Antonio Renzullo

https://github.com/Ariethel/Ok-il-prezzo-giusto-.git

Indice

Parte I
Introduzione

1 Cos'è Ok, il prezzo è giusto?

Si tratta di un progetto giocattolo, che potrebbe essere adattato per risolvere uno dei problemi della società moderna: l'eccessivo consumismo.
Trovo in particolar modo sconcertante la quantità di dispositivi mobile che ogni anno vengono gettati via anche se perfettamente funzionanti e funzionali al loro scopo. Questo progetto cerca di predire un prezzo "vincente" per i dispositivi usati, ovvero un prezzo che ne permetta la vendita in un lasso di tempo ragionevole senza svenderli, in modo da ridurre l'e-waste e generare maggiore interesse verso il mercato dell'usato.

1.1 Specifiche PEAS

L'acronimo PEAS sta per: Performance, Environment, Actuators e Sensors. Viene utilizzato per descrivere un ambiente.

- **Performance measure**: La precisione della previsione dei valori di output, nel caso in esame: MAE, MSE ed R^2.
- **Ambiente**: Il file CSV contenente i dati di input e di output.
- **Attuatori**: Il modello di regressione utilizzato per generare le previsioni.
- **Sensori**: Una piccola interfaccia grafica per leggere i dati dal file CSV e per generare le previsioni utilizzando il modello di regressione.

1.2 Caratteristiche dell'ambiente

- **Completamente osservabile**: L'agente ha una visione completa di tutti i dati che gli vengono forniti in ogni momento.
- **Deterministico**: lo stato successivo dell'ambiente è completamente determinato dallo stato corrente e dall'azione eseguita dall'agente, dunque, è un ambiente che non cambia nel tempo.
- **Statico**: L'ambiente resta invariato durante le elaborazione dell'agente
- **Discreto**: L'ambiente fornisce un numero limitato di percezioni e predizioni.
- **Episodico**: Le azioni dell'agente influiscono soltanto a breve termine, senza influenzare quelle future.
- **Singolo agente**: L'ambiente permette l'esistenza di un solo agente.

Parte II
Tecniche di risoluzione

2 Machine Learning

Il Machine Learning (ML) è un ramo dell'Intelligenza Artificiale (IA) che si concentra sullo sviluppo di algoritmi e modelli matematici che consentono a un computer di apprendere e migliorare automaticamente dall'esperienza, senza essere esplicitamente programmati.
In sintesi, il Machine Learning è una parte dell'IA che si concentra sull'apprendimento automatico dai dati, mentre l'IA è un campo più ampio che comprende anche altre tecniche per creare sistemi intelligenti.

Durante lo sviluppo di questa demo ci si è concentrati su una tipologia di machine learning detto **apprendimento supervisionato**, questo consiste nel fornire all'agente dei dati etichettati, ovvero comprensivi di variabili indipendenti (quelli che si usano per la predizioni) e della variabile dipendente (quella che si vuole predire).
I modelli che si andranno ad utilizzare per la risoluzione del problema in questione sono basati su **algoritmi di regressione**, ovvero tecniche in grado di predire il valore di variabili continue a partire dagli input forniti.

2.1 Modello di approccio al problema

Per progettare una soluzione incentrata sul machine learning si fa necessario un approccio di tipo ingegneristico.
CRISP-DM è un acronimo che sta per "Cross Industry Standard Process for Data Mining", è una metodologia standard per la conduzione di progetti di data mining e di business intelligence. La metodologia CRISP-DM è stata sviluppata per essere indipendente dalla specifica industria o dal tipo di dati, e può essere utilizzata per qualsiasi progetto di data mining.

La metodologia CRISP-DM è composta da sei fasi:

- Comprensione del business e del contesto
- Data understanding
- Preparazione dei dati
- Modellazione
- Evaluazione
- Deployment (utilizzo dei risultati)

La fase di comprensione del business e del contesto consente di definire gli obiettivi e le aspettative del progetto. La fase di data understanding consente di acquisire una conoscenza dei dati disponibili, mentre la fase di preparazione dei dati consente di preparare i dati per la modellazione. La fase di modellazione consiste nell'applicazione di algoritmi di data mining per creare modelli predittivi. La fase di valutazione consente di valutare l'accuratezza dei modelli creati, mentre la fase di deployment consente di utilizzare i risultati ottenuti per risolvere i problemi di business.

2.2 Comprensione del business e del contesto

Durante questa fase si raccolgono i requisiti e gli obiettivi di business.

