Shreeyash Shripad Dongarkar

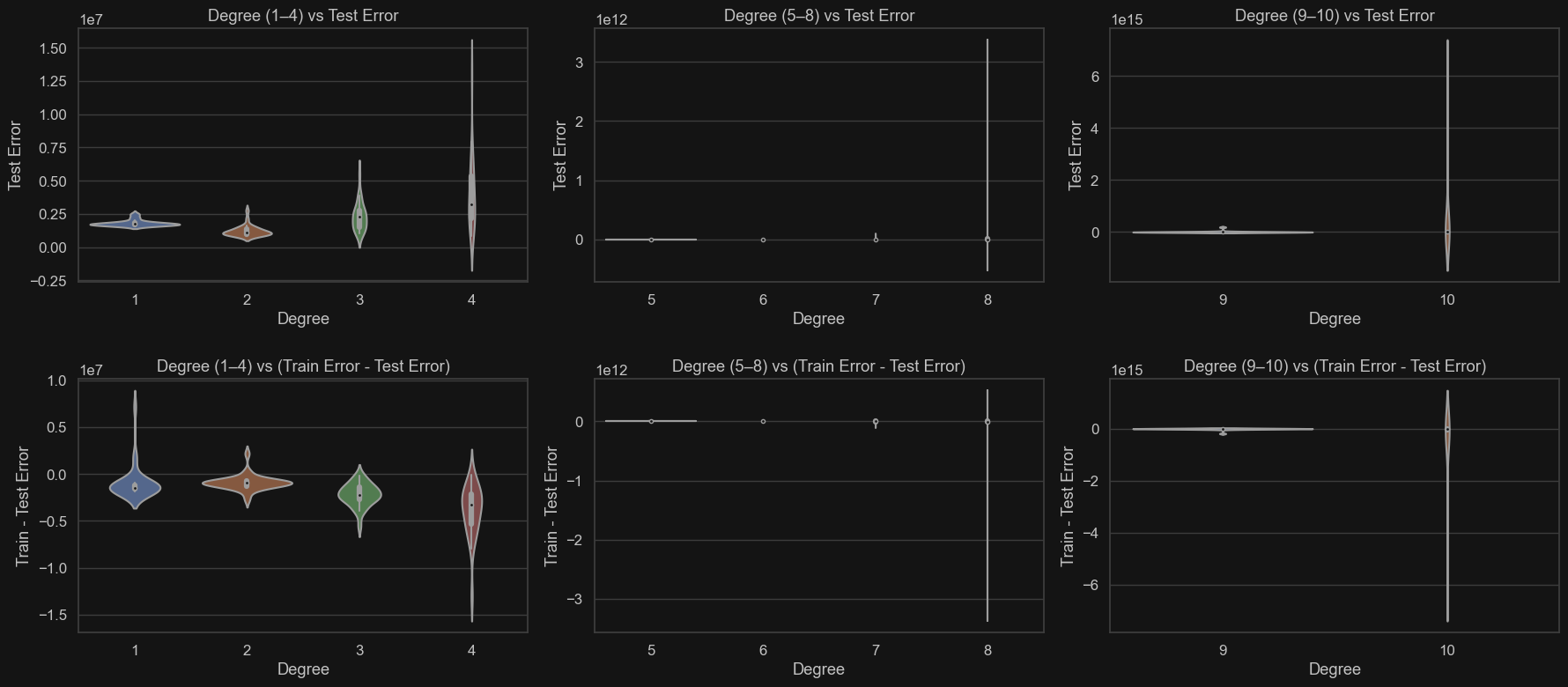
22510025

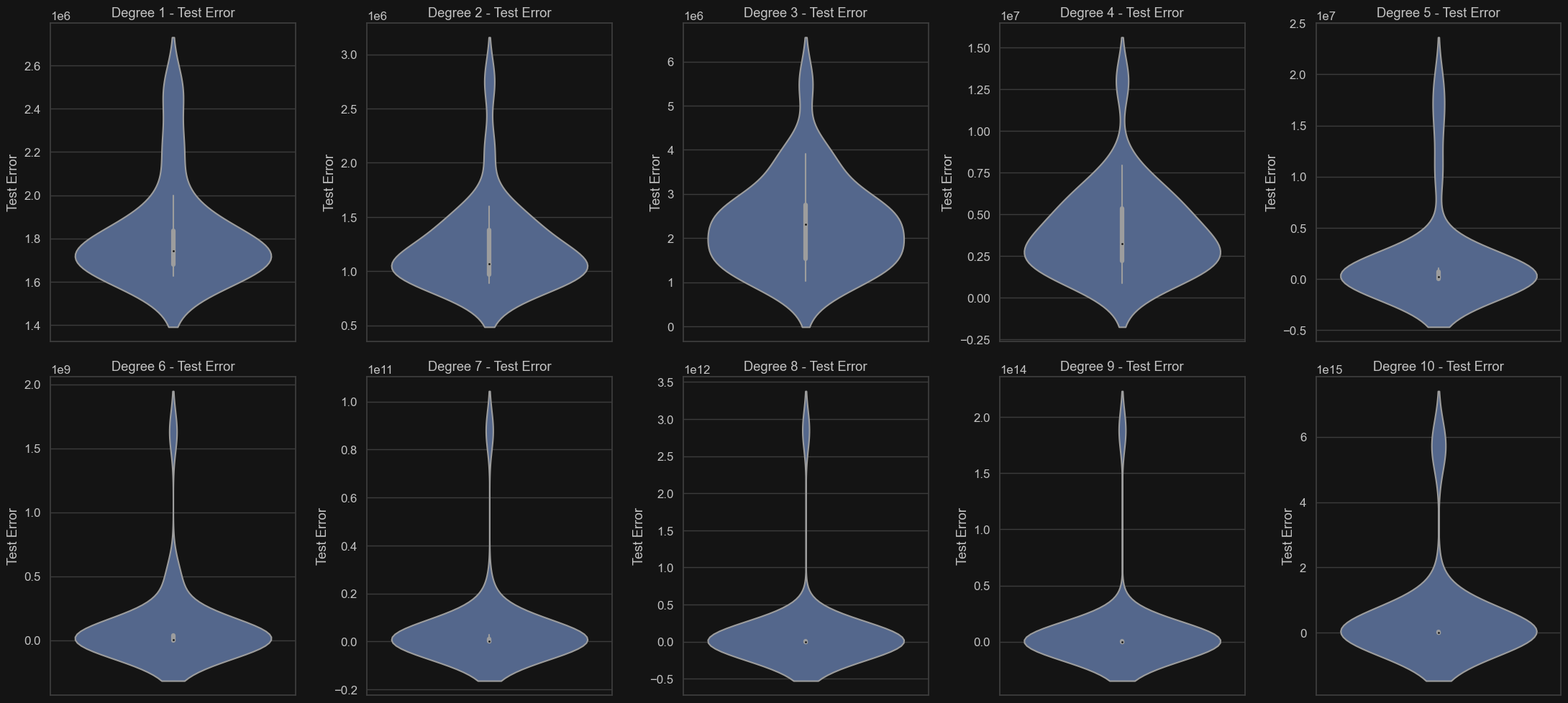
TY CSE BTech

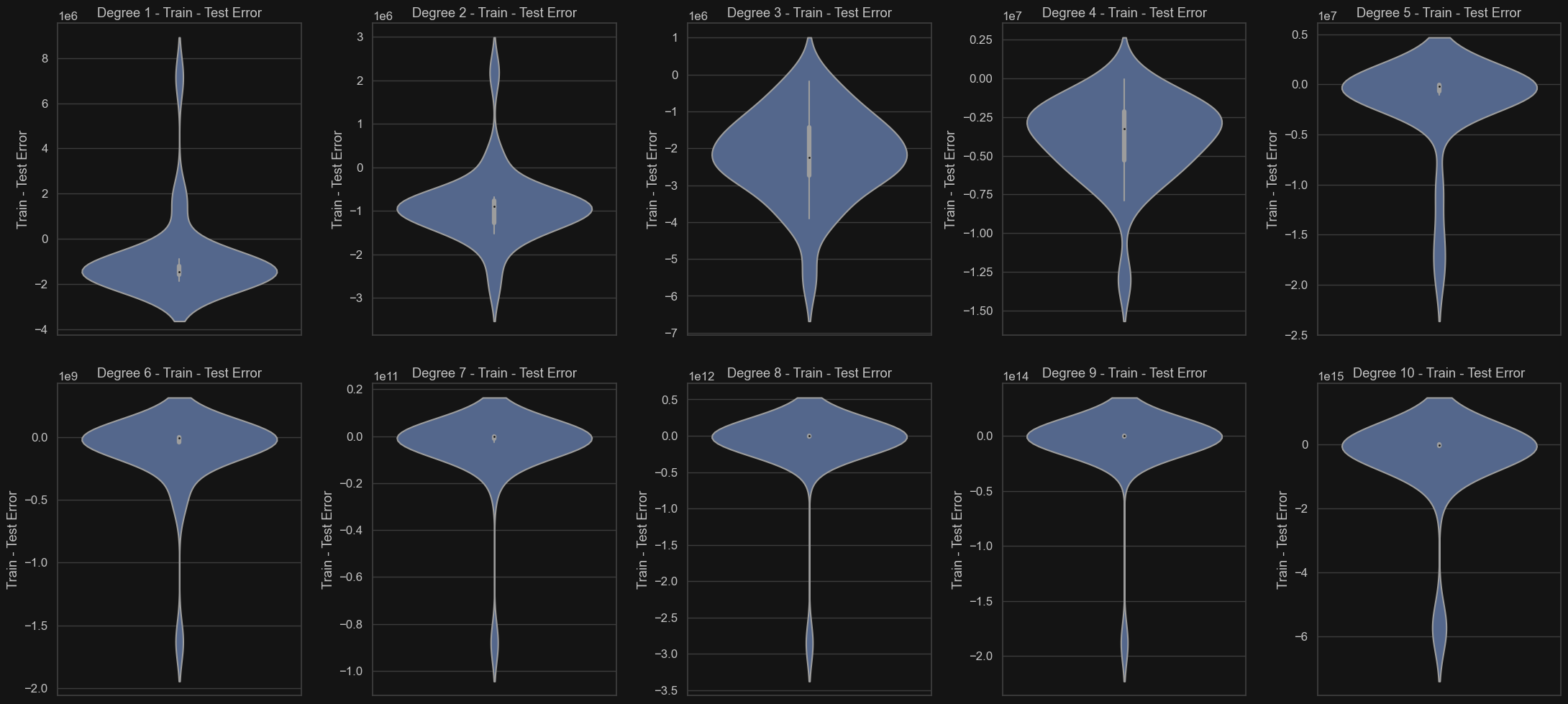
Machine Learning Lab

Assignment 8: Polynomial Regression

1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. import seaborn as sns
5. from sklearn.linear\_model import LinearRegression, RidgeCV, LassoCV
6. from sklearn.metrics import mean\_squared\_error, r2\_score
7. from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold
8. from sklearn.preprocessing import PolynomialFeatures, StandardScaler
9. from sklearn.pipeline import make\_pipeline
10. df = pd.read\_csv("polynomial\_regression.csv")
11. X = df[['x']]
12. Y = df['y']
13. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)
14. np.random.seed(42)
15. sample\_size = 20
16. num\_samples = 30
17. degrees = range(1, 11)
18. test\_errors\_by\_degree = {deg: [] for deg in degrees}
19. train\_minus\_test\_errors\_by\_degree = {deg: [] for deg in degrees}
20. for \_ in range(num\_samples):
21. idx = np.random.choice(len(X\_train), size=sample\_size, replace=False)
22. X\_sample = X\_train.iloc[idx]
23. Y\_sample = Y\_train.iloc[idx]
24. for degree in degrees:
25. poly = PolynomialFeatures(degree=degree, include\_bias=False)
26. X\_sample\_poly = poly.fit\_transform(X\_sample)
27. X\_test\_poly = poly.transform(X\_test)
29. model = LinearRegression()
30. model.fit(X\_sample\_poly, Y\_sample)
31. Y\_sample\_pred = model.predict(X\_sample\_poly)
32. Y\_test\_pred = model.predict(X\_test\_poly)
33. train\_error = mean\_squared\_error(Y\_sample, Y\_sample\_pred)
