso di addestramento e una scorretta rappresentazione dei dati di test. Questo solitamente avviene con basso bias e varianza elevata.

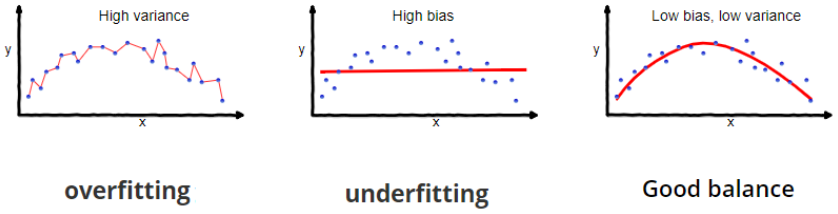


Figure 12: Bias and Variance

Nella regressione lineare, quando si aggiungono delle features al modello, la sua complessità naturalmente cresce, facendo aumentare conseguentemente la varianza.
Come precedentemente esposto, se la varianza aumenta si sfocierà in una condizione di overfitting. Per ovviare questo problema si possono utilizzare **tecniche di regolarizzazione** dove le grandezze delle variabili indipendenti vengono ridotte mantenendole però tutte per fare prediction. In particolare, viene aggiunto un termine di penalità (shrinkage) alla funzione di costo del modello.
Ci sono prevalentemente due tipi di regressione regolarizzata: Ridge Regression e Lasso Regression. La principale differenza fra le due tecniche è che nella regressione Ridge i pesi associati alle feature possono essere notevolmente ridotti, ma mai annullati. Nella regressione Lasso, invece, le feature possono anche acquisire peso 0, motivo per il quale è anche utilizzata come tecnica di feature selection.
Siccome nel mio dataset non è presente un elevato numero di features, ho deciso di utilizzare la regressione Ridge, non avendo necessità di ridurre utileriormente l'insieme di features utilizzate per predirre la variabile target.

3.4.1 Ridge Regression

La regressione Ridge riduce la funzione di costo aggiungendo un termine di penalità chiamato **norma L-2** che moltiplica l'iperparametro α con il peso al quadrato di ciascuna feature:

$$\sum_{i=0}^{N} \left\{ y_i - \sum_{j=0}^{M} \beta_j x_{ij} \right\}^2 + \alpha \sum_{j=0}^{M} \beta_j^2 \tag{2}$$

Dunque, il parametro α regolarizza i coefficienti in modo tale che se questi assumono valori elevati la funzione di ottimizzazione viene penalizzata. Quando α si avvicinerà a zero, la funzione di costo sarà approssimabile a quella della regressione lineare.
Il vincolo imposto dalla regressione Ridge è un vincolo convesso, ovvero risolvibile in forma chiusa, a differenza del vincolo imposto dalla regressione Lasso. Questo implica che l'algoritmo può essere risolto in forma chiusa e non iterativamente.
La scelta dell'iperparametro α verrà discussa nel capitolo 4.1.3

3.5 Ensemble

Ensembling è la strategia utilizzata in Machine Learning per cercare le migliori prestazioni possibili combinando le previsioni di più modelli detti "weak learner" o "base estimator". I modelli vengono addestrati per eseguire lo stesso task e in seguito combinati in un unico modello detto "strong model" o "ensemble model".
Quando vengono addestrati in parallelo modelli dello stesso tipo si parla di **ensemble omogeneo**. Successivamente vengono uniti e valutati per voto o media.
Quando vengono addestrati modelli diversi si parla di **ensemble eterogeneo**. Vengono addestrati singolarmente e poi combinati.
I weak learner possono essere combinati in tre possibili modi:

- Bagging: vengono utilizzati weak lerner omogenei. Vengono addestrati in parallelo e poi combinati;
- Boosting: vengono utilizzati weak learner omogenei. Vengono addestrati sequenzialmente in maniera adattativa. In seguito vengono combinati;
- Stacking: vengono utilizzati weak learner eterogenei. Vengono addestrati indipendentemente e poi combinati;

Per il progetto ho scelto di utilizzare Bagging applicato alla variante migliore di ogni modello considerando come metrica principale R square (vedi sezione 4.2). I risultati completi sono riportati nella tabella 6. I modelli selezionati sono **KNN with preprocessing**, **SVR with preprocessing** e **Ridge without preprocessing**.