Obiettivi di business L'agente avrà come obiettivo quello di stimare un prezzo vincente per la vendita di uno smartphone in un tempo relativamente breve e senza recare una eccessiva perdita economica al proprietario. La previsione andrà fatta sulla base delle caratteristiche del dispositivo.

Dataset Il dataset utilizzato per questa demo è stato trovato su Kaggle al link `https://www.kaggle.com/datasets/ahsan81/used-handheld-device-data`

Tecnologie impiegate Tutto è stato scritto utilizzando il linguaggio Python. Per il caricamento del dataset e per alcune operazioni di pulizia è stato scelto di utilizzare Pandas, mentre per la generazione di grafici si è optato per PyPlot. Tutta la parte di modellazione del dataset ed addestramento del modello è invece stata realizzata con l'ausilio di scikit-learn. Inoltre, per la semplicissima interfaccia grafica si è fatto riferimento alla libreria Tkinter.

2.3 Data Understanding

Il dataset di partenza è piuttosto semplice. Questo contiene **3454** entry di smartphone usati con 15 caratteristiche per riga:

- Nome del produttore
- Sistema operativo
- Misura dello schermo in cm
- Se dotato di tecnologia 4g (y/n)
- Se dotato di tecnologia 5g (y/n)
- Megapixel camera frontale
- Megapixel camera posteriore
- Ammontare di memoria interna in GB
- Ammontare di RAM in GB
- Dimensione della batteria in mAh
- Peso in grammi
- Anno di uscita sul mercato
- Numero di giorni di utilizzo del dispositivo
- Prezzo da nuovo (normalizzato)
- Prezzo da usato (normalizzato)

Sfortunatamente, nonostante le richieste di diversi utenti, non ci è stato dato sapere che tipo di normalizzazione il creatore di questo dataset abbia applicato agli ultimi due campi. Speculativamente parlando sembra essere stata utilizzata una **boxcox transformation**, ma questa operazione risulta impossibile da invertire senza il parametro lambda utilizzato per la sua applicazione.
Trattandosi di un Dataset costruito a scopo didattico, questo aveva pochissimi problemi, per cui la fase di preparazione dei dati è risultata piuttosto veloce.
Prima di tutto, grazie ad uno dei comandi della libreria Pandas ho stampanto alcune informazioni sul database, tra le quali eventuali entry mancanti.

```
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   device_brand           3454 non-null   object
 1   os                     3454 non-null   object
 2   screen_size            3454 non-null   float64
 3   4g                     3454 non-null   object
 4   5g                     3454 non-null   object
 5   rear_camera_mp         3275 non-null   float64
 6   front_camera_mp        3452 non-null   float64
 7   internal_memory        3450 non-null   float64
 8   ram                    3450 non-null   float64
 9   battery                3448 non-null   float64
 10  weight                 3447 non-null   float64
 11  release_year           3454 non-null   int64
 12  days_used              3454 non-null   int64
 13  normalized_used_price  3454 non-null   float64
 14  normalized_new_price   3454 non-null   float64
dtypes: float64(9), int64(2), object(4)
memory usage: 404.9+ KB
```

Come si può vedere in figura ci sono in totale **202** valori NULL. Questi sono stati integrati su un dataset fantoccio attraverso un SimpleImputer preso da scikit-learn. La tecnica di **data imputation** utilizzata è quella "most-frequent", ovvero la sostituzione dei valori nulli con quelli di frequenza maggiore all'interno della colonna.
Una volta terminato questo passaggio si è voluta verificare la linearità dei dati al fine di scegliere, in un successivo momento, il modello più adatto a risolvere il problema in questione. Per effettuare questo controllo sono state utilizzate due tecniche:

