Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

Analisi Predittiva del Successo Accademico: Un Approccio di Machine Learning

Relatore:
Chiar.ma Prof.ssa
Elena Loli Piccolomini

Presentata da:
Simone Magli

Sessione II
Anno Accademico 2024-2025

Contents

Introduzione

Riuscire a prevedere il rendimento accademico degli studenti è un tema che sta suscitando sempre più interesse sia all'interno delle organizzazioni scolastiche, sia all'interno del campo dell'analisi dei dati. Il GPA (Grade Point Average) è un indicatore della performance universitaria, prevalentemente usato oltreoceano, più precisamente negli Stati Uniti. Come però ci viene spesso detto, il rendimento scolastico non dipende solamente dalle abilità cognitive, ma anche da una moltitudine di fattori legati allo stile di vita, come per esempio, il numero di ore dedicate allo studio, la qualità e la quantità del sonno, il livello di stress, le ore di attività fisica. Perciò riuscire ad avere modelli predittivi che si basano su queste informazioni e vanno a trovare dei pattern comportamentali da cui deriva poi una stima di risultati accademici può essere un indicatore molto importante per l'evoluzione delle istituzioni scolastiche. I motivi per cui studiare questi modelli sono principalmente due: da un lato della medaglia abbiamo l'ottenimento dello strumento in se da poter utilizzare in campo educativo, dall'altro lato abbiamo la possibilità di andare ad aiutare gli studenti attraverso interventi mirati al loro benessere per andare poi ad aumentare la loro riuscita scolastica, fornendo delle raccomandazioni precise e specifiche. Data questa prospettiva, il presente lavoro si colloca tra data science e psicologia dell'educazione, cercando di ottenere dei risultati convincenti dagli algoritmi di machine learning quando andiamo a modellizzare la relazione tra il GPA di uno studente e le sue abitudine quotidiane di vita, usando algoritmi più o meno complessi per cercare l'approccio migliore.

L'obiettivo di questa tesi è duplice: effettuare l'analisi del dataset denominato"student_lifestyle_dataset.csv" che sarà il nostro punto di inizio di cui andremo a studiare la struttura, la qualità e le relazioni tra le varie colonne, ovvero le variabili, contenute al suo interno attraverso tecniche di Exploratory Data Analysis. Il secondo obiettivo è quello di andare a predirre una variabile, nel nostro caso sarà il GPA attraverso l'uso di 3 algoritmi di predizione diversi: il primo è un modello semplice, ovvero la regressione lineare multipla; il secondo è il Random Forest Regressor, un algoritmo ensemble prima eseguito con i parametri di default e poi eseguito nuovamente dopo un tuning, il terzo è invece il Support Vector Regression (SVR) con kernel RBF prima sempre eseguito con i parametri di default e poi eseguito nuovamente dopo un tuning.

L'obiettivo è verificare se una maggiore complessità dei modelli di predizione, sia prima che dopo il tuning, si traduca in un incremento delle prestazioni

e di conseguenza in una maggiore precisione dei risultati rispetto ai modelli più semplici. I modelli verranno valutati e confrontati sulla base di metriche standard (R^2, MSE), attraverso l'analisi dei residui e attraverso dei grafici per dare anche una dimensione visiva di ciò che stiamo facendo.

Le domande di ricerca che questo lavoro si prefissa di rispondere sono le seguenti: Esistono relazioni tra il GPA e le variabili che descrivono lo stile di vita di uno studente? Se sì, quali sono? Un modello di regressione lineare semplice, in che misura riesce a descrivere le relazioni individuate e a fornirne una predizione robusta e affidabile? Se andiamo ad utilizzare modelli più complessi, come Random Forest Regressor o Support Vector Regression, il fatto che siano appunto più complessi influisce sull'accuratezza della predizione?

Il progetto è stato realizzato in Python, utilizzando la piattaforma open-source Anaconda che semplifica la gestione dei pacchetti e anche degli ambienti virtuali. All'interno di Anaconda, come ambiente di sviluppo è stato scelto Spyder, siccome era già stato utilizzato precedentemente e anche per un discorso di continuità e coerenza è stata la scelta finale. Inoltre Spyder è particolarmente adatto quando si tratta di dover effettuare analisi numeriche in quanto proprio l'ambiente di sviluppo nella sua totalità ha molte funzioni utili come ad esempio la console in cui è possibile effettuare debug e visualizzare le prestazioni dei modelli usati oppure la sezione plots dove compariranno tutti i grafici che andremo a generare. All'interno del progetto sono state utilizzate anche alcune librerie molto comuni ed utilizzate, come ad esempio: pandas e numpy per la gestione dei dati; matplotlib e seaborn per la generazione dei grafici; scikit-learn usata per l'implementazione degli algoritmi di regressione e la validazione di essi. Unendo tutte queste cose il lavoro è stato sviluppato in un ambiente di lavoro ordinato e sicuro. All'interno del codice è possibile vedere ogni sezione separata dalle altre, in questo modo è possibile avere sempre un riferimento visivo alle fasi del progetto e che cosa sta venendo svolto in quel momento. Abbiamo perciò l'introduzione e la descrizione del dataset, poi l'analisi esplorativa dei dati,implementazione dei modelli di predizione e confronto dei loro risultati attraverso diverse metodologie. Infine i risultati vengono discussi e si arriva a dedurre una valutazione critica dell'approccio adottato per lo sviluppo complessivo del progetto.

Chapter 1

Studio dell'EDA e Splitting

1.1 Descrizione del dataset

Il dataset che è stato scelto, denominato `student_lifestyle_dataset.csv`, contiene al suo interno informazioni su specifiche parti della vita e delle abitudini di vari studenti in modo che si possa andare ad analizzare come esse hanno impatto sul rendimento accademico. Il rendimento accademico indicato come **GPA (Grade Point Average)** è la variabile dipendente che rappresenta la media ponderata dei voti ottenuti in materie diverse e si calcola attraverso la seguente formula:

$$GPA = \frac{\sum_{i=1}^{n}(Voto_i \times Crediti_i)}{\sum_{i=1}^{n} Crediti_i}$$

dove abbiamo $Voto_i$ che è il punteggio ottenuto nella materia e $Crediti_i$ è il peso associato alla materia stessa. Fornisce una rappresentazione numerica dei risultati accademici di uno studente, in genere su una scala da 0 a 4.0, in cui ogni voto in lettere corrisponde a un valore in punti specifico. [1] All'interno del dataset abbiamo tante variabili diverse che ci aiutano a comprendere la vita di ciascuno studente come:

- **Study_Hours_Per_Day**: numero di ore di studio giornaliere.
- **Sleep_Hours_Per_Day**: numero di ore di sonno giornaliere.
- **Physical_Activity_Hours_Per_Day**: numero di ore di attività fisica giornaliere.
- **Stress_Level**: livello di stress che ogni studente percepisce (inizialmente una variabile categorica).

- **Social_Hours_Per_Day**: numero di ore di social network giornaliere.

Inoltre per identificare ogni studente, all'interno del dataset era presente anche la colonna *Student_Id* che forniva un ID univoco ad ogni studente, che è stata eliminata in quanto per il nostro progetto avere un riconoscimento per ogni studente non è necessario.

1.2 Pre-Processing dei dati

Il pre-processing è la fase iniziale in cui andiamo a ispezionare il dataset e a prepararlo per i futuri passaggi. Perciò ho seguito i seguenti passi:

1. **Gestione dei valori mancanti:** tutte le righe all'interno del dataset che avessero come valore `NaN` sono state eliminate attraverso la funzione `dropna()`. Questo ci permette di avere già subito un dataset che contiene solo ed esclusivamente valori reali che possono essere utilizzati, anche se ciò però comporta ad una riduzione dei campioni totali.

2. **Rimozione delle variabili non utili:** come detto in precedenza la colonna `Student_Id` è stata rimossa dal dataset con il comando `drop(columns=['Student_ID'])` perchè non aveva nessun valore a livello predittivo.

3. **Codifica delle variabili categoriche:** la variabile categorica *Stress_Level* è stata trasformata utilizzando il **one-hot encoding**, che consiste nel prendere una variabile categorica ed andare a generare variabili binarie per ogni valore che assumeva (nel nostro ci saranno variabili con i seguenti nominativi: *Stress_Level_High,Stress_Level_Medium, ecc.* con valore 1 quando si tratta del livello di stress che lo studente possiede, 0 negli altri casi).

```
# ==============================
# FASE 2 - PULIZIA
# ==============================
# Rimuovo righe con valori mancanti
df_cleaned = df.dropna()

# Rimuovo ID (inutile per la predizione)
df_cleaned = df_cleaned.drop(columns=['Student_ID'])

# One-hot encoding per Stress_Level (variabile categorica)
df_cleaned = pd.get_dummies(df_cleaned, columns=['Stress_Level'], drop_first=False)

print(f"Numero di righe dopo la pulizia: {df_cleaned.shape[0]}")
```

1.3 Analisi esplorativa dei dati (EDA)

L'Exploratory Data Analysis o più semplicemente EDA è il processo che prevede l'utilizzo di riepiloghi numerici e visualizzazioni per studiare i dati e identificare potenziali relazioni tra le variabili.

Si tratta di un processo investigativo che si avvale del riepilogo statistico e di strumenti grafici per studiare i dati e capire che cosa si può trarne.

L'EDA consente di rilevare anomalie nei dati, come outlier od osservazioni insolite, identificare pattern, capire le potenziali relazioni tra le variabili e formulare domande o ipotesi interessanti che possono essere verificate in seguito usando metodi statistici più formali.

L'analisi esplorativa dei dati è simile a un'attività di indagine: si cercano indizi e informazioni che possano portare all'identificazione delle cause principali del problema che si vuole risolvere. Inizialmente si studia una sola variabile alla volta, poi se ne studiano due e infine molte contemporaneamente.[2]

Statistiche descrittive

Attraverso il comando `describe()` sono state calcolate le statistiche descrittive di due variabili in particolare ovvero **Study_Hours** e **GPA**. Vediamo che sono stati descritti i parametri:*count, mean, std(deviazione standard), min, primo quartile, mediana, terzo quartile e max.* In questo modo otteniamo un brevissimo sguardo iniziale a quelle che sono le due variabili più rilevanti del dataset.

```
=== Statistiche descrittive ===
       Study_Hours_Per_Day  ...          GPA
count          2000.000000  ...  2000.000000
mean              7.475800  ...     3.115960
std               1.423888  ...     0.298674
min               5.000000  ...     2.240000
25%               6.300000  ...     2.900000
50%               7.400000  ...     3.110000
75%               8.700000  ...     3.330000
max              10.000000  ...     4.000000
```

1.3.1 Distribuzioni univariate

Per le variabili più interessanti (*GPA, Study_Hours_Per_Day, Sleep_Hours_Per_Day, Physical_Activity_Hours_Per_Day* sono stati generati due tipologie di grafici che le vanno a descrivere:

- **Istogramma:** ci aiuta a capire i valori della variabile che assume all'interno del dataset.
- **Boxplot:** ci consente di visualizzare la mediana, l'intervallo interquartile e gli outlier.

Distribuzione del GPA

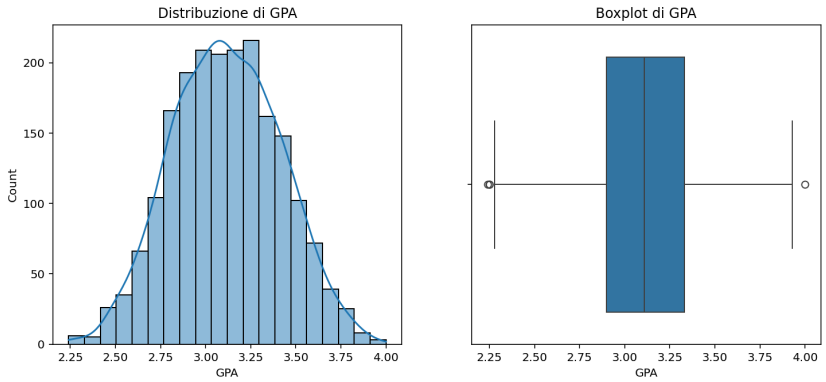


Figure 1.1: Distribuzione del GPA: istogramma con curva di densità (sinistra) e boxplot (destra).

Istogramma

Dal grafico notiamo che la distribuzione del GPA si concentra maggiormente nell'intervallo che è compreso tra i valori 2.8 e 3.5 con un picco che coincide con il valore 3.25. La distribuzione richiama una campana gaussiana lievemente asimmetrica verso sinistra.

Boxplot

La mediana si posiziona nell'intervallo dove abbiamo il picco di valori infatti ha un valore vicino al 3.10. L'intervallo interquartile è ristretto andando ad indicare presenza di GPA molto simile tra gli studenti. Notiamo la presenza di outlier specialmente nella parte bassa il che sta a indicare la presenza di alcuni studenti con un GPA molto basso. I whisker infine sono molto ampi e vanno a ricoprire praticamente tutti i valori.

Interpretazione

Il campione è caratterizzato da una maggior parte di studenti con performance medio-alte e pochi casi di rendimento medio-basso. Il che ci fa presupporre che il campione di dati preso rappresenti studenti che posseggono ottimi rendimenti scolastici.

