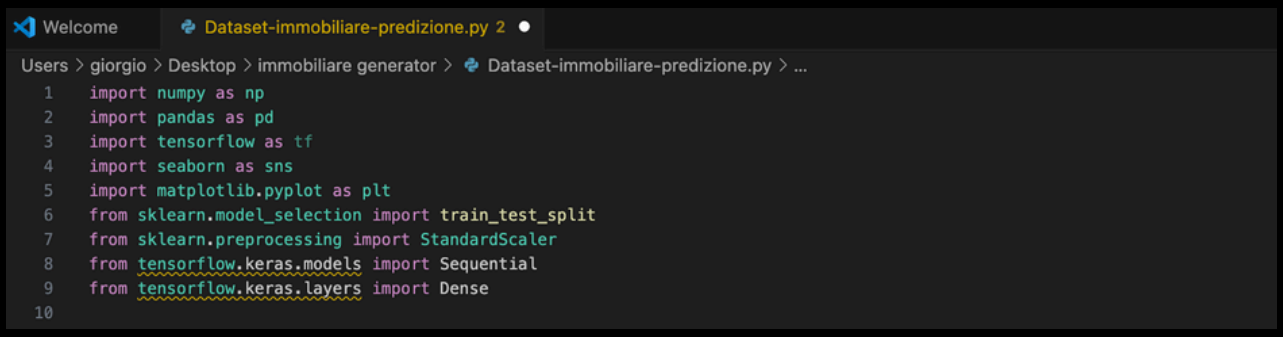
Creazione di un dataset immobiliare e la relativa predizione di un prezzo dati parametri

Questo codice serve a stimare il prezzo di case usando il machine learning. Creiamo un dataset “finto” che simula il mercato immobiliare, con dati come superficie, numero di stanze, età dell’edificio e distanza in km dal centro città.
Prepariamo i dati per addestrare un modello di rete neurale, che impara a calcolare il prezzo delle case basandosi su queste caratteristiche.

Una volta pronto, il modello permette di inserire i dati di una nuova casa per stampare all’utente una stima del relativo prezzo.
Il codice confronta il prezzo predetto con la distribuzione dei prezzi del dataset e calcola come si posiziona rispetto alla media.
È un progetto pratico che combina analisi e interattività, mostrando come il machine learning può essere applicato al settore immobiliare.

Per poter avere dati attendibili, si sarebbe dovuto usare un dataset con valori reali, questo codice serve a mostrare anche come si genera un dataset casuale.



1. Importazione delle librerie

Questa sezione importa librerie essenziali per la manipolazione dei dati, la visualizzazione e l'addestramento del modello:

- numpy: utilizzato per creare array numerici e generare dati sintetici.
- pandas: utilizzato per creare e manipolare il dataset in formato tabellare.
- tensorflow: utilizzato per costruire e addestrare la rete neurale.
- seaborn e matplotlib: utilizzati per creare grafici e visualizzazioni.
- sklearn: utilizzato per dividere il dataset in training e test set e per normalizzare i dati.

```python
# -- CREAZIONE DEL DATASET IMMOBILIARE --

# Generazione di un dataset sintetico con 2000 campioni
n_samples = 2000
superficie = np.random.randint(29, 200, n_samples)  # Superficie in m²
stanze = np.random.randint(1, 4, n_samples)  # Numero di stanze
età_fabbricato = np.random.randint(0, 100, n_samples)  # Età del fabbricato
distanza_centro = np.random.randint(1, 15, n_samples)  # Distanza in km dal centro

# Prezzo sintetico calcolato con una funzione lineare e rumore ridotto
prezzo = (
    100000
    + superficie * 1000
    + stanze * 5000
    - età_fabbricato * 500
    + distanza_centro * 1000
    + np.random.normal(0, 20000, n_samples)  # Rumore ridotto
)

# Creazione del DataFrame
data = pd.DataFrame({
    'Superficie': superficie,
    'Numero di stanze': stanze,
    'Età fabbricato': età_fabbricato,
    'Distanza dal centro': distanza_centro,
    'Prezzo': prezzo
})

# Arrotondamento della colonna Prezzo a 2 decimali
data['Prezzo'] = data['Prezzo'].round(2)

# Visualizza il dataset
print(data)

# Rimozione degli outlier
limite_superiore = data['Prezzo'].quantile(0.99)
data = data[data['Prezzo'] <= limite_superiore]

# Salvataggio del dataset su file CSV
data.to_csv('dataset_case.csv', index=False)