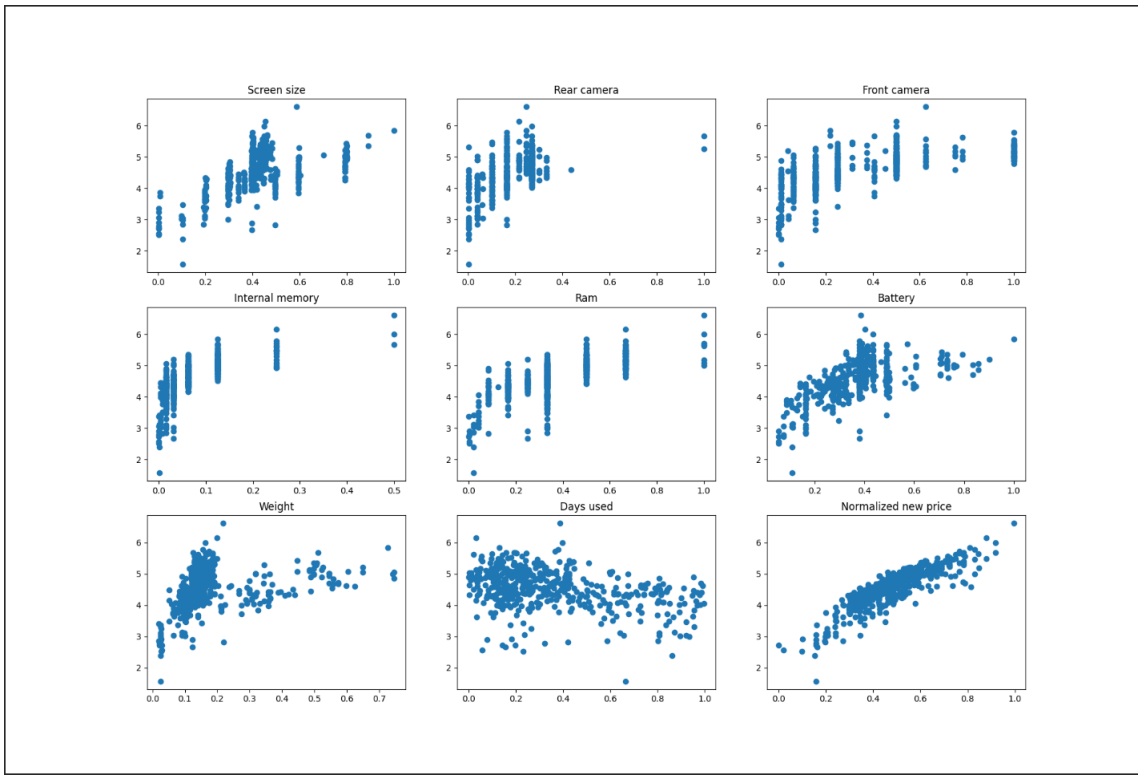
- Plot di alcuni valori su grafico.
- Test di Pearson.

2.3.1 Plot su grafico

Questo test non era strettamente necessario, ma è risultato utile per rendersi conto visivamente della distribuzione di alcune features su grafico. Sono quindi state scelte:

- Misura dello schermo
- Megapixel camera frontale
- Megapixel camera posteriore
- Ammontare di memoria interna
- Ammontare di RAM
- Dimensione della batteria
- Peso
- Numero di giorni di utilizzo del dispositivo
- Prezzo da nuovo

Queste caratteristiche sono, per loro natura, distribuite su scale numeriche differenti. Per tale ragione si è scelto di normalizzarle attraverso un algoritmo MinMaxScaler, risultando in valori che variano al massimo tra 0 ed 1.



Dal grafico si vede (abbastanza) chiaramente la linearità dei dati, nonostante gli inevitabili **outiliers**.

2.3.2 Test di linearità di Pearson

In statistica, l'indice di correlazione di Pearson (anche detto coefficiente di correlazione lineare) tra due variabili statistiche è un indice che esprime un'eventuale relazione di linearità tra esse.
Questo deve risultare in un valore compreso tra [-1,+1], dove +1 corrisponde alla perfetta correlazione lineare positiva, 0 corrisponde a un'assenza di correlazione lineare e -1 corrisponde alla perfetta correlazione lineare negativa.
Matematicamente parlando, date due variabili statistiche X ed Y, l'indice di correlazione di Pearson è definito come la loro covarianza divisa per il prodotto delle deviazioni standard delle due variabili:

$$\rho XY = \frac{\sigma XY}{\sigma_X \sigma_Y}$$

Per fortuna il test è incapsulato in una funzione corr(), quindi si può facilmente riportare il suo esito in un file csv, per comodità chiamato "correlation.csv".
Analizzando questo file si può facilmente notare la correlazione lineare positiva tra le variabili in esame, ad eccezione (giustamente) dei giorni di utilizzo del dispositivo, che vanno a far decrescere il valore di questo.

2.4 Data Preparation

Verificata la linearità dei dati numerici si è passato a lavorare sul dataset reale. Grazie, infatti, alla funzione *get_dummies*() di pandas si può ottenere un dataset le cui variabili categoriche vengono **scalate** con una tecnica di **OneHot Encoding**, ovvero codificate in stringhe binarie univocamente distinte.
Vengono infine riapplicate le tecniche di Data Imputation di cui sopra, che convertono il dataframe in un array NumPy, problema facilmente aggirabile passando l'array come parametro alla funzione di creazione dataframe di Pandas. Per comodita' il dataframe e' poi stato splittato in:

- Un dataframe dal quale è stata **rimossa** la variabile dipendente
- Un dataframe contenente **soltanto** la variabile dipendente

Questo passaggio è necessario per evitare fenomeni di **data leakage** in fase di apprendimento.
Trattandosi di un dataset didattico, non sono state necessarie ulteriori manipolazioni come **Data Balancing** o **Feature Selection**.

