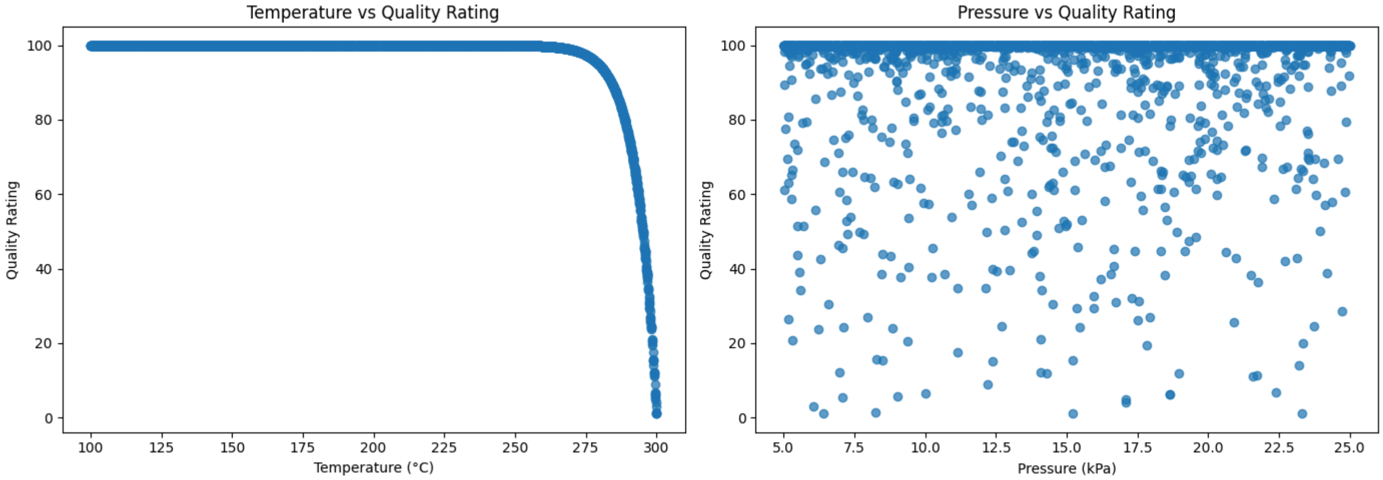
```
# ------------------------------
# Step 1: Import Libraries
# ------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.pipeline import Pipeline

# ------------------------------
# Question 1: Load Data and Visualize
# ------------------------------
# (a) Load dataset
file_path = 'manufacturing.csv'  # UPDATE this
manufacturing_df = pd.read_csv(file_path)
print(manufacturing_df.head())
print(manufacturing_df.info())

# (b) Scatter plots
plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)
plt.scatter(manufacturing_df['Temperature (°C)'], manufacturing_df['Quality Rating'], alpha=0.7)
plt.xlabel("Temperature (°C)")
plt.ylabel("Quality Rating")
plt.title("Temperature vs Quality Rating")

plt.subplot(1, 2, 2)
plt.scatter(manufacturing_df['Pressure (kPa)'], manufacturing_df['Quality Rating'], alpha=0.7)
plt.xlabel("Pressure (kPa)")
plt.ylabel("Quality Rating")
plt.title("Pressure vs Quality Rating")

plt.tight_layout()
plt.show()
```

