Polynomial_Regression

May 13, 2024

[32]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

data = pd.read_csv(r'E:\DUK\Predictive␣
↪Analytics\Polynomial_Regression\Position_Salaries.csv')
display(data)
```

```
            Position  Level   Salary
0   Business Analyst      1    45000
1  Junior Consultant      2    50000
2  Senior Consultant      3    60000
3            Manager      4    80000
4    Country Manager      5   110000
5     Region Manager      6   150000
6            Partner      7   200000
7     Senior Partner      8   300000
8            C-level      9   500000
9                CEO     10  1000000
```

Exploratory Data Analytics

[33]:
```python
data = pd.DataFrame({
    'Position': ['Business Analyst', 'Junior Consultant', 'Senior Consultant',␣
↪'Manager', 'Country Manager',
                 'Region Manager', 'Partner', 'Senior Partner', 'C-level',␣
↪'CEO'],
    'Level': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Salary': [45000, 50000, 60000, 80000, 110000, 150000, 200000, 300000,␣
↪500000, 1000000]
})
print(data)
```

```
            Position  Level   Salary
0   Business Analyst      1    45000
1  Junior Consultant      2    50000
2  Senior Consultant      3    60000
3            Manager      4    80000
```

```
4    Country Manager      5   110000
5     Region Manager      6   150000
6            Partner      7   200000
7     Senior Partner      8   300000
8            C-level      9   500000
9                CEO     10  1000000
```

[34]:
```python
X = data.iloc[:, 1:2].values
y = data.iloc[:, 2].values

print("Independent variable (Level):")
print(X)
print("\nDependent variable (Salary):")
print(y)
```

```
Independent variable (Level):
[[ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]]

Dependent variable (Salary):
[  45000   50000   60000   80000  110000  150000  200000  300000  500000
 1000000]
```

[35]:
```python
missing_values = data.isnull().sum()
print("Missing values in the dataset:")
print(missing_values)
```

```
Missing values in the dataset:
Position    0
Level       0
Salary      0
dtype: int64
```

[36]:
```python
summary_stats = data.describe()
print("Summary Statistics:")
print(summary_stats)
```

```
Summary Statistics:
          Level          Salary
count  10.00000       10.000000
mean    5.50000   249500.000000
```

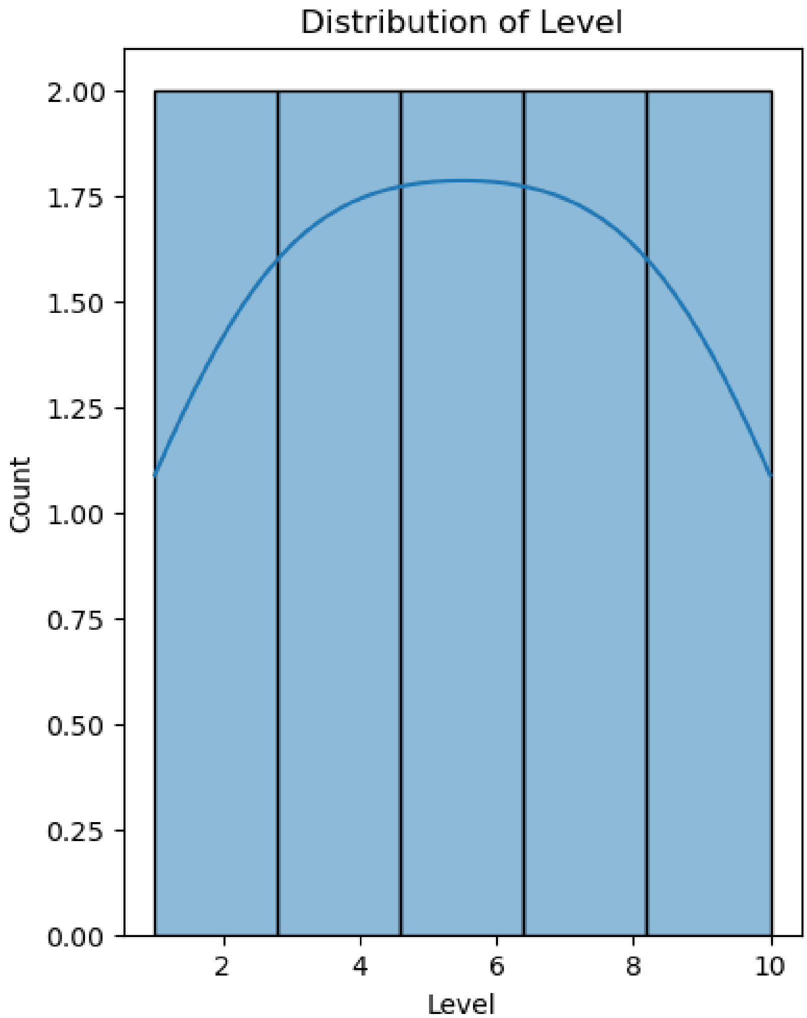
```
std     3.02765   299373.883668
min     1.00000    45000.000000
25%     3.25000    65000.000000
50%     5.50000   130000.000000
75%     7.75000   275000.000000
max    10.00000  1000000.000000
```

[37]:
```python
data.replace([np.inf, -np.inf], np.nan, inplace=True)
```

[38]:
```python
data_cleaned = data.dropna(subset=['Level'])
```

[41]:
```python
plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
sns.histplot(data['Level'], bins=5, kde=True)
plt.title('Distribution of Level')
plt.show()
```

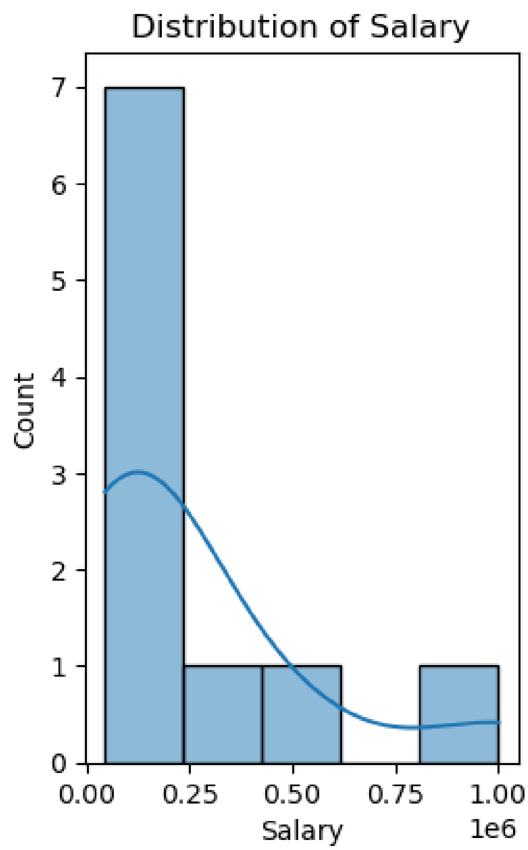
```
C:\Users\Harsh Vardhan\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[47]:

```
plt.subplot(1, 2, 2)
sns.histplot(data['Salary'], bins=5, kde=True)
plt.title('Distribution of Salary')
plt.show()
```