2.5 Modelling

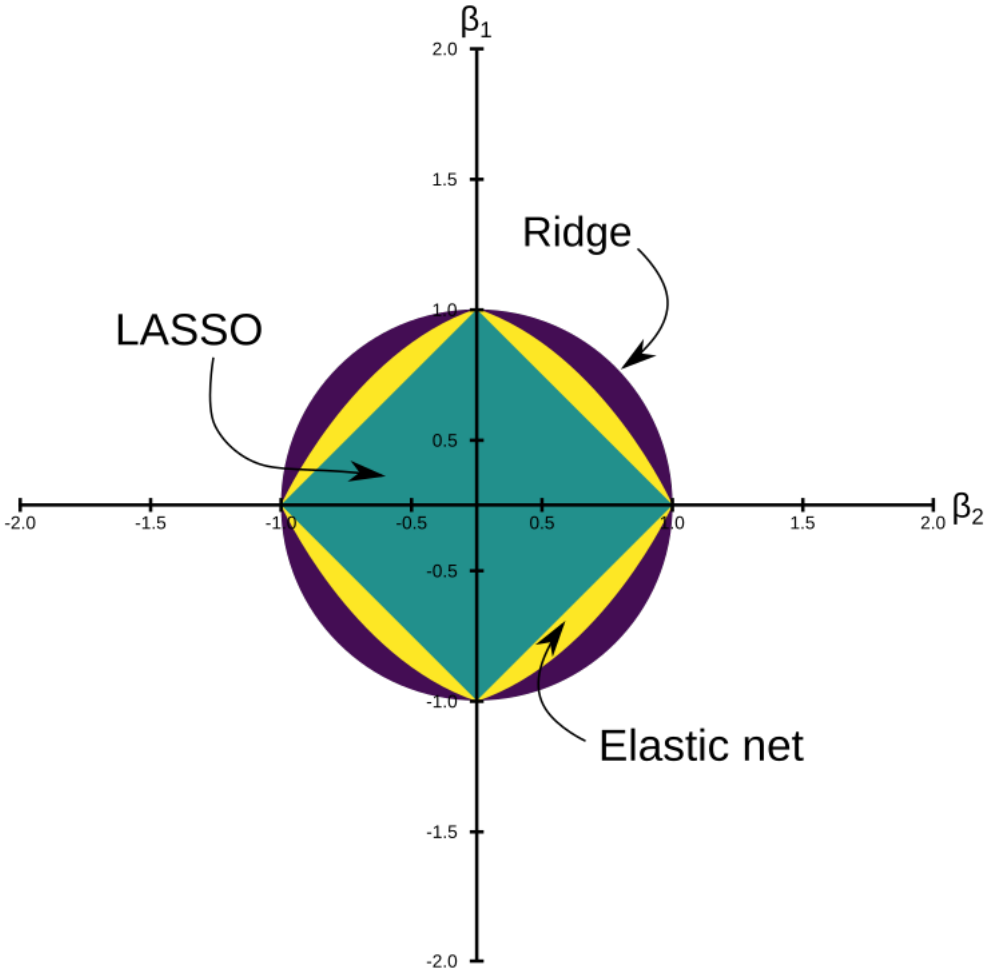
Si procede adesso a valutare il migliore approccio per risolvere il problema. Come già detto in precedenza, questo è un problema di regressione, si andranno quindi a valutare diversi algoritmi di regressione.
La regressione si può distinguere in **singola** e **multipla**. Nel caso in esame si parla di regressione multipla poichè la variabile dipendente viene predetta con l'ausilio di più variabili indipendenti.