34. test\_error = mean\_squared\_error(Y\_test, Y\_test\_pred)
35. test\_errors\_by\_degree[degree].append(test\_error)
36. train\_minus\_test\_errors\_by\_degree[degree].append(train\_error - test\_error)
37. test\_error\_df = pd.DataFrame([
38. {"Degree": deg, "Test Error": err}
39. for deg, errors in test\_errors\_by\_degree.items()
40. for err in errors
41. ])
42. train\_minus\_test\_df = pd.DataFrame([
43. {"Degree": deg, "Train - Test Error": err}
44. for deg, errors in train\_minus\_test\_errors\_by\_degree.items()
45. for err in errors
46. ])
47. sns.set\_theme(style="whitegrid", palette="muted", font\_scale=1.1)
48. plt.figure(figsize=(12, 4))
49. sns.violinplot(data=test\_error\_df, x="Degree", y="Test Error")
50. plt.title("Degree vs Test Error (30 Samples)")
51. plt.tight\_layout()
52. plt.show()
53. plt.figure(figsize=(12, 6))
54. sns.violinplot(data=train\_minus\_test\_df, x="Degree", y="Train - Test Error")
55. plt.title("Degree vs (Train Error - Test Error) (30 Samples)")
56. plt.tight\_layout()
57. plt.show()
58. idx = np.random.choice(len(X\_train), size=20, replace=False)
59. X\_sample = X\_train.iloc[idx]
60. Y\_sample = Y\_train.iloc[idx]
61. mean\_cv\_scores = []
62. for degree in degrees:
63. pipeline = make\_pipeline(PolynomialFeatures(degree, include\_bias=False), LinearRegression())
64. scores = cross\_val\_score(pipeline, X\_sample, Y\_sample, cv=5, scoring='neg\_mean\_squared\_error')
65. mean\_cv\_scores.append(scores.mean())
66. best\_degree = degrees[np.argmax(mean\_cv\_scores)]
67. print(f"Best degree from 5-fold CV on sample: {best\_degree}")
68. final\_poly = PolynomialFeatures(degree=best\_degree, include\_bias=False)
