CAR.CSV

Analisi Statistica

CONTENTS

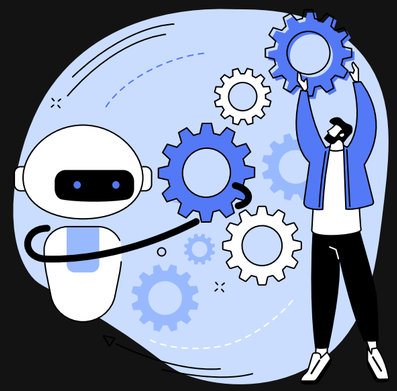
	Buying_Price	Maintenance_Price	No_of_Doors	Person_Capacity	Size_of_Luggage	Safety	Car_Acceptability
2	vhigh	vhigh	2	2	small	low	unacc
3	vhigh	vhigh	2	2	small	med	unacc
4	vhigh	vhigh	2	2	small	high	unacc
5	vhigh	vhigh	2	2	med	low	unacc
6	vhigh	vhigh	2	2	med	med	unacc
7	vhigh	vhigh	2	2	med	high	unacc
8	vhigh	vhigh	2	2	big	low	unacc
370	vhigh	low	3	4	big	high	acc
371	vhigh	low	3	more	small	low	unacc
372	vhigh	low	3	more	small	med	unacc
373	vhigh	low	3	more	small	high	acc
374	vhigh	low	3	more	med	low	unacc
375	vhigh	low	3	more	med	med	acc
376	vhigh	low	3	more	med	high	acc
377	vhigh	low	3	more	big	low	unacc
378	vhigh	low	3	more	big	med	acc

```
[5 rows x 7 columns]
Buying_Price          0
Maintenance_Price     0
No_of_Doors           0
Person_Capacity       0
Size_of_Luggage       0
Safety                0
Car_Acceptability     0
```

Presentazione del dataset, composizione:

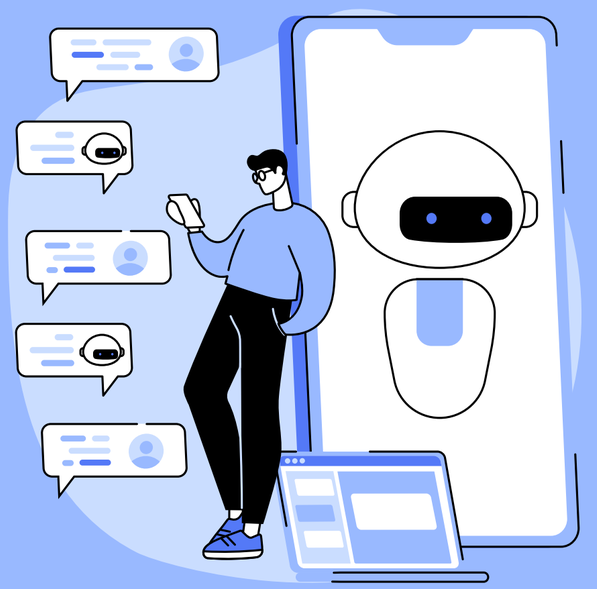
Fase PRE-PROCESSING:

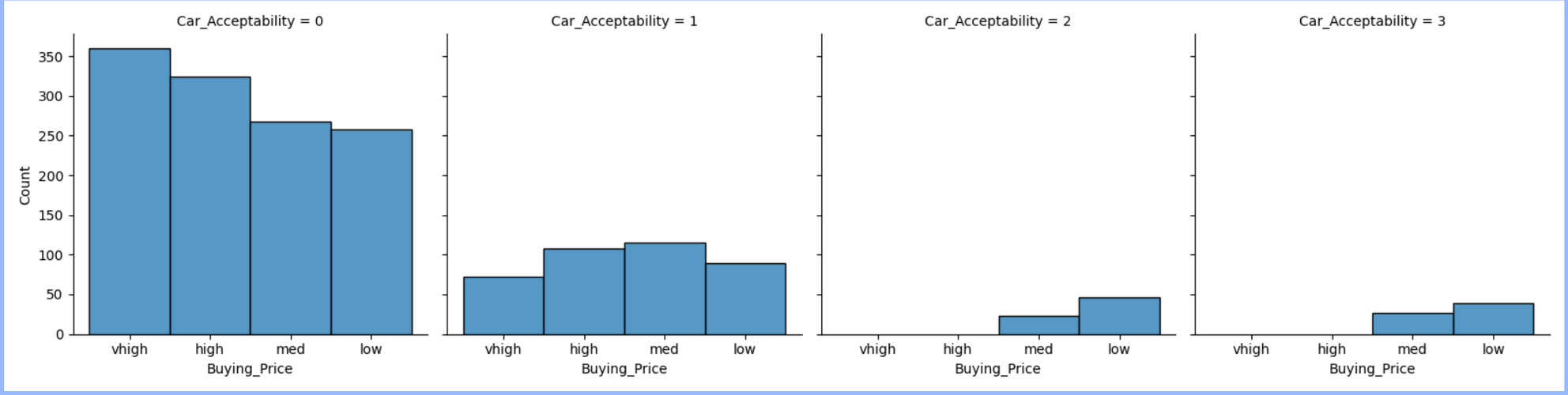
```python
print(dataframe.shape)
print(dataframe.head())
print(dataframe.isnull().sum())



# tutto ottimo, procediamo con il controllo dei valori fuori soglia

min_limit = 0  # Valore minimo accettabile generale, potremmo farne uno specifico ma teniamoci indicativi
max_limit = 5  # Valore massimo accettabile per un dataset sulle auto

# Seleziona solo le colonne numeriche
numeric_columns = ['No_of_Doors', 'Person_Capacity']

# Itera sulle colonne numeriche e controlla i valori fuori soglia
for column in numeric_columns:
    if column == 'No_of_Doors':
        outliers = dataframe[~dataframe[column].str.contains('5more')]
    else:
        outliers = dataframe[(dataframe[column] < min_limit) | (dataframe[column] > max_limit)]
        if not outliers.empty:
            print(f"Variabile: {column}")
            print(outliers)
            print("\n")
```