Algoritmi testati
Dall'analisi dei dati e dal fatto che il dataset stesso sia stato costruito puramente a scopo didattico, si potrebbe subito intuire che il miglior regressore per la soluzione del problema sia quello lineare. Tuttavia noi siamo studenti inesperti, quindi non ci fidiamo troppo e proviamo altri algoritmi:

- ElasticNet
- Decision Tree Regressor
- Linear Regression

2.5.1 ElasticNet

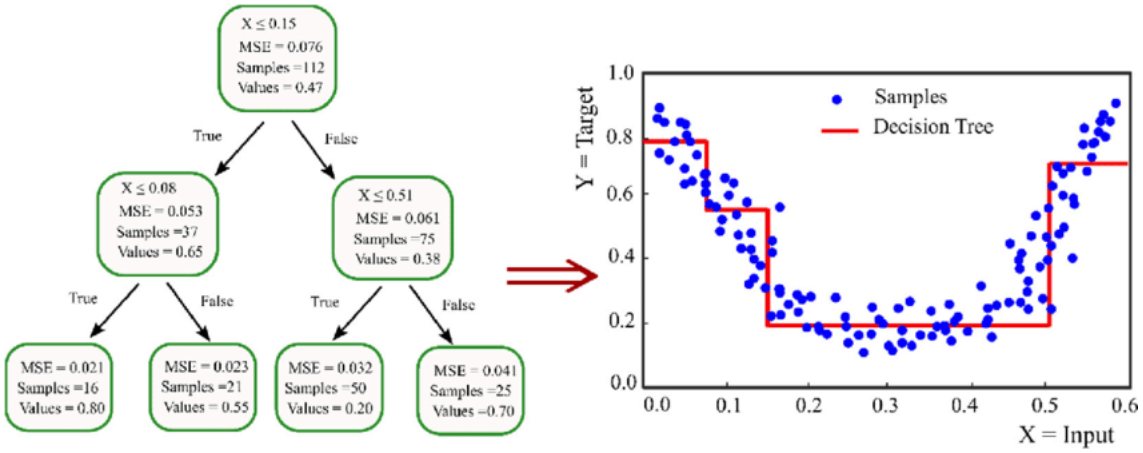
La regressione elastic net è una combinazione di due tecniche di regressione: la regressione **Lasso** (Least Absolute Shrinkage and Selection Operator) e la regressione **Ridge**.
La regressione Lasso utilizza una penalità L1 per selezionare automaticamente alcune caratteristiche e allo stesso tempo ridurre il valore dei coefficienti delle caratteristiche non selezionate a zero. Questo è utile per selezionare le caratteristiche più importanti, ma a volte può essere troppo aggressivo e generare una selezione di caratteristiche errata. Vista la modesta quantità di features per ogni smartphone, alcune di queste si sarebbero, probabilmente, potute scartare per portare ad un miglioramento delle performance.
La regressione Ridge utilizza una penalità L2 per limitare la grandezza dei coefficienti delle caratteristiche. Ciò consente di evitare overfitting, ma non seleziona automaticamente le caratteristiche.
La regressione elastic net combina le penalità L1 e L2 della regressione Lasso e Ridge. Ciò consente di ottenere i vantaggi della selezione automatica delle caratteristiche della regressione Lasso e della regolarizzazione dei coefficienti della regressione Ridge. Inoltre, l'utilizzo di entrambe le penalità consente di ottenere una soluzione intermedia tra la regressione Lasso e Ridge, evitando i loro svantaggi.
La regressione elastic net utilizza un parametro di equilibrio (chiamato **"alpha"**) per controllare la misura in cui la penalità L1 e L2 vengono utilizzate. Un alpha maggiore significa una maggiore enfasi sulla penalità L1, mentre un alpha minore significa una maggiore enfasi sulla penalità L2.



2.5.2 Decision Tree Regressor

È un modello che utilizza un **albero di decisione** per effettuare previsioni sulla base dei valori di alcune caratteristiche. L'albero di decisione è costruito mediante un processo di suddivisione ricorsiva dei dati in base al valore delle caratteristiche. Ogni nodo interno dell'albero rappresenta una caratteristica e ogni foglia rappresenta un valore predetto.
Il processo inizia dal nodo **radice**, che rappresenta l'intero dataset. Successivamente, l'algoritmo seleziona la caratteristica che suddivide i dati in modo migliore in due sottoinsiemi. Per questa selezione non è, chiaramente, possibile utilizzare l'**information gain** come per i problemi di classificazione, viene quindi utilizzato l'errore quadratico medio $(MSE)^2$. Questa caratteristica viene poi utilizzata come condizione di test al nodo interno. I sottoinsiemi di dati vengono quindi passati ai nodi figli sinistro e destro e il processo viene ripetuto fino a quando non viene soddisfatto un **criterio di arresto**. Il criterio di arresto può essere un numero minimo di campioni per foglia, una profondità massima dell'albero o un miglioramento minimo nella qualità delle suddivisioni.

