DEPRESSION PREDICTION AMONG STUDENTS (BINARY CLASSIFICATION)

Dindane Jean Baptiste (Matricola: 0001100694)

INTRODUZIONE

In questo progetto verra condotta un'analisi predittiva, per identificare la presenza di sintomi depressivi negli studenti.
Il dataset contiene diciotto colonne, dieci di tipo quantitativo(numerico) e otto di tipo qualitativo (categorico).
Il dataset è disponibile sul portale kaggle al seguente indirizzo: https://www.kaggle.com/datasets/hopesb/student-depression-dataset

PRESENTAZIONE DEL DATASET

COLONNE NUMERICHE

Id => Identificatore unico per ogni studente

Age => Indica l'eta dello studente.

Academic Pressure => Misura il livello di stress o pressione a cui uno studente è sottoposto a livello accademico.

Study Satisfaction => Indica il grado di soddisfazione di uno studente verso il proprio percorso di studio.

Work/Study Hours => Indica il numero medio di ore che lo studente dedica allo studio e al lavoro.

PRESENTAZIONE DEL DATASET

COLONNE NUMERICHE

Job Satisfaction => Indica il grado di soddisfazione a livello lavorativo di uno studente lavoratore.

CGPA => Rappresenta la media complessiva di tutti i voti ottenuti da uno studente.

Work Pressure => Misura il livello di stress o pressione a cui uno studente è sottoposto a livello lavorativo.

Finiancial Stress => Misura il livello di difficoltà economica percepita da uno studente.

Depression => Variabile target, indica se lo studente soffre di depressione (1) oppure non soffre di depressione (0).

PRESENTAZIONE DEL DATASET

COLONNE CATEGORICHE

Gender => Indica il genere dello studente.

Sleep Duration => Indica il numero medio di ore di sonno dello studente.

Dietary Habits => Descrive il comportamento alimentare dello studente.

Have you ever had Suicidal Thoughts => Indica se lo studente ha mai avuto pensieri suicidi o autolesionistici.

PRESENTAZIONE DEL DATASET

COLONNE CATEGORICHE

Family History of Mental => La laurea dello studente, nel caso egli fosse gia in possesso di un titolo universitario.

City => Città dello studente

Profession => Descrive la professione, nel caso lo studente fosse anche lavoratore.

Degree => Il titolo di studio posseduto dallo studente, nel caso fosse in possesso di altri titoli universitari.

COLONNE DEL DATASET

id	Gender
Unique_id	Gender
 2 141k	Male 56% Female 44%

Age	City
Age	City
 18 59	Kalyan 6% Srinagar 5% Other (24959) 89%

Profession	Academic Pressure
Profession	Academic Pressure
Student 100% Architect 0% Other (23) 0%	 0 5

COLONNE DEL DATASET

✓ Family History of ...	# Depression
History	Depression
 true 13.5k 48% false 14.4k 52%	 0 1

A Degree	✓ Have you ever ha...
Degree	Suicidal thoughts
Class 12 22% B.Ed 7% Other (19954) 72%	 true 17.7k 63% false 10.2k 37%

# Work/Study Hours	# Financial Stress
Study/work hours	Financial stress
 0 12	 1 5

COLONNE DEL DATASET

# Study Satisfaction	# Job Satisfaction
Study Satisfaction	Job satisfaction
	
0 5	0 4

# Work Pressure	# CGPA
Work Pressure	Grade
	
0 5	0 10

A Sleep Duration		A Dietary Habits	
Sleep duration range		Dietary	
Less than 5 hours	30%	Unhealthy	37%
7-8 hours	26%	Moderate	36%
Other (12245)	44%	Other (7663)	27%

#2 PREPROCESSING

```
#1.  ____________________________CARICAMENDO DATASET_____________________
df = pd.read_csv('Student_Depression_Dataset.csv')
print(df.info())


#2.  ____________________________PRE_PROCESSING_____________________________
#In questa fase andiamo a preparare i dati per le fasi successive

#EEsplorazione a livello generale dei miei del dataset
print(df.head())
print(df.describe())

# Controllo valori mancanti
print(df.isnull().sum())
#Solo "Financial Stress" risulta null in tre righe


#Non sapendo se le persone non hanno risposto alla domanda
#Perché hanno tanto o poco stress finanziario, utilizziamo la mediana (in questo caso 3)
median_value = df['Financial Stress'].median()
df['Financial Stress'] = df['Financial Stress'].fillna(median_value)
```

```
[8 rows x 10 columns]
id                                     0
Gender                                 0
Age                                    0
City                                   0
Profession                             0
Academic Pressure                      0
Work Pressure                          0
CGPA                                   0
Study Satisfaction                     0
Job Satisfaction                       0
Sleep Duration                         0
Dietary Habits                         0
Degree                                 0
Have you ever had suicidal thoughts ?  0
Work/Study Hours                       0
Financial Stress                       3
Family History of Mental Illness       0
Depression                             0
dtype: int64
```