Fase di SPLITTING:

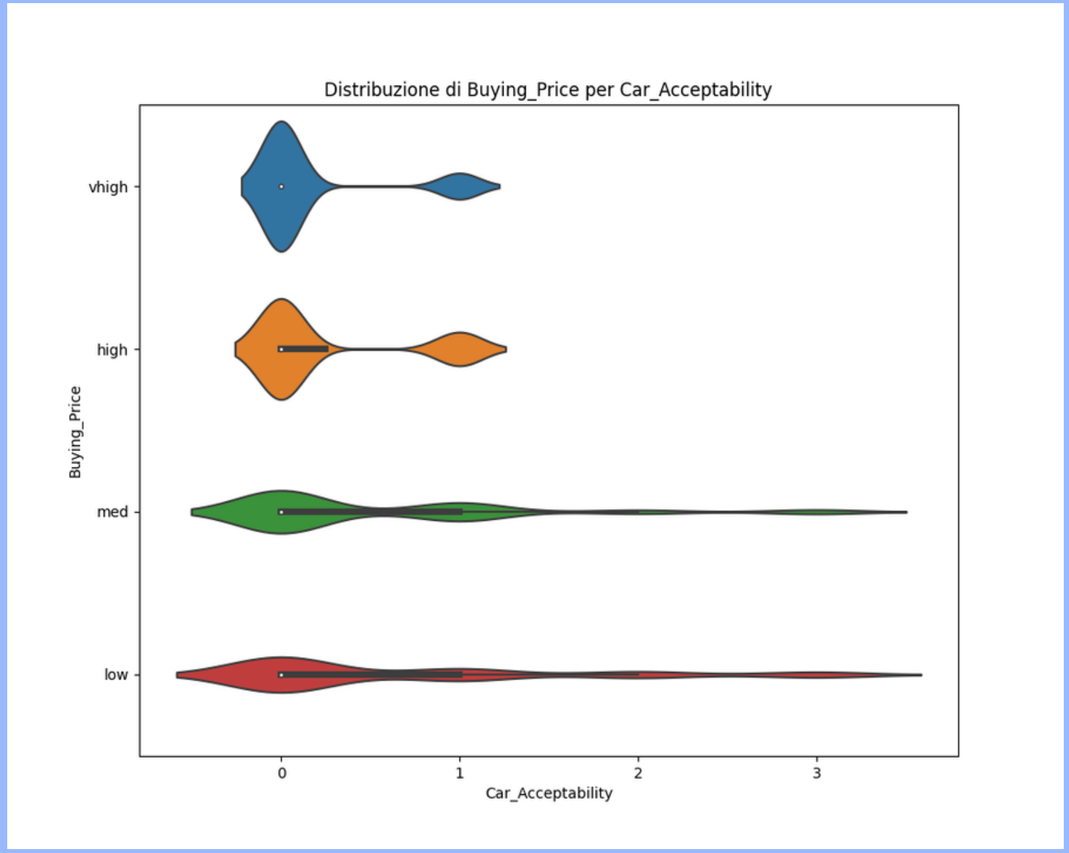
```
132  # tutto andato a buon fine, ma prima di procedere con lo splitting
133  print(dataframe.head())
134
135  X = dataframe.drop(columns = 'Car_Acceptability')
136  y = dataframe['Car_Acceptability']
137  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 42)
138
```





FASE EDA:

Possiamo notare con una rapida analisi EDA con i boxplot che al diminuire della possibilità di acquisto da parte di un utente finale corrisponda un'accettabilità del veicolo stesso più elevata, una visione più dinamica può esser vista nel seguente grafico interattivo: localhostlink



Addestramento dei modelli e confronto dei due algoritmi:

```python
#modello regressione logistica
modelloLogistica = LogisticRegression(solver='lbfgs', max_iter=1000)
modelloLogistica.fit(X_train, y_train)
# Definisci il modello SVC
model = SVC()
model.fit(X_train,y_train)

# Effettua predizioni sui dati di test
y_pred_logistic = modelloLogistica.predict(X_test)
#Adesso predizioni Svc
y_pred_svc = model.predict(X_test)
# Valuta le prestazioni del modello
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
print("Accuracy del modello di regressione logistica:", accuracy_logistic)
classification_report_logistic = classification_report(y_test, y_pred_logistic)
print("Report di classificazione del modello di regressione logistica:")
print(classification_report_logistic)
#ADESSO PARTE SVC
accuracy_SVC= accuracy_score(y_test,y_pred_svc)
print("Accuracy del modello SVC:", accuracy_SVC)
classification_report_SVC = classification_report(y_test, y_pred_svc)
print("Report di classificazione del modello SVC:")