Distribuzione delle ore di studio giornaliere

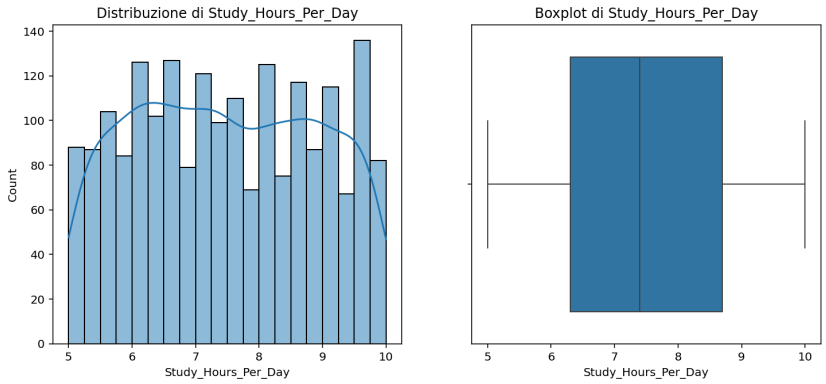


Figure 1.2: Distribuzione delle ore di studio giornaliere: istogramma con curva di densità (sinistra) e boxplot (destra).

Istogramma
Le ore di studio giornaliere si collocano in un intervallo che va dalle 5 alle 10 ore. Il grafico è molto irregolare e presenta molte oscillazioni di valori con picchi isolati tra di loro.
Boxplot
La mediana si trova intorno alle 7 ore e mezza di studio, l'intervallo interquartile è abbastanza ampio coprendo un range di quasi 3 ore indicando appunto i diversi picchi. Mentre i whisker come per il GPA coprono quasi tutti i valori e non sono stati registrati outlier.
Interpretazione
Il fatto di avere un grafico molto irregolare è sintomo delle diverse abitudini degli studenti, infatti c'è chi ha bisogno di più tempo di studio e chi invece ha bisogno di meno tempo. Guardando il boxplot però possiamo dire che comunque le ore di studio si concentrano maggiormente tra le 6 ore e mezza e le 9 ore giornaliere.

Distribuzione delle ore di sonno giornaliere

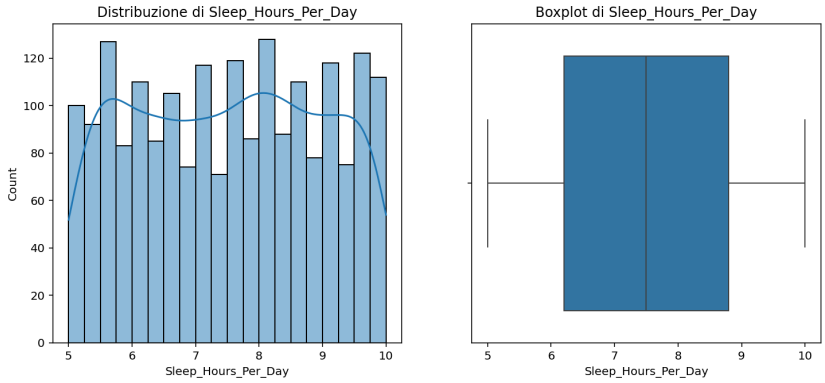


Figure 1.3: Distribuzione delle ore di sonno giornaliere: istogramma con curva di densità (sinistra) e boxplot (destra).

Istogramma
Così come le ore di studio, le ore di sonno hanno un grafico irregolare con picchi isolati che coprono tutte le fasce di sonno che vanno dalle 5 alle 10 ore.
Boxplot
Come il boxplot delle ore di studio la mediana si trova in un valore approssimativo di 7.5 ore, che è un valore che ci aspettavamo. L'intervallo interquartile invece restringe il range dalle 6 alle 9 ore. I whisker coprono quasi la totalità dei dati e non vengono registrati outlier.
Interpretazione
Gli studenti presentano abitudini di sonno diverse ed eterogenee rimanendo però in un range di sonno che coincide con quello fisiologico. Possiamo quindi dedurre che nonostante le ore di sonno siano variabili rimangono comunque entro un limite di "normalità" per la maggior parte di essi.

Distribuzione delle ore di attività fisica giornaliere

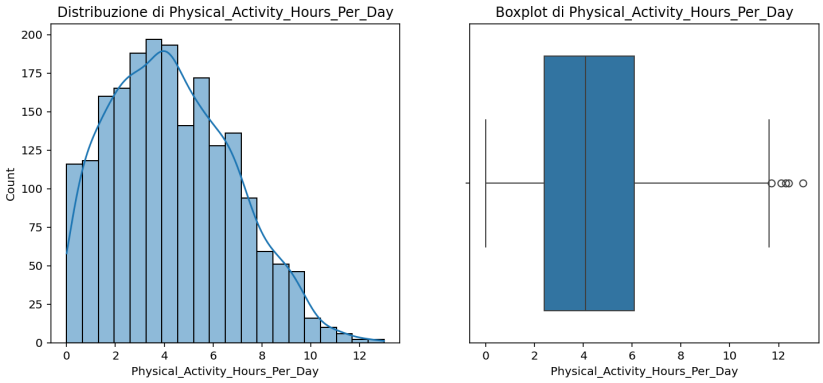


Figure 1.4: Distribuzione delle ore di attività fisica giornaliere: istogramma con curva di densità (sinistra) e boxplot (destra).

Istogramma
In questo grafico come era lecito aspettarsi abbiamo una simmetria particolarmente evidente verso sinistra, infatti il picco delle ore di attività fisica va dalle 2 alle 4, con un altro picco un po' più isolato sulle 6 ore per poi decrescere in modo abbastanza veloce.
Boxplot
Il boxplot riflette bene la situazione dell'istogramma infatti il suo range interquartile copre dalle 2 alle 6 ore con una mediana che si colloca sulle 4 ore. I whisker coprono anche in questo caso la quasi totalità dei valori. A sorpresa però troviamo tanti outlier che praticano più di 10 ore di attività fisica, dato che possiamo provare ad interpretare come studenti con attività fisiche intensive o persino atleti agonisti.
Interpretazione
Il campione presenta molti studenti che praticano attività fisica, anche con abbastanza ore giornaliere a carico. La parte sicuramente particolare qua e da tenere in particolare attenzione è la presenza degli outlier.

1.3.2 Distribuzioni bivariate

Una volta effettuate le analisi sulle variabili in modo singolo, il passaggio successivo è stato quello di combinare due variabili tra di loro e vedere l'interazione tra esse. Abbiamo perciò utilizzato, come fatto in precedenza, due tipologie di grafici:

- **Doppio boxplot:** ci servirà per analizzare la stessa variabile in due contesti diversi

- **Violinplot:** è un grafico che combina il boxplot con la densità dei dati per certe soglie.

Distribuzione del GPA in base al livello di stress

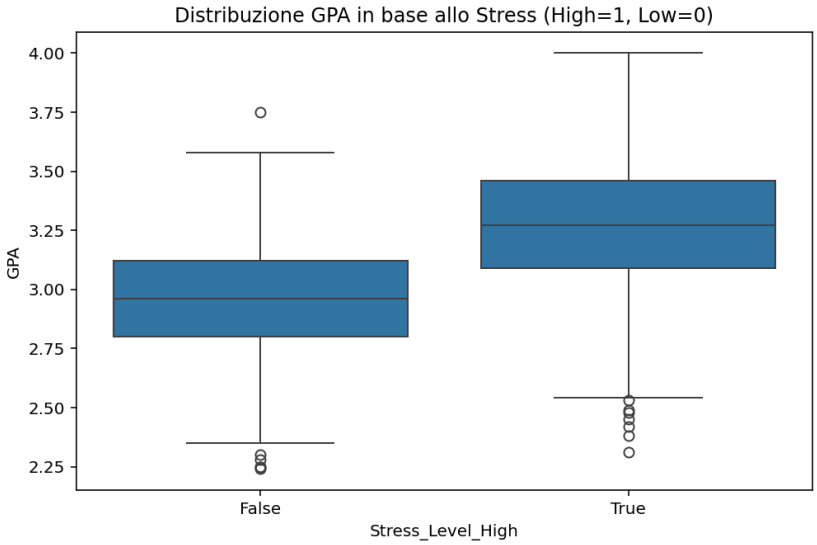


Figure 1.5: Distribuzione del GPA in base al livello di stress (alto = 1, basso = 0).

Boxplot (sinistra)
Il boxplot di sinistra fa riferimento a come cambia il GPA quando gli studenti non hanno un alto livello di stress. Possiamo notare che la mediana si sistema ad un valore di poco inferiore al 3, il range interquartile è abbastanza ristretto perciò ci sono dati abbastanza omogenei, i whisker vanno da una soglia di 2.35 ad una soglia di 3.60 senza però ricoprire tutti i dati o la maggior parte di essi e abbiamo degli outlier che si concentrano per la maggior parte nella parte inferiore, andando ad indicare che con lo stress medio/basso gli studenti che hanno risultati diversi dalla media tendono ad averli diversi in negativo piuttosto che in positivo siccome c'è un solo outlier che lo mostra.
Boxplot (destra)
In questo caso vediamo l'impatto di uno stress alto sul GPA che sembra dare più risultati, difatti la mediana si posiziona ad un valore di 3.25, il range interquartile è poco più largo indicando una maggiore diversità nei dati e i whisker coprono un GPA che va da 4 a 2.50. Per quello che invece riguarda gli outlier vediamo che si posizionano tutti verso il basso.
Interpretazione
In entrambi i grafici vediamo che la tendenza degli outlier è di stabilizzarsi

verso il basso, inoltre una particolarità che poteva non essere facilmente pronosticabile è il fatto che mediamente il GPA di studenti sotto stress alto sia più alto rispetto a chi è meno stressato, probabilmente lo stress è derivante dalle maggiore ore di studio.

Distribuzione del GPA per fasce di studio

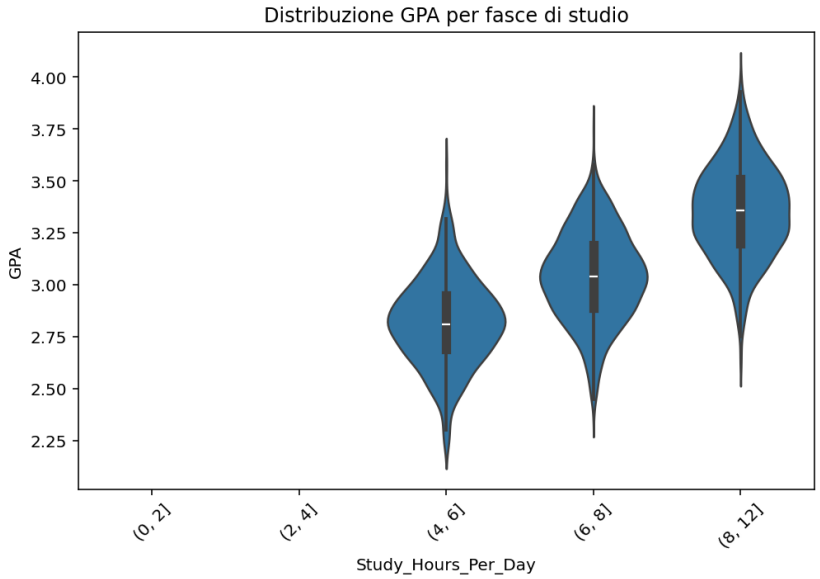


Figure 1.6: Distribuzione del GPA per fasce di ore di studio giornaliere.

Violinplot

Possiamo vedere come varia il GPA all'aumentare delle fasce di studio raccolte per coppie di ore. Vediamo che la mediana cresce in modo progressivo al crescere delle ore di studio, si passa da un valore di 2.8 nella fascia (4-6) ad un valore di 3.4 nella fascia (8-12). Nonostante ciò vediamo che la maggiore densità dei dati la troviamo nella fascia con GPA 2.8 dove il grafico è chiaramente più largo degli altri, per il secondo e il terzo invece si nota sempre un'ampiezza in corrispondenza della mediana ma meno evidente rispetto al primo.

Interpretazione

Notiamo che gli studenti che studiano più di 8 ore al giorno hanno maggiori risultati e più compattezza, ciò significa che un maggiore studio si traduce in migliori risultati con una bassa variabilità. Nelle fasce inferiori invece c'è maggiore dispersione.

Distribuzione del GPA per fasce di sonno

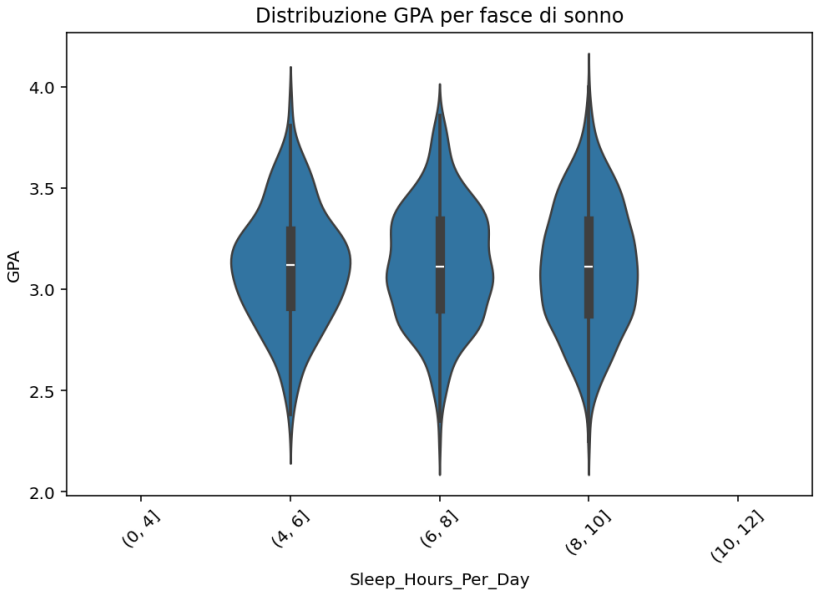


Figure 1.7: Distribuzione del GPA per fasce di sonno giornaliere.