69. X\_sample\_poly = final\_poly.fit\_transform(X\_sample)
70. X\_test\_poly = final\_poly.transform(X\_test)
71. model = LinearRegression()
72. model.fit(X\_sample\_poly, Y\_sample)
73. Y\_test\_pred = model.predict(X\_test\_poly)
74. final\_test\_error = mean\_squared\_error(Y\_test, Y\_test\_pred)
75. print(f"Test error of best degree model from CV: {final\_test\_error:.4f}")
76. alphas = np.logspace(-3, 3, 100)
77. ridge\_pipeline = make\_pipeline(
78. PolynomialFeatures(degree=best\_degree, include\_bias=False),
79. StandardScaler(),
80. RidgeCV(alphas=alphas, cv=10, scoring='neg\_mean\_squared\_error')
81. )
82. ridge\_pipeline.fit(X\_train, Y\_train)
83. ridge\_test\_error = mean\_squared\_error(Y\_test, ridge\_pipeline.predict(X\_test))
84. ridge\_r2\_score = r2\_score(Y\_test, ridge\_pipeline.predict(X\_test))
85. best\_ridge\_model = ridge\_pipeline.named\_steps['ridgecv']
86. print(f"Ridge best alpha: {best\_ridge\_model.alpha\_:.4f}, Test error: {ridge\_test\_error:.4f}, R^2: {ridge\_r2\_score:.4f}")
87. lasso\_pipeline = make\_pipeline(
88. PolynomialFeatures(degree=best\_degree, include\_bias=False),
89. StandardScaler(),
90. LassoCV(alphas=alphas, cv=10, max\_iter=100000)
91. )
92. lasso\_pipeline.fit(X\_train, Y\_train)
93. lasso\_test\_error = mean\_squared\_error(Y\_test, lasso\_pipeline.predict(X\_test))
94. lasso\_r2\_score = r2\_score(Y\_test, lasso\_pipeline.predict(X\_test))
95. best\_lasso\_model = lasso\_pipeline.named\_steps['lassocv']
96. print(f"Lasso best alpha: {best\_lasso\_model.alpha\_:.4f}, Test error: {lasso\_test\_error:.4f}, R^2: {lasso\_r2\_score:.4f}")