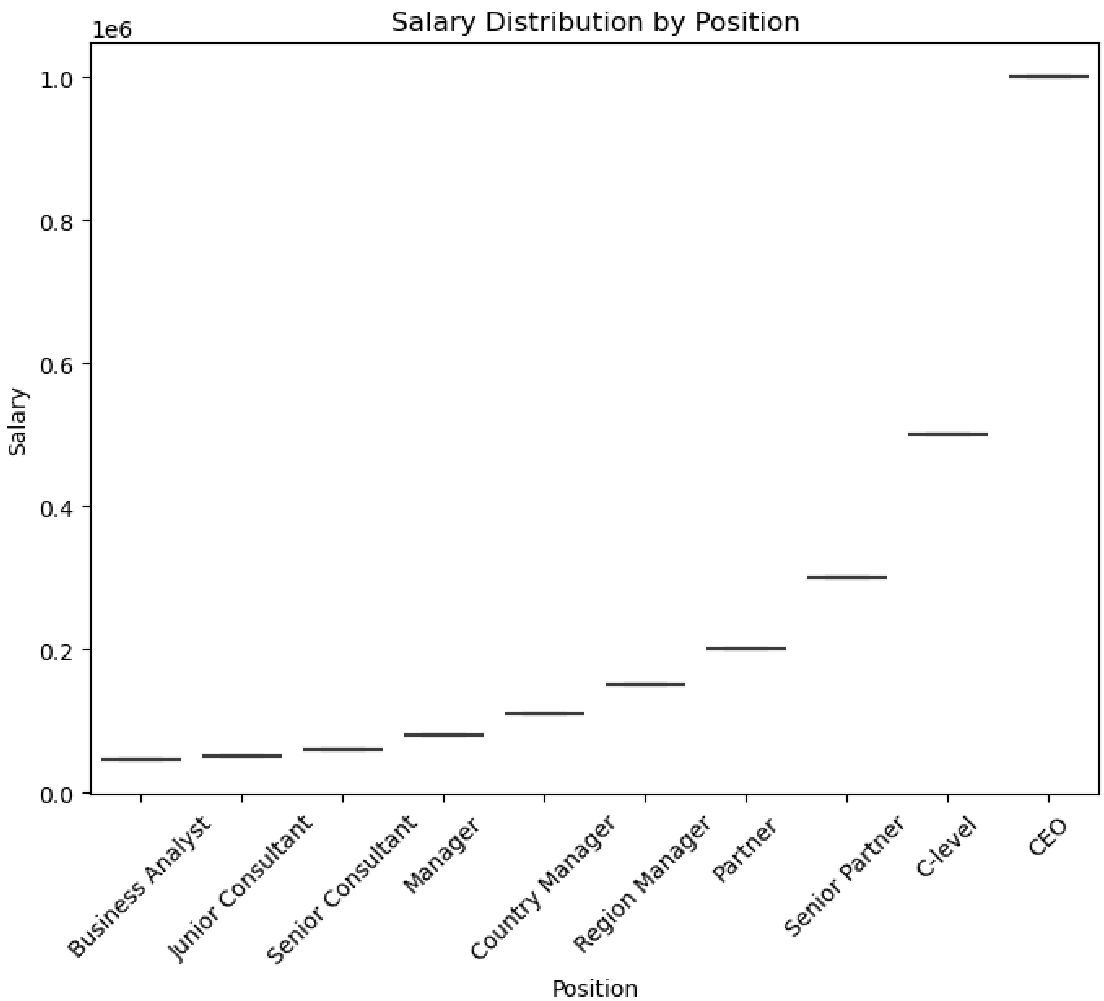
```
C:\Users\Harsh Vardhan\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



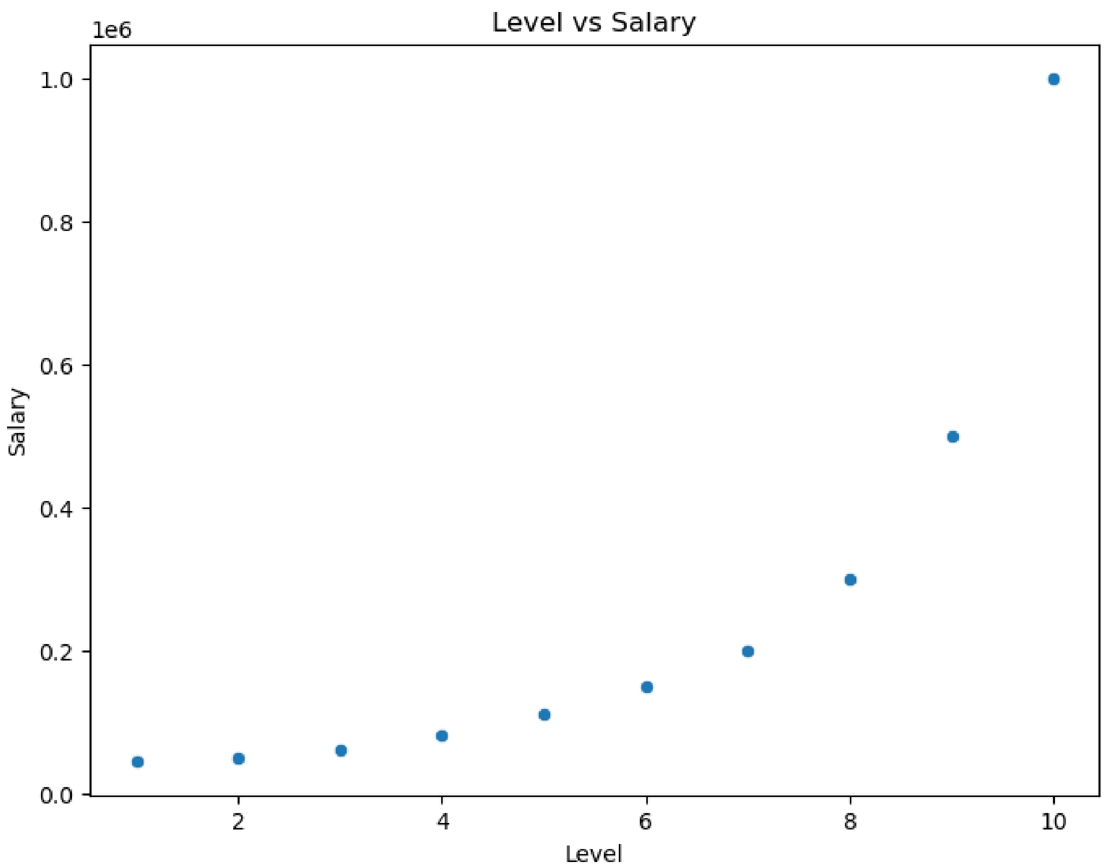
[45]:
```
plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

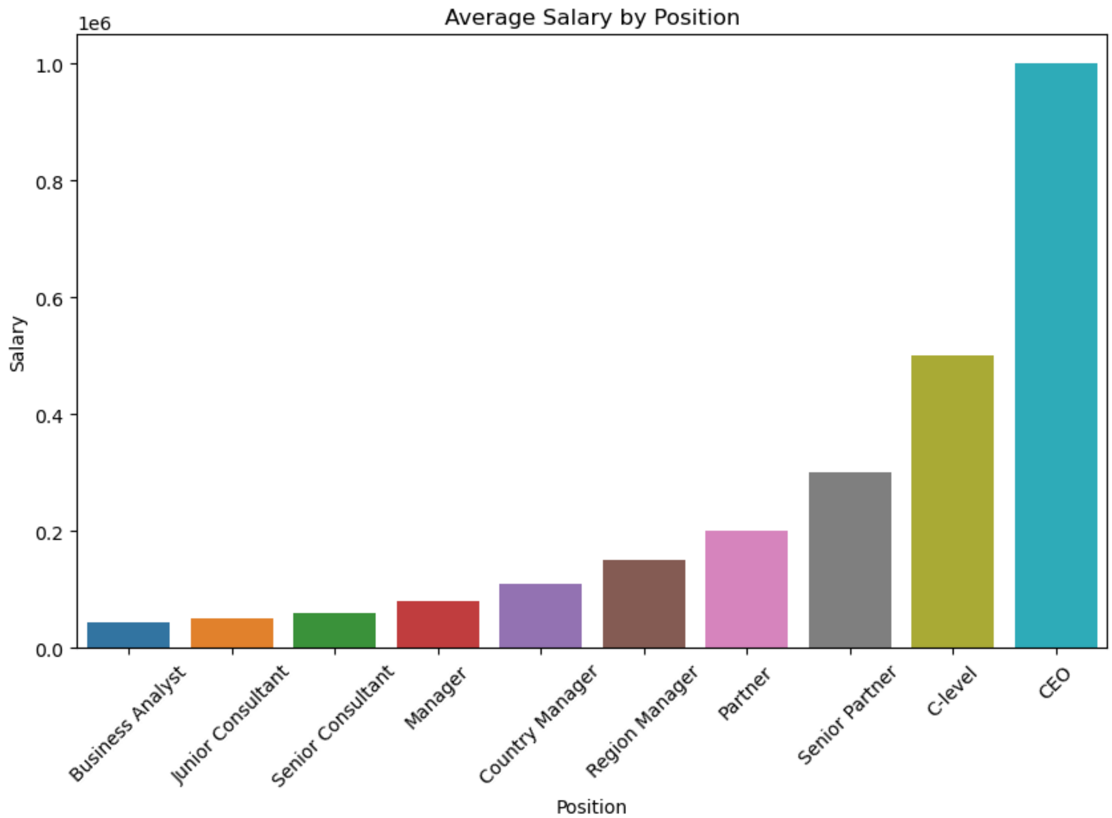
[48]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Position', y='Salary', data=data)
plt.xticks(rotation=45)
plt.title('Salary Distribution by Position')
plt.show()
```



[49]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Level', y='Salary', data=data)
plt.title('Level vs Salary')
plt.show()
```