Violinplot
Questo grafico mostra come varia il GPA al variare delle fasce di ore di sonno. I dati sono concentrati tra le 4 e le 10 ore di sonno, è subito evidente che chi dorme di meno ha performance più variabili rispetto a chi ne dorme di più. Tutte le varie fasce comunque non hanno impatto rilevante sul GPA.
Interpretazione
In generale la mediana non ha variazione con l'aumento delle ore di sonno, si intravede però una zona ottimale tra le 6 e le 8 dove il rendimento sembra essere meno variabile. Si può dire perciò che il sonno eccessivo o insufficiente influenza in modo negativo le prestazioni.

1.3.3 Distribuzioni multivariate

Per quello che invece riguarda le distribuzioni multivariate abbiamo usato diversi grafici per andare ad intercettare più relazioni tra le variabili:

- **Pairplot:** è un grafico a scacchiera dove vengono comparate tra di loro molte variabili.
- **Scatterplot:** di base un grafico di confronto tra due variabili, con l'aggiunta però dei vari livelli di stress.

- **Matrice di correlazione:** matrice che indica i vari livelli di correlazione tra le variabili del dataset.

Pairplot delle variabili numeriche

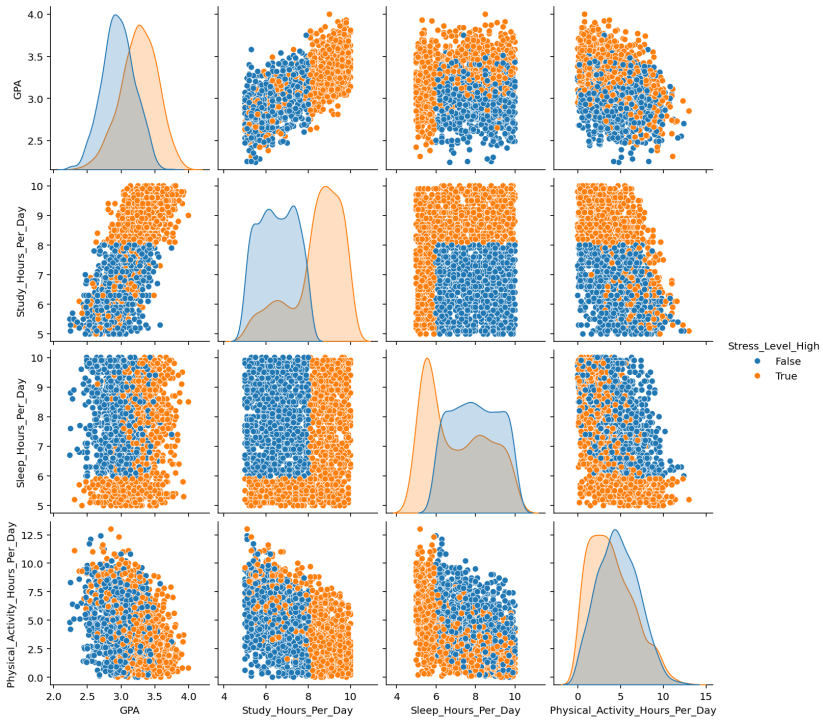


Figure 1.8: Pairplot delle variabili numeriche, colorato per livello di stress

Interpretazione

Il pairplot ci mostra in generale come il livello di stress condiziona i comportamenti degli studenti: notiamo che quelli con maggiore stress tendono a studiare di più e a dormire di meno e come conseguenza di ciò ottengono un GPA leggermente maggiore. Come già mostrato in precedenza anche il pairplot mostra una relazione positiva tra le ore di studio e il rendimento, mentre le altre variabili risultano trascurabili facendoci intendere la possibile presenza di relazioni non lineari.

Relazione tra GPA e attività fisica

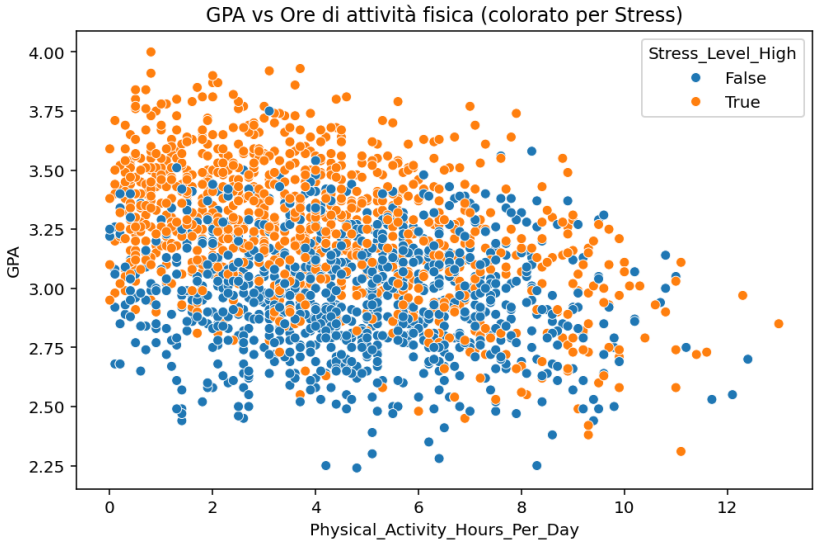


Figure 1.9: Distribuzione del GPA e attività fisica considerando anche il livello di stress.

Scatterplot

Il grafico ci mostra la relazione che c'è tra le ore giornaliere dedicate all'attività fisica e il GPA degli studenti, con l'aggiunta dei punti colorati in modi diversi che vanno ad indicare il livello di stress dello studente (colore arancione = stress alto, blu = stress basso). I dati appaiono piuttosto dispersi senza che delineino una chiara correlazione tra attività fisica e rendimento. La maggior parte degli studenti come già assunto in precedenza pratica dalle 0 alle 8 ore di attività fisica con valori di GPA che si collocano tra il 2.8 e il 3.5. Chi però ha GPA maggiore ha anche uno stress alto e questo lo si nota molto chiaramente dal colore dei pallini.

Interpretazione

Possiamo dire che l'attività fisica non ha un diretto impatto sul GPA . Nonostante ciò possiamo comunque fare una deduzione importante, che ci va a confermare le ipotesi fatte con i grafici precedenti: esiste un'effettiva differenza di rendimento tra gli studenti con più o meno stress, inoltre un alto livello di stress se combinato in maniera giusta con l'attività fisica sembra portare dei grandi benefici.

Matrice di correlazione

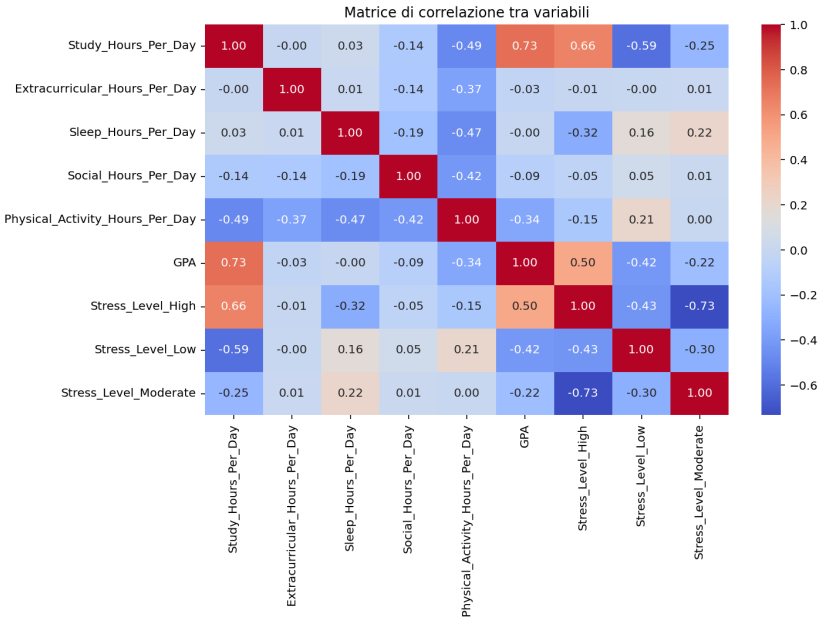


Figure 1.10: Matrice di correlazione del dataset

Interpretazione
La matrice di correlazione funge da conferma finale di tutto quello che è stato ipotizzato durante l'EDA. Si evince facilmente come le uniche due correlazioni forti con il GPA siano *Study_Hours_Per_Day* e *Stress_Level_High*, mentre le relazioni con *Sleep_Hours_Per_Day* e *Physical_Activity_Hours_Per_Day* non abbiano correlazione positiva con il GPA così come emerso durante tutto lo studio delle variabili del dataset.

1.4 Suddivisione del dataset e strategia di validazione

Per poter lavorare con i modelli di predizione è necessaria un'attenta e precisa preparazione dei dati. Il dataset complessivo è stato suddiviso in tre sottoinsiemi distinti chiamati *set*, ed ognuno di essi sarà usato e sarà fondamentale in diverse parti del progetto:

- **Suddivisione del Dataset**: Il dataset iniziale è stato suddiviso come segue:
  - **Training set** (70% dei dati): Utilizzato per addestrare i modelli e per l'ottimizzazione degli iperparametri (tuning).

  - **Validation set** (15% dei dati): Impiegato per il confronto delle prestazioni e la selezione preliminare tra i modelli.
  - **Test set** (15% dei dati): Riservato esclusivamente alla fase finale per ottenere una stima conclusiva, non contaminata, delle predizioni su dati mai visti.

- **Strategia di Validazione e Uso dei Set**: Per trovare gli iperparametri migliori per i modelli complessi sono state utilizzate tecniche di **cross-validation** applicate esclusivamente al **training set**.

- **Valutazione e Riserva**: Le prestazioni di tutti i modelli vengono calcolate e valutate in via preliminare sul **validation set**. Il **test set** è mantenuto rigorosamente separato e viene usato solo ed esclusivamente per la stima conclusiva, garantendo così un confronto finale non influenzato dal processo di selezione e tuning.

- **Analisi Finale**: Infine, sia il training set che il test set vengono utilizzati congiuntamente per l'analisi statistica conclusiva.

Questa metodologia di suddivisione e validazione è ottimale per i problemi di predizione poiché riduce significativamente il rischio di **overfitting da tuning** e fornisce una valutazione più realistica della robustezza e precisione dei modelli.

Conclusioni del Capitolo 2

L'analisi esplorativa che è stata fatta ci ha permesso di riconoscere alcune tendenze:

- Le **ore di studio** sono il predittore più potente di GPA, sia a livello di correlazione sia a livello di grafici.

- **Il sonno e l'attività fisica** hanno un legame debole se non quasi nullo, pertanto non possiamo considerarli buoni predittori.

- Il **livello di stress** degli studenti non agisce come predittore ma va a modificare e ad interagire con la vita e le abitudini degli studenti che hanno conseguenze più o meno dirette sul loro rendimento scolastico.

Per capire meglio le relazioni che ci sono all'interno del dataset e capirne la complessità è perciò necessario passare all'utilizzo di metodi predittivi che siano in grado di stimare il GPA sulla base di alcune variabili date in input e sulla base delle loro relazioni. Andremo perciò ad utilizzare tre approcci con un grado di complessità sempre crescente:

1. **Regressione lineare Multipla:** come modello di base e riferimento per i modelli successivi.
2. **Random Forest Regressor:** come modello intermedio capace però di catturare relazioni non lineari.
3. **Support Vector Regression (SVR):** come modello complesso capace sempre di catturare relazioni non lineari attraverso un kernel RBF.

Chapter 2

Algoritmi di predizione di machine learning

2.1 Modelli predittivi considerati

Per effettuare la predizione del rendimento accademico sono stati usati 3 modelli di machine learning diversi tra di loro per struttura e complessità intrinseca. Siamo così in grado di poter confrontare i vari approcci partendo da quelli semplici arrivando a quelli più complessi andando ad analizzare risultati, accuratezza e complessità.

- **Regressione Lineare Multipla**
  - Modelle di base, semplice e facilmente interpretabile.
  - Ci permette di fare una valutazione sull'effetto che hanno insieme più variabili predittive sul GPA.
  - Funziona come modello iniziale per poi confrontare le sue prestazioni con quelli degli altri algoritmi.
- **Random Forest Regressor**
  - Modello *ensemble* basato sulla combinazione dei risultati di vari alberi decisionali.
  - Essendo in grado di catturare relazioni non lineari ci permette di avere uno sguardo diverso alle relazioni tra variabili.
  - Generalmente offre prestazioni superiori ad una regressione lineare andando però a perdere la facilità di interpretazione di quest'ultima.
- **Support Vector Regression**

  - Modello *kernel-based*, è adatto a rappresentare relazioni difficilmente catturabili e complesse tra variabili indipendenti e la variabile target.
  - Richiede la standardizzione dei dati e una ricerca precisa per trovare gli iperparametri adatti.
  - Più flessibile della regressione lineare a costo però di un costo computazionale maggiore andando come il Random Forest Regressor a perdere di immediatezza nell'interpretazione.