```

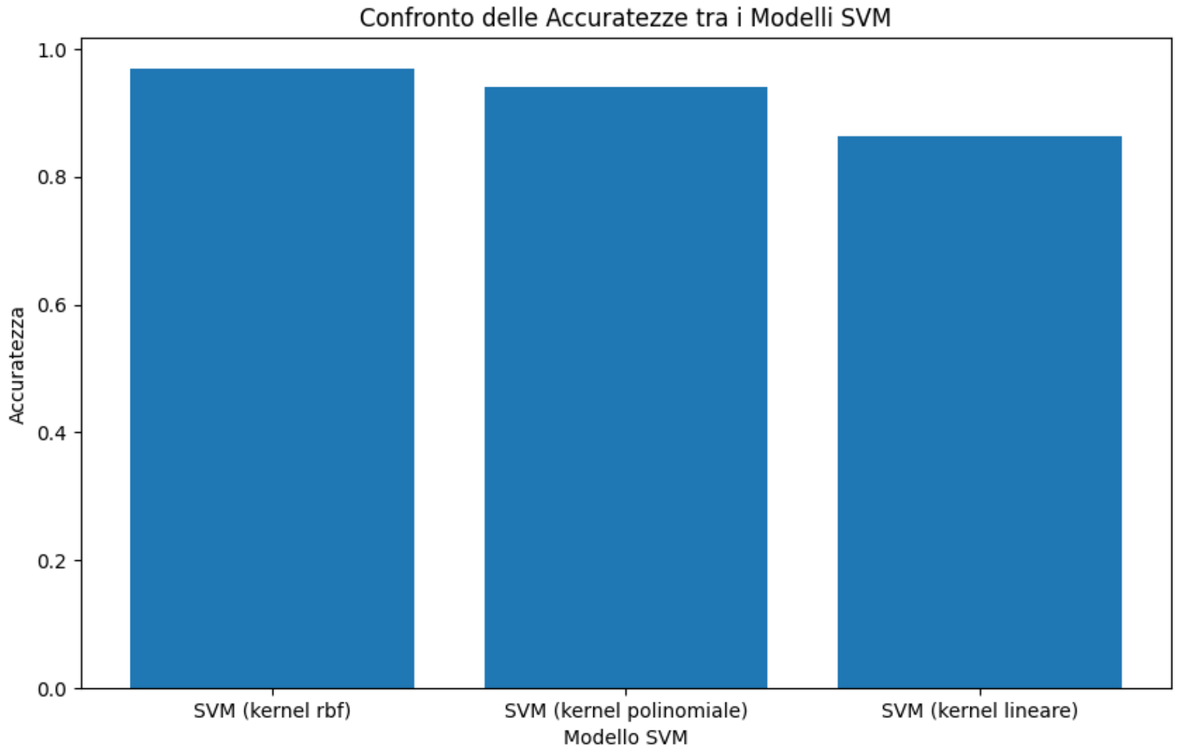
```
[5 rows x 7 columns]
Accuracy del modello di regressione logistica: 0.8458574181117534
Report di classificazione del modello di regressione logistica:
              precision    recall  f1-score   support

           0       0.90      0.94      0.92       358
           1       0.70      0.64      0.67       118
           2       0.62      0.42      0.50        19
           3       0.83      0.79      0.81        24

    accuracy                           0.85       519
   macro avg       0.76      0.70      0.72       519
weighted avg       0.84      0.85      0.84       519

Accuracy del modello SVC: 0.928709055876686
Report di classificazione del modello SVC:
Media: 0.8895
Mediana: 0.8786
Minimo: 0.8266
Massimo: 0.9827
Intervallo: 0.1561
Deviazione standard: 0.0435
```

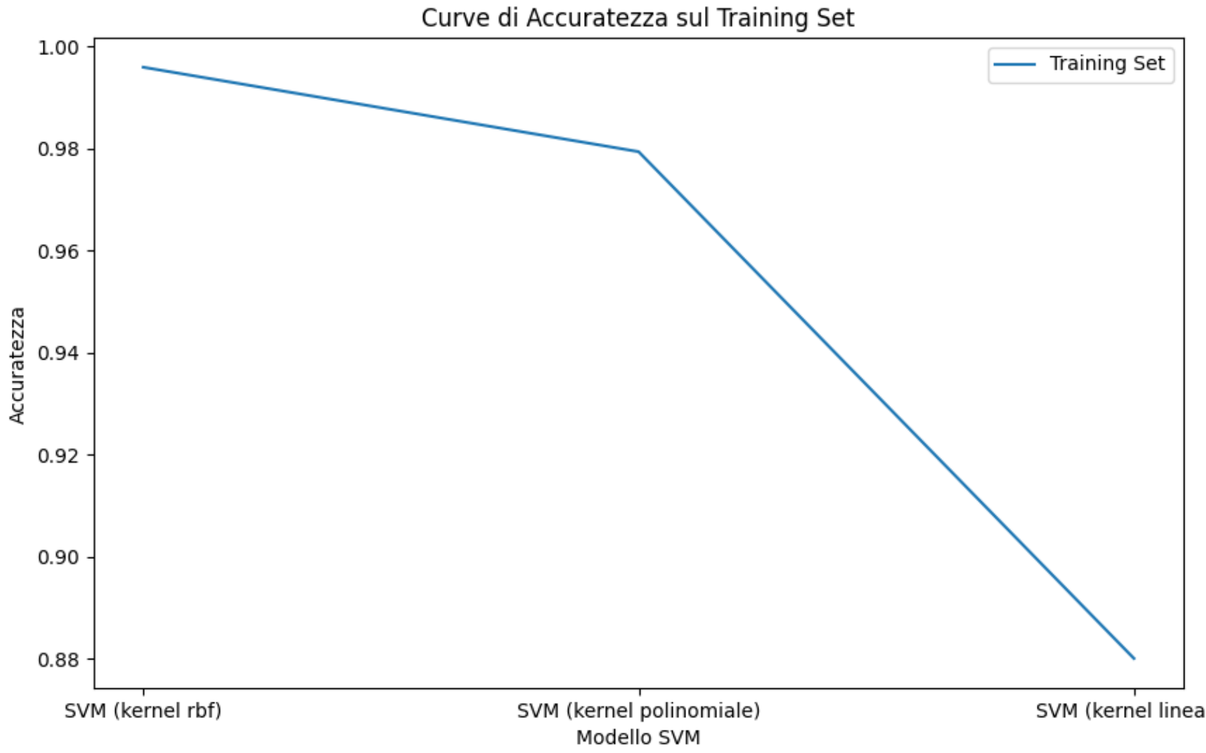
Hyperparameter tuning:

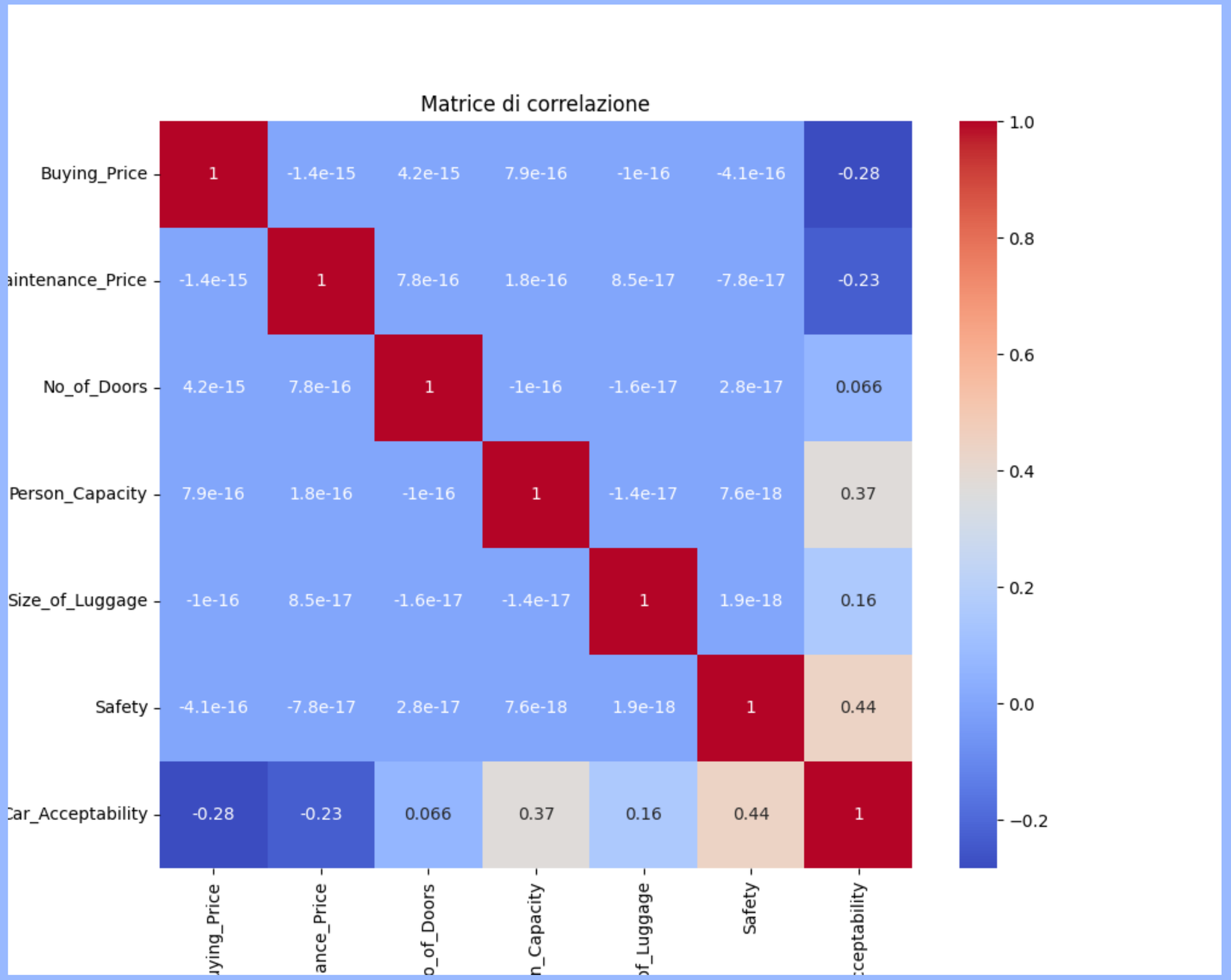
```
#Addestramento del modello SVM con kernel rbf
param_grid_rbf = {
    'C': [0.1, 1.0, 10.0],
    'gamma': ['scale', 'auto']
}
grid_search_rbf = GridSearchCV(estimator=SVC(kernel='rbf'), param_grid=param_grid_rbf, cv=5)
grid_search_rbf.fit(X_train, y_train)
best_model_rbf = grid_search_rbf.best_estimator_
accuracy_rbf = best_model_rbf.score(X_test, y_test)

# Addestramento del modello SVM con kernel polinomiale
param_grid_poly = {
    'C': [0.1, 1.0, 10.0],
    'gamma': ['scale', 'auto'],
    'degree': [2, 3, 4]
}
grid_search_poly = GridSearchCV(estimator=SVC(kernel='poly'), param_grid=param_grid_poly, cv=5)
grid_search_poly.fit(X_train, y_train)
best_model_poly = grid_search_poly.best_estimator_
accuracy_poly = best_model_poly.score(X_test, y_test)

# 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)
grid_search_linear.fit(X_train, y_train)
best_model_linear = grid_search_linear.best_estimator_
accuracy_linear = best_model_linear.score(X_test, y_test)