[51]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Position', y='Salary', data=data, errorbar=None)
plt.xticks(rotation=45)
plt.title('Average Salary by Position')
plt.show()
```



[52]:
```
correlation = data['Level'].corr(data['Salary'])
print("Correlation between Level and Salary:", correlation)
```

```
Correlation between Level and Salary: 0.8179494074776199
```

0.0.1 Based on the exploratory data analysis, answer the following

Q1. Can you use a simple linear regression model to fit this data (Yes/No)? Ans. Yes

Q2. * If you use a simple linear model to fit this data, will it fit well (Yes/N? Ans. No

Q3. * Explain why you said yes or no to the above question Ans. Yes, you can use a simple linear regression model to fit this data. In a simple linear regression model, you have one independent variable (predictor) and one dependent variable (response). In the given dataset, the 'Level' can be considered as the independent variable, and 'Salary' as the dependent variable. You can fit a linear regression model to predict the salary based on the level of the position.

No, a simple linear regression model may not fit the data well. Based on the exploratory data analysis, we observed that the relationship between the 'Level' and 'Salary' is not purely linear. Instead, it seems to follow a non-linear pattern, possibly a polynomial relationship.er

[57]:
```
X = data[['Level']].values
y = data['Salary'].values
```

```
X_poly = np.c_[np.ones_like(X), X, X**2]  # Include constant term, X, and X^2

coefficients = np.linalg.inv(X_poly.T.dot(X_poly)).dot(X_poly.T).dot(y)

intercept, w1, w2 = coefficients

print("Intercept (b):", intercept)
print("Coefficients (w1, w2):", w1, w2)
```

```
Intercept (b): 232166.66666667152
Coefficients (w1, w2): -132871.21212121303 19431.818181818264
```

0.0.2 By analysing the cost function, answer the followingQ1.

Can you use mean squared error as the cost function (Yes/No)?Ans: Yes eQ2. r Explain why you said yes or no to the above questionAns: We can use mean squared error (MSE) as the cost function for various machine learning models, including linear regression models. The MSE measures the average squared difference between the actual and predicted values. It is widely used as a measure of the quality of an estimator—it quantifies the difference between the estimator and the estimated value.

In the context of linear regression, the goal is to minimize the MSE by adjusting the model parameters (coefficients) such that the predicted values closely match the actual values. This optimization process, often referred to as training or fitting the model, involves adjusting the coefficients iteratively using techniques like gradient descent until the MSE is minimized.swer

[102]:
```
def mean_squared_error(y_true, y_pred):

    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    squared_errors = (y_true - y_pred) ** 2
    mse = np.mean(squared_errors)
    return mse
```

[113]:
```
y_true = np.array([3, 5, 7, 9, 11])
y_pred = np.array([2.8, 5.2, 6.9, 8.5, 10.7])
mse_value = mean_squared_error(y_true, y_pred)
print("Mean Squared Error:", mse_value)
```

```
Mean Squared Error: 0.0860000000000001
```

[114]:
```
def gradient_descent(X, y, theta, learning_rate, num_iterations):
    m = len(y)

    cost_history = np.zeros(num_iterations)
    for i in range(num_iterations):
        y_pred = np.dot(X, theta)
        errors = y_pred - y
```

```
        gradients = (1 / m) * np.dot(X.T, errors)
        theta -= learning_rate * gradients
        cost = np.mean(errors ** 2)
        cost_history[i] = cost
        print("Iteration:", i+1, "Cost:", cost)

    return theta, cost_history
```

[115]:
```
import numpy as np
import matplotlib.pyplot as plt

# Define the gradient_descent function (from the previous code snippet)

# Step 1: Data Preparation
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([3, 5, 7, 9, 11])

# Add a column of ones to X to account for the intercept term
X_with_intercept = np.hstack((np.ones((X.shape[0], 1)), X))

# Step 2: Model Training
# Initialize parameters (theta) with zeros
theta_initial = np.zeros(X_with_intercept.shape[1])

# Set hyperparameters for gradient descent
learning_rate = 0.01
num_iterations = 1000

# Perform gradient descent to estimate parameters (theta)
theta_optimized, cost_history = gradient_descent(X_with_intercept, y,␣
↪theta_initial, learning_rate, num_iterations)
```

```
Iteration: 1 Cost: 57.0
Iteration: 2 Cost: 44.3174
Iteration: 3 Cost: 34.45820132
Iteration: 4 Cost: 26.79385308418399
Iteration: 5 Cost: 20.835733603941723
Iteration: 6 Cost: 16.20399907696077
Iteration: 7 Cost: 12.603367247978003
Iteration: 8 Cost: 9.804292156879507
Iteration: 9 Cost: 7.628329800064098
Iteration: 10 Cost: 5.936761716015316
Iteration: 11 Cost: 4.621751209818714
Iteration: 12 Cost: 3.599468395741636
Iteration: 13 Cost: 2.8047457558661444
Iteration: 14 Cost: 2.1869234883901805
Iteration: 15 Cost: 1.7066197723655434
Iteration: 16 Cost: 1.3332200422222633
```

```
Iteration: 17 Cost: 1.042925204757756
Iteration: 18 Cost: 0.8172343657596748
Iteration: 19 Cost: 0.6417653349041729
Iteration: 20 Cost: 0.5053377120895901
Iteration: 21 Cost: 0.3992600988286402
Iteration: 22 Cost: 0.31677599199166623
Iteration: 23 Cost: 0.25263303373512463
Iteration: 24 Cost: 0.20274815582754419
Iteration: 25 Cost: 0.16394727018038616
Iteration: 26 Cost: 0.13376290996637785
Iteration: 27 Cost: 0.110276920255669
Iteration: 28 Cost: 0.09199816916076882
Iteration: 29 Cost: 0.07776748315820772
Iteration: 30 Cost: 0.06668374588908968
Iteration: 31 Cost: 0.058046448984757894
Iteration: 32 Cost: 0.05131103233654675
Iteration: 33 Cost: 0.04605416659933678
Iteration: 34 Cost: 0.04194676456997769
Iteration: 35 Cost: 0.03873300082380447
Iteration: 36 Cost: 0.03621400203943921
Iteration: 37 Cost: 0.03423516821440106
Iteration: 38 Cost: 0.03267631645572755
Iteration: 39 Cost: 0.03144401897861062
Iteration: 40 Cost: 0.030465646834294625
Iteration: 41 Cost: 0.029684739634561225
Iteration: 42 Cost: 0.029057406076922525
Iteration: 43 Cost: 0.028549525791715224
Iteration: 44 Cost: 0.028134574119298927
Iteration: 45 Cost: 0.027791931139472165
Iteration: 46 Cost: 0.027505567147965572
Iteration: 47 Cost: 0.0272630207746453
Iteration: 48 Cost: 0.02705460459497746
Iteration: 49 Cost: 0.026872787589771224
Iteration: 50 Cost: 0.02671171508290197
Iteration: 51 Cost: 0.026566835551406133
Iteration: 52 Cost: 0.026434610515817532
Iteration: 53 Cost: 0.026312289015263744
Iteration: 54 Cost: 0.026197732289342662
Iteration: 55 Cost: 0.026089277489648975
Iteration: 56 Cost: 0.025985631732095178
Iteration: 57 Cost: 0.025885789735505066
Iteration: 58 Cost: 0.02578896979565663
Iteration: 59 Cost: 0.025694564012901034
Iteration: 60 Cost: 0.0256020996002033
Iteration: 61 Cost: 0.02551120880485503
Iteration: 62 Cost: 0.02542160552627193
Iteration: 63 Cost: 0.025333067139179134
Iteration: 64 Cost: 0.025245420363349248
```