Attraverso l'utilizzo di questi tre modelli andremo ad osservare se, e nel caso di riscontri positivi in che modo,la complessità del modello influisca sulle capacità predittive.

2.2 Metriche e criteri di confronto

Per valutare le prestazioni dei vari modelli utilizzati, vengono considerati vari indicatori numeri e grafici, in questo modo è possibile affiancare ai risultati numerici anche dei risultati visivi dei vari comportamenti predittivi.

- **Metriche scalari**
  - R^2**:** Il coefficiente di determinazione R quadro è un indice statistico legato alle tecniche di regressione dei dati. Molto spesso vogliamo cercare una formula matematica che permetta di trovare un'approssimazione del valore di una variabile target a partire dai valori di altre variabili di input dette regressori. Il coefficiente R quadro è uno dei modi possibili per valutare quanto queste approssimazioni sono accurate e quindi quale sarà il valore predittivo generale del modello di regressione.[3]
  - **MSE (Mean Squared Error):** L'errore quadratico medio (MSE) è una metrica ampiamente utilizzata in statistica e analisi dei dati che quantifica la differenza media al quadrato tra valori previsti e valori effettivi. Serve come misura della qualità di uno stimatore o di un modello predittivo, fornendo informazioni su quanto bene funziona il modello. Più basso è l'MSE, più vicino è l'adattamento del modello ai punti dati effettivi, rendendolo un componente cruciale nella valutazione delle prestazioni del modello in varie applicazioni, tra cui l'analisi di regressione e l'apprendimento automatico.[4]
- **Visualizzazioni diagnostiche**

- **Predicted vs Actual:** E' il confronto tra i valori osservati e quelli predetti attraverso la retta bisettrice. In questo modo è facile vedere la dispersione delle previsioni.
- **Residual plot:** Grafico che rappresenta i valori residui rispetto ai valori predetti.
- **Distribuzione dei residui:** analisi dei residui tramite istogrammi o curve di densità.
- **Barplot comparativi:** Grafico che mette a confronto diretto i vari modelli attraverso la rappresentazione grafica di R^2 e MSE.

In questo capitolo il nostro scopo sarà quello di andare ad analizzare il funzionamento degli algoritmi di machine learning che andremo ad usare per andare a predirre la nostra variabile target **GPA**.

2.3 Regressione Lineare Multipla

La regressione lineare multipla è una tecnica che viene utilizzata per prevedere il valore di una variabile in base al valore di un'altra variabile. La variabile che si desidera prevedere è chiamata variabile dipendente. La variabile che si utilizza per prevedere il valore dell'altra variabile è chiamata variabile indipendente. Questa forma di analisi stima i coefficienti dell'equazione lineare, coinvolgendo una o più variabili indipendenti che meglio prevedono il valore della variabile dipendente. La regressione lineare si inserisce su una linea retta o una superficie che riduce al minimo le discrepanze tra i valori di output previsti e quelli effettivi. Esistono semplici calcolatori di regressione lineare che utilizzano un metodo dei "minimi quadrati" per scoprire la linea più adatta per un insieme di dati abbinati. Quindi si stima il valore di X (variabile dipendente) da Y (variabile indipendente).[5] Nel nostro codice perciò questo modello andrà a cercare di prevedere il valore della variabile dipendente **GPA** andando ad utilizzare le altre variabili del dataset. Il modello esegue la predizione assumendo che ci sia una relazione lineare tra le variabili indipendenti e quella dipendente che si esprime attraverso la seguente formula:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

Dove Y è nel nostro caso il GPA, $[X_1], [X_2], \ldots, [X_p]$ sono le variabili indipendenti, β_0 è l'intercetta e $\beta_1, \ldots, \beta_p$ sono i coefficienti da stimare.[6] Il modello viene quindi allenato sul training set e usato per fare previsioni sul validation set. Le prestazioni che ottiene il modello vengono registrate attraverso i due

coefficienti accennati in precedenza: R^2, che ci indica quanto bene il modello riesce a spiegare la variabile dipendente, e l'errore quadratico medio (MSE), che invece misura la differenza media tra i valori predetti e i valori reali.

```
# Modello multivariato con tutte le feature
linreg_model = LinearRegression()
linreg_model.fit(X_train, y_train)

# Predizione sul validation set
y_lin_pred = linreg_model.predict(X_val)

print("\n=== Regressione Lineare Multivariata ===")
print("Coefficiente/i:")
for col, coef in zip(X_train.columns, linreg_model.coef_):
    print(f"  {col}: {coef:.4f}")
print(f"Intercetta: {linreg_model.intercept_:.4f}")

# Metriche
r2_lin = r2_score(y_val, y_lin_pred)
mse_lin = mean_squared_error(y_val, y_lin_pred)
print(f"R^2 (Validation): {r2_lin:.4f}")
print(f"MSE (Validation): {mse_lin:.4f}")

# Scatter plot Predicted vs Actual
plt.figure(figsize=(10, 6))
plt.scatter(y_val, y_lin_pred, color='blue', alpha=0.6, label='Predizioni')
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--', label='Perfetta predizione (y=x)')
plt.xlabel('Valori reali GPA')
plt.ylabel('Valori predetti GPA')
plt.title('Regressione Lineare Multivariata: Predicted vs Actual')
plt.legend()
plt.show()

# Analisi residui
residui_lin = y_val - y_lin_pred
plt.figure(figsize=(10, 6))
sns.histplot(residui_lin, kde=True, bins=20, color="blue")
plt.title("Distribuzione residui - Regressione Lineare Multivariata")
plt.xlabel("Residui")
plt.show()
```

Il codice soprastante è la parte del progetto dove si svolgono tutte le operazioni di predizione e conseguente generazione di grafici per mostrare i risultati e la stampa di essi in console. Andando ad analizzarlo più attentamente possiamo dire che le prime 3 righe sono il cuore del modello:

- *linreg_model = LinearRegression()*: questa riga di codice utilizzando il metodo LinearRegression importato da sci-kit learn va a creare un modello di regressione lineare.
- *linreg_model.fit(X_train, y_train)*: Questo passaggio è uno dei più importanti siccome prendiamo il modello appena creato ed usando il training set andiamo ad allenarlo con i dati che si trovano al suo interno. In questo passaggio vengono calcolati i coefficienti delle linee (le beta che troviamo nella formula vista precedentemente) che vanno a minimizzare l'MSE.

- *y_lin_pred = linreg_model.predict(X_val)*: In questa riga avviene la predizione, al nostro modello andiamo a passare tutte le variabili (esclusa ovviamente quella target che è *y_val*) del validation set, quindi su dati diversi da quelli su cui lo abbiamo fatto allenare. Ogni valore dell'array *y_lin_pred* contiene la stima del valore di y per un determinato esempio di *X_val*.

L'ultima parte del codice per cui vale la pena andare più a fondo è l'utilizzo delle due funzioni del package scikit-learn, più precisamente del modello *sklearn.metrics* per calcolare, appunto, le metriche del modello.

- *r2_lin = r2_score(y_val, y_lin_pred):* Come parametri della funzione vengono passati y_val che rappresenta i valori reali del validation set e y_lin_pred che invece rappresenta i valori predetti. La funzione calcola: la varianza totale dei dati e la somma dei quadrati tra i valori predetti e i valori veri. In seguito esegue la formula:

$$R^2 = 1 - \frac{\text{Somma degli errori quadrati}}{\text{Varianza totale}}$$

  che da come output un numero float che è compreso tra 0 e 1, con la possibilità di essere anche negativo, questo output più è vicino ad 1 più ci indica che il modello spiega bene, con il valore vicino allo 0 invece ci indica il contrario[7].

- *mse_lin = mean_squared_error(y_val, y_lin_pred):* anche qua abbiamo i due valori di input che hanno lo stesso lo stesso significato che hanno per la funzione r2_score(). La differenza dei due metodi, ovviamente, sta nel loro funzionamento: il primo passaggio è il calcolo della differenza tra ogni valore reale e il corrispondente valore che è stato predetto, questa differenza viene elevata al quadrato e infine viene fatta la media di tutti questi quadrati. L'output è un numero che indica quanto è grande l'errore perciò più è basso più il modello si comporta correttamente. La formula del calcolo dell'output è la seguente:

$$\text{MSE} = \frac{\text{Somma degli errori al quadrato}}{\text{Numero delle osservazioni}}$$

  [4]

Ora che abbiamo visto le parti principali del codice andiamo ad analizzare i risultati visivi e del terminale che descrivono la predizione.

```
=== Regressione Lineare Multivariata ===
Coefficiente/i:
  Study_Hours_Per_Day: 0.1248
  Extracurricular_Hours_Per_Day: -0.0412
  Sleep_Hours_Per_Day: -0.0287
  Social_Hours_Per_Day: -0.0269
  Physical_Activity_Hours_Per_Day: -0.0280
  Stress_Level_High: 0.0080
  Stress_Level_Low: 0.0117
  Stress_Level_Moderate: -0.0197
Intercetta: 2.6750
R^2 (Validation): 0.5251
MSE (Validation): 0.0462
```

Figure 2.1: Risultati del terminale della regressione lineare Multipla

I coefficenti che sono associati a ciascuna variabile rappresentano il peso medio della variabile nella previsione ovvero quanto il modello si aspetta che il GPA aumenti se la variabile aumenta di 1. L'unica variabile di cui vale la pena parlare è sempre quella che ci racconta le ore di studio giornaliere che ha un peso del 0.1248 sul GPA, ciò significa che un'ora in più di studio aumenterebbe di 0.12 il GPA dello studente. Tutte le altre variabili hanno pesi negativi o quasi nulli che non vale la pena analizzare. Per intercetta invece si intende il valore atteso del GPA quando tutte le altre variabili sono a 0 ed infatti il risultato è 2.6750 che è molto basso. La parte che è interessante andare ad analizzare sono le metriche di valutazione di cui abbiamo parlato prima.
R^2 ha un valore di 0.5251 il che ci dice che il modello riesce a spiegare circa il 52.5% della variabilità. In sostanza il modello riesce a spiegare poco più della metà delle differenze di voto negli studenti, l'altra metà dipende da fattori che non sono stati considerati e non sono stati inseriti nel modello come ad esempio il metodo di studio, la motivazione personale, la qualità degli insegnanti e molti altri ancora.
L'MSE invece ha come valore 0.0462 che è un valore basso, sta perciò ad indicare che le previsione che sono state fatte sono vicine ad i valori reali e gli errori commessi sono in generale di piccola portata.

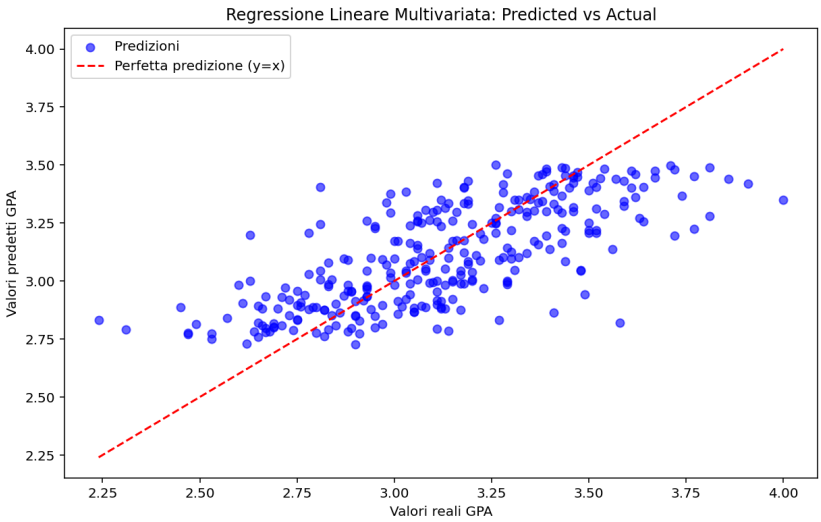


Figure 2.2: Regressione lineare Multipla - Predetti vs Reali

Osservazione dei risultati

Grazie alla legenda presente nel grafico, capiamo che i punti blu che vediamo sono tutte le predizioni che il modello ha fatto, di questi punti, tutti quelli che si trovano esattamente sulla retta bisettrice y=x sono le predizioni corrette, come possiamo notare il modello fatica a predirre perfettamente ma ha un ottimo risultato in quanto la densità delle predizione è notevolmente vicina alla retta di perfezione.