```

2. Creazione del dataset sintetico

il dataset immobiliare è generato utilizzando valori casuali:

- Superficie, stanze, età fabbricato e distanza dal centro: variabili indipendenti generate casualmente con limiti realistici.
- Prezzo: calcolato usando una funzione lineare che simula dati reali.
- Rimozione degli outliers.

Output:

- Dataset salvato in un file CSV (dataset_case.csv) e caricato in un DataFrame pandas.

```python
# Visualizzazione della distribuzione dei prezzi
plt.figure(figsize=(10, 6))
sns.histplot(data['Prezzo'], kde=True, bins=30, color='skyblue')
plt.title("Distribuzione dei Prezzi degli Immobili")
plt.xlabel("Prezzo (€)")
plt.ylabel("Frequenza")
plt.show()
```

```
# -- PREPARAZIONE DEI DATI --

# Separazione di variabili indipendenti (X) e dipendenti (Y)
X = data[['Superficie', 'Numero di stanze', 'Distanza dal centro', 'Età fabbricato']].values
Y = data[['Prezzo']].values

# Suddivisione in training e test set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Normalizzazione dei dati
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

3. Preparazione dei dati

- Separazione: il dataset è diviso in variabili indipendenti (X)
- e dipendenti (Y).
- Divisione train-test: il dataset è suddiviso in training set (80%)
- e test set (20%).
- Normalizzazione: le variabili indipendenti sono scalate usando StandardScaler.

```
# -- CREAZIONE DEL MODELLO RETE NEURALE --

# Definizione del modello di rete neurale
model = Sequential([
    Dense(128, activation='relu', input_dim=4),  # Input layer
    Dense(64, activation='relu'),  # Hidden layer
    Dense(32, activation='relu'),  # Hidden layer
    Dense(1)  # Output layer
])
```

4. Costruzione del modello di rete neurale

Questo codice crea una rete neurale con più strati:

- Input Layer: Accetta 4 caratteristiche (Superficie, numero di stanze, distanza dal centro, età fabbricato) con 128 neuroni e attivazione ReLU.
- Hidden Layer 1: 64 neuroni con attivazione ReLU.
- Hidden Layer 2: 32 neuroni con attivazione ReLU.
- Output Layer: Un neurone senza funzione di attivazione per predire il prezzo come valore continuo.

```python
# -- ADDESTRAMENTO E VALUTAZIONE MODELLO --

# Compilazione del modello con una funzione di perdita meno sensibile agli outlier
model.compile(optimizer='adam', loss='mean_absolute_error')

# Addestramento del modello
model.fit(X_train, Y_train, epochs=500, batch_size=64, validation_data=(X_test, Y_test), verbose=1)

# Valutazione sul test set
loss = model.evaluate(X_test, Y_test, verbose=0)
print(f"Mean Absolute Error sul Test Set: {loss:.2f}€")

```

5. Addestramento e valutazione del modello

- Compilato: Si definisce l'ottimizzatore (adam) e la funzione di perdita (mean_absolute_error), adatta per problemi di regressione e meno sensibile agli outlier rispetto alla media quadratica.
- Addestrato: Il modello apprende dai dati normalizzati del training set, iterando per 500 epoche con batch da 64 elementi. Durante l'addestramento, viene monitorata la performance sui dati di validazione (test set).
- Valutato: Dopo l'addestramento, il modello viene testato sul test set per calcolare il Mean Absolute Error (MAE), che indica l'errore medio nella predizione dei prezzi degli immobili.