Il regressore decision tree è un modello in grado di gestire relazioni non lineari tra caratteristiche e target, quindi non particolarmente indicato nel caso in esame.
Funziona anche bene con caratteristiche sia categoriche che numeriche. Tuttavia, è un modello suscettibile a problemi di overfitting se l'albero viene fatto crescere troppo profondamente. Possono essere utilizzate tecniche di **potatura** per evitare overfitting rimuovendo i rami che non migliorano l'accuratezza del modello.



2.5.3 Linear Regression

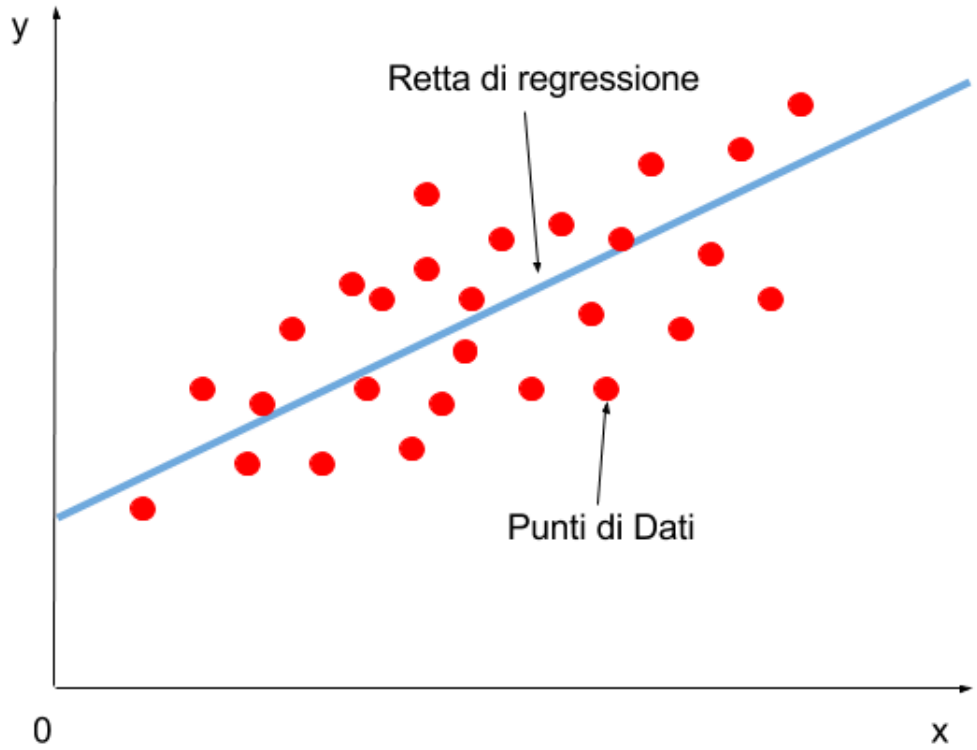
La regressione lineare è un metodo statistico utilizzato per individuare la relazione tra una variabile indipendente (x) e una o più variabili dipendenti (y). Viene utilizzato per costruire una linea di regressione, che rappresenta il modello matematico che descrive la relazione tra x e y. La regressione lineare è utilizzata in molti campi, tra cui l'economia, la finanza e l'ingegneria.
Per il caso in esame risulta essere l'algoritmo di regressione piu' efficiente, e quello che in generale approssima meglio la nostra funzione.

I vantaggi della regressione lineare includono:

- **Facilità di interpretazione**: i coefficienti della regressione possono essere facilmente interpretati come cambiamenti delle risposte in relazione ai cambiamenti delle variabili indipendenti.
- **Affidabilità**: la regressione lineare è un modello affidabile quando i dati soddisfano le ipotesi di linearità e di indipendenza delle osservazioni.
- **Semplicità**: la regressione lineare è un modello semplice e facile da utilizzare.

