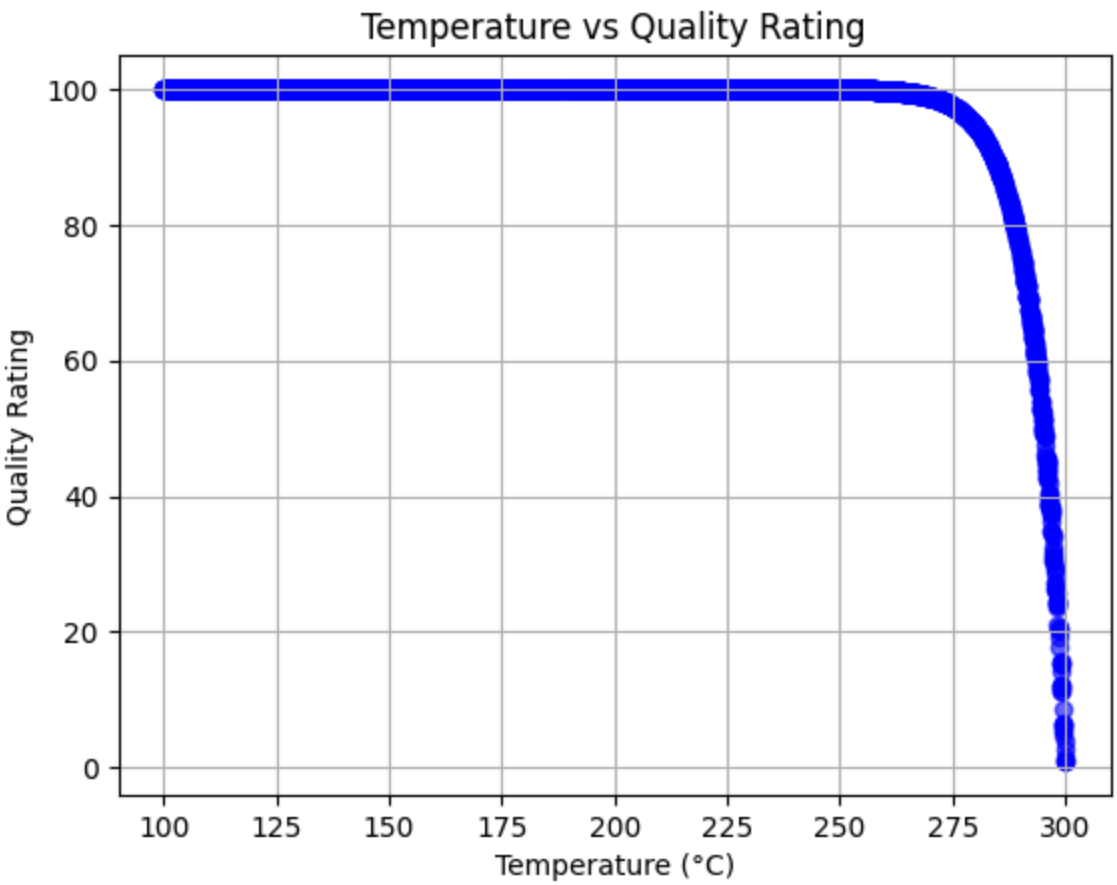
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score, mean_squared_error
```

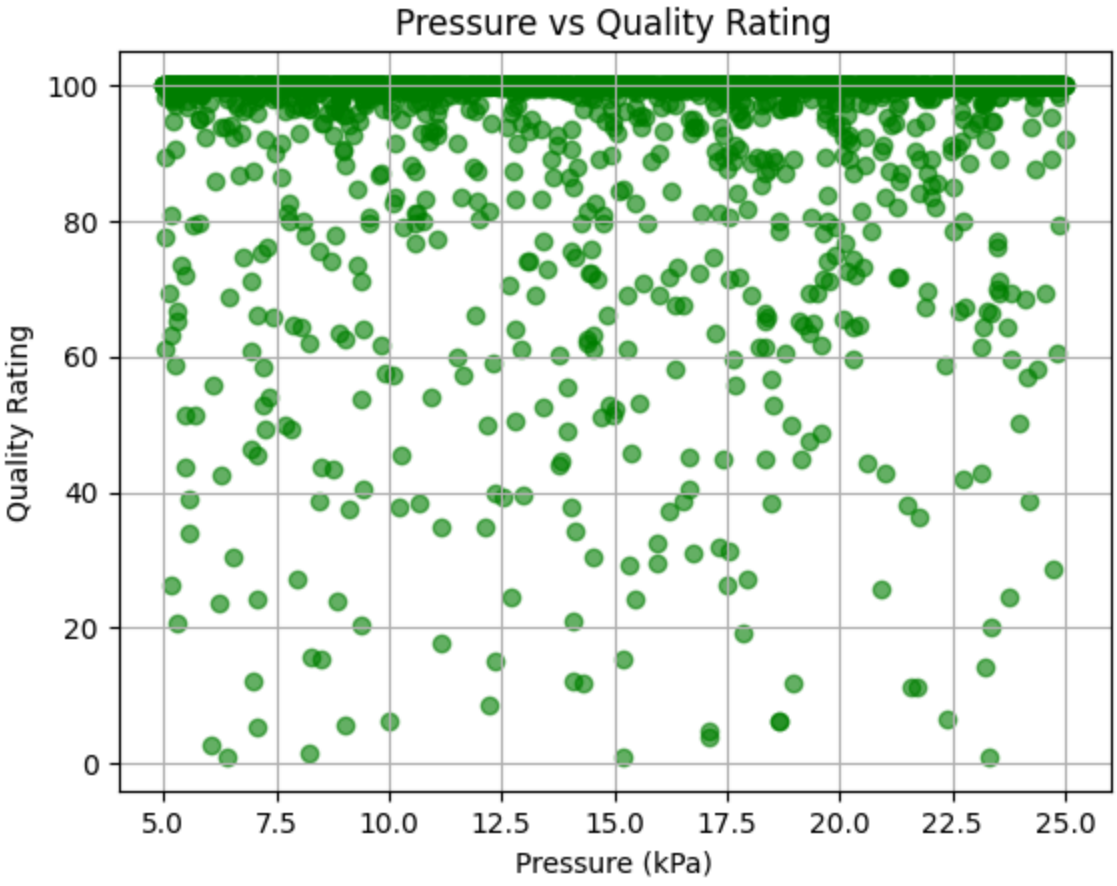
```
file_path = 'manufacturing.csv'  # Change this!
manufacturing_df = pd.read_csv(file_path)