```python
# -- PREDIZIONE CON INPUT UTENTE --

# Funzione per fare predizioni basate su input utente
def input_utente():
    superficie = float(input("Inserisci la superficie in mq: "))
    stanze = int(input("Inserisci il numero di stanze: "))
    età_fabbricato = int(input("Inserisci l'età del fabbricato: "))
    distanza_centro = float(input("Inserisci la distanza dal centro in km: "))
    return np.array([[superficie, stanze, distanza_centro, età_fabbricato]])

# Predizione per un nuovo immobile
nuovo = input_utente()
nuovo_normalizzato = scaler.transform(nuovo)
predizione = model.predict(nuovo_normalizzato)
print(f"Prezzo predetto: {predizione[0][0]:.2f}€")
```

6. Predizione con input utente

Funzione input_utente:

- Chiede all'utente di inserire i dettagli di una proprietà.

Normalizzazione:

- I dati immessi sono normalizzati con lo stesso scaler usato in precedenza.

Predizione:

- Il modello predice un prezzo basandosi sui dati in input.

```python
# -- ANALISI: PREDIZIONE VS DATASET  --

plt.figure(figsize=(10, 6))
sns.histplot(data['Prezzo'], kde=True, bins=30, color='green', label='Prezzi dataset')
plt.axvline(predizione[0][0], color='red', linestyle='--', linewidth=2, label=f"Predizione Utente: {predizione[0][0]:.2f}€")
plt.title("Distribuzione dei Prezzi Reali e Prezzo Predetto")
plt.xlabel("Prezzo (€)")
plt.ylabel("Frequenza")
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

```

7. Analisi: Distribuzione dei Prezzi Reali vs Prezzo Predetto

Questo blocco di codice crea un grafico che confronta la distribuzione dei prezzi reali del dataset con il prezzo predetto dall'utente.

- Obiettivo: Mostrare visivamente come il prezzo predetto si posiziona rispetto alla distribuzione dei prezzi reali.
- Funzionalità principali:
- Usa un istogramma con KDE (stima della densità Kernel) per rappresentare i prezzi del dataset.
- Disegna una linea verticale per evidenziare il prezzo predetto.

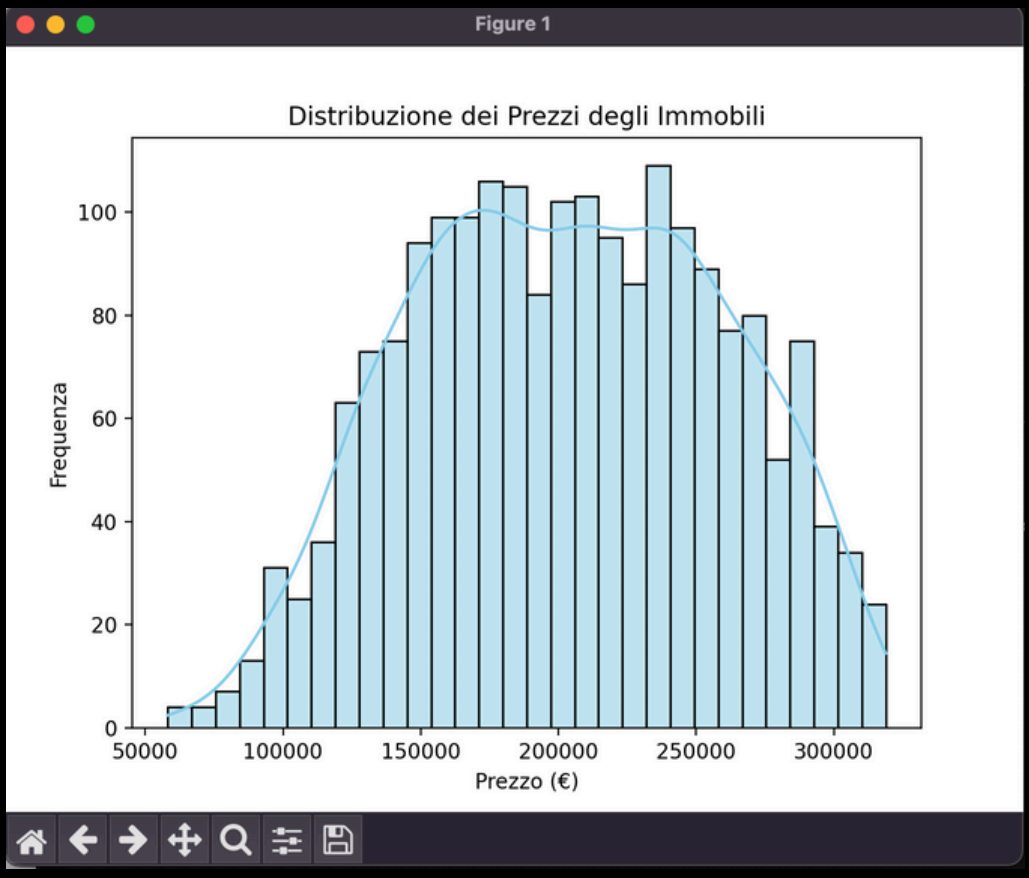
```python
# --  PREZZO PREDETTO RISPETTO ALLA MEDIA --