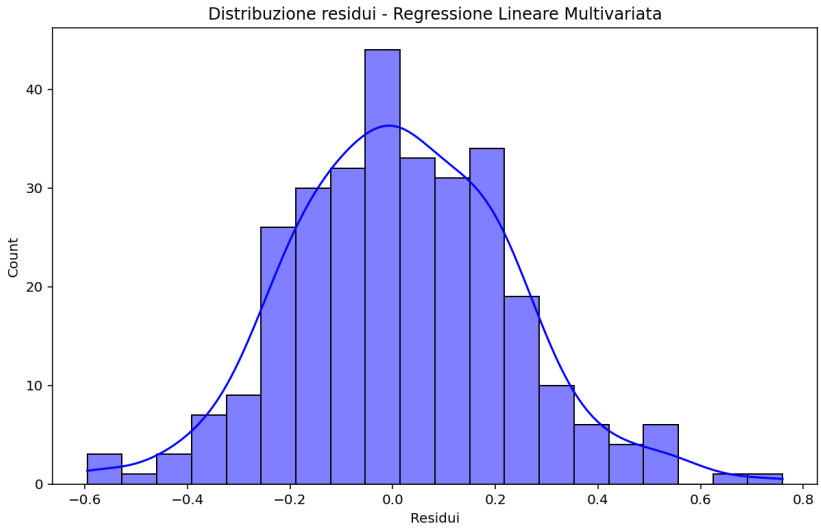


Figure 2.3: Regressione lineare Multipla - Analisi dei residui

Questo grafico mostra la distribuzione dei residui che sono definiti come le differenze tra i valori osservati effettivi e i valori previsti da un modello statistico. L'analisi di questi residui fornisce informazioni sulle prestazioni del modello, aiutando a determinare se cattura adeguatamente le tendenze dei dati sottostanti o se sono necessari[8]. Matematicamente, questo può essere espresso come:

$$\text{Residuo} = y_i - \hat{y}_i$$

dove y_i indica il valore reale e $\hat{y}_i$ il valore predetto. Una distribuzione ottimale dei residui implica che essi siano distribuiti in modo simmetrico in modo approssimativo attorno allo 0. Dal grafico notiamo che la distribuzione ha una forma regolare a campana, con il picco intorno allo 0 e un'assenza di valori estremi/outlier, ciò stando alle implicazioni base di interpretazione del grafico ci conferma ciò che ci aveva anticipato il risultato dell'MSE, infatti i risultati ottenuti indicano una buona capacità di adattamento senza particolari errori rilevanti.

2.4 Random Forest Regressor - Parametri di default

Il Random Forest è un algoritmo di tipo supervisionato che può essere usato sia per task di classificazione sia di regressione. Questo algoritmo, fa parte dei metodi ensemble, ovvero un modello che al proprio interno mette insieme altri modelli più semplici. Quando decido di utilizzare un Random Forest, adotto la teoria del "più teste sono meglio di una". Il mio obiettivo sarà infatti quello di creare tanti classificatori "deboli" che uniti insieme possano darmi un risultato eccellente. La caratteristica di questo algoritmo è che gli alberi sono indipendenti, non si condizionano a vicenda. Questo permette di evitare alcuni degli errori più comuni degli alberi: sappiamo infatti che gli alberi rischiano di essere molti instabili a causa del loro modo intrinseco di fare prediction, ovvero il metodo verticale "ad albero". Altra importante peculiarità è quella di scegliere ogni volta un campione casuale di variabili da utilizzare per il train interno, il che rende ancora più indipendenti (e differenti) i vari alberi, permettendo ad ognuno una diversa specializzazione. Questa tecnica implica che alcuni dati possono comparire contemporaneamente in più modelli mentre altri potrebbero non comparire mai.[9] Ogni albero esegue una sola predizione ed arriva ad un risultato che viene preso e inserito all'interno della formula:

$$\hat{Y}(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

che prende ogni predizione di ogni albero e fa la media. Il Random Forest Regressor ci permette di andare a catturare collegamenti complessi e relazioni non lineari tra variabili, inoltre come detto in precedenza la struttura a foresta di esso permette predizioni più stabili. Infine come per la regressione lineare andremo ad utilizzare come metriche di valutazione R^2 e MSE.

```
# ------------------------------
# Random Forest Regressor
# ------------------------------
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

y_rf_pred = rf_model.predict(X_val)

print("\n=== Random Forest Regressor ===")
print(f"R^2: {r2_score(y_val, y_rf_pred):.4f}")
print(f"MSE: {mean_squared_error(y_val, y_rf_pred):.4f}")

plt.figure(figsize=(10, 6))
plt.scatter(y_val, y_rf_pred, color='green', alpha=0.6, label="Predizioni")
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--', label="Perfetta predizione (y=x)")
plt.xlabel('Valori reali GPA')
plt.ylabel('Predizioni GPA')
plt.title('Random Forest: Valori reali vs Predetti')
plt.legend()
plt.show()

# --- Analisi residui RF ---
residui_rf = y_val - y_rf_pred


plt.figure(figsize=(10, 6))
sns.histplot(residui_rf, kde=True, bins=20, color="blue")
plt.title("Distribuzione residui - Random Forest")
plt.xlabel("Residui")
plt.show()
```

Le parti fondamentali del codice, anche qua come per la regressione lineare, sono le prime righe di codice.

- *rf_model = RandomForestRegressor(n_estimators=100, random_state=42):* La funzione viene importata dal pacchetto sklearn.ensamble e praticamente ci va a creare il modello del Random Forest, il parametro fondamentale è *n_estimators* che indica il numero di alberi decisionali che vengono creati, mentre *random_state* serve per generare sempre gli stessi risultati ad ogni esecuzione del programma.

- *rf_model.fit(X_train, y_train):* semplicemente addestra il modello creato usando il training set, andando perciò a far crescere i 100 alberi generati.

- *y_rf_pred = rf_model.predict(X_val):* il metodo predict() fa in modo che ogni albero vada ad effettuare la propria predizione, salvandola e

andando poi a calcolare la media di tutte le predizioni utilizzando la formula vista in precedenza.

Come per la regressione lineare abbiamo i risultati delle metriche di valutazione:

```
=== Random Forest Regressor ===
R^2: 0.4210
MSE: 0.0564
```

Figure 2.4: Risultati del terminale del Random Forest Regressor

I risultati che abbiamo sono peggiori della regressione lineare multi variata, perciò possiamo dire che il modello con i parametri di default è peggiore, i risultati rimangono comunque soddisfacenti, infatti un valore di R^2 equivalente a 0.42 è comunque un risultato che indica una capacità alta del modello di spiegazione e l'MSE è basso.

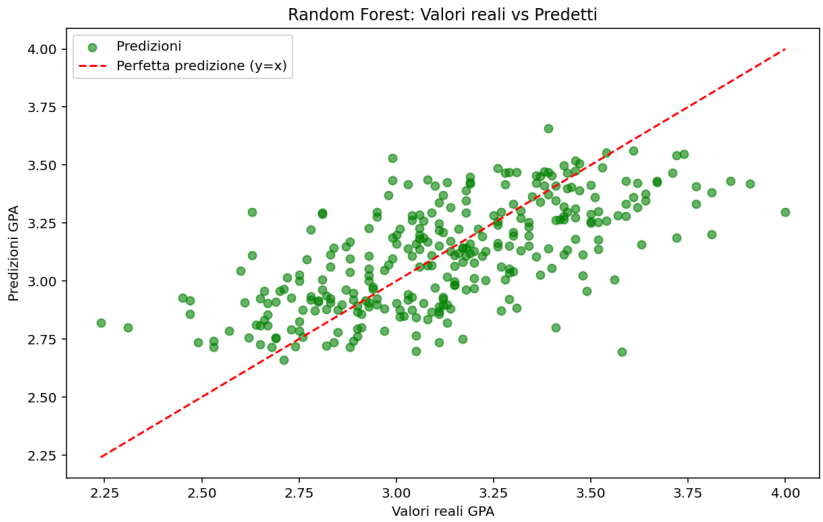


Figure 2.5: Random Forest Regressor - Predetti vs Reali

Osservazione dei risultati

Il grafico dei predetti vs reali in questo caso ci è utile per fare una comparazione diretta tra i risultati del terminale e una prima visualizzazione grafica di essi. Notiamo innanzitutto che il modello si comporta bene, la maggior

parte dei punti sono vicini alla linea di predizione perfetta, rispetto alla regressione lineare sono un po' più sparsi e da qua interpretiamo la maggior distanza dei punti come il motivo dei risultati di poco inferiori di questo modello. Notiamo comunque che gli outlier sono aumentati perciò ci sono alcune predizioni che non sono state fatte correttamente.

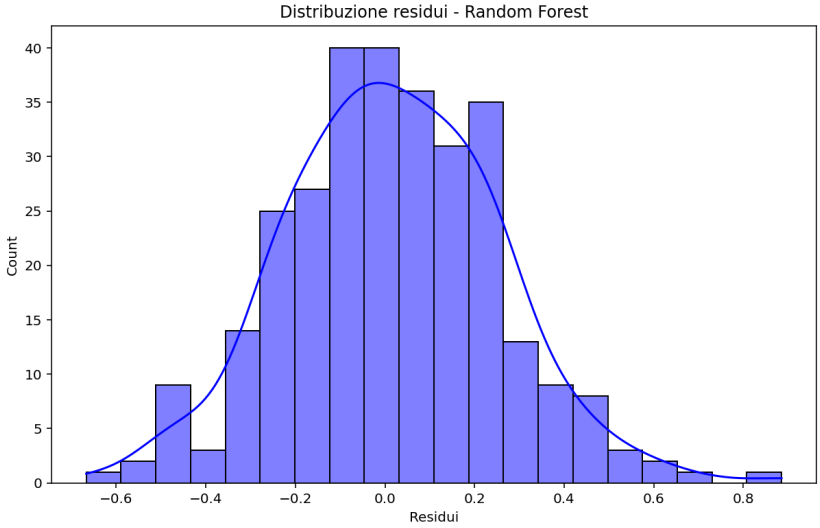


Figure 2.6: Random Forest Regressor - Analisi dei residui

L'analisi dei residui rimane coerente con ciò che è stato osservato in precedenza, il modello effettua una buona predizione infatti è facile notare il picco nell'area dello 0. La distribuzione però è tutt'altro che uniforme e la forma a campana gaussiana è poco riconoscibile infatti anche i valori vicini a 0.2 hanno un picco molto alto di residui a loro carico, perciò il modello riesce comunque a predire in modo soddisfacente senza però fornire le prestazioni rimarchevoli che avevamo pronosticato andando a sottintendere di riuscire ad individuare le relazioni non lineari tra variabili, pronostico che però con il primo modello complesso non ha trovato riscontro.

2.5 Support Vector Regression - Parametri di default

Le macchine a vettori di supporto (SVM, dall'inglese support-vector machines o networks) sono dei modelli di apprendimento supervisionato associati ad algoritmi di apprendimento per la regressione e la classificazione. Dato un insieme di esempi per l'addestramento, ognuno dei quali etichettato

con la classe di appartenenza fra le due possibili classi, un algoritmo di addestramento per le SVM costruisce un modello che assegna i nuovi esempi a una delle due classi, ottenendo quindi un classificatore lineare binario non probabilistico. Un modello SVM è una rappresentazione degli esempi come punti nello spazio, mappati in modo tale che gli esempi appartenenti alle due diverse categorie siano chiaramente separati da uno spazio il più possibile ampio. I nuovi esempi sono quindi mappati nello stesso spazio e la predizione della categoria alla quale appartengono viene fatta sulla base del lato nel quale ricade. Oltre alla classificazione lineare è possibile fare uso delle SVM per svolgere efficacemente la classificazione non lineare utilizzando il metodo kernel, mappando implicitamente i loro ingressi in uno spazio delle caratteristiche multi-dimensionale. Le SVM possono essere usate anche per la regressione, utilizzando opportune funzioni-obiettivo ε-sensibili.[10]. La regressione a vettori di supporto (SVR) è un'estensione delle SVM, che viene applicata ai problemi di regressione (ovvero, il risultato è continuo). Analogamente alle SVM lineari, la SVR trova un iperpiano con il margine massimo tra i punti dati.[11] La funzione di predizione dell'SVR con kernel RBF è:

$$\hat{Y}(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)\, K(x_i, x) + b$$

dove:

- α_i, α_i^* sono i moltiplicatori di Lagrange che rappresentano i **support vectors**, ovvero i punti più rilevanti del dataset,
- $K(x_i, x)$ è il kernel RBF.
- b è il termine di bias che serve per 'spostare' la funzione verso il basso o verso l'alto evitando che passi per l'origine se non necessario.

Solo i support vectors, i punti vicini al tubo ε, influenzano la predizione e ne vanno a determinare la forma finale. Come per entrambi i modelli precedenti andremo a valutare le sue performance usando R^2 e MSE.

```python
# ------------------------------
# Support Vector Regression (SVR con kernel RBF)
# ------------------------------
svr_model = make_pipeline(StandardScaler(), SVR(kernel='rbf', C=100, epsilon=0.1))
svr_model.fit(X_train, y_train)

y_svr_pred = svr_model.predict(X_val)

print("\n=== Support Vector Regression (RBF Kernel) ===")
print(f"R^2: {r2_score(y_val, y_svr_pred):.4f}")
print(f"MSE: {mean_squared_error(y_val, y_svr_pred):.4f}")

plt.figure(figsize=(10, 6))
plt.scatter(y_val, y_svr_pred, color='purple', alpha=0.6, label="Predizioni")
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--', label="Perfetta predizione (y=x)")
plt.xlabel('Valori reali GPA')
plt.ylabel('Predizioni GPA')
plt.title('SVR: Valori reali vs Predetti')
plt.legend()
plt.show()

# --- Analisi residui SVR ---
residui_svr = y_val - y_svr_pred

plt.figure(figsize=(10, 6))
sns.histplot(residui_svr, kde=True, bins=20, color="orange")
plt.title("Distribuzione residui - SVR")
plt.xlabel("Residui")
plt.show()
```

Il codice del Support Vector Regression funziona in modo leggermente diverso rispetto ai precedenti modelli, andiamo a visualizzare più nel dettaglio i metodi principali:

- *svr_model = make_pipeline(StandardScaler(), SVR(kernel='rbf', C=100, epsilon=0.1)):* Per creare il modello SVR il primo passo che andiamo a fare è il metodo *make_pipeline()* che va a creare una sequenza di passaggi che verranno applicati ai dati, il primo è il metodo *StandardScaler()* che normalizza i nostri dati ovvero imposta la media = 0 e la varianza = 1 perchè in questo modo andiamo a mettere tutti i dati sullo stesso piano siccome il modello SVR (ma più in generale tutti i modelli che si basano sull'utilizzo di kernel) non deve avere feature con valori troppo grandi altrimenti andrebbero a condizionare la predizione in modo troppo significativo. Il secondo passaggio è il metodo *SVR()* che va a creare il modello con paramenti specifici, *kernel='rbf'* indica il kernel Radial Basis Function che come detto nella spiegazione teorica del modello trasforma i dati per catturare relazioni non lineari; *C=100* è il parametro di penalizzazione ovvero è il parametro attraverso il quale andiamo a specificare la tolleranza che vogliamo sugli errori, 100 è un valore alto quindi significa che il modello cerca di ridurre di molto gli errori rischiando anche l'overfitting; infine *epsilon=0.1* è il margine in insensibilità ovvero ci dice che il modello prende in considerazione solo

deviazioni maggiori di 0.1, per i valori che hanno una epsilon inferirore l'errore non viene contato.