**For Degree vs Test Error:**

* For degree 1-4 there are high and relatively consistent test errors which means model can be underfitting.
* For degree 5 there is sharp drop in test error compared to previous degree and implies improved generalization.
* For degree 6 there is lowest test error and distribution in narrow and centred around a very low error value.
* For greater degrees extremely wide distributions imply high variance and test error increase drastically

**For Degree vs Train-Test Error:**

* The **Train - Test Error** represents how much better (or worse) the model performs on training data compared to unseen test data. **Positive values** suggest the model is overfitting (train error is low, test error is high). **Negative or near-zero values** imply more generalization or underfitting.
* For low degrees plots show relatively narrower spread indicating consistent error difference across trials.
* For moderate degrees error difference remains moderate
* For higher degrees distribution widens dramatically and value explodes to very large magnitudes indicate extreme overfitting.

1. **Polynomial Regression (Best Degree from CV):**
   * **Best Degree:** 6 (from 5-fold CV on a sample)
   * **Test MSE:** **4,598,251.42** (extremely high)
   * **Observation:**
     + The high test MSE suggests severe **overfitting** when using a 6th-degree polynomial on the small sample (20 points).
     + Polynomials of high degree can fit training data perfectly but generalize poorly to unseen data.
     + The model likely **memorized noise** in the small sample, leading to poor test performance.
2. **Ridge Regression (L2 Regularization):**
   * **Best Alpha (λ):** 0.2656
   * **Test MSE:** **10,456.22** (much lower than polynomial regression)
   * **R²:** **0.9965** (near-perfect fit)
   * **Observation:**
     + Ridge regression **dramatically improved generalization** compared to the unregularized polynomial model.
     + The small **α (λ)** suggests **mild regularization** was sufficient to prevent overfitting.
     + The high **R²** indicates the model explains **99.65% of variance** in the test set.
3. **Lasso Regression (L1 Regularization):**
   * **Best Alpha (λ):** 0.0756
   * **Test MSE:** **10,456.14** (slightly better than Ridge)
   * **R²:** **0.9965** (same as Ridge)
   * **Observation:**
     + Lasso performed **slightly better** than Ridge in terms of MSE.
     + The optimal **α** is smaller than Ridge’s, suggesting **less aggressive shrinkage**.
     + Since Lasso can **zero out coefficients**, it might have selected a **sparse model** (fewer features).