```
Iteration: 65 Cost: 0.02515853027904565
Iteration: 66 Cost: 0.02507229178786462
Iteration: 67 Cost: 0.024986622974579265
Iteration: 68 Cost: 0.02490145994678048
Iteration: 69 Cost: 0.02481675282332232
Iteration: 70 Cost: 0.02473246261582742
Iteration: 71 Cost: 0.024648558804433063
Iteration: 72 Cost: 0.02456501745322826
Iteration: 73 Cost: 0.024481819745232357
Iteration: 74 Cost: 0.024398950843518088
Iteration: 75 Cost: 0.024316399005869953
Iteration: 76 Cost: 0.024234154896537893
Iteration: 77 Cost: 0.02415221105120661
Iteration: 78 Cost: 0.02407056146107195
Iteration: 79 Cost: 0.02398920124950852
Iteration: 80 Cost: 0.023908126420717362
Iteration: 81 Cost: 0.023827333664325274
Iteration: 82 Cost: 0.02374682020348429
Iteration: 83 Cost: 0.023666583676784706
Iteration: 84 Cost: 0.023586622046453266
Iteration: 85 Cost: 0.023506933526988422
Iteration: 86 Cost: 0.023427516529678737
Iteration: 87 Cost: 0.023348369619471503
Iteration: 88 Cost: 0.023269491481440092
Iteration: 89 Cost: 0.023190880894715167
Iteration: 90 Cost: 0.023112536712216393
Iteration: 91 Cost: 0.02303445784489423
Iteration: 92 Cost: 0.02295664324947784
Iteration: 93 Cost: 0.022879091918947988
Iteration: 94 Cost: 0.022801802875128292
Iteration: 95 Cost: 0.022724775162925263
Iteration: 96 Cost: 0.022648007845845898
Iteration: 97 Cost: 0.022571500002512977
Iteration: 98 Cost: 0.022495250723952996
Iteration: 99 Cost: 0.022419259111485623
Iteration: 100 Cost: 0.02234352427508125
Iteration: 101 Cost: 0.02226804533208234
Iteration: 102 Cost: 0.02219282140620519
Iteration: 103 Cost: 0.02211785162676469
Iteration: 104 Cost: 0.022043135128067503
Iteration: 105 Cost: 0.021968671048938536
Iteration: 106 Cost: 0.021894458532350856
Iteration: 107 Cost: 0.021820496725136522
Iteration: 108 Cost: 0.021746784777757988
Iteration: 109 Cost: 0.021673321844129616
Iteration: 110 Cost: 0.02160010708147723
Iteration: 111 Cost: 0.021527139650224424
Iteration: 112 Cost: 0.021454418713905212
```

```
Iteration: 113 Cost: 0.02138194343909172
Iteration: 114 Cost: 0.021309712995336298
Iteration: 115 Cost: 0.021237726555125805
Iteration: 116 Cost: 0.021165983293841784
Iteration: 117 Cost: 0.02109448238972955
Iteration: 118 Cost: 0.021023223023870236
Iteration: 119 Cost: 0.020952204380158816
Iteration: 120 Cost: 0.02088142564528338
Iteration: 121 Cost: 0.020810886008707632
Iteration: 122 Cost: 0.020740584662655843
Iteration: 123 Cost: 0.02067052080209762
Iteration: 124 Cost: 0.020600693624735954
Iteration: 125 Cost: 0.02053110233099408
Iteration: 126 Cost: 0.020461746124004143
Iteration: 127 Cost: 0.02039262420959709
Iteration: 128 Cost: 0.020323735796290744
Iteration: 129 Cost: 0.02025508009528118
Iteration: 130 Cost: 0.020186656320431567
Iteration: 131 Cost: 0.020118463688263068
Iteration: 132 Cost: 0.020050501417945378
Iteration: 133 Cost: 0.01998276873128748
Iteration: 134 Cost: 0.019915264852727956
Iteration: 135 Cost: 0.01984798900932596
Iteration: 136 Cost: 0.01978094043075321
Iteration: 137 Cost: 0.01971411834928317
Iteration: 138 Cost: 0.019647521999783944
Iteration: 139 Cost: 0.01958115061970812
Iteration: 140 Cost: 0.019515003449084543
Iteration: 141 Cost: 0.01944907973051004
Iteration: 142 Cost: 0.0193833787091395
Iteration: 143 Cost: 0.019317899632678114
Iteration: 144 Cost: 0.01925264175137257
Iteration: 145 Cost: 0.01918760431800231
Iteration: 146 Cost: 0.019122786587871192
Iteration: 147 Cost: 0.01905818781879833
Iteration: 148 Cost: 0.018993807271110313
Iteration: 149 Cost: 0.01892964420763252
Iteration: 150 Cost: 0.018865697893680616
Iteration: 151 Cost: 0.01880196759705193
Iteration: 152 Cost: 0.018738452588017235
Iteration: 153 Cost: 0.01867515213931282
Iteration: 154 Cost: 0.018612065526131445
Iteration: 155 Cost: 0.018549192026114037
Iteration: 156 Cost: 0.018486530919342263
Iteration: 157 Cost: 0.018424081488329632
Iteration: 158 Cost: 0.018361843018013285
Iteration: 159 Cost: 0.018299814795745918
Iteration: 160 Cost: 0.01823799611128776
```

```
Iteration: 161 Cost: 0.018176386256798172
Iteration: 162 Cost: 0.018114984526827548
Iteration: 163 Cost: 0.018053790218309926
Iteration: 164 Cost: 0.017992802630553632
Iteration: 165 Cost: 0.017932021065234414
Iteration: 166 Cost: 0.01787144482638723
Iteration: 167 Cost: 0.017811073220397492
Iteration: 168 Cost: 0.017750905555994144
Iteration: 169 Cost: 0.01769094114424135
Iteration: 170 Cost: 0.017631179298530433
Iteration: 171 Cost: 0.017571619334571926
Iteration: 172 Cost: 0.01751226057038858
Iteration: 173 Cost: 0.017453102326306064
Iteration: 174 Cost: 0.01739414392494699
Iteration: 175 Cost: 0.017335384691221478
Iteration: 176 Cost: 0.01727682395232081
Iteration: 177 Cost: 0.017218461037708375
Iteration: 178 Cost: 0.017160295279113134
Iteration: 179 Cost: 0.017102326010521862
Iteration: 180 Cost: 0.01704455256817042
Iteration: 181 Cost: 0.01698697429053727
Iteration: 182 Cost: 0.016929590518336146
Iteration: 183 Cost: 0.01687240059450682
Iteration: 184 Cost: 0.01681540386420954
Iteration: 185 Cost: 0.016758599674816382
Iteration: 186 Cost: 0.016701987375904028
Iteration: 187 Cost: 0.016645566319246442
Iteration: 188 Cost: 0.016589335858807192
Iteration: 189 Cost: 0.016533295350732606
Iteration: 190 Cost: 0.016477444153343495
Iteration: 191 Cost: 0.016421781627128897
Iteration: 192 Cost: 0.01636630713473768
Iteration: 193 Cost: 0.016311020040971996
Iteration: 194 Cost: 0.016255919712779794
Iteration: 195 Cost: 0.016201005519247375
Iteration: 196 Cost: 0.016146276831592414
Iteration: 197 Cost: 0.016091733023156883
Iteration: 198 Cost: 0.016037373469399277
Iteration: 199 Cost: 0.01598319754788826
Iteration: 200 Cost: 0.015929204638294718
Iteration: 201 Cost: 0.015875394122385412
Iteration: 202 Cost: 0.015821765384015418
Iteration: 203 Cost: 0.015768317809121253
Iteration: 204 Cost: 0.01571505078571387
Iteration: 205 Cost: 0.01566196370387117
Iteration: 206 Cost: 0.015609055955732268
Iteration: 207 Cost: 0.015556326935488718
Iteration: 208 Cost: 0.015503776039379363
```