```

```
plt.bar(labels, accuracies)
plt.xlabel('Modello SVM')
plt.ylabel('Accuratezza')
plt.title('Confronto delle Accuratezze tra i Modelli SVM')
plt.show()

# Calcolo delle accuratezze dei modelli sul training set
accuracy_rbf_train = best_model_rbf.score(X_train, y_train)
accuracy_poly_train = best_model_poly.score(X_train, y_train)
accuracy_linear_train = best_model_linear.score(X_train, y_train)

# Creazione dei vettori di accuratezza
accuracies_train = [accuracy_rbf_train, accuracy_poly_train, accuracy_linear_train]


# Etichette per i modelli
labels = ['SVM (kernel rbf)', 'SVM (kernel polinomiale)', 'SVM (kernel lineare)']

# Grafico delle curve di accuratezza per il training set
plt.figure(figsize=(10, 6))
plt.plot(accuracies_train, label='Training Set')
plt.xticks(range(len(labels)), labels)
plt.xlabel('Modello SVM')
plt.ylabel('Accuratezza')
plt.title('Curve di Accuratezza sul Training Set')
plt.legend()
plt.show()


# Esegui la validazione incrociata con k ripetizioni (k >= 10)
k = 10
scores = cross_val_score(model, X, y, cv=k, scoring='accuracy')
```

Plot in cui risulta evidente una maggiore accuratezza del modello kernel rbf per il nostro dataset.