#2 PREPROCESSING

```
#Esploro tutte le colonne che potrebbero non risultare rilevanti per lo studio per esempio
#Esempio con id (ma ho esplorato anche Profession, city, Degree ecc...)
print(df['id'].value_counts())
print(df['Profession'].value_counts())


#Elimino le colonne irrilevaniti per lo studio,
#id non contiene alcuna informazione utile ma serve soltanto come identificatore
#Profession non influisce molto sullo studio visto che su 27.901 persone 27.870 sono student
#Rimuoviamo "Work Pressure" perchè è correlato a profession
#Rimuoviamo  "Job Satisfaction" per la stessa ragione
df = df.drop(columns=['id'])
df = df.drop(columns=['Profession'])
df = df.drop(columns=['Work Pressure'])
df = df.drop(columns=['Job Satisfaction'])

#Ricontrolliamo le colonne su cui verra condotto lo studio, dopo aver rimosso
#quelle che si ritengono non rilevanti ai fini dello studio
print(df.info())
```

```
Profession
Student                  27870
Architect                    8
Teacher                      6
Digital Marketer             3
Content Writer               2
Chef                         2
Doctor                       2
Pharmacist                   2
Civil Engineer               1
UX/UI Designer               1
Educational Consultant       1
Manager                      1
Lawyer                       1
Entrepreneur                 1
```

#3 EXPLORATORY DATA ANALYSYS

Si è condotta un'analisi esplorativa dei dati per comprendere meglio le caratteristiche dei dati e le relazioni tra le varibili.
I punti chiave di questa fase sono:

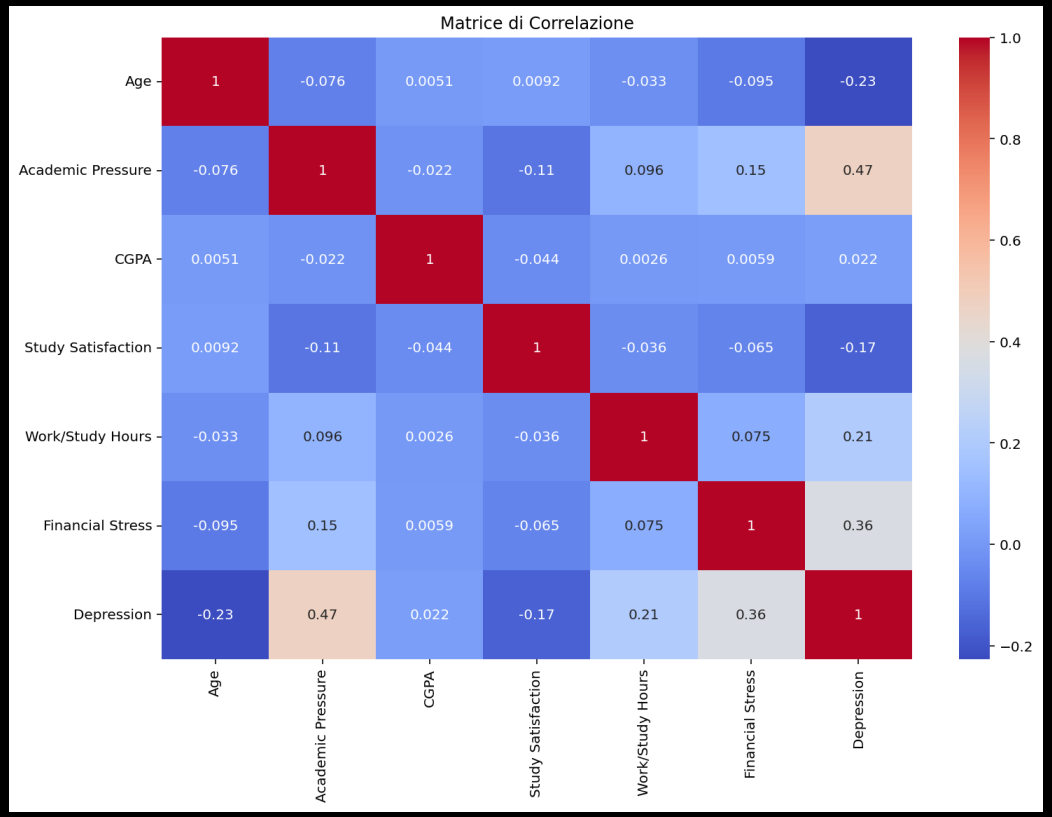
1. **Analisi Bivariata**: Si esamina la relazione tra due variabilii.

2. **Matrice di Correlazione**: Calcolo e visualizzazione delle relazioni lineari tra le variabili numeriche tramite una heatmap, evidenziando colori forti e debili.

3. **Relazione con la variabile target**:

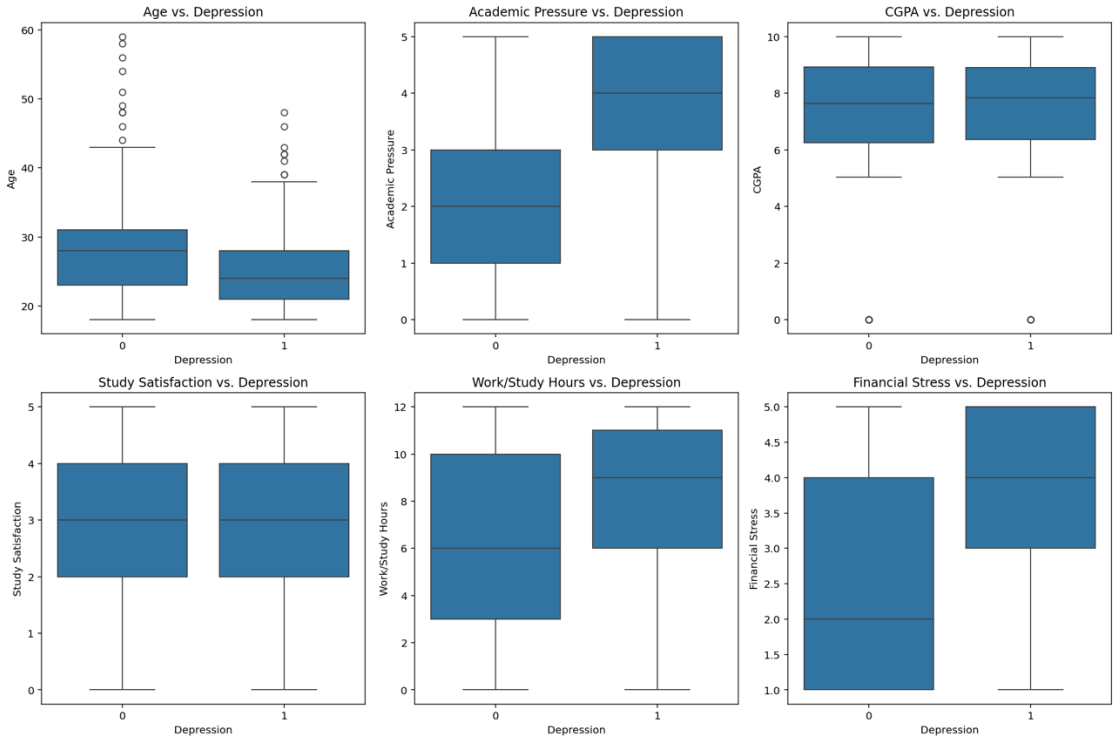
    - **Boxplot**, per analizzare relazione tra variabili numeriche e variabile target.
    - **Grafici a barre**, per le variabili categoriche.

#3 EXPLORATORY DATA ANALYSYS

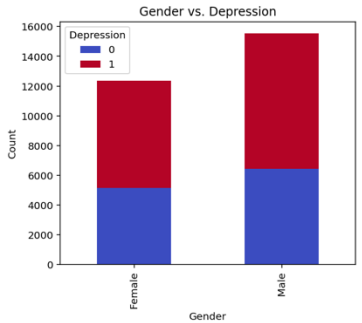


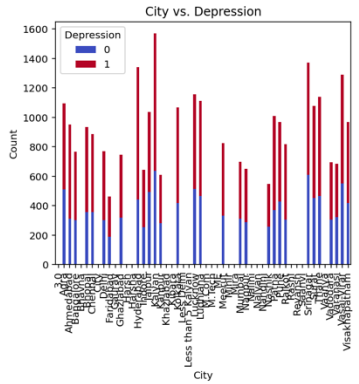
La matrice di correlazione è uno strumento statistico che consente di analizzare la relazione tra più variabili quantitative in un dataset.

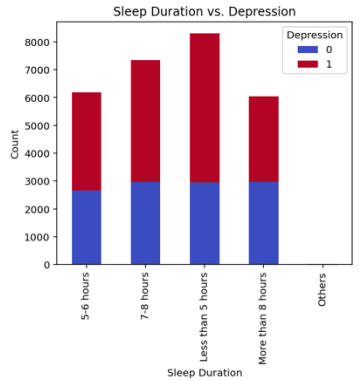
- Il valore **1** indica una correlazione perfetta positiva, quandouna variabile aumenta, anche l'altra aumenta proporzionalmente.
- Il valore **-1** inidica una correlazione perfetta negativa, quando una variabile aumenta, l'altra diminuisce proporzionalmente.
- Il valore **0** significa che non abbiamo nessuna correlazione (le variabili non hanno una relazione lineare.

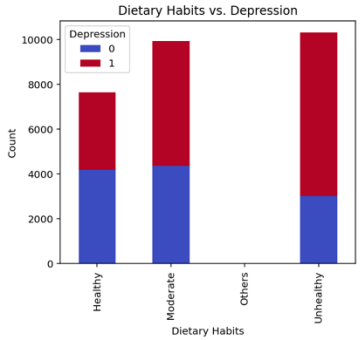


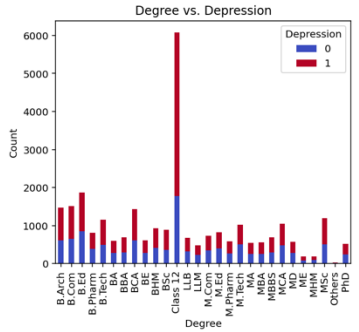
#3 EXPLORATORY DATA ANALYSYS

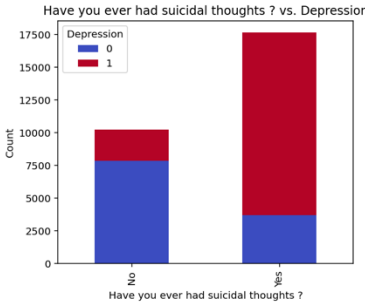


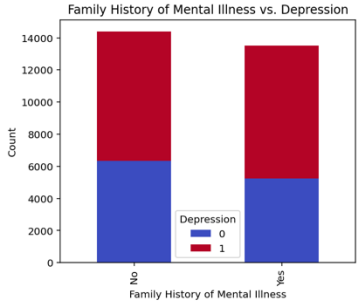












#3 EXPLORATORY DATA ANALYSYS

#4 SPLITTING

In questa fase, il dataset viene preparato per l'addestramento di un modello di classificazione. Il dataset suddiviso in **Training Set** e **Test Set** (il quale verra diviso nuovamente per otterenere il **Validation set**):

1. **Training Set** (70% del dataset): Verrà utilizzato per addestrare il modello.
2. **Validation Set** (15% del dataset): Utilizzato per ottimizzare i parametri del modello.
3. **Test Set** (15 % del dataset): Utilizzato per valutare le performance finali del

#4 SPLITTING

L'encoding trasforma le variabili categoriche in numeriche, perciò utilizzabili dai modelli.
Le tecniche di encoding utilizzate sono:

1. **One-Hot-Encoding**: per variabili senza un ordine specific.