```
Iteration: 209 Cost: 0.015451402665682143
Iteration: 210 Cost: 0.015399206214707759
Iteration: 211 Cost: 0.015347186088792935
Iteration: 212 Cost: 0.015295341692293216
Iteration: 213 Cost: 0.015243672431576066
Iteration: 214 Cost: 0.015192177715014892
Iteration: 215 Cost: 0.015140856952981077
Iteration: 216 Cost: 0.015089709557838102
Iteration: 217 Cost: 0.015038734943934498
Iteration: 218 Cost: 0.014987932527597109
Iteration: 219 Cost: 0.014937301727124944
Iteration: 220 Cost: 0.0148868419627813
Iteration: 221 Cost: 0.01483655265678845
Iteration: 222 Cost: 0.014786433233320323
Iteration: 223 Cost: 0.014736483118495954
Iteration: 224 Cost: 0.01468670174037318
Iteration: 225 Cost: 0.014637088528941689
Iteration: 226 Cost: 0.01458764291611676
Iteration: 227 Cost: 0.014538364335733103
Iteration: 228 Cost: 0.014489252223537535
Iteration: 229 Cost: 0.014440306017183014
Iteration: 230 Cost: 0.01439152515622245
Iteration: 231 Cost: 0.0143429090821019
Iteration: 232 Cost: 0.01429445723815398
Iteration: 233 Cost: 0.014246169069592283
Iteration: 234 Cost: 0.014198044023504077
Iteration: 235 Cost: 0.014150081548844725
Iteration: 236 Cost: 0.014102281096430903
Iteration: 237 Cost: 0.014054642118934835
Iteration: 238 Cost: 0.014007164070876894
Iteration: 239 Cost: 0.013959846408621088
Iteration: 240 Cost: 0.013912688590367006
Iteration: 241 Cost: 0.01386569007614539
Iteration: 242 Cost: 0.013818850327810384
Iteration: 243 Cost: 0.01377216880903413
Iteration: 244 Cost: 0.01372564498530075
Iteration: 245 Cost: 0.01367927832390008
Iteration: 246 Cost: 0.013633068293921174
Iteration: 247 Cost: 0.01358701436624674
Iteration: 248 Cost: 0.01354111601354705
Iteration: 249 Cost: 0.01349537271027341
Iteration: 250 Cost: 0.013449783932652884
Iteration: 251 Cost: 0.013404349158681453
Iteration: 252 Cost: 0.013359067868118909
Iteration: 253 Cost: 0.013313939542482223
Iteration: 254 Cost: 0.01326896366503987
Iteration: 255 Cost: 0.013224139720805956
Iteration: 256 Cost: 0.013179467196534312
```

```
Iteration: 257 Cost: 0.013134945580712373
Iteration: 258 Cost: 0.01309057436355573
Iteration: 259 Cost: 0.01304635303700187
Iteration: 260 Cost: 0.013002281094704854
Iteration: 261 Cost: 0.012958358032029058
Iteration: 262 Cost: 0.012914583346043507
Iteration: 263 Cost: 0.012870956535516214
Iteration: 264 Cost: 0.01282747710090873
Iteration: 265 Cost: 0.012784144544369413
Iteration: 266 Cost: 0.012740958369729063
Iteration: 267 Cost: 0.012697918082494297
Iteration: 268 Cost: 0.01265502318984208
Iteration: 269 Cost: 0.012612273200614702
Iteration: 270 Cost: 0.012569667625312947
Iteration: 271 Cost: 0.012527205976091499
Iteration: 272 Cost: 0.012484887766753355
Iteration: 273 Cost: 0.01244271251274318
Iteration: 274 Cost: 0.012400679731143331
Iteration: 275 Cost: 0.012358788940666849
Iteration: 276 Cost: 0.012317039661653177
Iteration: 277 Cost: 0.012275431416061532
Iteration: 278 Cost: 0.012233963727466466
Iteration: 279 Cost: 0.012192636121051825
Iteration: 280 Cost: 0.012151448123605274
Iteration: 281 Cost: 0.012110399263513118
Iteration: 282 Cost: 0.012069489070754955
Iteration: 283 Cost: 0.01202871707689807
Iteration: 284 Cost: 0.011988082815091991
Iteration: 285 Cost: 0.011947585820063594
Iteration: 286 Cost: 0.011907225628111258
Iteration: 287 Cost: 0.01186700177710016
Iteration: 288 Cost: 0.011826913806456077
Iteration: 289 Cost: 0.011786961257161218
Iteration: 290 Cost: 0.011747143671747862
Iteration: 291 Cost: 0.011707460594294231
Iteration: 292 Cost: 0.011667911570418186
Iteration: 293 Cost: 0.011628496147272715
Iteration: 294 Cost: 0.011589213873540736
Iteration: 295 Cost: 0.011550064299429643
Iteration: 296 Cost: 0.011511046976666122
Iteration: 297 Cost: 0.011472161458491482
Iteration: 298 Cost: 0.011433407299655985
Iteration: 299 Cost: 0.01139478405641399
Iteration: 300 Cost: 0.011356291286519105
Iteration: 301 Cost: 0.011317928549218679
Iteration: 302 Cost: 0.011279695405248983
Iteration: 303 Cost: 0.011241591416830352
Iteration: 304 Cost: 0.011203616147661735
```

```
Iteration: 305 Cost: 0.011165769162916093
Iteration: 306 Cost: 0.011128050029235257
Iteration: 307 Cost: 0.011090458314725124
Iteration: 308 Cost: 0.01105299358895023
Iteration: 309 Cost: 0.011015655422929416
Iteration: 310 Cost: 0.01097844338913077
Iteration: 311 Cost: 0.010941357061466275
Iteration: 312 Cost: 0.010904396015287646
Iteration: 313 Cost: 0.010867559827380798
Iteration: 314 Cost: 0.010830848075961593
Iteration: 315 Cost: 0.01079426034067062
Iteration: 316 Cost: 0.010757796202568237
Iteration: 317 Cost: 0.010721455244130464
Iteration: 318 Cost: 0.0106852370492434
Iteration: 319 Cost: 0.010649141203199193
Iteration: 320 Cost: 0.01061316729269027
Iteration: 321 Cost: 0.010577314905805968
Iteration: 322 Cost: 0.010541583632026688
Iteration: 323 Cost: 0.010505973062219828
Iteration: 324 Cost: 0.010470482788634652
Iteration: 325 Cost: 0.010435112404898018
Iteration: 326 Cost: 0.0103998615060095
Iteration: 327 Cost: 0.010364729688336875
Iteration: 328 Cost: 0.010329716549611179
Iteration: 329 Cost: 0.010294821688922713
Iteration: 330 Cost: 0.010260044706715838
Iteration: 331 Cost: 0.010225385204784833
Iteration: 332 Cost: 0.010190842786268913
Iteration: 333 Cost: 0.010156417055648272
Iteration: 334 Cost: 0.010122107618738834
Iteration: 335 Cost: 0.010087914082688469
Iteration: 336 Cost: 0.010053836055971831
Iteration: 337 Cost: 0.010019873148386479
Iteration: 338 Cost: 0.009986024971047858
Iteration: 339 Cost: 0.009952291136385231
Iteration: 340 Cost: 0.00991867125813716
Iteration: 341 Cost: 0.009885164951346946
Iteration: 342 Cost: 0.00985177183235834
Iteration: 343 Cost: 0.00981849151881113
Iteration: 344 Cost: 0.009785323629636836
Iteration: 345 Cost: 0.009752267785054081
Iteration: 346 Cost: 0.009719323606564613
Iteration: 347 Cost: 0.009686490716948624
Iteration: 348 Cost: 0.009653768740260811
Iteration: 349 Cost: 0.00962115730182576
Iteration: 350 Cost: 0.00958865602823342
Iteration: 351 Cost: 0.009556264547335894
Iteration: 352 Cost: 0.00952398248824161
```