4 Dettagli implementativi

4.1 Scelta degli iperparametri

Per ottenere modelli il più performanti possibile è auspicabile utilizzare tecniche di **cross validazione** sia per scegliere i migliori iperparametri per un modello (model selection) che per testare le performance del modello su dati nuovi (model assesment). In questo progetto si procederà con l'utilizzo di cross validazione per fare model selection.
L'idea generale è quella di utilizzare il training set non solo per il training del modello ma anche per la ricerca dei migliori parametri da poter utilizzare.
È possibile utiizzare una fra le seguenti tecniche di cross validazione:

- Holdout: il training set viene ulteriormente diviso in training e validation set. La porzione di training viene utilizzata per addestrare il modello mentre la porzione di validation viene utilizzata per valutare le prestazioni. È un approccio semplice, utilizzato come standar per modelli complessi, ma i risultati sono dipendenti da quali dati sono presenti nel training set e quali nel validation set;
- K-fold: il dataset viene diviso in k parti uguali. k-1 folds vengono utilizzate come training set mentre il k-esimo fold viene utilizzato come validation set. Questo procedimento di divisione viene effettuato k volte cambiando di volta in volta il k riservato al validation set. Alla fine dei k-run le prestazioni del modello verranno valutate facendo la media. Di default viene tenuto k = 5 folds;
- Leave-One-Out: il training set contiene n-1 sample e il validation set contiene solamente l'n-esimo sample. L'addestramento viene fatto sul training set e la validazione viene fatta sull'n-esimo sample. È una tecnica che è meglio utilizzare in dataset di dimensioni ridotte;
- Stratificazione: training set e validation set vengono suddivisi in modo tale che contengano un numero di sample di una classe proporzionalmente alla numerosità delle classi. Questo approccio è utilizzato specialmente in dataset sbilanciati.

In python grazie al metodo *GridSearchCV* è possibile implementare il metodo K-fold. Come imput il metodo accetta:

- Un modello;
- La metrica di valutazione (legata ovviamente al tipo di problema che si sta affrontando);
- Un dizionario contentente gli iperparametri del modello;
- Il numero di folds in cui si vuole dividere il training set. Se viene passato un numero intero il metodo effettuerà automaticamente una stratified cross validation;

4.1.1 K-Nearest Neighbors hyperparameters

Per il modello K-Nearest Neighors, gli iperparametri presi in considerazione sono **K** (numero di neighbors) e **weight**.

- K: numero di neighbors presi in considerazione. All'aumentare di K l'accuratezza della previsione aumenta, nel momento in cui si scelgono K troppo elevati non vi sono più vantaggi. Per i problemi di classificazione in generale è auspicabile tenere K dispari per evitare di assegnare la stessa classe ad un sample. Non è questo il mio caso;
- Weight: che peso assegnare a ciascun neighbor. I valori accettati possono essere "uniform", ovvero tutti i vicini assumono lo stesso peso, oppure "distance", ovvero i punti più vicini al sample query assumono un peso maggiore.

Ho deciso di far variare K da 1 a 15 e per Weight ho considerato entrambe le possibilità, ovvero "uniform" e "distance".
Sono stati valutati gli iperparametri sul modello con dati non preprocessati e sul modello con dati preprocessati ottenendo i seguenti risultati:

	K	**Weight**
KNN NO PP	12	distance
KNN PP	15	distance

Table 3: Best KNN hyperparameters

4.1.2 SVR hyperparameters

Per il modello Support Vector Regressor, gli iperparametri presi in considerazione sono **C**, il **kernel** ed **epsilon**.