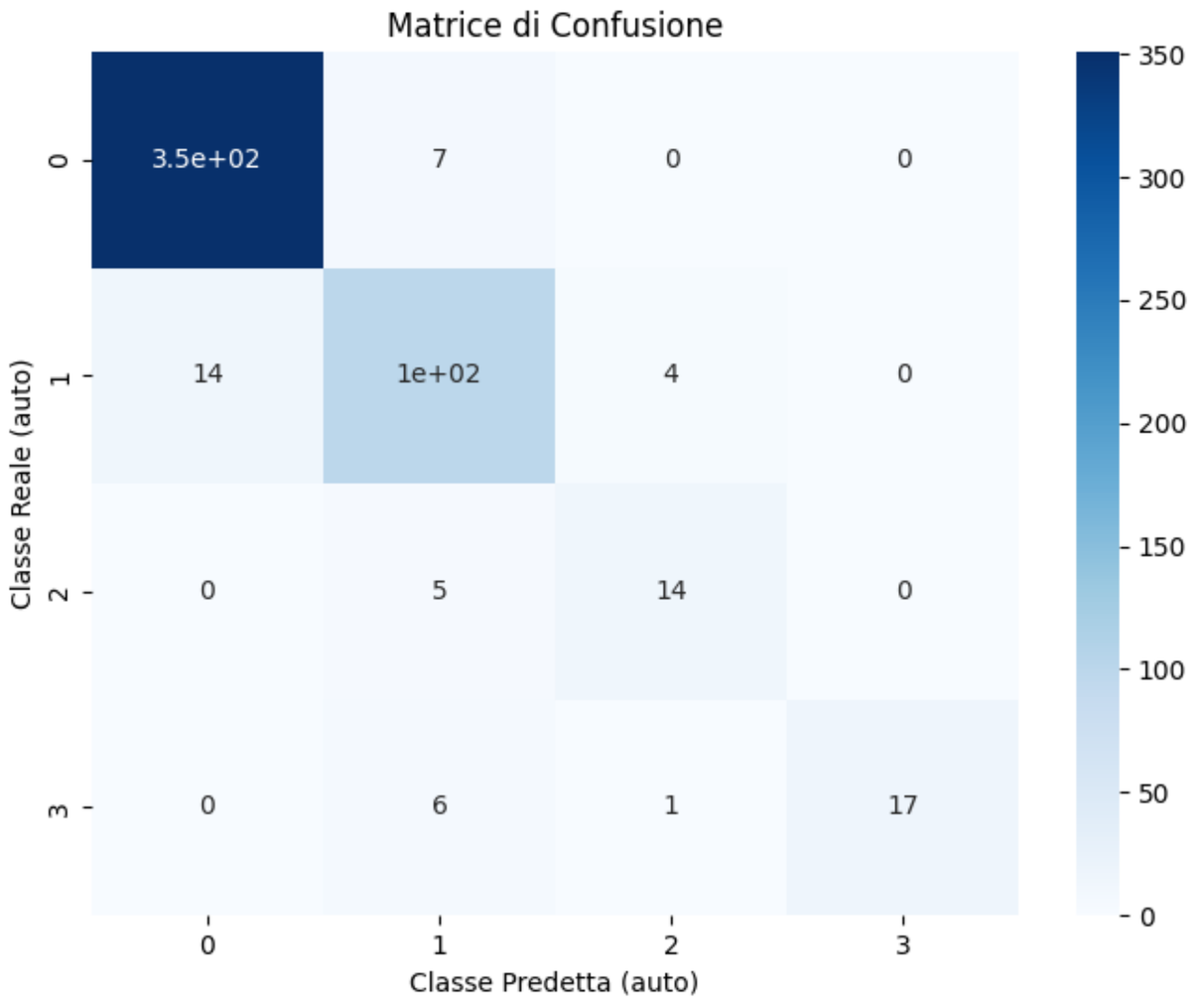


La matrice di correlazione ci consente di individuare le variabili che sono correlate tra loro. Una correlazione positiva significa che le variabili aumentano o diminuiscono insieme, mentre una correlazione negativa significa che le variabili si muovono in direzioni opposte.

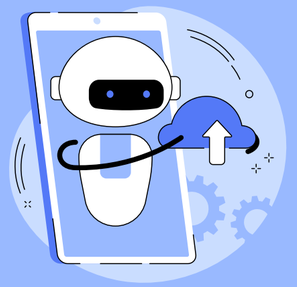
Notiamo che il colore rosso indica correlazione positiva perfetta mentre blu scuro quella negativa perfetta, ovvero rispettivamente 1 e -1

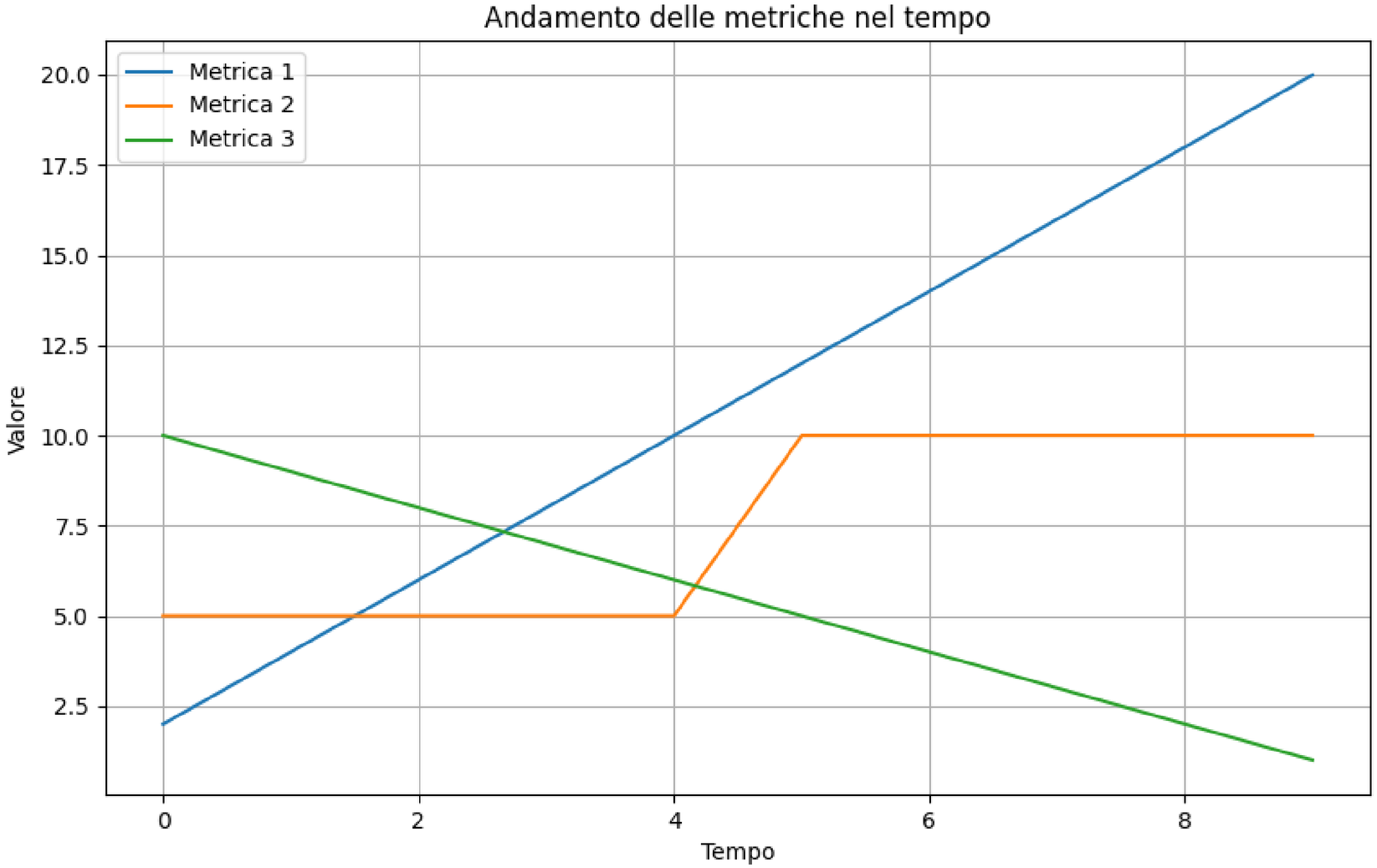
la correlazione non implica causalità. È importante considerare il contesto del tuo dataset e valutare la relazione tra le variabili in base alla conoscenza di dominio e agli obiettivi dell'analisi.

Valutazione delle performance con matrice di confusione.