2. **Ordinal-Encoding**: per variabili con un ordine specifico.

Viene eseguito solamente dopo aver diviso il dataset in Training e Test Set per evitare Data Leakage.

```
#DIVISIONE DEL DATASET IN TRAINING E TEST SET
# Separiamo X (feature) e y (target)
X = df.drop(columns=['Depression'])  # Tutte le feature
y = df['Depression']  # Target

# Split iniziale
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

```
#                                   ------------------CODIFICA TRAINING SET ----------------------
X_train = pd.get_dummies(X_train, columns=['Gender', 'City', 'Dietary Habits', 'Degree',
       'Have you ever had suicidal thoughts ?', 'Family History of Mental Illness'], drop_first=True)

#print(X_TempEncode['Sleep Duration'].value_counts())

#Non possiamo utilizzare one-hot encoding per "Sleep Duration", perché non tiene conto dell'ordine
# Definire l'ordine


# converto "Sleep Duration" in una colonna categoriale con ordine specificato
X_train['Sleep Duration'] = pd.Categorical(
    X_train['Sleep Duration'],
    categories=['Less than 5 hours', '5-6 hours', '7-8 hours', 'More than 8 hours', 'Others'],
    ordered=True
)

# Sostituisco i valori categorici con i valori interi(mantenendo l'ordine)
X_train['Sleep Duration'] = X_train['Sleep Duration'].cat.codes + 1  # Sommiamo 1 per partire da 1

#print(X_train.info())

# Salviamo tutte le colonne del training set
train_columns = X_train.columns
```

#5 REGRESSIONE

```
# Le colonne numeriche più fortemente correlate (positivamente) sono
#"Academic Pressure" e "Financial Stress", per questo vengono utilizzate
#per la regressione lineare
X = 'Academic Pressure' # Variabile indipendente
y = 'Financial Stress'  # Variabile dipendente

#preparazione delle variabili per la regressione
X_reg = df[[X]].values
y_reg = df[y].values
#creazione e addestramento del modello di regressione lineare
reg_model = LinearRegression()
reg_model.fit(X_reg, y_reg)
```

Le variabili numeriche più fortemente correlate(positivamente), sono, **Academic Pressure** e **Financial Stress**.

Inizialmente, si era scelto di utilizzare una **Regressione logica** tra Depression e Academic Pressure, vista la forte correlazione e essendo Depression una variabile binaria.
Si è optato infine per una **Regressione Lineare**.

#5 REGRESSIONE

STIMA DEI COEFFICIENTI

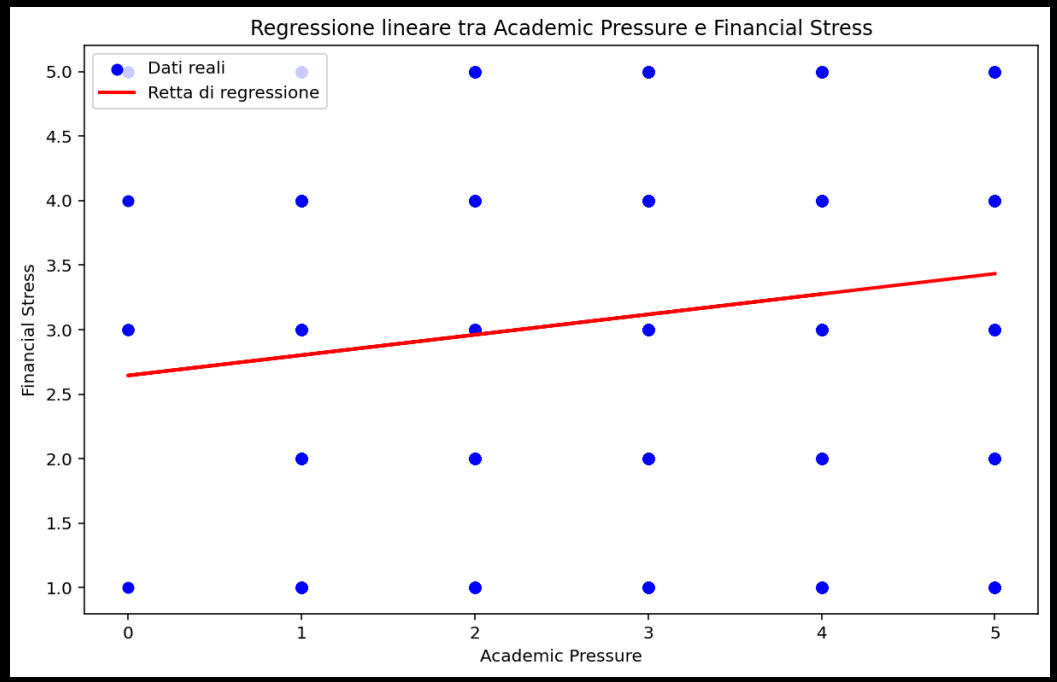
```
#5.1: Stima Dei Coefficienti
intercept = reg_model.intercept_
coef = reg_model.coef_[0]
print(f'Intercetto: {intercept:.2f}')
print(f'Coefficiente: {coef:.2f}')
#predizione dei valori di y utilizzando il modello addestrato
y_pred_reg = reg_model.predict(X_reg)
```