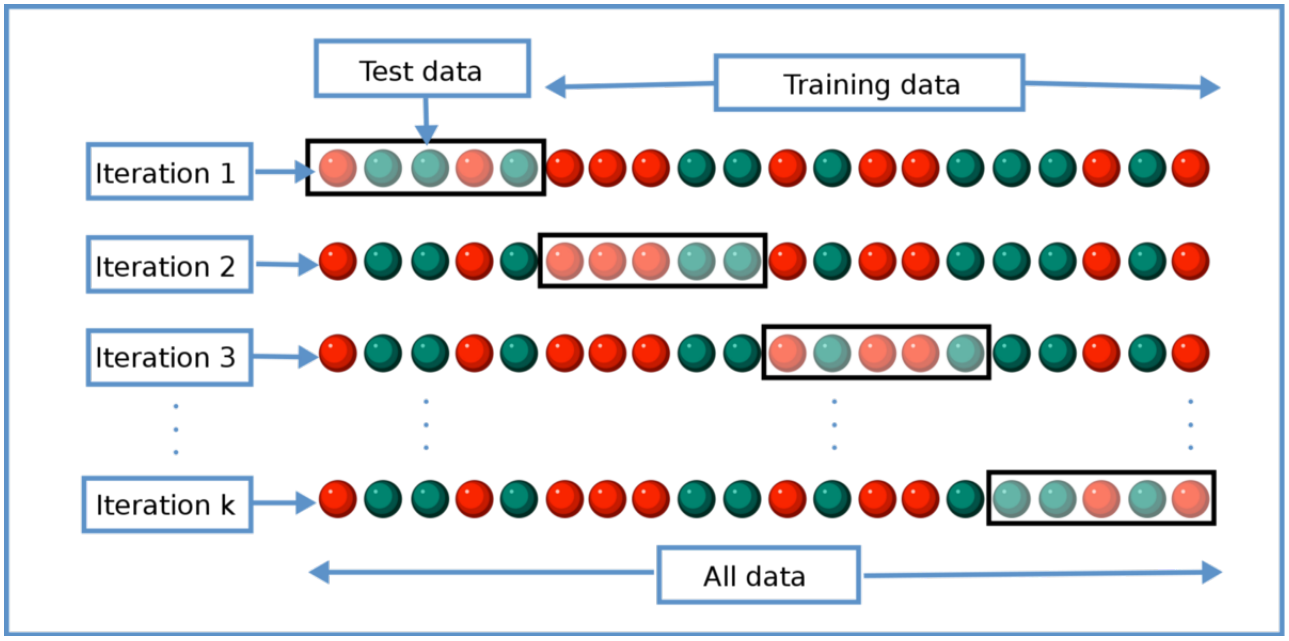
Gli svantaggi della regressione lineare includono:

- **Assunzioni di linearità**: la regressione lineare richiede che i dati soddisfino l'ipotesi di linearità, il che può non essere sempre il caso nei dati reali.
- **Indipendenza delle osservazioni**: la regressione lineare richiede che le osservazioni siano indipendenti.
- **Non adatta alle relazioni non lineari**: la regressione lineare è limitata alle relazioni lineari tra le variabili.



2.6 Validazione

La **cross-validation** è una tecnica utilizzata per valutare la performance di un modello di machine learning su un insieme di dati di test indipendente. Consiste nel dividere i dati di input in più parti (chiamate "fold"), addestrare il modello su una parte di essi e testarlo sull'altra parte.
Questo processo viene ripetuto più volte, utilizzando una diversa combinazione di fold come set di addestramento e di test ogni volta, e i risultati vengono quindi combinati per ottenere una stima più precisa della performance del modello. A seconda del numero **k** di fold scelto, la codifica prende il nome di **k-fold cross validation**.
In questo caso si è scelto di usare una 10-fold cross validation.



2.7 Valutazione dei modelli

Una volta addestrati i modelli, sono necessarie delle metriche oggettive per verificarne la bontà, sono quindi state utilizzate:

- MAE
- MSE
- R^2

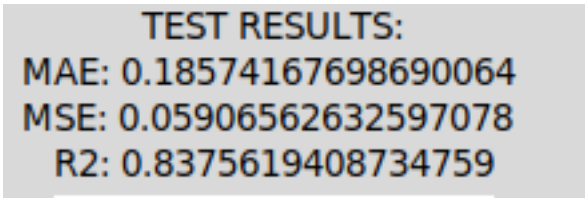
MAE: Questa misura la differenza media assoluta tra i valori previsti e i valori effettivi. Più precisamente, per ogni punto di dati, si calcola la differenza tra la previsione e il valore reale, si prende il valore assoluto di questa differenza e poi si calcola la media di tutte queste differenze. Più piccola è la MAE, migliore è la precisione del modello.

$$MAE = \frac{\sum_{i=1}^{n}(\tilde{y_i} - y_i)}{n}$$

MSE: Essa misura la differenza media quadratica tra i valori previsti e i valori effettivi. Quindi, per ogni punto di dati, si calcola la differenza tra la previsione e il valore reale, si eleva al quadrato questa differenza e poi si calcola la media di tutte queste differenze. La MSE è spesso utilizzata perché penalizza maggiormente gli errori rispetto ad una metrica come la MAE. Più piccola è la MSE, migliore è la precisione del modello.