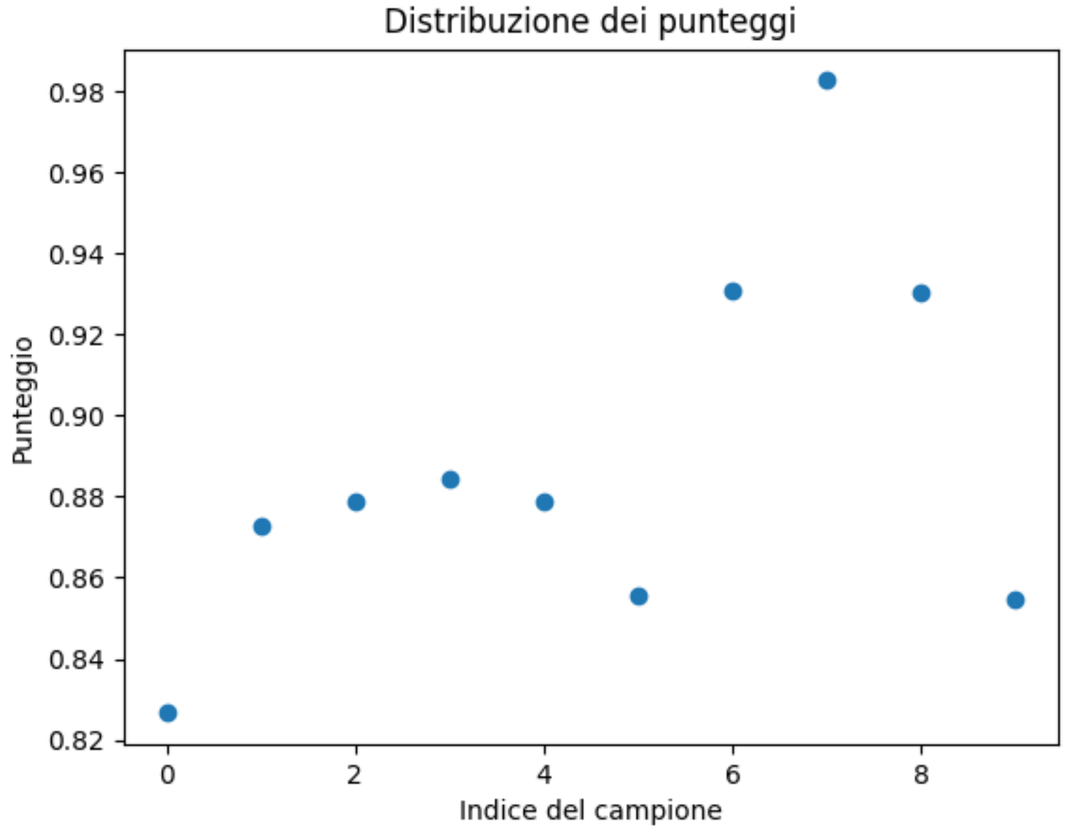
Come notiamo abbiamo in posizione 0-0 ben 351 istanze che appartengono alla classe reale "unacc" e sono state predette correttamente come "unacc" (True Negatives, TN). Ci sono anche 7 istanze di classe "unacc" che sono state predette erroneamente come altre classi. Non ci sono istanze di altre classi che sono state predette come "unacc" (valori 0 nella riga).