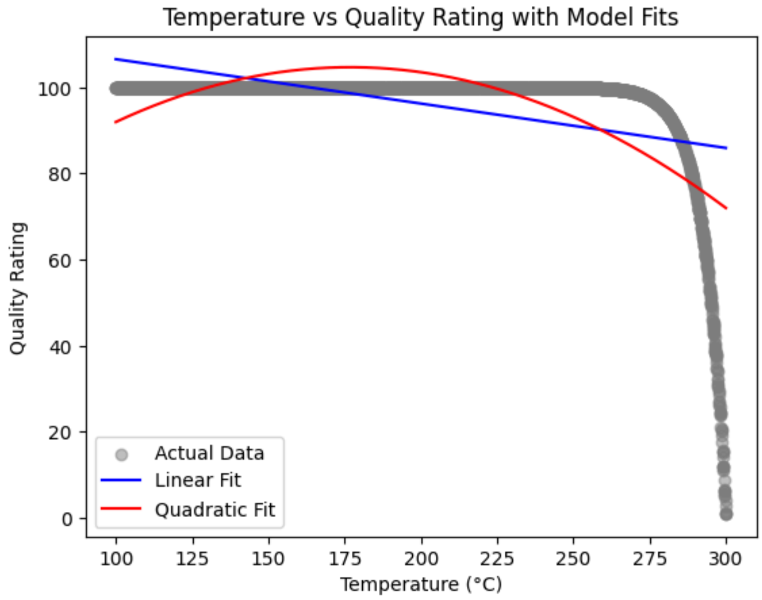
```
   Temperature (°C)  Pressure (kPa)  Temperature x Pressure  \
0        209.762701        8.050855             1688.769167
1        243.037873       15.812068             3842.931469
2        220.552675        7.843130             1729.823314
3        208.976637       23.786089             4970.736918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



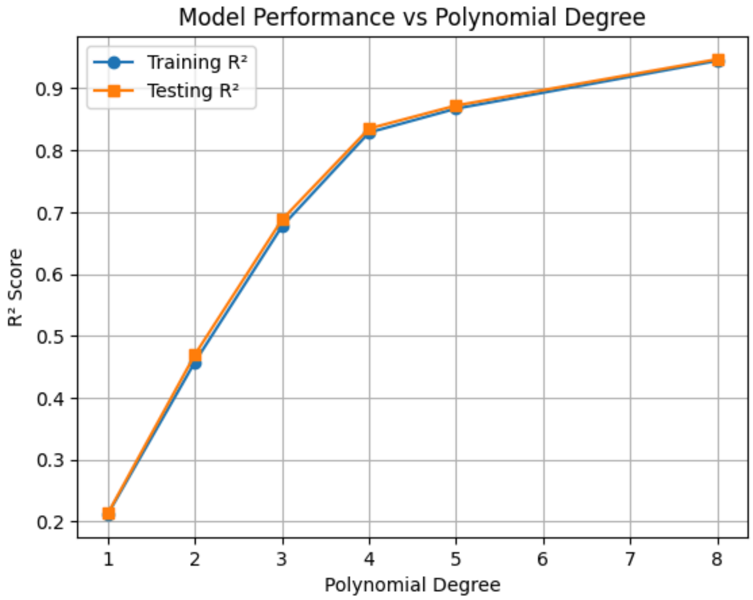
```
# ------------------------------
# Question 2: Linear vs Quadratic Model for Temperature
# ------------------------------
X = manufacturing_df[['Temperature (°C)']]
y = manufacturing_df['Quality Rating']

# (a) Linear Regression (degree 1)
linear_model = LinearRegression()
linear_model.fit(X, y)
y_pred_linear = linear_model.predict(X)
r2_linear = r2_score(y, y_pred_linear)
mse_linear = mean_squared_error(y, y_pred_linear)
print("Linear Model - R²:", r2_linear, "MSE:", mse_linear)

# (b) Quadratic Regression (degree 2)
poly2 = PolynomialFeatures(degree=2)
X_poly2 = poly2.fit_transform(X)
quadratic_model = LinearRegression()
quadratic_model.fit(X_poly2, y)
y_pred_quad = quadratic_model.predict(X_poly2)
r2_quad = r2_score(y, y_pred_quad)
mse_quad = mean_squared_error(y, y_pred_quad)
print("Quadratic Model - R²:", r2_quad, "MSE:", mse_quad)

# (c) Plot both predictions
plt.scatter(X, y, color='gray', alpha=0.5, label='Actual Data')
sorted_idx = X.values.flatten().argsort()
```