- *svr_model.fit(X_train, y_train):* questo come negli altri modelli è il metodo utilizzato per addrestare il modello sul training set.
- *y_svr_pred = svr_model.predict(X_val):* sempre come nei modelli precedenti il metodo predict() va ad applicare il modello su un nuovo set di dati.

Come i modelli visti in precedenza anche il modello SVR usa le stesse metriche di valutazione con i seguenti risultati:

```
=== Support Vector Regression (RBF Kernel) ===
R^2: 0.4034
MSE: 0.0581
```

Figure 2.7: Risultati del terminale del Support Vector Regression

I risultati delle metriche del modello SVR con i parametri di default per ora sono i peggiori registrati, questo deriva da più ragioni: la più importante è sicuramente il fatto che un modello complesso come SVR per funzionare adeguatamente e al massimo delle potenzialità ha bisogno di una fase di tuning dei propri parametri, un altra causa del risultato inferiore alla regressione Multipla potrebbe essere il fatto che in realtà non ci sono molte relazioni non lineari da intercettare.

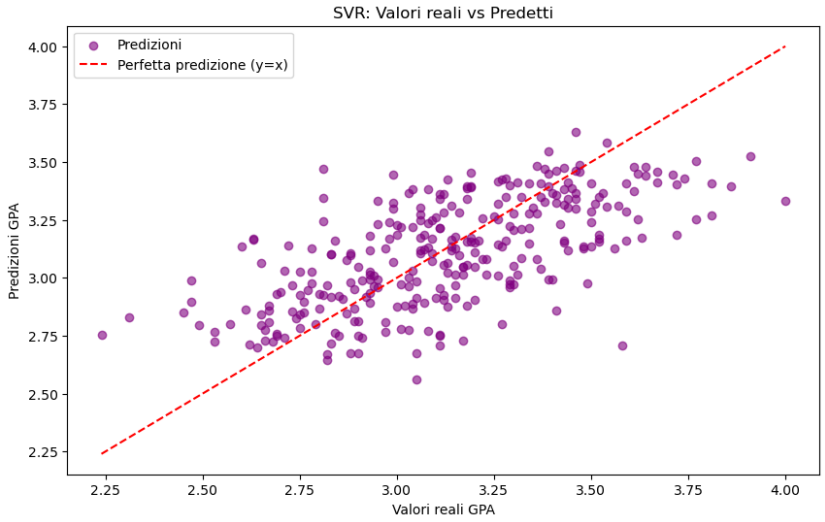


Figure 2.8: Support Vector Regression - Predetti vs Reali

Osservazione dei risultati

Come notiamo appena guardiamo il grafico troviamo immediatamente una correlazione tra ciò che abbiamo visto nei risultati del terminale e la distribuzione del grafico. Infatti confrontando tutti e tre i grafici dei valori reali vs i valori predetti notiamo che questo grafico è quello che ha i punti che più si allontanano dalla retta di predizione perfetta, andando a dare una visualizzazione grafica alle prestazioni inferiori del modello rispetto ai modelli precedentemente utilizzati. Notiamo anche una presenza più numerosa di outlier sintomo di predizioni poco accurate.

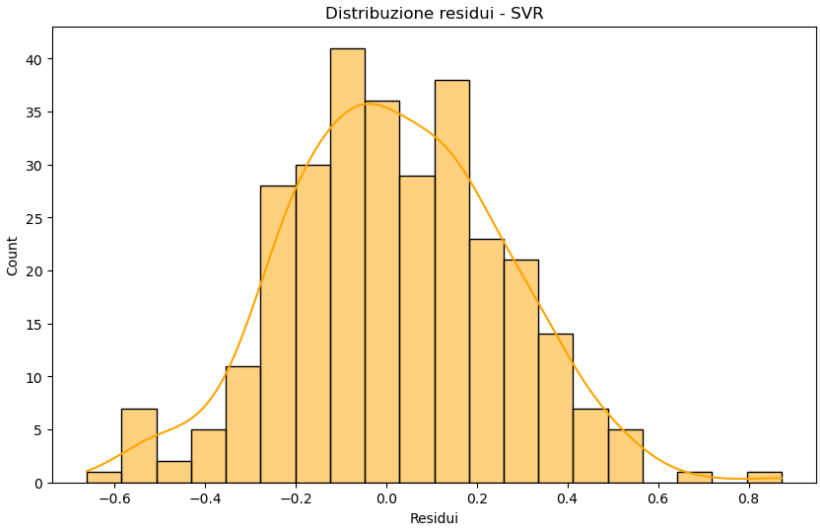


Figure 2.9: Support Vector Regression - Analisi dei residui

Il grafico dell'analisi dei residui anche qua rispecchia e spiega chiaramente i risultati ottenuti, per la prima volta abbiamo il picco dei residui non allineato con il valore 0, ciò ci indica in modo palese la minore accuratezza del modello, difatti i residui che si trovano in concomitanza con il valore di 0, ovvero quelli predetti in modo perfetto non sono nemmeno i secondi per quantità ma sono i terzi, questo significa che il modello ha prodotto molti dati con errori, molti di questi sono dati con un errore molto piccolo, però da questo grafico abbiamo un chiaro segnale che il modello non è sulla giusta via di predizione.

In seguito alle osservazioni e ai risultati ottenuti con i due modelli di predizione complessi che non sono stati sufficienti si è deciso di effettuare un hyperparameter tuning su entrambi i modelli per andare a sfruttare il loro massimo potenziale e vedere le differenze con il modello di regressione lineare

semplice e con loro stessi con i paramentri di default.

2.6 HyperParameter Tuning

Quando si costruisce un modello di Machine Learning esistono due tipi di parametri che vanno considerati:

- **Parametri del modello (parameters):** ovvero i valori che il modello impara durante l'addestramento dai dati su cui si allena.
- **Iperparamentri (hyperparameters):** sono dei valori che devono essere fissati prima dell'addestramento dall'utente, valori da cui dipende in modo enorme il comportamento e le prestazioni del modello.

L'ottimizzazione degli iperparametri (Hyperparameter Tuning) è un processo cruciale nel machine learning che mira a trovare la combinazione migliore di iperparametri per migliorare le prestazioni del modello[12]. Il modello SVR inoltre è un modello che è molto sensibile agli iperparametri, motivo per cui questo passaggio è strettamente necessario per visualizzare le sue vere capacità di predizione. Esistono diverse tecniche per scegliere gli iperparametri ottimali, quella che è stata adottata all'interno del progetto è la **Grid Search**: una delle tecniche più semplici e comuni per l'ottimizzazione degli iperparametri. Consiste nella ricerca esaustiva su una griglia predefinita di combinazioni di iperparametri[12]. Il vantaggio e lo svantaggio della tecnica sono facilmente intuibili, la garanzia di testare tutte le opzioni possibili va ad aumentare il costo in modo significativo, specialmente se la griglia definita contiene molti iperparametri. Per dare una valutazione sulla qualità del tuning non basta osservare l'errore sul training set, nel nostro progetto andremo ad usare i modelli tuned su un test set, ovvero un set di dati mai visto prima e in seguito andremo ad eseguire una cross-validation, Fare cross-validazione significa dividere i nostri dati di addestramento in diverse porzioni e testare il nostro modello su parti alcune di queste parti. Le performance del modello vengono valutate sulle porzioni generate dalla cross-validazione. Questa metodica viene chiamata K-Fold.Dopo aver interato attraverso ogni split, avremo come risultato finale la media delle performance. Questo aumenta la validità delle performance, in quanto un modello "nuovo" viene addestrato su ogni porzione del dataset di addestramento. Avremo quindi un punteggio finale che riassume la performance del modello in tanti step di validazione - una metodica molto affidabile rispetto alla performance in una singola iterazione.[13]

2.7 Random Forest Regressor & Support Vector Regression - Parametri tuned

2.7.1 Random Forest Regressor - Parametri tuned

```
# ================================
# PARAMETER TUNING - RANDOM FOREST
# ================================
param_grid_rf = {
    'n_estimators': [100, 200, 500],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': [None, 'sqrt', 'log2']
}

rf = RandomForestRegressor(random_state=42)

grid_search_rf = GridSearchCV(
    estimator=rf,
    param_grid=param_grid_rf,
    cv=5,                # 5-fold cross-validation
    scoring='r2',        # valuta con R^2
    n_jobs=-1,           # usa tutti i core
    verbose=2
)

grid_search_rf.fit(X_train, y_train)

print("\n=== Best Parameters Random Forest ===")
print(grid_search_rf.best_params_)
print(f"Best R^2 (CV): {grid_search_rf.best_score_:.4f}")

best_rf = grid_search_rf.best_estimator_

# Valutazione sul validation set
y_rf_best = best_rf.predict(X_val)
print(f"R^2 (Validation): {r2_score(y_val, y_rf_best):.4f}")
print(f"MSE (Validation): {mean_squared_error(y_val, y_rf_best):.4f}")
```

Per effettuare il tuning dei parametri andiamo ad utilizzare *param_grid_rf* che crea un dizionario con tutti gli iperparametri da provare:

- *n_estimators:* numero degli alberi che compongono la foresta.
- *max_depth:* profondità massima degli alberi.
- *min_samples_split:* numero minimo di campioni per dividere un nodo. Più è alto il valore più l'albero riduce overfitting.

- *min_samples_leaf:* numero minimo di campioni presenti all'interno di una foglia.
- *max_features:* numero di features da considerare ad ogni split (none usa tutte le feature, sqrt usa il numero sotto radice e log2 usa il logaritmo di due del numero delle features).

In questa griglia abbiamo 3x4x3x3x3 = 324 combinazioni da provare, considerando che cv=5 quindi vengono fatti i test su 5 campioni di dati differenti verranno fatti 324x5 = 1620 addestramenti. In seguito come nei modelli di default viene creato il modello con *RandomForestRegressor()*, successivamente viene impostata la Grid Search cross-validation usando il metodo *GridSearchCV(...)* con i suoi parametri:

- *GridSearchCV(...):* metodo che prova tutte le combinazioni della griglia.
- *estimator=rf:* il modello che va ottimizzato è il Random Forest.
- *param_grid=param_grid_rf:* viene passata la griglia creata in precedenza.
- *cv=5:* vengono usati 5 fold, ciò significa che ogni combinazione di iperparametri viene valutata su 5 split diversi del training.
- *scoring='r2':* la metrica di valutazione che vogliamo massimizzare è R^2 su ogni fold.
- *n_jobs=-1:* utilizza tutti i core della CPU in parallelo.
- *verbose=2:* stampa un log dettagliato nella console.

Attraverso il metodo *fit()* per ogni combinazione viene addestrato il modello sui 5 fold e viene calcolato l'R^2 per ogni combinazione. Tra tutte, infine viene memorizzata quella con il miglior valore di R^2 medio. Successivamente con *best_params_* mostriamo gli iperparametri che hanno dato il risultato migliore che visualizziamo con *best_score_*. Attraverso *best_estimator_* indichiamo la Random Forest già addestrata con gli iperparametri migliori.

```
=== Best Parameters Random Forest ===
{'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators':
200}
Best R^2 (CV): 0.5173
R^2 (Validation): 0.4778
MSE (Validation): 0.0508
```

Figure 2.10: Risultati metriche di valutazione Random Forest Regressor - best parameters

I risultati del Random Forest Regressor con i migliori iperparametri evidenziano un miglioramento delle prestazioni del modello. Sintomo del fatto che i modelli complessi necessitano di Hyperparameter tuning per performare al meglio.

2.7.2 Support Vector Regression - Parametri tuned

```python
# ==============================
# PARAMETER TUNING - SVR
# ==============================
param_grid_svr = {
    'svr__C': [0.1, 1, 10, 100],
    'svr__epsilon': [0.01, 0.1, 0.2, 0.5],
    'svr__kernel': ['rbf', 'poly'],
    'svr__gamma': ['scale', 'auto']
}

svr_pipeline = make_pipeline(StandardScaler(), SVR())

grid_search_svr = GridSearchCV(
    estimator=svr_pipeline,
    param_grid=param_grid_svr,
    cv=5,
    scoring='r2',
    n_jobs=-1,
    verbose=2
)

grid_search_svr.fit(X_train, y_train)

print("\n=== Best Parameters SVR ===")
print(grid_search_svr.best_params_)
print(f"Best R^2 (CV): {grid_search_svr.best_score_:.4f}")

best_svr = grid_search_svr.best_estimator_

# Valutazione sul validation set
y_svr_best = best_svr.predict(X_val)
print(f"R^2 (Validation): {r2_score(y_val, y_svr_best):.4f}")
print(f"MSE (Validation): {mean_squared_error(y_val, y_svr_best):.4f}")
```

Anche per il modello SVR andiamo a creare la griglia degli iperparametri:

- *svr__C:* quanta importanza decidiamo di dare al modello sul minimizzare gli errori sui dati di training rispetto ad avere un modello più generale (C basso vuol dire che il modello rimane semplice, C alto vuol dire che viene più complesso nonostante segua meglio i dati).
- *svr__epsilon:* margine di tolleranza per l'errore, valori maggiori permettono di ignorare gli errori piccoli rendendo il modello meno sensibile

al rumore.