```
Intercetto: 2.64
Coefficiente: 0.16
```

- **Intercetto**: punto in cui la retta di Regressione intercetta l'asse y.
- **Coefficiente**: rappresenta la pendenza della retta di regressione

#5 REGRESSIONE

Grafico dei punti e della retta



Il grafico ci permette di confrontare visivamente, come la variabile Financial Stress varia in funzione della variabile Academic Pressure.
La retta di regressione rappresenta una linea retta che meglio rappresenta la relazione tra queste due variabili.

Osservando il grafico è possibile concludere:

- **Scarsa correlazione** tra le due variabili.
- **Pendenza quasi piatta**: minima variazione di Financial stress al variare di Academic Pressure

#5 REGRESSIONE

Calcolo del coefficiente R^2

Detto anche **coefficiente di determinazione** indica quanto bene i dati si adattano al Modello.

- $R^2 = 1$ il modello spiega completamente la variabilità dei dati.
- $R^2 = 0$ il modello non spiega affatto la variabilità dei dati.

```
#5.3: Calcolo del Coefficiente r^2
r2 = r2_score(y_reg, y_pred_reg)
print(f'R^2: {r2:.2f}')
```

```
R^2: 0.02
```

Calcolo del valore di MSE

Mean Squared Error, misura la differenza tra i valori predetti dal modello e i valori reali.
Valuta la **qualità delle previsioni** di un modello.

```
#5.4: Calcolo del Valore di MSE
mse = mean_squared_error(y_reg, y_pred_reg)
print(f'MSE: {mse:.2f}')
```

```
MSE: 2.02
```

#5 REGRESSIONE

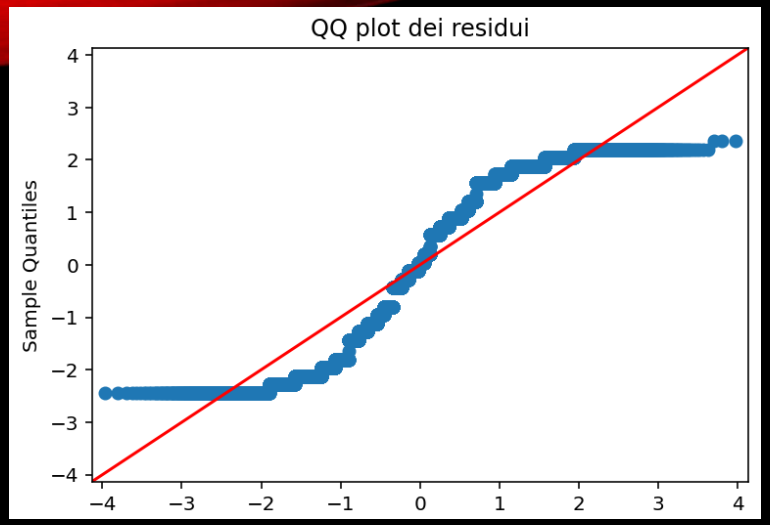
Analisi di normalità dei residui

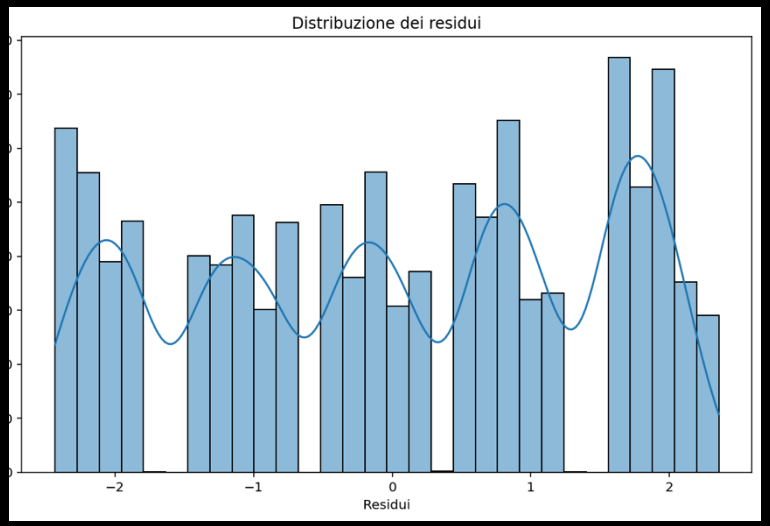
Kolmogorov-Smirnov test, verifica se un campione di dati segue una distribuzione normale.