print(manufacturing_df.head())
print(manufacturing_df.info())
```

```
   Temperature (°C)  Pressure (kPa)  Temperature x Pressure  \
0        209.762701        8.050855             1688.769167
1        243.037873       15.812068             3842.931469
2        220.552675        7.843130             1729.823314
3        208.976637       23.786089             4970.736918
4        184.730960       15.797812             2918.345014

   Material Fusion Metric  Material Transformation Metric  Quality Rating
0            44522.217074                    9.229576e+06       99.999971
1            63020.764997                    1.435537e+07       99.985703
2            49125.950249                    1.072839e+07       99.999758
3            57128.881547                    9.125702e+06       99.999975
4            38068.201283                    6.303792e+06      100.000000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3957 entries, 0 to 3956
Data columns (total 6 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Temperature (°C)                3957 non-null   float64
 1   Pressure (kPa)                  3957 non-null   float64
 2   Temperature x Pressure          3957 non-null   float64
 3   Material Fusion Metric          3957 non-null   float64
 4   Material Transformation Metric  3957 non-null   float64
 5   Quality Rating                  3957 non-null   float64
dtypes: float64(6)
memory usage: 185.6 KB
None
```

```
# Temperature vs Quality
plt.scatter(manufacturing_df['Temperature (°C)'], manufacturing_df['Quality Rating'], color=
plt.title('Temperature vs Quality Rating')
plt.xlabel('Temperature (°C)')
```

```
    plt.ylabel('Quality Rating')
    plt.grid(True)
    plt.show()

    # Pressure vs Quality
    plt.scatter(manufacturing_df['Pressure (kPa)'], manufacturing_df['Quality Rating'], color='g
    plt.title('Pressure vs Quality Rating')
    plt.xlabel('Pressure (kPa)')
    plt.ylabel('Quality Rating')
    plt.grid(True)
    plt.show()
```





(c) Visual Inspection (Write your answers) Based on scatter plots:

Temperature: Appears non-linear (curved trend); polynomial may suit.

Pressure: Some curvature or variation — might benefit from polynomial too.

