The Marble Run FFNN Regressor

Not Just Yummy or Meh

Prompts

- Separate features and target
- Scale the data
- Build a feed-forward neural network with "star_rating" as the target

Strategy

- Prepare the data
- Build a neural network, compile the model, train it, evaluate, and predict
- Visualize training
- Check the Actual vs Predicted

Code & Results

1. Prepare the data

This code separates features and target, scales numeric data, and performs train-test split.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# 1. Separate features and target
X = data.drop(columns=['star_rating'])
y = data['star_rating']

# 2. Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

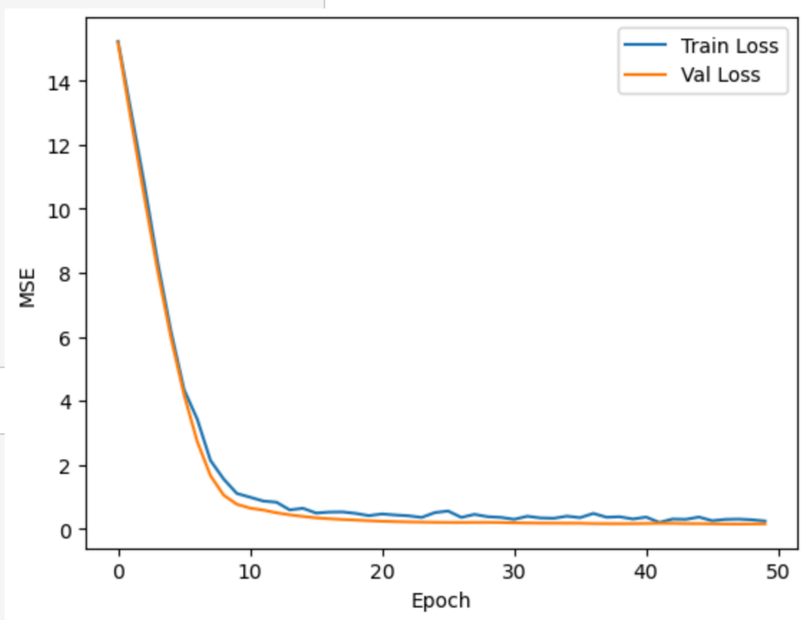
2. Build and evaluate the Model

This code builds a simple neural network.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

# 5. Build the FFNN Regressor
model = Sequential([
    Dense(64, input_dim=X_train.shape[1], activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dense(1, activation='linear')  # Linear output for regression
])

# 6. Compile model
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='mse',          # Mean Squared Error for regression
    metrics=['mae']       # Mean Absolute Error for readability
)

# 7. Train the model
history = model.fit(
    X_train, y_train,
    epochs=50,
    batch_size=16,
    validation_split=0.2,
    verbose=1
)

# 8. Evaluate
loss, mae = model.evaluate(X_test, y_test)
print(f"Test MAE: {mae:.2f}")
```

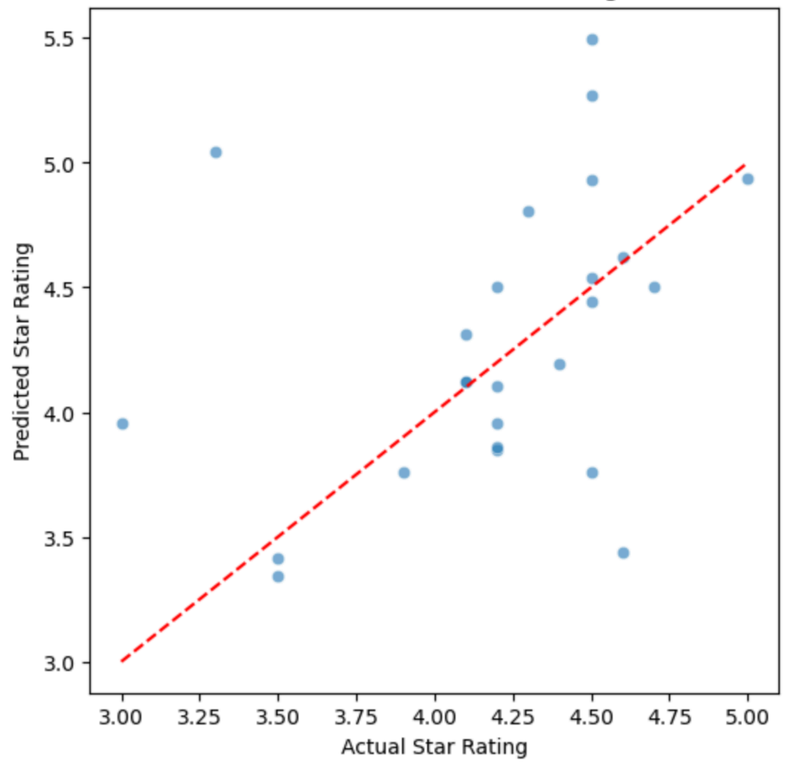
This code visualizes training.

```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.xlabel('Epoch')
plt.ylabel('MSE')
plt.legend()
plt.show()
```



This code plots actual vs predicted data.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Predict on test data
y_pred = model.predict(X_test).flatten()

# Scatter plot
plt.figure(figsize=(6,6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  # perfect line
plt.xlabel("Actual Star Rating")
plt.ylabel("Predicted Star Rating")
plt.title("Actual vs Predicted Star Ratings")
plt.show()
```



Made by: @katerynakononova
Website: https://katerynakononova.github.io/meowlearning/
Book: https://www.amazon.com/dp/BOCW9SFYXF