```
plt.plot(X.values[sorted_idx], y_pred_linear[sorted_idx], color='blue', label='Linear Fit')
plt.plot(X.values[sorted_idx], y_pred_quad[sorted_idx], color='red', label='Quadratic Fit')
plt.xlabel("Temperature (°C)")
plt.ylabel("Quality Rating")
plt.title("Temperature vs Quality Rating with Model Fits")
plt.legend()
plt.show()
```

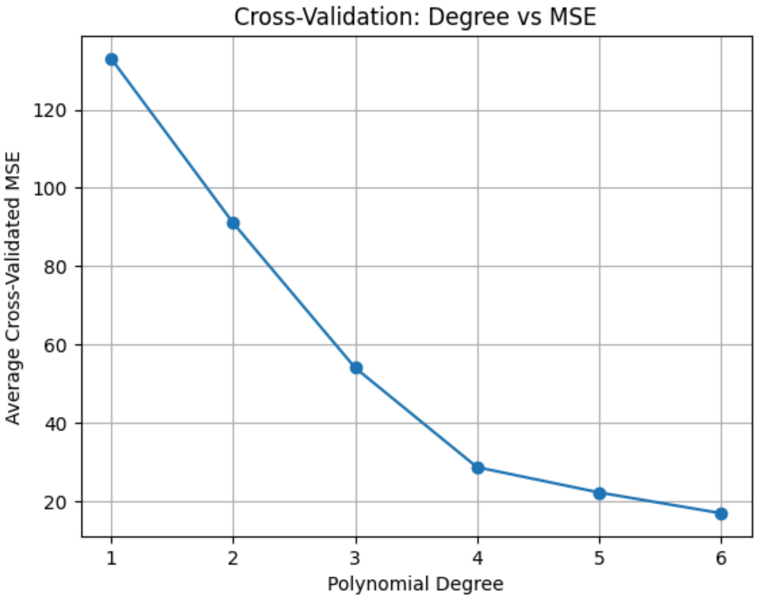
```
Linear Model - R²: 0.2127778634271571 MSE: 132.84863630633382
Quadratic Model - R²: 0.4613061729741008 MSE: 90.90793688626003
```



```
# ------------------------------
# Question 3: Higher-Degree Polynomials and Overfitting
# ------------------------------
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)
    train_r2.append(model.score(X_train_poly, y_train))
    test_r2.append(model.score(X_test_poly, y_test))

# (c) Plot R² scores
plt.plot(degrees, train_r2, marker='o', label='Training R²')
plt.plot(degrees, test_r2, marker='s', label='Testing R²')
plt.xlabel("Polynomial Degree")
plt.ylabel("R² Score")
plt.title("Model Performance vs Polynomial Degree")
plt.legend()
plt.grid(True)
plt.show()
```



```
# ------------------------------
# Question 4: Optimal Degree via Cross-Validation
# ------------------------------
cv_degrees = list(range(1, 7))
mean_mse_scores = []

for d in cv_degrees:
    pipeline = Pipeline([
        ('poly', PolynomialFeatures(degree=d)),
        ('model', LinearRegression())
    ])
    neg_mse_scores = cross_val_score(pipeline, X, y, cv=5, scoring='neg_mean_squared_error')
    mean_mse_scores.append(-neg_mse_scores.mean())

# (b) Plot MSE vs degree
plt.plot(cv_degrees, mean_mse_scores, marker='o')
plt.xlabel("Polynomial Degree")
plt.ylabel("Average Cross-Validated MSE")
plt.title("Cross-Validation: Degree vs MSE")
plt.grid(True)
plt.show()
```



```
# ------------------------------
# Question 5: Final Model and Prediction
# ------------------------------
# Assume best degree (based on CV plot) is 2 or replace with your result
optimal_degree = cv_degrees[np.argmin(mean_mse_scores)]
final_poly = PolynomialFeatures(degree=optimal_degree)
X_final_poly = final_poly.fit_transform(X)
final_model = LinearRegression()
final_model.fit(X_final_poly, y)

# (b) Predict for Temperature = 215°C
temp_val = np.array([[215]])
temp_poly = final_poly.transform(temp_val)
```