```
# 5.5: Analisi di Normalità dei Residui con Kolmogorov-Smirnov Test
residui = y_reg - y_pred_reg

# Parametri della distribuzione normale basati sui residui
mean_resid = residui.mean()
std_resid = residui.std()

# Test K-S per verificare la normalità
ks_stat, p_value_ks = kstest(residui, 'norm', args=(mean_resid, std_resid))
if p_value_ks > 0.05:
    print(f"I residui seguono una distribuzione normale (p-value: {p_value_ks:.2f})")
else:
    print(f"I residui non seguono una distribuzione normale (p-value: {p_value_ks:.2f})")
```

```
I residui non seguono una distribuzione normale (p-value: 0.00)
```

#5 REGRESSIONE





I grafici mostrano che I residui seguono una distribuzione non normale essendo p-value uguale a 0.00 (dovrebbe essere almeno 0.05 per avere una distribuzione normale.

- **Residui**: differenza tra dati osservati e valori predetti.
- **Quantile-Quantile plot**: grafico utilizzate per verificare la normalità di una distribuzione di dati.

È possible osservare, che I punti in blu si allontano dalla retta rossa, specialmente alle estremità, ciò suggerisce che residui non seguono una distribuzione normale.

#6 ADDESTRAMENTO DEL MODELLO

```
logistic_model = LogisticRegression(solver='lbfgs', max_iter=1000)
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_val)
#valutazione delle prestazioni del modello di regressione logistica
acc_logistic = accuracy_score(y_val, y_pred_logistic)
print("Acc regressione logistica: {:.3f}".format(acc_logistic))


#SVM con kernel lineare
svm_linear_model = SVC(kernel='linear', C=1)
svm_linear_model.fit(X_train, y_train)
y_pred_svm_linear = svm_linear_model.predict(X_val)
#valutazione delle prestazioni del modello SVC lineare
acc_SVM_linear = accuracy_score(y_val, y_pred_svm_linear)
print("Acc SVM lineare: {:.3f}".format(acc_SVM_linear))

#SVM con kernel polinomiale
svm_poly_model = SVC(kernel='poly', degree=3, C=1)
svm_poly_model.fit(X_train, y_train)
y_pred_svm_poly = svm_poly_model.predict(X_val)
#valutazione delle prestazioni del modello SVC polinomiale
acc_SVM_poly = accuracy_score(y_val, y_pred_svm_poly)
print("Acc SVM polinomiale: {:.3f}".format(acc_SVM_poly))

#SVM con kernel RBF
svm_rbf_model = SVC(kernel='rbf', C=1, gamma='scale')
svm_rbf_model.fit(X_train, y_train)
y_pred_svm_rbf = svm_rbf_model.predict(X_val)
#valutazione delle prestazioni del modello SVC RBF
acc_SVM_rbf = accuracy_score(y_val, y_pred_svm_rbf)
print("Acc SVM rbf: {:.3f}".format(acc_SVM_rbf))
```

Possiamo osservare che i modelli hanno ottenuto risultati simili.
I modelli che hanno performato meglio sono, SVM con kernel Polinomiale e SVM con kernel rbf, entrambi con un'accuratezza del 85,3%.

```
Acc regressione logistica: 0.849
Acc SVM lineare: 0.852
Acc SVM polinomiale: 0.853
Acc SVM rbf: 0.853
```

#7 HYPERPARAMETER TUNING

```
#7.  ____________________________HYPERPARAMETER TUNING____________________________

#addestramento del modello SVM con kernel lineare
param_grid_linear = {
    'C': [0.1,1.0, 10.0]
}
grid_search_linear = GridSearchCV(estimator=SVC(kernel='linear'), param_grid=param_grid_linear, cv=5, n_jobs=-1)
grid_search_linear.fit(X_train, y_train)
best_model_linear = grid_search_linear.best_estimator_
accuracy_linear = best_model_linear.score(X_val, y_val)
print("Acc SVM lineare ottimizzato: {:.3f}".format(accuracy_linear))

#addestramento del modello SVM con kernel polinomiale
param_grid_poly = {
    'C': [0.1,1.0,10.0],
    'gamma': ['scale'],
    'degree': [2,3,4]
}
grid_search_poly = GridSearchCV(estimator=SVC(kernel='poly'), param_grid=param_grid_poly, cv=5, n_jobs=-1)
grid_search_poly.fit(X_train, y_train)
best_model_poly = grid_search_poly.best_estimator_
accuracy_poly = best_model_poly.score(X_val, y_val)
print("Acc SVM polinomiale ottimizzato: {:.3f}".format(accuracy_poly))

#addestramento del modello SVM con kernel rbf
param_grid_rbf = {
    'C': [0.1,1.0,10.0],
    'gamma': ['scale']
}
grid_search_rbf = GridSearchCV(estimator=SVC(kernel='rbf'), param_grid=param_grid_rbf, cv=5)
grid_search_rbf.fit(X_train, y_train)
best_model_rbf = grid_search_rbf.best_estimator_
accuracy_rbf = best_model_rbf.score(X_val, y_val)
print("Acc SVM rbf ottimizzato: {:.3f}".format(accuracy_rbf))
```