```
X = manufacturing_df[['Temperature (°C)']]
y = manufacturing_df['Quality Rating']

model_lin = LinearRegression()
model_lin.fit(X, y)

y_pred_lin = model_lin.predict(X)

r2_lin = r2_score(y, y_pred_lin)
mse_lin = mean_squared_error(y, y_pred_lin)

print(f"Linear Model - R²: {r2_lin:.4f}, MSE: {mse_lin:.4f}")
```

```
Linear Model - R²: 0.2128, MSE: 132.8486
```

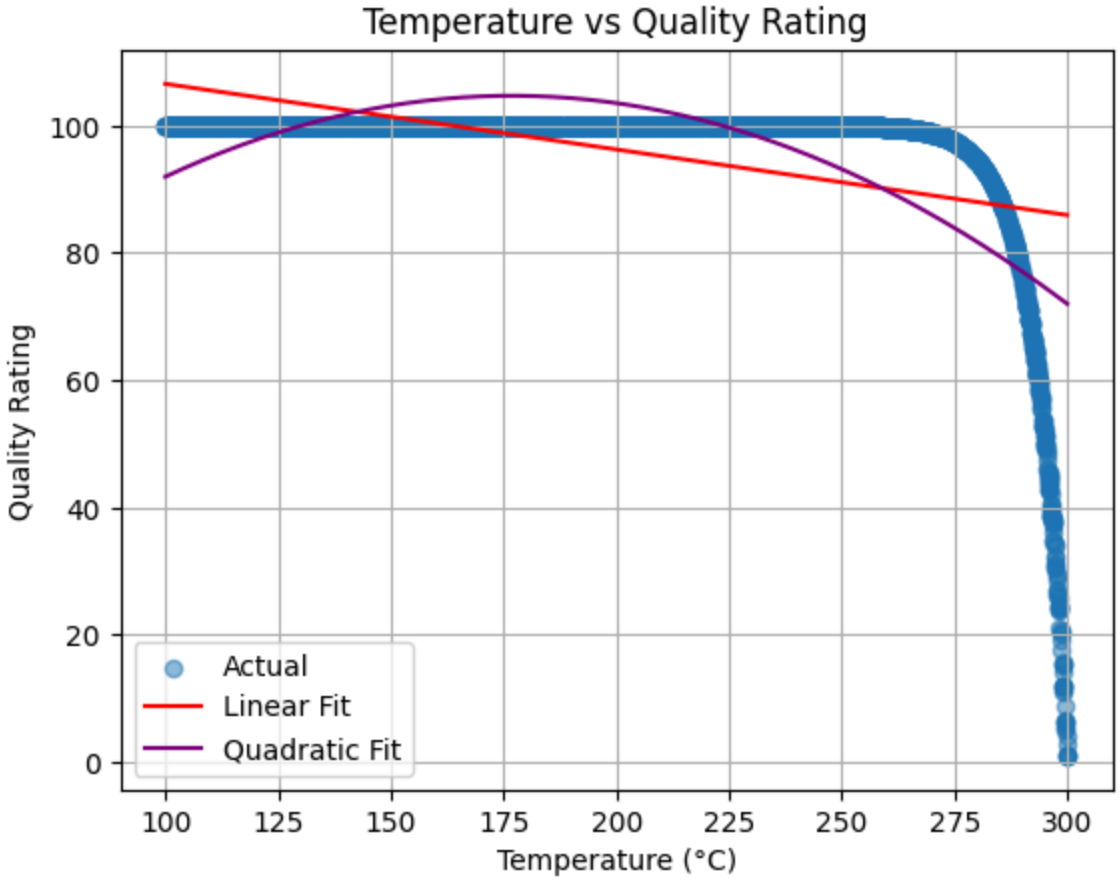
```
poly = PolynomialFeatures(degree=2)
X_poly2 = poly.fit_transform(X)

model_quad = LinearRegression()
model_quad.fit(X_poly2, y)

y_pred_quad = model_quad.predict(X_poly2)

r2_quad = r2_score(y, y_pred_quad)
mse_quad = mean_squared_error(y, y_pred_quad)

print(f"Quadratic Model - R²: {r2_quad:.4f}, MSE: {mse_quad:.4f}")
```

```
Quadratic Model - R²: 0.4613, MSE: 90.9079
```

```
plt.scatter(X, y, alpha=0.5, label='Actual')

# Sort for smooth curve
X_sorted = np.sort(X.values, axis=0)
y_lin_plot = model_lin.predict(X_sorted)
y_quad_plot = model_quad.predict(poly.transform(X_sorted))

plt.plot(X_sorted, y_lin_plot, color='red', label='Linear Fit')
plt.plot(X_sorted, y_quad_plot, color='purple', label='Quadratic Fit')

plt.title('Temperature vs Quality Rating')
plt.xlabel('Temperature (°C)')
plt.ylabel('Quality Rating')
plt.legend()
```