media_prezzo = data['Prezzo'].mean()
scarto = predizione[0][0] - media_prezzo
if scarto > 0:
    print(f"Il prezzo predetto è superiore alla media del dataset di {scarto:.2f}€ (Media: {media_prezzo:.2f}€).")
elif scarto < 0:
    print(f"Il prezzo predetto è inferiore alla media del dataset di {abs(scarto):.2f}€ (Media: {media_prezzo:.2f}€).")
else:
    print(f"Il prezzo predetto è esattamente uguale alla media del dataset ({media_prezzo:.2f}€).")

```

8. Analisi: Prezzo Predetto rispetto alla Media del Dataset

Questo blocco di codice calcola la differenza tra il prezzo predetto e la media del dataset, quindi fornisce un'interpretazione testuale.

- Informa l'utente se il prezzo predetto è maggiore, minore o uguale alla media e di quanto.
- Calcola la media dei prezzi nel dataset.
- Determina lo scarto tra il prezzo predetto e la media.
- Stampa lo scarto tramite un “ if “ a seconda che sia maggiore, minore o uguale della media del dataset.

Esempio funzionamento del codice:



Distribuzione dei Prezzi delle proprietà

Asse X: Rappresenta il prezzo delle proprietà in euro.

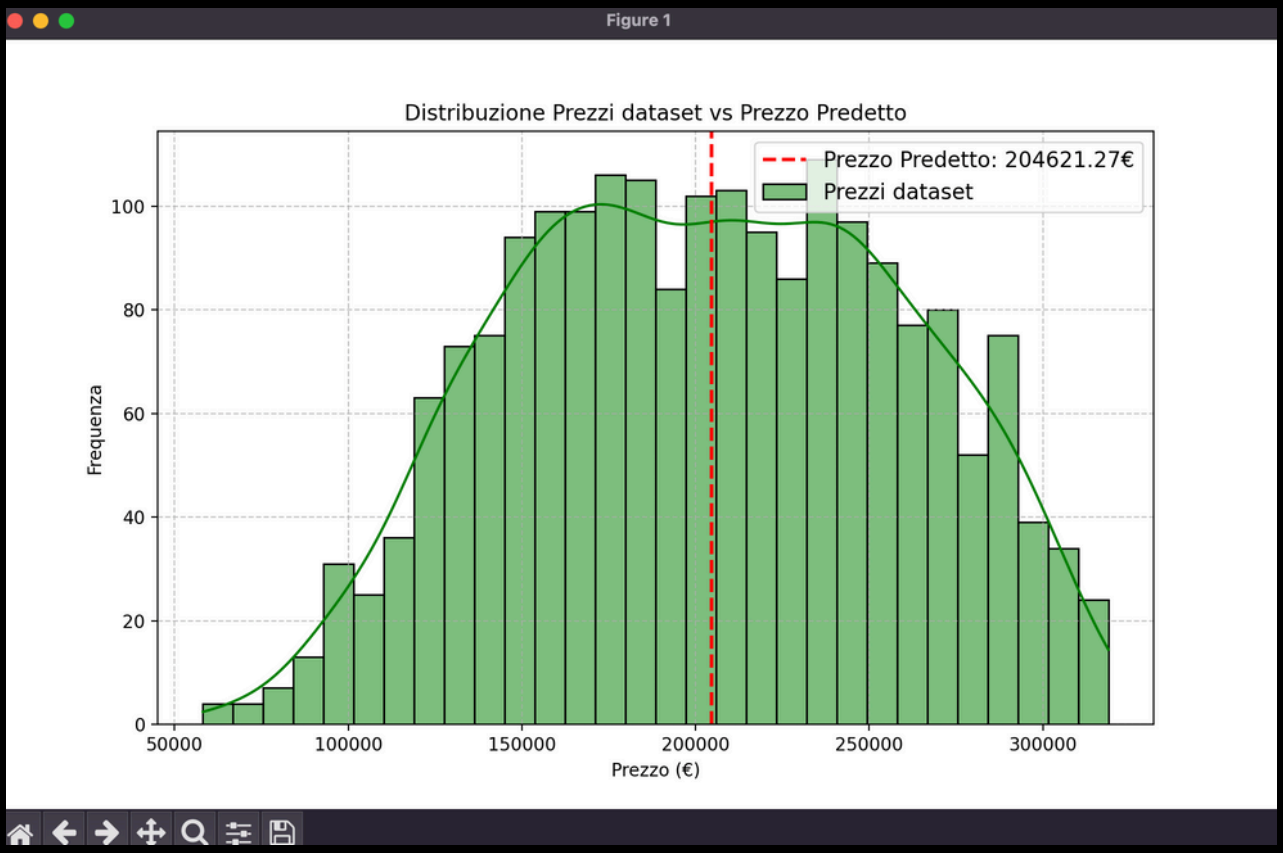
Asse Y: Mostra la frequenza delle proprietà per ciascun intervallo di prezzo.

Osservazioni Principali:

La maggior parte dei prezzi delle proprietà si concentra tra 150.000€ e 250.000€.

