TITANIC

Prevedere la Sopravvivenza con il Machine Learning

INTRODUZIONE AL PROBLEMA

Il progetto si basa sul famoso dataset del Titanic. L'obiettivo è costruire un modello di Machine Learning per prevedere se un passeggero è sopravvissuto o meno al naufragio, basandosi su alcune delle sue caratteristiche.

IL PROCESSO DI MACHINE LEARNING

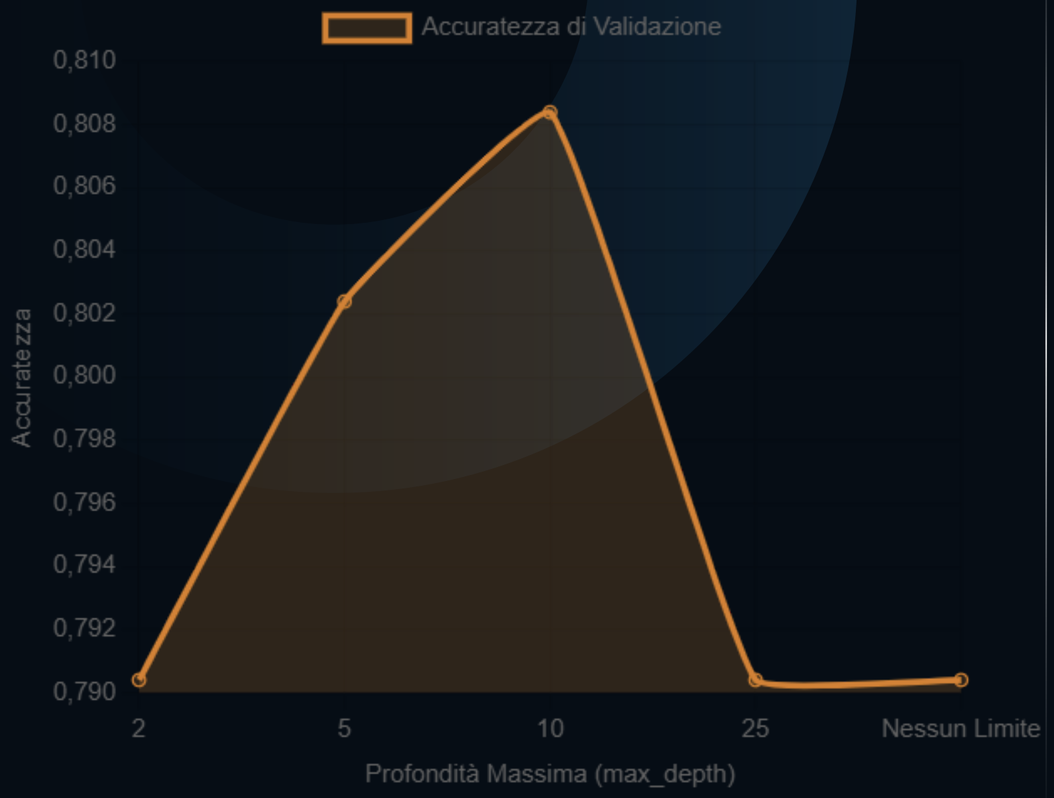
Caricamento e Analisi dei Dati

Ho iniziato caricando il dataset titanic_sub.csv su Pandas e analizzando le sue caratteristiche, inclusa la presenza di valori mancanti nelle colonne "Age" (età) ed "Embarked" (porto di imbarco). La colonna PassengerId è stata usata come indice.

Preprocessing dei Dati

Questa è stata una fase cruciale per rendere i dati utilizzabili dal modello dove ho:

- Gestito i valori mancanti in "Age" con la mediana.
- Gestito i valori mancanti in "Embarked" con la moda.
- Convertito le variabili categoriche. Per "Sex" ho usato LabelEncoder (male = 1, female = 0), mentre per "Embarked" ho utilizzato One-Hot Encoding con get_dummies per creare colonne binarie (es. Embarked_C, Embarked_Q, Embarked_S).



Suddivisione del Dataframe

Per garantire una valutazione robusta del modello, i dati sono stati divisi in:

- Training Set (75%): Usato per addestrare il modello.
- Test Set (25%): Usato per la valutazione finale, mantenuto separato fino alla fine.
- Il Training Set è stato a sua volta suddiviso in Training Set e Validation Set per la fase di ottimizzazione.

Modellazione e Ottimizzazione

- Ho creato un modello DecisionTreeClassifier.
- Per trovare la migliore profondità dell'albero (max_depth), abbiamo testato i valori 2, 5, 10, 25 e la profondità illimitata, utilizzando l'accuratezza come metrica di valutazione sul Validation Set. La profondità ottimale è stata scelta in base a questa analisi, raggiungendo un picco con profondità 10.

CODICE

Preprocessing dei Dati

```
# Gestione valori mancanti in 'Age'
imputer = SimpleImputer(strategy='median')
df['Age'] = imputer.fit_transform(df[['Age']])


# Gestione valori mancanti in 'Embarked'
imputer_cat = SimpleImputer(strategy='most_frequent')
df['Embarked'] = imputer_cat.fit_transform(df[['Embarked']]).ravel()
```

Preprocessing dei Dati

```
# LabelEncoder per 'Sex'
le = LabelEncoder()
df['Sex_encoded'] = le.fit_transform(df['Sex'])
# 'male' diventa 1, 'female' diventa 0


# One-hot encoding per 'Embarked'
dummies = pd.get_dummies(df['Embarked'], prefix='Embarked')

# Aggiungo la colonna one-hot al dataframe originale
df = pd.concat([df, dummies], axis=1)
```

CODICE

Addestramento e Tuning del Modello

```
# Determino i valori per la profondità del decision tree
depth_values = [2, 5, 10, 25, None] # imposto none per profondità massima
results= {}

for depth in depth_values:
    model = DecisionTreeClassifier(max_depth=depth, random_state=0)
    model.fit(X_train, y_train)

    val_preds = model.predict(X_val)
    val_acc = accuracy_score(y_val, val_preds)

    results[depth] = val_acc
    print(f'max depth: {depth}, Validation_accuracy: {val_acc:.4f}')
```

Valutazione del Modello

```
# Addestro il modello sulla profondità migliore e la prima ripartizione del dataset
final_model = DecisionTreeClassifier(max_depth=best_depth, random_state=0)
final_model.fit(X_train_full, y_train_full)

# Valuto il modello finale su test
test_preds = final_model.predict(X_test)
test_acc = accuracy_score(y_test, test_preds)
print(f'Test accuracy with best depth = {best_depth}: {test_acc:.4f}')
```

ACCURATEZZA FINALE DEL MODELLO

78.92 %

Questa è la precisione del nostro albero decisionale nel prevedere la sopravvivenza dei passeggeri sul set di dati di test, dimostrando l'efficacia del modello.

Notebook

https://drive.google.com/file/d/13Dc6thlK_H10X7wN7mQ-eWg_f_tOZ-GL/view?usp=sharing

THANK YOU