```
plt.grid(True)
plt.show()
```

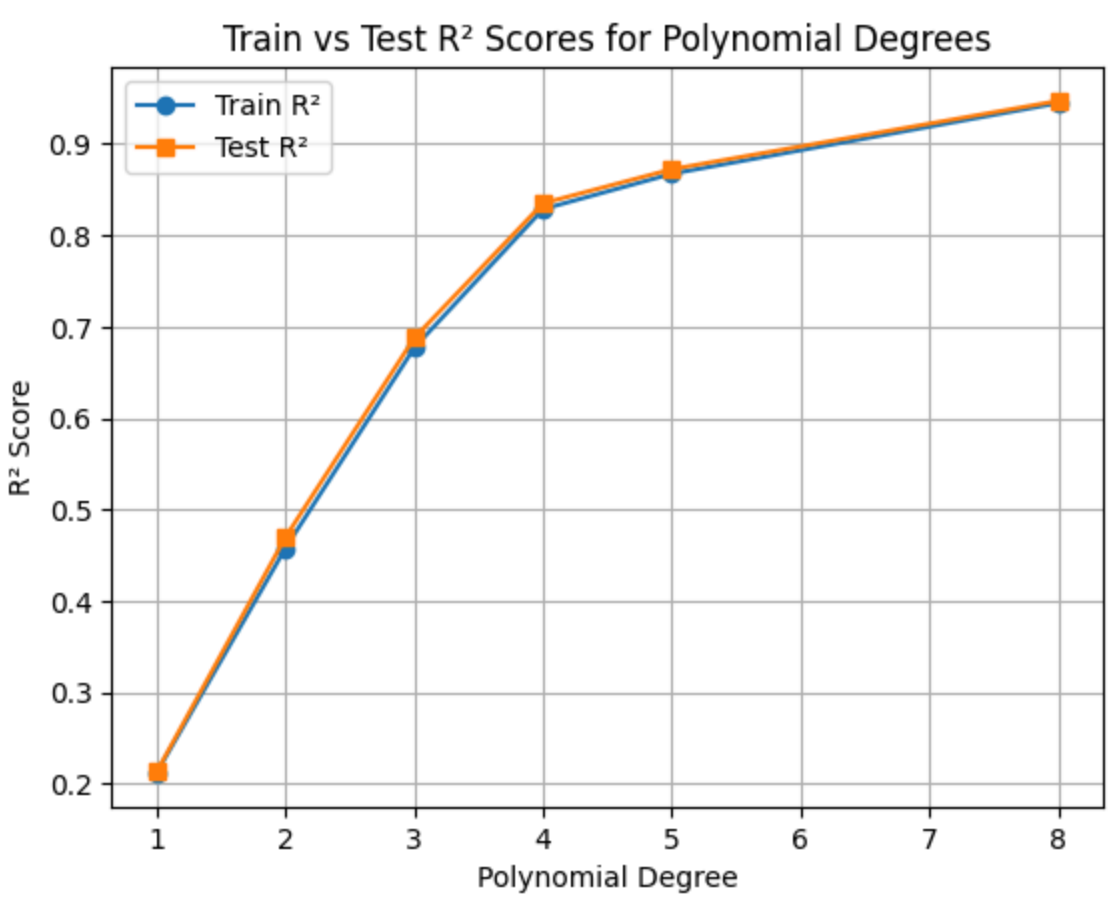
```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: ❯
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: ❯
  warnings.warn(
```



(d) Answer Quadratic model has higher R^2 and lower MSE, visually fits the curve better → better representation.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


degrees = [1, 2, 3, 4, 5, 8]
train_r2 = []
test_r2 = []

for d in degrees:
    poly = PolynomialFeatures(degree=d)
    X_train_poly = poly.fit_transform(X_train)
    X_test_poly = poly.transform(X_test)

    model = LinearRegression()
    model.fit(X_train_poly, y_train)
```