- C: è un parametro di regolarizzazione che controlla il trade-off fra l'ampiezza del margine e l'errore effettuato sul sample. Per C piccoli si dà meno importanza agli errori e ci si focalizza maggiormente sul margine. Per C ampi si dà più importanza agli errori e si riduce l'apiezza del margine;
- Kernel: specifica la funzione che sposta i dati da un piano a dimensione più bassa ad uno di dimensioni maggiori che meglio includerà i sample. Il kernel può essere "linear" ossia lineare, "poly" ossia curve polinomiali, "rbs" ossia radial basis function cioè curve di complessità maggiore;
- Epsilon: serve per settare l'ampiezza del tubo che meglio approssima i sample.

Ho deciso di utilizzare come parametri di C i valori 0.1, 1 e 10. I kernel da me considerati sono "poly", "linear" e "rbf mentre i valori di epsilon sono 0.01, 0.1, 0.2, 0.5, 1, 2.

Osservazione: non ho incluso nel range di epsilon il valore zero in quanto, nonostante porti dei buoni risultati, potrebbe aumentare molto la probabilità di sfociare in una condizione di over-fitting.

Sono stati valutati gli iperparametri sul modello con dati non preprocessati e sul modello con dati preprocessati ottenendo i seguenti risultati:

	C	**Kernel**	**Espilon**
SVR NO PP	10	rbf	0.01
SVR PP	10	rbf	0.01

Table 4: Best SVR hyperparameters

NOTA BENE: Nella scelta degli iperparametri per dati non preprocessati, i dati sono stati comunque scalati (vedi sezione 2.6).

4.1.3 Ridge Regression hyperparameters

Per il modello di Ridge Regression, l'unico iperparametro preso in considerazione è il parametro **alpha**. α è il parametro regolatore che cerca di bilanciare l'adattamento del modello ai dati e i pesi dei coefficienti del modello.

$$RSS + \alpha \sum_{j=0}^{M} \beta_j^2 \tag{3}$$

Per α bassi non viene imposta penalità ai coefficienti, mentre quando α aumenta i coefficienti vengono maggiornamente penalizzati. Come precedentemente esposto, con la regressione Ridge i coefficienti non verranno annullati, a differenza della regressione Lasso, nei valori di alpha non sarò incluso lo zero. Ho deciso di utilizzare come parametri: 0.001, 0.01, 0.1, 0.5, 1, 10, 100.
È stato valutato l'iperparametro sul modello con dati non preprocessati e sul modello con dati preprocessati ottenendo i seguenti risultati:

	alpha
Ridge NO PP	0.001
Ridge PP	0.5

Table 5: Best Ridge hyperparameters

4.2 Metriche utilizzate

A questo punto è necessario definire le metriche utilizzate per valutare le prestazioni di un algoritmo di regressione. Ne ho selezionate alcune che ritengo essere le più esplicative.

4.2.1 R Square

R square è la metrica impiegata per valutare quanto il modello è in grado di esplicitare la variabilità della variabile target. È calcolato in questo modo:

$$1 - \frac{RSS}{TSS} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2} \tag{4}$$

RSS è il Residual Sum of Squares, ovvero la differenza fra valori predetti e valori veri elevati al quadrato. La differenza è chiamata residuo. TSS è il Total Sum of Squares, ovvero la differenza fra valore predetto e valore vero medio.
R square varia fra 0 e 1 è **più è alto più c'è corrispondenza fra valore predetto e valore effettivo.**
Lo svantaggio di R square è che non tiene conto dell'over-fitting pur essendo un buon indicatore per valutare le prestazioni.