```
Acc SVM lineare ottimizzato: 0.852
Acc SVM polinomiale ottimizzato: 0.851
Acc SVM rbf ottimizzato: 0.852
```

L'obbiettivo di questa fase, è ottimizzare i parametri, al fine di migliorare le prestazioni del modello.

SVM lineare e SVM rbf, sono i modelli che hanno performato meglio dopo questa fase con 85,2%.

Hyperparameter tuning con più combinazione di parametri risulterebbe troppo costoso a livello computazionale.

#8 VALUTAZIONE DELLE PERFORMANCE

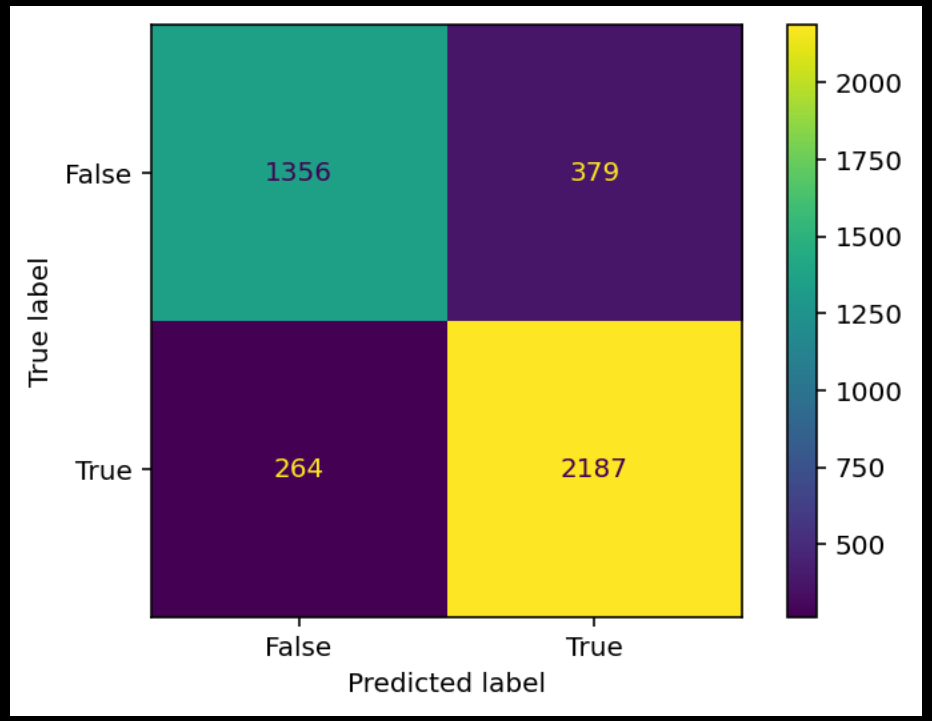
```
acc = {
    "regressione logistica": acc_logistic,
    "SVM lineare": accuracy_linear,
    "SVM polinomiale": accuracy_poly,
    "SVM rbf": accuracy_rbf
}
#il nome del miglior modello e di conseguenza la sua accuratezza
best_model_name = max(acc, key=acc.get)
best_model_acc = acc[best_model_name]
print("Il miglior modello è " + best_model_name + " con un'accuratezza di {:.3f}".format(best_model_acc))
#valutazione delle performance con il miglior modello
if best_model_name == "regressione logistica":
    best_model = logistic_model
elif best_model_name == "SVM lineare":
    best_model = grid_search_linear
elif best_model_name == "SVM polinomiale":
    best_model = grid_search_poly
elif best_model_name == "SVM rbf":
    best_model = grid_search_rbf
y_test_pred = best_model.predict(X_final_test)
```

Dopo aver determinato il modello che ha performato meglio, lo si è utilizzato per fare previsioni sul **Test Set**(fino ad ora nascosto ai modelli).

Il miglior modello è SVM con **kernel rbf**.

```
Il miglior modello è SVM rbf con un'accuratezza di 0.852
```

#8 VALUTAZIONE DELLE PERFORMANCE



La **Confusion Matrix**, è utilizzata per valutare le performance di un modello.

- **True Positives**: casi positivi correttamente classificati come tali.
- **True Negatives**: casi negativi correttamente classificati come tali.
- **False Positives**: casi negativi erroneamente classificati come positivi.
- **False Negatives**: casi positivi erroneamente classificati come negativi.

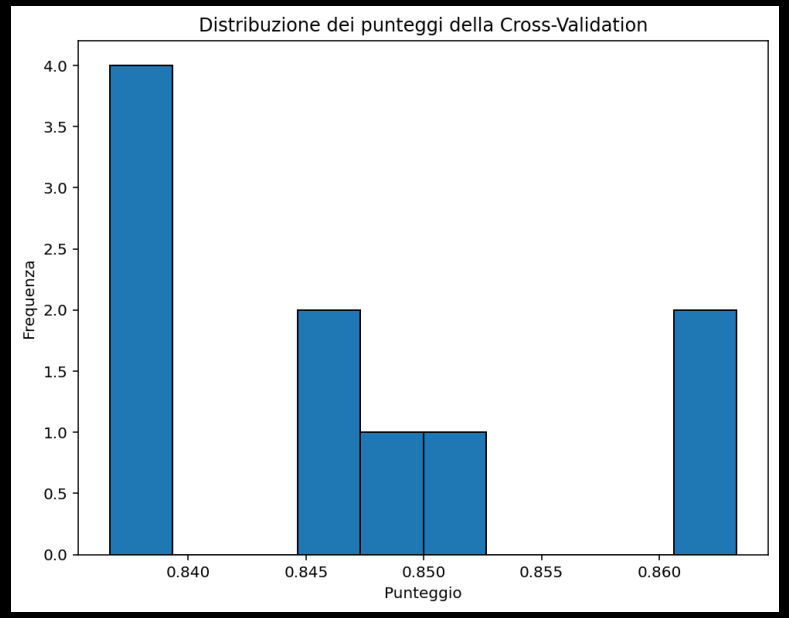
#9 STUDIO STATISTICO DEI RISULTATI DELLA VALUTAZIONE

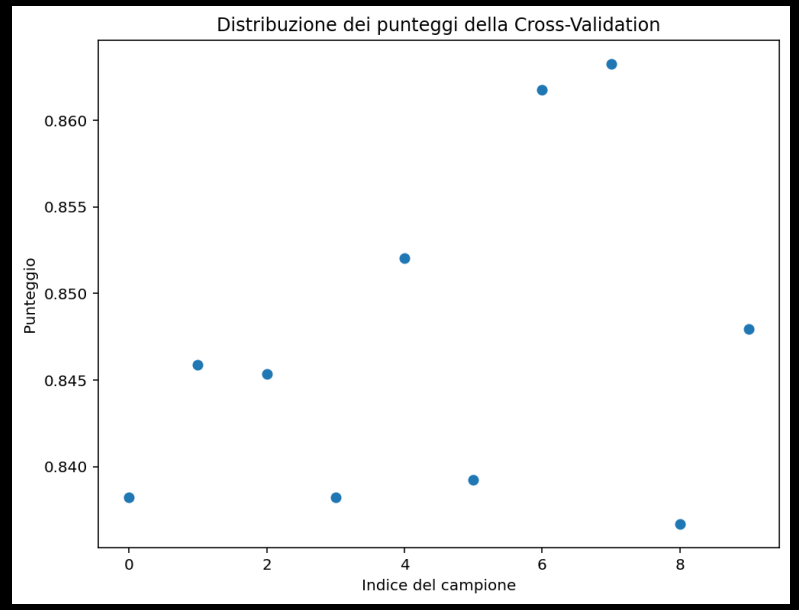
```python
#9: Studio Statistico dei Risultati della Valutazione
#esecuzione della cross-validation con k ripetizioni (k >= 10)
k = 10
scores = cross_val_score(best_model_linear, X_train, y_train, cv=k,