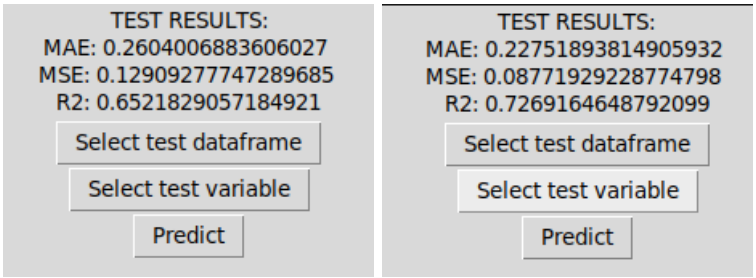
$$MSE = \frac{\sum_{i=1}^{n}(\tilde{y_i} - y_i)^2}{n}$$

R^2 : L'indice R-squared è una metrica comunemente utilizzata per valutare la bontà di adattamento di un modello di regressione. Esso indica la percentuale di varianza nei dati spiegata dal modello.
L'indice R^2 varia tra 0 e 1, dove un valore di 1 indica che il modello spiega perfettamente la varianza nei dati, mentre un valore di 0 indica che il modello non spiega affatto la varianza nei dati. In generale, un indice R^2 più alto è preferibile, poiché significa che il modello spiega una maggiore percentuale della varianza nei dati. Tuttavia, è importante notare che un alto valore di R^2 non garantisce necessariamente che il modello sia adatto per la previsione o la spiegazione di fenomeni futuri.
Infatti l'indice R^2 potrebbe essere influenzato da diversi fattori quali: **Overfitting, Non linearità, Multicollinearità e Outlier**.



Come si vede dal test, utilizzando la **regressione lineare** si ottengono MAE ed MSE piuttosto bassi, mentre l'indice R^2 è molto vicino a 1, questo ci dice che il modello così costruito riesce ad approssimare particolarmente bene i dati.

La stessa cosa non accade invece se utilizziamo rispettivamente **ElasticNet** oppure un **Decision Tree Regressor**.



2.8 Bonus

Per la creazione di questo progetto sono state impiegate diverse Intelligenze Artificiali. Principalmente questa scelta è stata fatta per rispondere ad una domanda che ultimamente preoccupa molte persone: **Le IA metteranno fine al ruolo dell'informatico così come lo conosciamo?**

In particolare quelle provate sono:

- ChatGPT
- Github Copilot
- LogoAI e Zyro IA Upscale
- DALL·E 2

Per quanto riguarda **ChatGPT**, questa è stata utilizzata prevalentemente per ottenere definizioni formali che racchiudessero un concetto chiave. Si è rivelato inoltre un ottimo tool per rispondere a domande "stupide" senza dover disturbare inutilmente colleghi o professori. Le sue risposte sono risultate quasi sempre corrette, con un margine minimo di errore facilmente integrabile attraverso una ricerca su libri o altre fonti.
Github Copilot è stato utilizzato per apprendere velocemente il linguaggio Python, con il quale l'autore di questo progetto non aveva mai scritto neanche una riga di codice. I suoi suggerimenti sono tuttavia piuttosto imprecisi, tende infatti a generare codice molto confusionario e non sempre coerente con quanto scritto in precedenza. Nonostante ciò ha rappresentato un notevole boost in velocità dal punto di vista dell'apprendimento, arrivando ad essere nelle fasi finali del progetto un'ottima alternativa al completamento automatico integrato nell'IDE.
LogoAI è stato molto banalmente impiegato per generare il logo utilizzato in pagina [1]. Un tool molto utile per progetti indipendenti che necessitano di un logo senza la possibilità di assumere un grafico, ma non è neanche lontanamente paragonabile al lavoro di creatività umana che potrebbe effettuare una persona esperta. Genera inoltre loghi a bassissima risoluzione se non si aderisce ad un piano a pagamento, ma il problema è facilmente aggirabile facendo uno screenshot, ripulendolo dai banner con Photoshop ed usando tool di **upscaling** come quello integrato in **Zyro**.
DALL·E 2 infine è stato utilizzato per generare alcune immagini utilizzate per le slide di presentazione.

Per rispondere dunque alla domanda, allo stato attuale non mi sento di dire che le IA possano minacciare **neanche minimamente** il ruolo dell'informatico, poichè questo risulta essere l'unico capace di dargli i giusti input e di portarle a rispondere in un certo modo (nonchè l'unico capace di capire come funzionano dietro le quinte). Ritengo però che queste possano essere, almeno al momento della stesura di questa documentazione, ottimi tool per gestire task ripetitivi, lenti e noiosi, ma ancora di più che possano essere integrate agli strumenti tradizionali per velocizzare notevolemente la fase di **apprendimento e verifica delle informazioni**.