4.2.2 RMSE - Root Mean Square Error

Root Mean Square Error è la radice quadrata del Mean Square Error (MSE):

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{5}$$

È usato più comunemente dell'MSE perché tramite la radice i valori più ampi di errore vengono maggiormente evidenziati. RMSE è una buona misura da utilizzare se vogliamo stimare la deviazione standard di un valore dato dalla previsione del nostro modello. **Più è basso il valore di RMSE più il modello predirrà accuratamente i dati**.

4.2.3 MAE - Mean Absolute Error

Mean Absolute Error è simile all'errore quadratico medio (MSE). Tuttavia, invece della somma dei quadrati dell'errore in MSE, MAE sta prendendo la somma del valore assoluto dell'errore:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \mid y_i - \hat{y}_i \mid \tag{6}$$

Rispetto a MSE o RMSE, MAE è una miglior rappresentazione della somma dei termini di errore. MSE penalizzazione maggiormente grandi errori di previsione elevando al quadrato mentre MAE tratta tutti gli errori allo stesso modo. **Più è basso il valore di MAE più il modello predirrà accuratamente i dati**.

Nella prossima sezione verranno analizzati i risultati secondo queste metriche.

5 Risultati e discussione

In questa sezione verranno inizialmente utilizzate tutte le metriche elencate nella sezione 4.2. Verranno invece riportati i risultati dell'ensemble facendo riferimento solamente all'R square. I controlli sono stati effettuati sui dati non preprocessati, su dati preprocessati e in ensemble con ciascuna variante migliore dei modelli.
Sono state inoltre monitorate le tempistiche necessarie per la ricerca degli iperparametri e le tempistiche di predizione dei modelli con i migliori iperparametri.

5.1 Risultati

5.1.1 Metriche

Riporto nella seguente tabella le metriche selezionate calcolate per ciascun modello:

	$\mathbf{R^2}$	**RMSE**	**MAE**
KNN NO PP	0.1202	0.1702	0.1219
KNN PP	0.2473	0.1575	0.1125
SVR NO PP	-0.0036	0.1819	0.1452
SVR PP	0.3429	0.1472	0.0949
Ridge NO PP	0.3702	0.1440	0.1048
Ridge PP	0.3678	0.1443	0.1049

Table 6: Metriche di valutazione sui modelli

Osservazioni: Per quanto riguarda R square, sia per KNN che per SVR si nota un leggero miglioramento a seguito del preprocessing (soprattutto in SVR). Per quanto riguarda RMSE e MAE c'è un miglioramento per KNN e SVR.

5.1.2 Tempi

Nella seguente tabella riporterò le tempistiche per il calcolo degli iperparametri e il process time:

	Cross validation Timing [s]	**Process Timing [s]**
KNN NO PP	0.9099	0.0040
KNN PP	0.6063	0.0059
SVR NO PP	7.3589	0.0185
SVR PP	7.4285	0.0207
Ridge NO PP	0.1560	0.0016
Ridge PP	0.0431	$6.7234 \cdot 10^{-5}$

Table 7: Tempistiche

5.1.3 Risultati Ensemble

Infine riporto i risultati a seguito dell'ensemble dei modelli selezionati:

	$\mathbf{R^2}$
KNN PP	0.2241
SVR PP	0.3563
Ridge NO PP	0.3698

Table 8: Risultati Ensemble

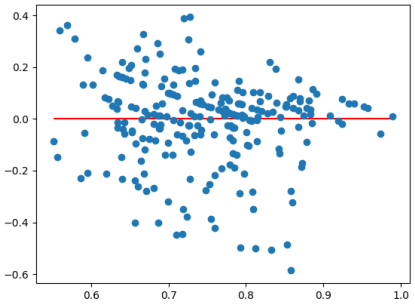
Come visibile nella tabella 8, rispetto ai risultati ottenuti precedentemente c'è un leggero miglioramento per quanto riguarda soprattutto Rigde e SVR.