- *svr__kernel:* tipo di funzione usata per proiettare i dati, *rbf* è la funzione gaussiana e la tecnica più comune, *poly* è la funzione polinomiale meno utilizzata.

- *svr__gamma:* controlla l'influenza dei singoli punti nella funzione risultato. Con valori bassi avremo un modello più liscio ma meno preciso che però si adatta meglio a più set di dati, con un gamma alto abbiamo un modello complesso che però funziona bene con i dati locali.

Abbiamo una griglia di 4x4x2x2 combinazioni per un totale di 64 modelli che moltiplicati per i 5 fold fanno 320 addestramenti. Andiamo poi a ripercorrere i passi che abbiamo eseguito per il modello di default andando a creare la pipeline e successivamente andiamo a definire, come per il modello analizzato in precedenza, la Grid Search Cross-Validation che non ha parametri diversi da quelli del Random Forest e che perciò non andremo ad analizzare nuovamente.

```
=== Best Parameters SVR ===
{'svr__C': 0.1, 'svr__epsilon': 0.1, 'svr__gamma': 'scale', 'svr__kernel': 'rbf'}
Best R^2 (CV): 0.5258
R^2 (Validation): 0.4988
MSE (Validation): 0.0488
```

Figure 2.11: Risultati metriche di valutazione Support Vector Regression - best parameters

I valori delle metriche finali vanno a migliorare in modo abbastanza significativo le prestazioni del modello base, ciò trova riscontro nell'affermazione che era stata fatta una volta visti i risultati del SVR con i valori di default ovvero che il modello essendo complesso per performare necessita di iperparametri selezionati in maniera molto specifica.

2.7.3 Confronto tra i modelli tuned

Ora che abbiamo visto come si comportano i modelli tuned attraverso i risultati numerici del terminale andiamo a visualizzare alcuni grafici per andare a comprendere meglio le loro prestazioni.

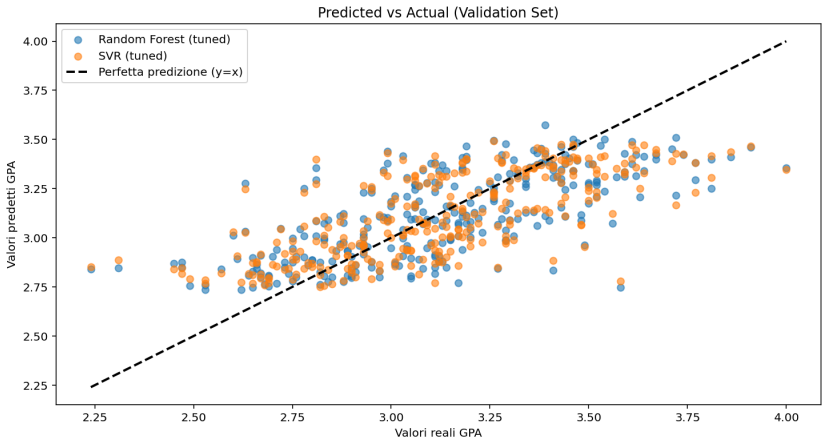


Figure 2.12: Confronto Predicted vs Actual - Modelli tuned

Visti l'uno insieme all'altro i due modelli si comportano in modo molto simile con il modello SVR che ha un comportamento leggermente maggiore nella fascia del GPA che va da 2.75 a 3.50. In generale però è evidente che entrambi i modelli siano entrambi dei buoni preditori in grado di fornire solide prestazioni senza che ci sia una netta supremazia di uno sull'altro.

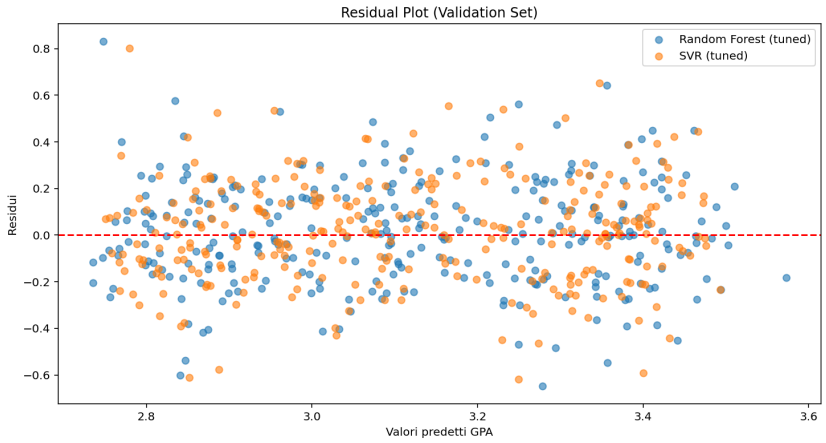
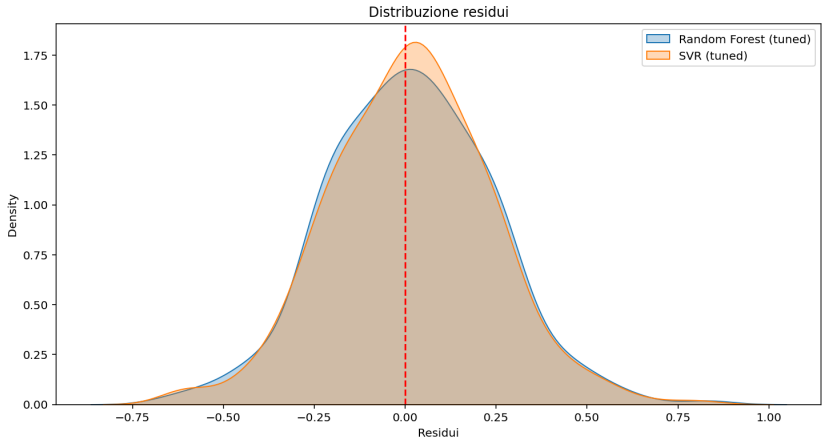


Figure 2.13: Sopra: Residual plot; Sotto: Grafico della distribuzione dei residui



Dal grafico dei residual plot non riusciamo bene ad avere un'idea di come i due modelli si comportano, sicuramente il grafico della loro distribuzione è molto più esplicativo e chiaro su di essi. Nel grafico iniziale SVR è leggermente più solido, specialmente intorno ai punti dove il GPA è intorno al 2.8 e 3.4, nella parte centrale del grafico invece i due modelli non hanno grandissima diversità nelle prestazioni. Entrambi comunque effettuano delle predizioni che si allontanano molto dalla retta di predizione perfetta, dimostrando comunque che esistono dei dati che i modelli non riescono ad interpretare nel modo corretto. Come detto inizialmente, il grafico della distribuzione è molto più esplicativo e di facile interpretazione. Si nota infatti con abbastanza facilità il fatto che i modelli facciano praticamente la stessa funzione gaussiana; l'unica differenza sta nel fatto che SVR riesce ad avere più predizioni perfette, che si notano dal fatto che il piccolo della sua funzione (arancione) abbia un punto di massimo globale maggiore di quello della funzione del Random Forest (blu).

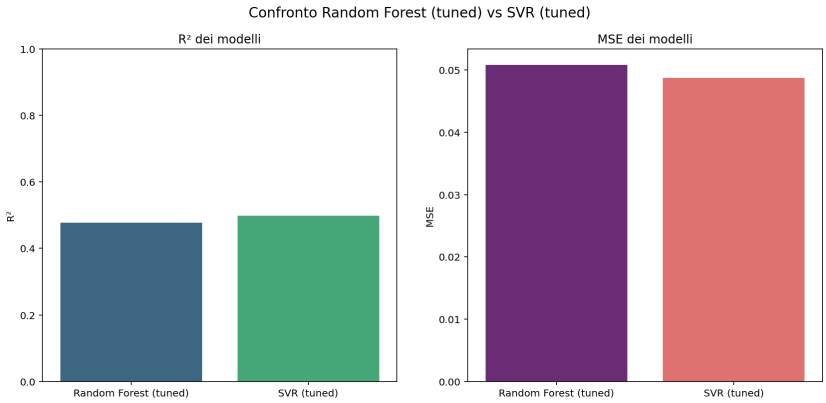


Figure 2.14: Confronto delle metriche di valutazione dei modelli tuned

Questo grafico semplicemente mette in relazione grafica i valori risultanti del terminale che abbiamo visto in precedenza, rispecchia perciò quanto visto, con il modello SVR (tuned) che ha un R^2 leggermente superiore al Random Forest (Tuned) e un MSE leggermente minore. Questo grafico simultaneamente trova conferma e ne da un'ulteriore di tutto quello che è stato osservato sui modelli con gli iperparametri migliori possibili.

2.8 Confronto tra i modelli migliori

Ora a questo punto abbiamo 5 diversi modelli che abbiamo utilizzato durante il nostro progetto, tutti diversi e che ci hanno fornito risultati e prestazioni diverse tra loro. Attraverso la tabella sottostante andiamo a raccogliere i loro risultati sulle metriche di valutazione, sceglieremo i 3 migliori e andremo a fare un confronto tra essi andandoli a testare su dati inediti usando il test set che dalla fase di split è rimasto ancora intatto. Successivamente andremo a fare un test finale su più ripetizioni per vedere la solidità dei due modelli che si sono dimostrati i migliori sulle prove con il test set in modo da andare a trovare il miglior modello di tutti.

Modello	$\mathbf{R^2}$	**MSE**
Linear Regression (Multipla)	0.5251	0.0462
Random Forest (default)	0.4210	0.0564
SVR (default)	0.4034	0.0581
Random Forest (tuned)	0.4778	0.0508
SVR (tuned)	0.4988	0.0488

Table 2.1: Confronto delle performance dei 5 modelli in termini di R^2 e MSE.

Dalla tabella si vede che i tre modelli con le metriche migliori sono la Linear Regression, il Random Forest (tuned) e l'SVR (tuned) perciò andiamo a prenderli tutti e tre ed andiamo a fare una prova ulteriore sul test set.

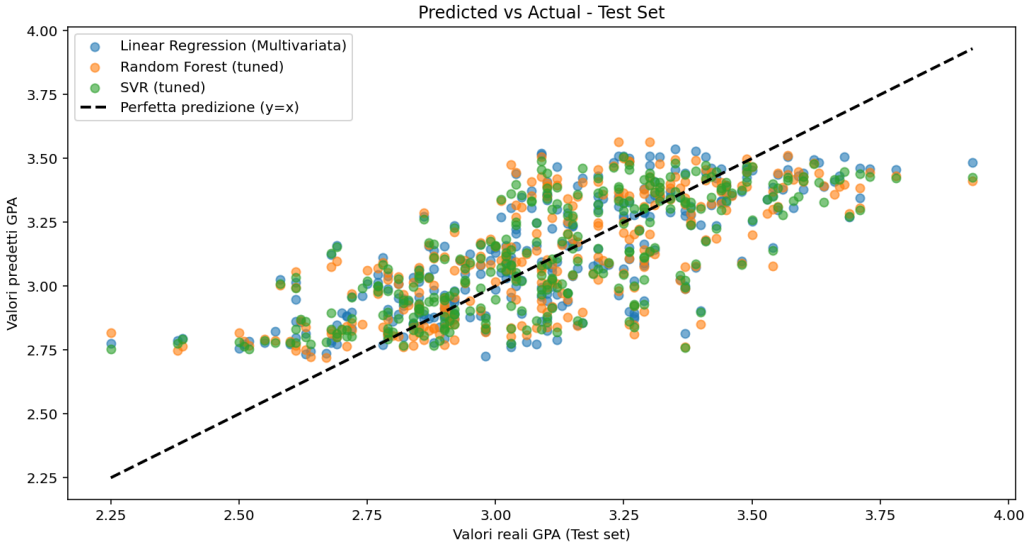


Figure 2.15: Predicted vs Actual - 3 modelli migliori

```
=== VALUTAZIONE FINALE SU TEST SET ===

Linear Regression (Multivariata)
  R²  = 0.5317
  MSE = 0.0391

Random Forest (tuned)
  R²  = 0.5073
  MSE = 0.0412

SVR (tuned)
  R²  = 0.5302
  MSE = 0.0393
```

Figure 2.16: Risultati delle metriche sul test set

Osservazione dei risultati

Tutti e tre i modelli mostrano una tendenza generale positiva, perciò possiamo dire che i modelli catturano in modo consistente la relazione, dal grafico Predicted vs Actual, non si riesce a definire un modello nettamente migliore dell'altro perciò andiamo a fare principalmente riferimento alle nuove metriche calcolate sul test set. I risultati ci spiegano esattamente il perchè il grafico risulti molto simile per i tre modelli, infatti le metriche sono tutte molto simili con il modello **Linear regression** e **SVR (tuned)** che primeggiano con risultati praticamente identici staccandosi leggermente dal Random Forest che rimane un ottimo modello ma che rimane indietro rispetto ai due nominati in precedenza. Ciò ci porta a dire che il Random Forest con i dati del test set è il peggiore dei 3, come si era visto dalla tabella visionata precedentemente. Per capire quale dei due modelli che si sono comportati meglio è effettivamente il migliore andiamo ad eseguire un ultimo passaggio. Andiamo ad eseguire uno studio statistico finale che si baserà sull'addestrare e testare i modelli per k volte (nel nostro caso 30) ad un intervallo di confidenza del 95%

(Questo significa che se si ripetesse per cento volte la stessa sperimentazione si troverebbero cento valori diversi della misura di efficacia considerata: 95 di questi valori si situerebbero entro l'intervallo di confidenza, mentre 5 si troverebbero al di fuori di esso[14]) per decretare il modello migliore.