#calcolo delle metriche statistiche
mean_score = np.mean(scores)
std_score = np.std(scores)
median_score = np.median(scores)
min_score = np.min(scores)
max_score = np.max(scores)
range_score = max_score - min_score

#calcolo dell'intervallo di confidenza al 95%
alpha = 0.05
n = len(scores)
mean_std = std_score / np.sqrt(n)
#valore critico per k=10, alpha=0.05
t_value = 2.262
lower_bound = mean_score - t_value * mean_std
upper_bound = mean_score + t_value * mean_std

#creazione del grafico istogramma
plt.figure(figsize=(8, 6))
plt.hist(scores, bins=10, edgecolor='black')
plt.xlabel("Punteggio")
plt.ylabel("Frequenza")
plt.title("Distribuzione dei punteggi della Cross-Validation")
plt.show()
```

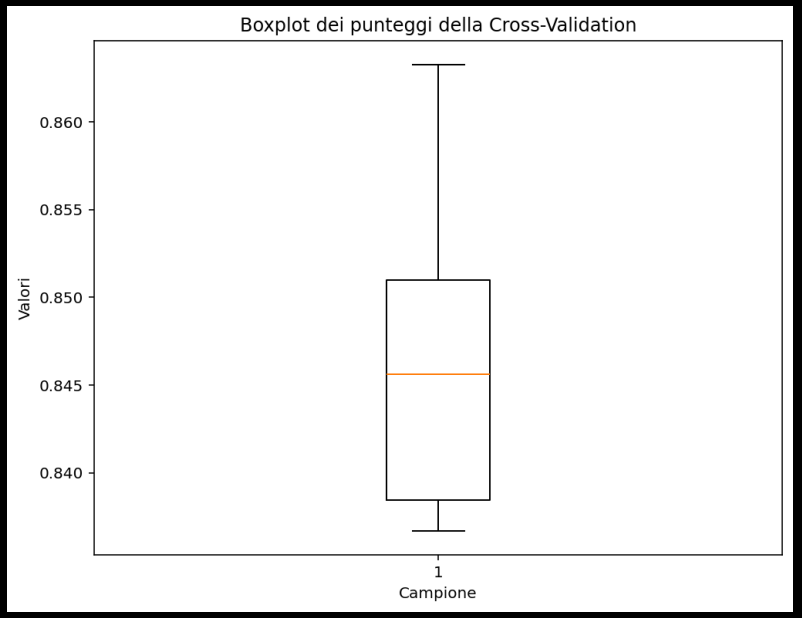
```python
#creazione del grafico scatter
plt.figure(figsize=(8, 6))
plt.scatter(range(len(scores)), scores)
plt.xlabel("Indice del campione")
plt.ylabel("Punteggio")
plt.title("Distribuzione dei punteggi della Cross-Validation")
plt.show()

#creazione del boxplot
plt.figure(figsize=(8, 6))
plt.boxplot(scores)
plt.xlabel("Campione")
plt.ylabel("Valori")
plt.title("Boxplot dei punteggi della Cross-Validation")
plt.show()

#stampa dei risultati statistici
print(f"Media: {mean_score:.3f}")
print(f"Deviazione standard: {std_score:.3f}")
print(f"Mediana: {median_score:.3f}")
print(f"Minimo: {min_score:.3f}")
print(f"Massimo: {max_score:.3f}")
print(f"Intervallo: {range_score:.3f}")
print(f"Intervallo di confidenza al 95%: [{lower_bound:.3f}, {upp
```

```
Media: 0.847
Deviazione standard: 0.009
Mediana: 0.846
Minimo: 0.837
Massimo: 0.863
Intervallo: 0.027
Intervallo di confidenza al 95%: [0.840, 0.853]
```

#9 STUDIO STATISTICO DEI RISULTATI DELLA VALUTAZIONE







I seguenti grafici aiutano ad avere una rappresentazione visiva della distribuzione e della variabilità dei punteggi della **cross-validation** (tecnica per valutare prestazioni di un modello).

FINE

DINDANE JEAN BAPTISTE
MATRICOLA (0001100694)