```
Iteration: 353 Cost: 0.009491809481312483
Iteration: 354 Cost: 0.009459745158158839
Iteration: 355 Cost: 0.00942778915163563
Iteration: 356 Cost: 0.009395941095837902
Iteration: 357 Cost: 0.009364200626096891
Iteration: 358 Cost: 0.009332567378975625
Iteration: 359 Cost: 0.009301040992264966
Iteration: 360 Cost: 0.009269621104979192
Iteration: 361 Cost: 0.009238307357352144
Iteration: 362 Cost: 0.009207099390832993
Iteration: 363 Cost: 0.009175996848082239
Iteration: 364 Cost: 0.009144999372967227
Iteration: 365 Cost: 0.00911410661055857
Iteration: 366 Cost: 0.009083318207125863
Iteration: 367 Cost: 0.009052633810133543
Iteration: 368 Cost: 0.00902205306823694
Iteration: 369 Cost: 0.00899157563127842
Iteration: 370 Cost: 0.008961201150283137
Iteration: 371 Cost: 0.00893092927745495
Iteration: 372 Cost: 0.008900759666172943
Iteration: 373 Cost: 0.00887069197098689
Iteration: 374 Cost: 0.008840725847613435
Iteration: 375 Cost: 0.008810860952932648
Iteration: 376 Cost: 0.008781096944983175
Iteration: 377 Cost: 0.008751433482959254
Iteration: 378 Cost: 0.008721870227206276
Iteration: 379 Cost: 0.008692406839216899
Iteration: 380 Cost: 0.008663042981627487
Iteration: 381 Cost: 0.008633778318213898
Iteration: 382 Cost: 0.008604612513887819
Iteration: 383 Cost: 0.008575545234693088
Iteration: 384 Cost: 0.008546576147801434
Iteration: 385 Cost: 0.008517704921508985
Iteration: 386 Cost: 0.00848893122523236
Iteration: 387 Cost: 0.00846025472950517
Iteration: 388 Cost: 0.00843167510597369
Iteration: 389 Cost: 0.008403192027393511
Iteration: 390 Cost: 0.0083748051676257
Iteration: 391 Cost: 0.008346514201633263
Iteration: 392 Cost: 0.008318318805476742
Iteration: 393 Cost: 0.008290218656311441
Iteration: 394 Cost: 0.00826221343238308
Iteration: 395 Cost: 0.00823430281302418
Iteration: 396 Cost: 0.008206486478650793
Iteration: 397 Cost: 0.008178764110758185
Iteration: 398 Cost: 0.008151135391917917
Iteration: 399 Cost: 0.008123600005773682
Iteration: 400 Cost: 0.008096157637037938
```

```
Iteration: 401 Cost: 0.008068807971488127
Iteration: 402 Cost: 0.00804155069596316
Iteration: 403 Cost: 0.008014385498360099
Iteration: 404 Cost: 0.007987312067629777
Iteration: 405 Cost: 0.00796033009377444
Iteration: 406 Cost: 0.00793343926784301
Iteration: 407 Cost: 0.007906639281928297
Iteration: 408 Cost: 0.007879929829163418
Iteration: 409 Cost: 0.00785331060371782
Iteration: 410 Cost: 0.007826781300794212
Iteration: 411 Cost: 0.007800341616624832
Iteration: 412 Cost: 0.00777399124846837
Iteration: 413 Cost: 0.00774772989460584
Iteration: 414 Cost: 0.007721557254337768
Iteration: 415 Cost: 0.007695473027980374
Iteration: 416 Cost: 0.007669476916862148
Iteration: 417 Cost: 0.0076435686233206675
Iteration: 418 Cost: 0.0076177478506990635
Iteration: 419 Cost: 0.007592014303342473
Iteration: 420 Cost: 0.0075663676865947395
Iteration: 421 Cost: 0.007540807706795315
Iteration: 422 Cost: 0.007515334071275642
Iteration: 423 Cost: 0.007489946488355484
Iteration: 424 Cost: 0.007464644667340472
Iteration: 425 Cost: 0.0074394283185175965
Iteration: 426 Cost: 0.007414297153153161
Iteration: 427 Cost: 0.007389250883488366
Iteration: 428 Cost: 0.007364289222736735
Iteration: 429 Cost: 0.007339411885080605
Iteration: 430 Cost: 0.007314618585667688
Iteration: 431 Cost: 0.007289909040608026
Iteration: 432 Cost: 0.007265282966970763
Iteration: 433 Cost: 0.007240740082780481
Iteration: 434 Cost: 0.007216280107014845
Iteration: 435 Cost: 0.007191902759600313
Iteration: 436 Cost: 0.0071676077614097515
Iteration: 437 Cost: 0.007143394834259018
Iteration: 438 Cost: 0.00711926370090335
Iteration: 439 Cost: 0.007095214085035044
Iteration: 440 Cost: 0.007071245711279292
Iteration: 441 Cost: 0.007047358305191829
Iteration: 442 Cost: 0.007023551593255402
Iteration: 443 Cost: 0.006999825302876841
Iteration: 444 Cost: 0.0069761791623835275
Iteration: 445 Cost: 0.006952612901021014
Iteration: 446 Cost: 0.006929126248949098
Iteration: 447 Cost: 0.00690571893723939
Iteration: 448 Cost: 0.006882390697871806
```

```
Iteration: 449 Cost: 0.006859141263731748
Iteration: 450 Cost: 0.006835970368606878
Iteration: 451 Cost: 0.006812877747184259
Iteration: 452 Cost: 0.0067898631350471845
Iteration: 453 Cost: 0.006766926268672058
Iteration: 454 Cost: 0.006744066885425631
Iteration: 455 Cost: 0.0067212847235617905
Iteration: 456 Cost: 0.006698579522218714
Iteration: 457 Cost: 0.0066759510214156655
Iteration: 458 Cost: 0.006653398962050157
Iteration: 459 Cost: 0.006630923085895152
Iteration: 460 Cost: 0.006608523135595745
Iteration: 461 Cost: 0.006586198854666445
Iteration: 462 Cost: 0.00656394998748838
Iteration: 463 Cost: 0.006541776279305809
Iteration: 464 Cost: 0.006519677476223944
Iteration: 465 Cost: 0.006497653325205431
Iteration: 466 Cost: 0.006475703574068022
Iteration: 467 Cost: 0.006453827971480889
Iteration: 468 Cost: 0.006432026266962589
Iteration: 469 Cost: 0.0064102982108777545
Iteration: 470 Cost: 0.00638864355443432
Iteration: 471 Cost: 0.006367062049680563
Iteration: 472 Cost: 0.006345553449502469
Iteration: 473 Cost: 0.006324117507620829
Iteration: 474 Cost: 0.006302753978588235
Iteration: 475 Cost: 0.00628146261778658
Iteration: 476 Cost: 0.006260243181424043
Iteration: 477 Cost: 0.0062390954265323095
Iteration: 478 Cost: 0.006218019110964093
Iteration: 479 Cost: 0.006197013993389631
Iteration: 480 Cost: 0.006176079833294775
Iteration: 481 Cost: 0.006155216390977798
Iteration: 482 Cost: 0.00613442342754663
Iteration: 483 Cost: 0.006113700704916197
Iteration: 484 Cost: 0.006093047985805879
Iteration: 485 Cost: 0.006072465033736268
Iteration: 486 Cost: 0.006051951613027157
Iteration: 487 Cost: 0.006031507488794381
Iteration: 488 Cost: 0.006011132426947091
Iteration: 489 Cost: 0.005990826194185489
Iteration: 490 Cost: 0.0059705885579976435
Iteration: 491 Cost: 0.005950419286657115
Iteration: 492 Cost: 0.005930318149220396
Iteration: 493 Cost: 0.005910284915524091
Iteration: 494 Cost: 0.005890319356182173
Iteration: 495 Cost: 0.005870421242583671
Iteration: 496 Cost: 0.005850590346889792
```