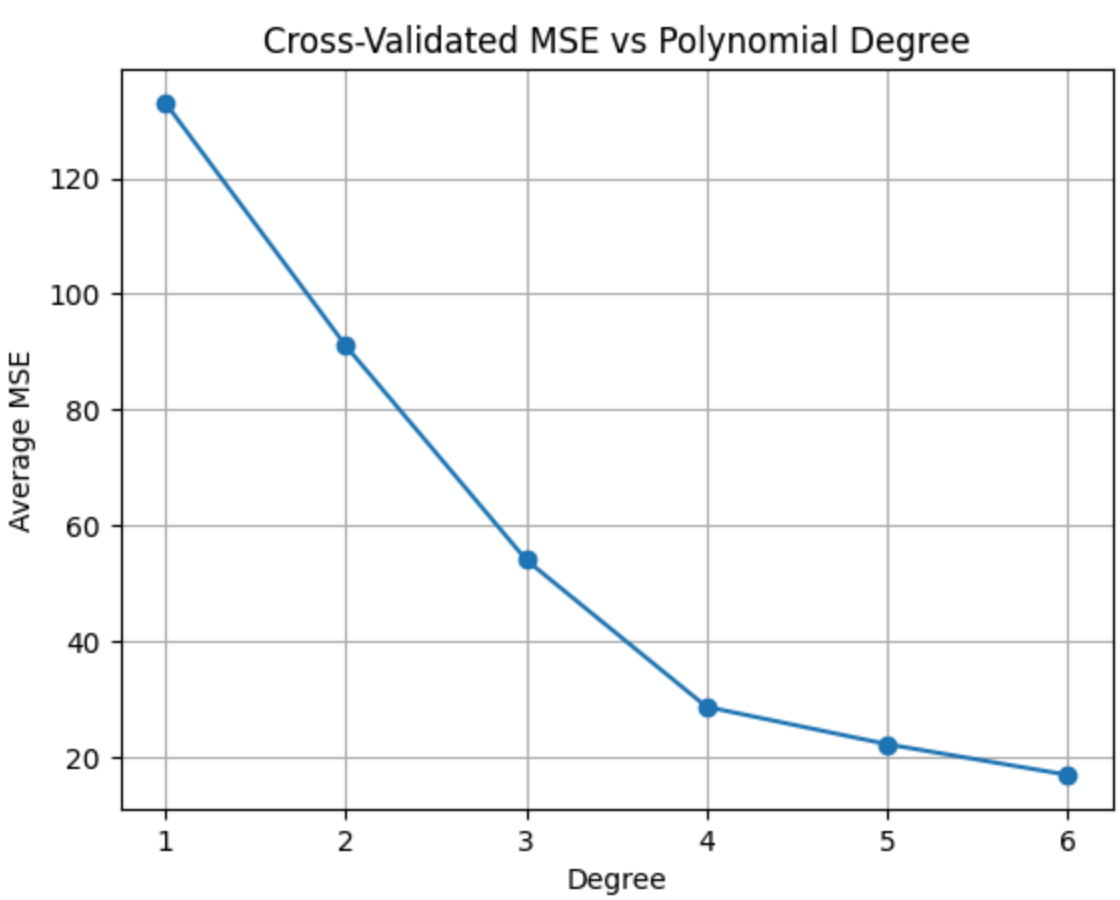
```
    y_train_pred = model.predict(X_train_poly)
    y_test_pred = model.predict(X_test_poly)

    train_r2.append(r2_score(y_train, y_train_pred))
    test_r2.append(r2_score(y_test, y_test_pred))


plt.plot(degrees, train_r2, label='Train R²', marker='o')
plt.plot(degrees, test_r2, label='Test R²', marker='s')

plt.xlabel('Polynomial Degree')
plt.ylabel('R² Score')
plt.title('Train vs Test R² Scores for Polynomial Degrees')
plt.legend()
plt.grid(True)
plt.show()
```



(d) Answer If test R² drops after a peak while train R² keeps increasing → overfitting. Usually occurs after degree 3–4 depending on your plot.

```
from sklearn.model_selection import cross_val_score

cv_mse = []
```

```
for d in range(1, 7):
    model = Pipeline([
        ('poly', PolynomialFeatures(degree=d)),
        ('linreg', LinearRegression())
    ])
    neg_mse = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')
    cv_mse.append(-np.mean(neg_mse))


plt.plot(range(1, 7), cv_mse, marker='o')
plt.title('Cross-Validated MSE vs Polynomial Degree')
plt.xlabel('Degree')
plt.ylabel('Average MSE')
plt.grid(True)
plt.show()
```



(c) Answer Choose the degree with lowest average MSE and reasonable complexity (avoid too high degree even if slightly better MSE).

```
final_poly = PolynomialFeatures(degree=3)
X_final_poly = final_poly.fit_transform(X)

final_model = LinearRegression()
final_model.fit(X_final_poly, y)
```

```
LinearRegression
LinearRegression()
```

```
temp_value = np.array([[215]])
temp_poly = final_poly.transform(temp_value)
predicted_quality = final_model.predict(temp_poly)

print(f"Predicted Quality Rating at 215°C: {predicted_quality[0]:.2f}")
```