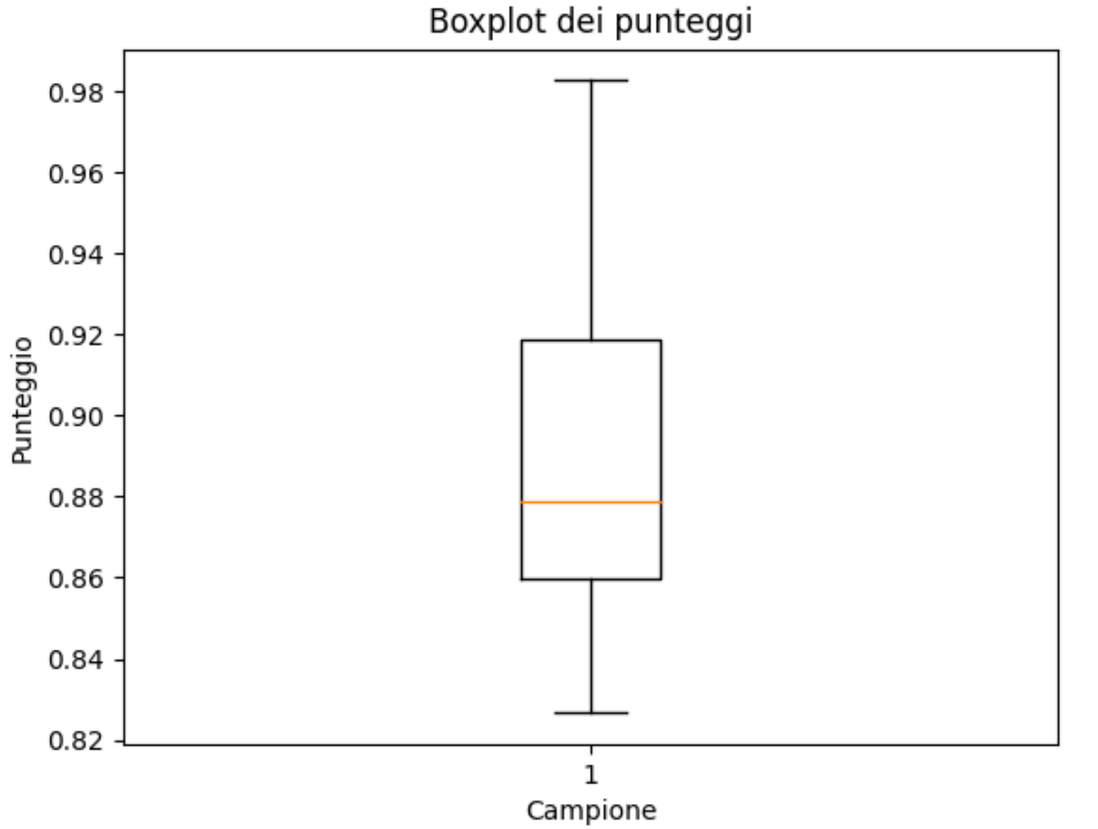



Andamento delle metriche nel tempo
Metrica 1
Metrica 2
Metrica 3
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
Valore
0
2
4
6
8
Tempo

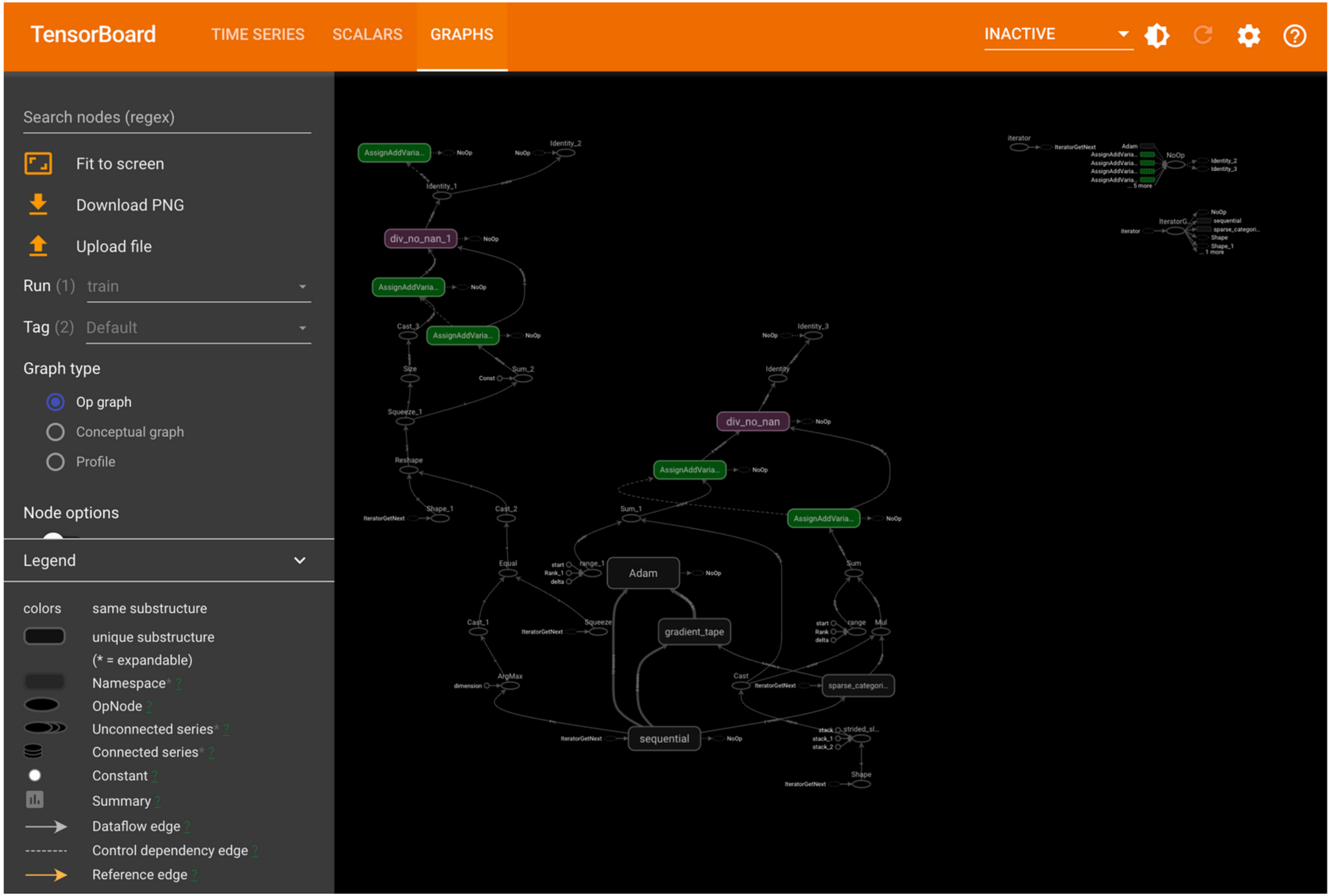
Studio statistico sui risultati della valutazione:

```
Media: 0.8895
Mediana: 0.8786
Minimo: 0.8266
Massimo: 0.9827
Intervallo: 0.1561
Deviazione standard: 0.0435
Intervallo di confidenza al 95%: [0.8584, 0.9206]
```





Applicazione di Rete Neurale con Tensorflow:



PROGETTO DI MASSIMO ANDRES ROJAS CAMACHO.

NUMERO MATRICOLA: 0001019759

MASSIMO.ROJASCAMACHO@STUDIO.UNIBO.IT