```
Iteration: 497 Cost: 0.005830826442031468
Iteration: 498 Cost: 0.005811129301706629
Iteration: 499 Cost: 0.005791498700377753
Iteration: 500 Cost: 0.005771934413269128
Iteration: 501 Cost: 0.005752436216364506
Iteration: 502 Cost: 0.005733003886404033
Iteration: 503 Cost: 0.005713637200882591
Iteration: 504 Cost: 0.005694335938046238
Iteration: 505 Cost: 0.005675099876890335
Iteration: 506 Cost: 0.0056559287971567705
Iteration: 507 Cost: 0.005636822479331687
Iteration: 508 Cost: 0.005617780704642358
Iteration: 509 Cost: 0.0055988032550555135
Iteration: 510 Cost: 0.005579889913274192
Iteration: 511 Cost: 0.00556104046273546
Iteration: 512 Cost: 0.0055422546876081665
Iteration: 513 Cost: 0.005523532372790014
Iteration: 514 Cost: 0.005504873303905447
Iteration: 515 Cost: 0.00548627726730321
Iteration: 516 Cost: 0.005467744050053555
Iteration: 517 Cost: 0.005449273439946259
Iteration: 518 Cost: 0.00543086522548782
Iteration: 519 Cost: 0.005412519195899202
Iteration: 520 Cost: 0.005394235141113427
Iteration: 521 Cost: 0.005376012851773161
Iteration: 522 Cost: 0.00535785211922827
Iteration: 523 Cost: 0.005339752735533469
Iteration: 524 Cost: 0.00532171449344597
Iteration: 525 Cost: 0.005303737186423002
Iteration: 526 Cost: 0.0052858206086195516
Iteration: 527 Cost: 0.005267964554886044
Iteration: 528 Cost: 0.005250168820765825
Iteration: 529 Cost: 0.005232433202492886
Iteration: 530 Cost: 0.005214757496989695
Iteration: 531 Cost: 0.0051971415018645965
Iteration: 532 Cost: 0.005179585015409696
Iteration: 533 Cost: 0.005162087836598524
Iteration: 534 Cost: 0.0051446497650838065
Iteration: 535 Cost: 0.005127270601194637
Iteration: 536 Cost: 0.005109950145935174
Iteration: 537 Cost: 0.005092688200981387
Iteration: 538 Cost: 0.0050754845686794255
Iteration: 539 Cost: 0.005058339052043089
Iteration: 540 Cost: 0.0050412514547514735
Iteration: 541 Cost: 0.005024221581147057
Iteration: 542 Cost: 0.005007249236233223
Iteration: 543 Cost: 0.004990334225672043
Iteration: 544 Cost: 0.004973476355782093
```

```
Iteration: 545 Cost: 0.004956675433536122
Iteration: 546 Cost: 0.004939931266559174
Iteration: 547 Cost: 0.004923243663125981
Iteration: 548 Cost: 0.004906612432158999
Iteration: 549 Cost: 0.004890037383226197
Iteration: 550 Cost: 0.004873518326538631
Iteration: 551 Cost: 0.004857055072948812
Iteration: 552 Cost: 0.004840647433948022
Iteration: 553 Cost: 0.0048242952216643485
Iteration: 554 Cost: 0.004807998248860475
Iteration: 555 Cost: 0.0047917563289318595
Iteration: 556 Cost: 0.0047755692759040395
Iteration: 557 Cost: 0.004759436904430828
Iteration: 558 Cost: 0.004743359029792345
Iteration: 559 Cost: 0.004727335467892609
Iteration: 560 Cost: 0.004711366035257405
Iteration: 561 Cost: 0.00469545054903249
Iteration: 562 Cost: 0.004679588826981268
Iteration: 563 Cost: 0.004663780687482677
Iteration: 564 Cost: 0.004648025949529393
Iteration: 565 Cost: 0.0046323244327252655
Iteration: 566 Cost: 0.004616675957283793
Iteration: 567 Cost: 0.004601080344025782
Iteration: 568 Cost: 0.0045855374143771
Iteration: 569 Cost: 0.004570046990367106
Iteration: 570 Cost: 0.0045546088946262575
Iteration: 571 Cost: 0.004539222950384248
Iteration: 572 Cost: 0.004523888981467801
Iteration: 573 Cost: 0.004508606812298864
Iteration: 574 Cost: 0.004493376267892437
Iteration: 575 Cost: 0.004478197173854814
Iteration: 576 Cost: 0.004463069356381166
Iteration: 577 Cost: 0.004447992642253919
Iteration: 578 Cost: 0.004432966858840609
Iteration: 579 Cost: 0.004417991834092165
Iteration: 580 Cost: 0.004403067396540207
Iteration: 581 Cost: 0.004388193375296144
Iteration: 582 Cost: 0.004373369600048321
Iteration: 583 Cost: 0.004358595901060526
Iteration: 584 Cost: 0.0043438721091699755
Iteration: 585 Cost: 0.004329198055785325
Iteration: 586 Cost: 0.004314573572884533
Iteration: 587 Cost: 0.0042999984930135425
Iteration: 588 Cost: 0.004285472649283565
Iteration: 589 Cost: 0.004270995875369868
Iteration: 590 Cost: 0.004256568005509579
Iteration: 591 Cost: 0.004242188874499485
Iteration: 592 Cost: 0.004227858317694911
```

```
Iteration: 593 Cost: 0.004213576171006975
Iteration: 594 Cost: 0.004199342270901258
Iteration: 595 Cost: 0.004185156454395768
Iteration: 596 Cost: 0.0041710185590591465
Iteration: 597 Cost: 0.004156928423008627
Iteration: 598 Cost: 0.0041428858849083565
Iteration: 599 Cost: 0.004128890783967549
Iteration: 600 Cost: 0.004114942959938422
Iteration: 601 Cost: 0.0041010422531147055
Iteration: 602 Cost: 0.004087188504329575
Iteration: 603 Cost: 0.0040733815549538095
Iteration: 604 Cost: 0.004059621246894254
Iteration: 605 Cost: 0.004045907422591523
Iteration: 606 Cost: 0.0040322399250186675
Iteration: 607 Cost: 0.004018618597679288
Iteration: 608 Cost: 0.004005043284605311
Iteration: 609 Cost: 0.003991513830355876
Iteration: 610 Cost: 0.003978030080015111
Iteration: 611 Cost: 0.003964591879190272
Iteration: 612 Cost: 0.003951199074010531
Iteration: 613 Cost: 0.0039378515111245654
Iteration: 614 Cost: 0.003924549037699205
Iteration: 615 Cost: 0.00391129150141765
Iteration: 616 Cost: 0.0038980787504772816
Iteration: 617 Cost: 0.003884910633588805
Iteration: 618 Cost: 0.003871786999973545
Iteration: 619 Cost: 0.003858707699362445
Iteration: 620 Cost: 0.003845672581994057
Iteration: 621 Cost: 0.00383268149861268
Iteration: 622 Cost: 0.0038197343004669365
Iteration: 623 Cost: 0.003806830839307866
Iteration: 624 Cost: 0.0037939709673874646
Iteration: 625 Cost: 0.0037811545374566795
Iteration: 626 Cost: 0.0037683814027638977
Iteration: 627 Cost: 0.0037556514170533524
Iteration: 628 Cost: 0.003742964434563323
Iteration: 629 Cost: 0.003730320310024294
Iteration: 630 Cost: 0.003717718898657725
Iteration: 631 Cost: 0.0037051600561740583
Iteration: 632 Cost: 0.003692643638771176
Iteration: 633 Cost: 0.003680169503132638
Iteration: 634 Cost: 0.003667737506426322
Iteration: 635 Cost: 0.0036553475063025266
Iteration: 636 Cost: 0.0036429993608923864
Iteration: 637 Cost: 0.0036306929288062928
Iteration: 638 Cost: 0.003618428069132267
Iteration: 639 Cost: 0.003606204641434358
Iteration: 640 Cost: 0.003594022505751183
```

```
Iteration: 641 Cost: 0.003581881522593757
Iteration: 642 Cost: 0.00356978155294472
Iteration: 643 Cost: 0.0035577224582561177
Iteration: 644 Cost: 0.003545704100448032
Iteration: 645 Cost: 0.0035337263419070765
Iteration: 646 Cost: 0.0035217890454846924
Iteration: 647 Cost: 0.003509892074495576
Iteration: 648 Cost: 0.003498035292716262
Iteration: 649 Cost: 0.0034862185643833475
Iteration: 650 Cost: 0.003474441754192168
Iteration: 651 Cost: 0.0034627047272949947
Iteration: 652 Cost: 0.0034510073492998346
Iteration: 653 Cost: 0.0034393494862683927
Iteration: 654 Cost: 0.0034277310047150732
Iteration: 655 Cost: 0.0034161517716052255
Iteration: 656 Cost: 0.003404611654353385
Iteration: 657 Cost: 0.0033931105208221206
Iteration: 658 Cost: 0.0033816482393204186
Iteration: 659 Cost: 0.0033702246786019713
Iteration: 660 Cost: 0.0033588397078640614
Iteration: 661 Cost: 0.003347493196745608
Iteration: 662 Cost: 0.0033361850153259876
Iteration: 663 Cost: 0.0033249150341235108
Iteration: 664 Cost: 0.0033136831240939144
Iteration: 665 Cost: 0.003302489156628706
Iteration: 666 Cost: 0.0032913330035540165
Iteration: 667 Cost: 0.0032802145371288493
Iteration: 668 Cost: 0.0032691336300437876
Iteration: 669 Cost: 0.0032580901554194385
Iteration: 670 Cost: 0.003247083986805051
Iteration: 671 Cost: 0.0032361149981770657
Iteration: 672 Cost: 0.003225183063937473
Iteration: 673 Cost: 0.0032142880589128833
Iteration: 674 Cost: 0.003203429858352419
Iteration: 675 Cost: 0.003192608337926839
Iteration: 676 Cost: 0.0031818233737266888
Iteration: 677 Cost: 0.003171074842261367
Iteration: 678 Cost: 0.0031603626204572493
Iteration: 679 Cost: 0.0031496865856564227
Iteration: 680 Cost: 0.0031390466156155606
Iteration: 681 Cost: 0.0031284425885040137
Iteration: 682 Cost: 0.003117874382902868
Iteration: 683 Cost: 0.00310734187780338
Iteration: 684 Cost: 0.003096844952605414
Iteration: 685 Cost: 0.0030863834871163584
Iteration: 686 Cost: 0.0030759573615497526
Iteration: 687 Cost: 0.003065566456523498
Iteration: 688 Cost: 0.003055210653059048
```