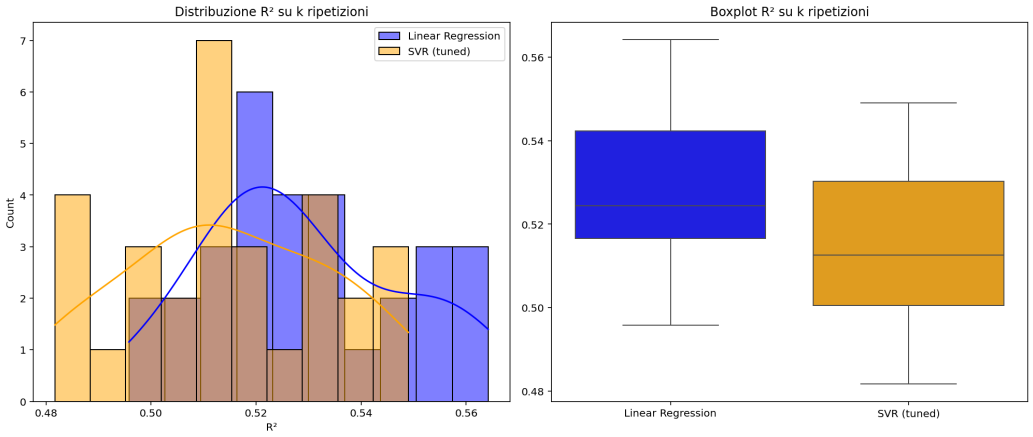


Figure 2.17: Confronto finale Linear Regression - SVR (tuned)

```
=== Linear Regression ===
R² medio: 0.5288 ± 0.0186
IC 95% R²: (np.float64(0.5217510833775572), np.float64(0.5359068427492231))
MSE medio: 0.0415 ± 0.0019
IC 95% MSE: (np.float64(0.04077330376950911), np.float64(0.042182891313184874))

=== SVR (tuned) ===
R² medio: 0.5146 ± 0.0192
IC 95% R²: (np.float64(0.5072626093359199), np.float64(0.5218711645460964))
MSE medio: 0.0427 ± 0.0021
IC 95% MSE: (np.float64(0.041944374671972935), np.float64(0.043541048059278366))
```

Figure 2.18: Risultati delle metriche di valutazione medie e con l'intervallo di confidenza al 95%

Osservazione dei risultati

Per quest'ultimo confronto tra i due modelli è stata condotta un'analisi basata su 30 ripetizioni (il valore della nostra k) di training test casuali valutati sui test set generati. Ogni volta sono stati calcolate le metriche di valutazione standard del nostro progetto R^2 e MSE, riportati alla fine con i valori medi e con i valori per l'intervallo di confidenza al 95%.

- La **regressione lineare Multipla** ha ottenuto un R^2 medio di 0.5288 (+-0.0186) con un intervallo di confidenza 95% compreso tra 0.5218

e 0.5360. L'MSE medio è risultato 0.0415(+-0.0019) con intervallo di confidenza 95% tra 0.0408 e 0.0421.

- Il modello **SVR (tuned)** invece ha registrato un valore per l'R^2 medio di 0.5146 (+-0.0192) perciò leggermente inferiore, con intervallo di confidenza al 95% tra 0.5073 e 0.5219 con MSE medio di 0.0427 (+-0.0021) con intervallo di confidenza al 95% tra 0.0419 e 0.0435.

Dal punto di vista prettamente numerico perciò possiamo dire che la Regressione Lineare ha registrato dei valori medi poco migliori per entrambe le metriche. Tuttavia gli intervalli di confidenza sono molto vicini e anche leggermente sovrapposti suggerendoci che le differenze tra i due modelli siano veramente molto piccole a livello di predizioni. A prova di ciò che abbiamo osservato anche l'istogramma ci conferma ciò che abbiamo detto, si nota chiaramente che le curve sono sovrapposte in larga parte tranne all'inizio che c'è una presenza singola di valori generati dall'SVR mentre alla fine di valori generati dalla Regressione Lineare. Il boxplot invece ci mostra che la mediana e i valori trovati in generale dalla regressione lineare (con mediana intorno allo 0.525) sono generalmente più alti di quelli dell'SVR (con mediana interno allo 0.510).

In conclusione entrambi i modelli mostrano prestazioni similari sul dataset che abbiamo preso in analisi con una leggera superiorità media della **Regressione Lineare**. Tuttavia tale differenza non è così ampia da risultare significativa. Perciò possiamo evincere che per il nostro progetto la regressione lineare è preferibile per la sua maggiore semplicità e interpretabilità, senza però scordarsi delle performance predittive molto vicine a quelle di un modello più complesso come SVR.

Chapter 3

Conclusioni e sviluppi futuri

L'obiettivo del lavoro di ricerca che questa tesi si era prefissata era quello di studiare l'impatto delle abitudini di vita degli studenti universitari sul rendimento accademico, che viene misurato attraverso il Grade Point Average (GPA). L'analisi si è concentrata su alcune variabili in particolare, quelle che sono state ritenute le più interessanti da osservare, ovvero le ore di studio, le ore di sonno, il livello di stress e le ore di attività fisica con lo scopo di verificare se questi fattori in qualche maniera possono essere dei predittori affidabili delle performance scolastiche.
Il progetto si è sviluppato in 3 macro-fasi principali:

- **Preprocessing, esplorazione dei dati e splitting:** questa è stata la prima fase del progetto, è stata effetuata una pulizia del dataset, eliminazione delle variabili non informative e una fase di esplorazione dati attraverso diversi grafici illustrativi. Per poi concludere con la fase di splitting in cui abbiamo creato i vari set per prepararci alla predizione.

- **Stima dei modelli predittivi:** in questo passaggio abbiamo utilizzato cinque algoritmi di predizione per andare a predirre la variabile target: *Regressione lineare Multipla*, *Random Forest Regressor*, *Support Vector Regression (SVR)* con kernerl RBF, *Random Forest Regressor (tuned) Support Vector Regression (SVR) (tuned).* Tutti e cinque gli algoritmi sono stati valutati secondo le stesse metriche: R^2 e MSE.

- **Analisi statistica:** dei due algoritmi più performanti: *regressione linare Multipla* e *Support Vector Regression (SVR) (tuned)* è stata effettuata un'analisi statistica improntata alla valutazione dei risultati di questi due algoritmi su 30 ripetizioni di splitting casuale del dataset e con il calcolo degli intervalli di confidenza al 95%.

Per riportare un'ultima volta i risultati ottenuti e dare una stima alle predizioni effettuate possiamo dire che i due modelli migliori sono stati:

- **Regressione lineare:** R^2 medio: 0.5288 +-0.0186 (IC95%: 0.5218 – 0.5360); MSE medio: 0.0415 +- 0.0019 (IC95%: 0.0408 – 0.0422).
- **SVR (tuned):** R^2 medio: 0.5146 +-0.0192 (IC95%: 0.5073 - 0.5219); MSE medio: 0.0427 +-0.0021 (IC95%: 0.0419 - 0.0435).

Questi numeri ci dimostrano che la regressione lineare ha un'accuratezza leggermente superiore ma anche risultati più stabili su più ripetizioni. Ciò può essere interpretato in diversi modi, il motivo principale potrebbe essere che la relazione tra le variabili indipendenti e il GPA sia fortemente lineare, perciò per un modello come SVR che si impegna a trovare relazioni non lineari ciò potrebbe aver pesato. Il secondo motivo potrebbe essere riscontrato nella natura propria del dataset, il campione preso in osservazione non ha dimensioni particolarmente grandi quindi è possibile che modelli complessi come il Random Forest e SVR non siano riusciti ad esprimere appieno il loro potenziale. Da ciò che abbiamo analizzato potremmo trarre degli spunti concreti per applicazioni pratiche, siccome il rendimento universitario non è solo determinato dalle capacità cognitive ma anche da fattori esterni su cui le varie università possono intervenire in maniera concreta con programmi mirati sulle varie abitudini dei loro studenti.

Nonostante gli aspetti positivi, il lavoro presenta alcune limitazioni, tra cui il dataset che copre una bassa percentuale di studenti che potrebbe non rispecchiare le tendenze comuni di altri gruppi di studenti non presenti all'interno del campione oppure il fatto di aver trovato poche correlazioni non lineari tra le variabili potrebbe suggerire una soluzione abbastanza superficiale dell'analisi svolta.

Per il futuro invece si potrebbero effettuare diverse implementazioni al progetto: innanzitutto si potrebbe pensare ad un ampliamento del dataset includendo molti più studenti diversi e con abitudini diverse; oltre agli studenti aggiungere al campione anche altre variabili psicologiche e socio-economiche (motivazione, autostima, ansia da prestazione, reddito familiare, accesso a risorse educative) e infine si potrebbe andare ad utilizzare altri algoritmi di predizione per avere ancora più punti di vista differenti sulle relazioni del dataset.

In conclusione il lavoro svolto ha mostrato che, al contrario di ciò che ci si aspettava inizialmente la regressione lineare Multipla si è dimostrata il

modello più adatto per predirre il rendimento accademico utilizzando delle variabili indipendenti dello stile di vita degli studenti. Non ha solo garantito le migliori performance sulle metriche di valutazione scelte ma ha anche mostrato la stabilità maggiore tra tutti gli algoritmi utilizzati. Questo risultato assegna alla regressione lineare un valore duplice: da un lato come strumento scientifico per comprendere le relazioni tra performance e abitudini ma anche come strumento utilizzabile dalle università in modo diretto per migliorare in modo attivo ed efficace il successo dei loro studenti.

Bibliography

[1] Redazione, "Che cosa è: Calcolo GPA", *https://it.statisticseasily.com/glossario/what-is-gpa-calculation-understanding-academic-performance/*, ultimo accesso: 9 ottobre 2025

[2] Redazione, "Analisi esplorativa dei dati", *https://www.jmp.com/it/statistics-knowledge-portal/exploratory-data-analysis*, ultimo accesso: 9 ottobre 2025

[3] Nicola Iantomasi, "Coefficiente di determinazione R quadro", *https://www.yimp.it/coefficiente-di-determinazione-r-quadro/*, ultimo accesso: 9 ottobre 2025.

[4] Redazione, "Che cosa è: Errore quadratico medio", *https://it.statisticseasily.com/glossario/che-cosa-%C3%A8-l%27errore-quadratico-medio-guida-completa/*, ultimo accesso 9 ottobre 2025.

[5] Redazione, "Cos'è la regressione lineare?" *https://www.ibm.com/it-it/think/topics/linear-regression*, ultimo accesso: 9 ottobre 2025.

[6] Redazione, "Regressione Lineare Multipla", *https://it.wikipedia.org/wiki/Regressione_lineare#Regressione_lineare_multipla*, ultimo accesso: 9 ottobre 2025.

[7] Paola Pozzolo, "R quadro: il coefficente di determinazione", 1 Agosto 2020, *https://paolapozzolo.it/coefficiente-determinazione-r-quadro/*, ultimo accesso: 9 ottobre 2025.

[8] Redazione, "Che cosa è: Metodo Residuo", *https://it.statisticseasily.com/glossario/what-is-residual-method-explained-in-detail/*, ultimo accesso: 9 ottobre 2025.

[9] Redazione, "Random Forest – Algoritmi di Machine Learning", 18 Dicembre 2021, *https://pulplearning.altervista.org/random-forest-algoritmi-di-machine-learning/*, ultimo accesso: 9 ottobre 2025

[10] Redazione, "Macchine a vettori di supporto", *https://it.wikipedia.org/wiki/Macchine_a_vettori_di_supporto*, ultimo accesso: 9 ottobre 2025

[11] Redazione, "Cosa sono le macchine a vettori di supporto (SVM)?", *https://www.ibm.com/it-it/think/topics/support-vector-machine*, ultimo accesso: 9 ottobre 2025

[12] Redazione, "Hyperparameter Tuning", 24 Giugno 2024, *https://algoretico.it/it/magazine/tech/hyperparameter-tuning*, ultimo accesso: 9 ottobre 2025

[13] Andrea D'agostino, "Cosa è la cross-validazione nel machine learning", 27 Agosto 2022, *https://www.diariodiunanalista.it/posts/cosa-e-la-cross-validazione-nel-machine-learning/*, ultimo accesso: 9 ottobre 2025

[14] Redazione, "Intervallo di confidenza", 9 Settembre 2022, *https://ilpunto.it/leggere-i-numeri/intervallo-di-confidenza/*, ultimo accesso: 9 ottobre 2025