```
Epoch 500/500
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 15526.1475 - val_loss: 16266.5850
Mean Absolute Error sul Test Set: 16266.58€
Inserisci la superficie in mq: 100
Inserisci il numero di stanze: 2
Inserisci l'età del fabbricato: 15
Inserisci la distanza dal centro in km: 5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
Prezzo predetto: 204621.27€
```

in questo esempio abbiamo dato in input:
100 come metrature, 2 come numero di stanze, 15 come età del fabbricato, 5 come i km di distanza dal centro.
l'output è stato un prezzo predetto di 201 810.20€



Distribuzione dei Prezzi Reali e Prezzo Predetto.

Asse X: Rappresenta i prezzi delle proprietà in euro.

Asse Y: Indica la frequenza delle proprietà per ciascun intervallo di prezzo.

Osservazioni Principali:

L'istogramma mostra come sono distribuiti i prezzi delle proprietà nel dataset. La maggior parte dei prezzi si concentra in un intervallo ben definito, evidenziando una tendenza centrale.

La linea rossa tratteggiata rappresenta il prezzo predetto dal modello, posizionandosi rispetto alla distribuzione generale.

Interpretazione:

La posizione della linea rossa rispetto alla distribuzione consente di valutare come il prezzo predetto si colloca rispetto ai prezzi medi e agli estremi del mercato. Ad esempio, se la linea è verso l'estremità destra, il prezzo predetto è relativamente alto rispetto al resto del dataset.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
Prezzo predetto: 204621.27€
Il prezzo predetto è superiore alla media del dataset di 1338.67€ (Media: 203282.59€).
```

Prezzo predetto rispetto alla media del dataset in questo caso risulta inferiore di 5013.21€