

Quantifying Yumminess

Linear Regression

Prompts



- Create a Linear regression using the preprocessed dataset 'data', with 'star_rating' as the target.
- Scale the features before training
- Plot the best fit

Strategy



- Prepare the data
- Train the Linear regression
- Check the Accuracy
- Visualise the best fit

Code & Results

1. Prepare the data

This code 1) separates features and target, 2) scales the features, and 3) splits train and test sets.

```
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)

# 2. 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. Train the Model

This code creates a linear regression model and trains it using the training set.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# 4. Train Linear Regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

3. Check the Accuracy

This code evaluates model accuracy.

```
# 5. Predictions
y_pred = model.predict(X_test_scaled)

# 6. Evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.4f}")
print(f"R² Score: {r2:.4f}")
```

Mean Squared Error: 0.1925
R² Score: 0.0656

✓ The MSE helps to see if your model is good enough. The smaller the MSE, the closer the model predictions are to the actual values.

✗ The variance of the features explains only about 6.6% of the cats' preferences. The remainder must be attributed to other factors.

4. Visualize the best fit

This code plots the actual test values (y_test) on the x-axis and the predicted values (y_pred) on the y-axis; and plotting a diagonal line (y=x) in red to visualise perfect predictions.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.7, color='green')
plt.xlabel('Actual Star Rating')
plt.ylabel('Predicted Star Rating')
plt.title('Actual vs Predicted Star Rating')
# Add a reference line
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linewidth=2)
plt.show()
```