5.2 Osservazioni conclusive

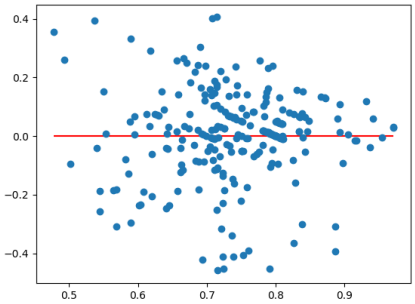
Dai risultati riportati in sezione 5.1 considerando dati preprocessati:

- Per quanto riguarda le metriche: a seguito del preprocessing c'è un miglioramento per K-Nearest Neighbors e per Support Vector Regression e un leggero peggioramento per quanto riguarda Ridge. In ensemble Ridge è il modello con un R square migliore;
- Per quanto riguarda le tempistiche: a seguito del preprocessing sia KNN che Support Vector Regression pagano in termini di tempo, mentre Ridge gode del preprocessing rispermiando sopratutto in process time;

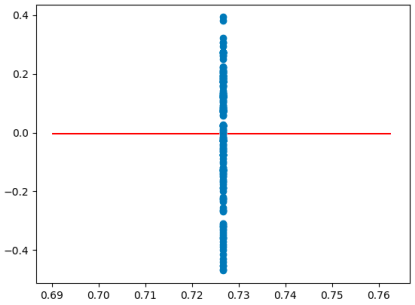
Riporto di seguito i grafici dei residui (differenza fra valore predetto e valore reale) di ciascun modello:



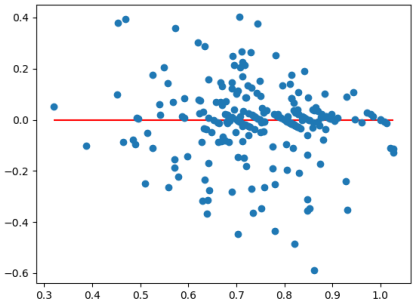
(a) KNN no preprocessing



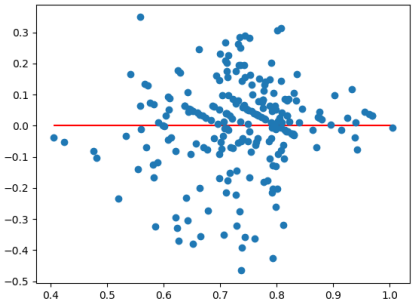
(b) KNN with preprocessing



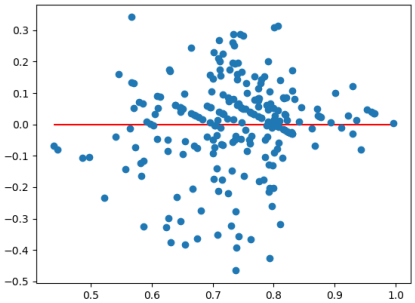
(c) SVR no preprocessing



(d) SVR with preprocessing



(e) Ridge no preprocessing



(f) Ridge with preprocessing

A parte SVR senza preprocessing, gli altri residual plots risultano essere abbastanza accettabili in quanto sono sufficientemente distribuiti simmetricamente, sono clusterizzati attorno alla prima cifra decimale dopo lo zero e, in generale, non segono un chiaro pattern ma seguono pattern random.
Per quanto riguarda il behavior del residual plot di SVR senza preprocessing, la ragione dell'andamento potrebbe essere legata al fatto che la predizione è stata fatta su dati di test non normalizzati. In generale

utilizzando questa tipologia di algoritmi è sicuramente auspicabile effettuare normalizzazione dei dati e, sicuramente per SVR e KNN, fare preprocessing.

References

[1] Abdullah Al Imran. Productivity prediction of garment employees data set. https://archive.ics.uci.edu/ml/datasets/Productivity+Prediction+of+Garment+Employees.

[2] Wikipedia. Exploratory data analysis. https://en.wikipedia.org/wiki/Exploratory_data_analysis.