```
Iteration: 689 Cost: 0.0030448898325795637
Iteration: 690 Cost: 0.0030346038769089095
Iteration: 691 Cost: 0.003024352668270132
Iteration: 692 Cost: 0.0030141360892841
Iteration: 693 Cost: 0.0030039540229683586
Iteration: 694 Cost: 0.0029938063527352515
Iteration: 695 Cost: 0.002983692962391556
Iteration: 696 Cost: 0.0029736137361359148
Iteration: 697 Cost: 0.0029635685585586093
Iteration: 698 Cost: 0.002953557314639702
Iteration: 699 Cost: 0.002943579889747669
Iteration: 700 Cost: 0.002933636169638294
Iteration: 701 Cost: 0.002923726040453374
Iteration: 702 Cost: 0.002913849388719275
Iteration: 703 Cost: 0.0029040061013456088
Iteration: 704 Cost: 0.0028941960656240487
Iteration: 705 Cost: 0.0028844191692271176
Iteration: 706 Cost: 0.0028746753002067246
Iteration: 707 Cost: 0.0028649643469929272
Iteration: 708 Cost: 0.0028552861983926385
Iteration: 709 Cost: 0.002845640743588571
Iteration: 710 Cost: 0.0028360278721375333
Iteration: 711 Cost: 0.0028264474739696475
Iteration: 712 Cost: 0.0028168994393867356
Iteration: 713 Cost: 0.0028073836590612094
Iteration: 714 Cost: 0.0027979000240348382
Iteration: 715 Cost: 0.0027884484257175088
Iteration: 716 Cost: 0.0027790287558858194
Iteration: 717 Cost: 0.0027696409066819837
Iteration: 718 Cost: 0.0027602847706126405
Iteration: 719 Cost: 0.002750960240547536
Iteration: 720 Cost: 0.0027416672097182668
Iteration: 721 Cost: 0.0027324055717170732
Iteration: 722 Cost: 0.002723175220495803
Iteration: 723 Cost: 0.002713976050364373
Iteration: 724 Cost: 0.0027048079559898026
Iteration: 725 Cost: 0.002695670832394944
Iteration: 726 Cost: 0.002686564574957303
Iteration: 727 Cost: 0.0026774890794078465
Iteration: 728 Cost: 0.002668444241829522
Iteration: 729 Cost: 0.0026594299586566853
Iteration: 730 Cost: 0.0026504461266732954
Iteration: 731 Cost: 0.002641492643011981
Iteration: 732 Cost: 0.0026325694051530972
Iteration: 733 Cost: 0.00262367631092301
Iteration: 734 Cost: 0.0026148132584934516
Iteration: 735 Cost: 0.0026059801463801636
Iteration: 736 Cost: 0.0025971768734415314
```

```
Iteration: 737 Cost: 0.002588403338877796
Iteration: 738 Cost: 0.002579659442229517
Iteration: 739 Cost: 0.002570945083376754
Iteration: 740 Cost: 0.002562260162537809
Iteration: 741 Cost: 0.0025536045802679935
Iteration: 742 Cost: 0.00254497823745846
Iteration: 743 Cost: 0.0025363810353353595
Iteration: 744 Cost: 0.0025278128754584106
Iteration: 745 Cost: 0.00251927365971983
Iteration: 746 Cost: 0.0025107632903432426
Iteration: 747 Cost: 0.002502281669882772
Iteration: 748 Cost: 0.0024938287012214906
Iteration: 749 Cost: 0.00248540428757064
Iteration: 750 Cost: 0.0024770083324684084
Iteration: 751 Cost: 0.0024686407397788344
Iteration: 752 Cost: 0.0024603014136908062
Iteration: 753 Cost: 0.0024519902587167197
Iteration: 754 Cost: 0.0024437071796916043
Iteration: 755 Cost: 0.002435452081772004
Iteration: 756 Cost: 0.002427224870434807
Iteration: 757 Cost: 0.0024190254514762704
Iteration: 758 Cost: 0.002410853731010805
Iteration: 759 Cost: 0.0024027096154699674
Iteration: 760 Cost: 0.0023945930116015474
Iteration: 761 Cost: 0.002386503826468117
Iteration: 762 Cost: 0.0023784419674464266
Iteration: 763 Cost: 0.0023704073422259582
Iteration: 764 Cost: 0.0023623998588081533
Iteration: 765 Cost: 0.002354419425505078
Iteration: 766 Cost: 0.0023464659509387195
Iteration: 767 Cost: 0.0023385393440395552
Iteration: 768 Cost: 0.002330639514045874
Iteration: 769 Cost: 0.0023227663705025206
Iteration: 770 Cost: 0.0023149198232597736
Iteration: 771 Cost: 0.0023070997824726326
Iteration: 772 Cost: 0.0022993061585994826
Iteration: 773 Cost: 0.002291538862401273
Iteration: 774 Cost: 0.00228379780494032
Iteration: 775 Cost: 0.0022760828975794995
Iteration: 776 Cost: 0.0022683940519809516
Iteration: 777 Cost: 0.002260731180105416
Iteration: 778 Cost: 0.002253094194210871
Iteration: 779 Cost: 0.0022454830068518493
Iteration: 780 Cost: 0.0022378975308781
Iteration: 781 Cost: 0.0022303376794340036
Iteration: 782 Cost: 0.0022228033659571385
Iteration: 783 Cost: 0.002215294504177562
Iteration: 784 Cost: 0.0022078110081168536
```

```
Iteration: 785 Cost: 0.0022003527920869577
Iteration: 786 Cost: 0.002192919770689256
Iteration: 787 Cost: 0.0021855118588136714
Iteration: 788 Cost: 0.002178128971637626
Iteration: 789 Cost: 0.0021707710246250808
Iteration: 790 Cost: 0.002163437933525516
Iteration: 791 Cost: 0.0021561296143731193
Iteration: 792 Cost: 0.0021488459834855993
Iteration: 793 Cost: 0.002141586957463502
Iteration: 794 Cost: 0.0021343524531889694
Iteration: 795 Cost: 0.0021271423878249855
Iteration: 796 Cost: 0.0021199566788143794
Iteration: 797 Cost: 0.002112795243878801
Iteration: 798 Cost: 0.002105658001017974
Iteration: 799 Cost: 0.0020985448685084305
Iteration: 800 Cost: 0.0020914557649029993
Iteration: 801 Cost: 0.00208439060902948
Iteration: 802 Cost: 0.0020773493199898805
Iteration: 803 Cost: 0.0020703318171596784
Iteration: 804 Cost: 0.002063338020186452
Iteration: 805 Cost: 0.002056367848989481
Iteration: 806 Cost: 0.0020494212237582485
Iteration: 807 Cost: 0.0020424980649521233
Iteration: 808 Cost: 0.0020355982932991
Iteration: 809 Cost: 0.0020287218297948954
Iteration: 810 Cost: 0.0020218685957021565
Iteration: 811 Cost: 0.002015038512549459
Iteration: 812 Cost: 0.0020082315021305794
Iteration: 813 Cost: 0.002001447486503368
Iteration: 814 Cost: 0.0019946863879890383
Iteration: 815 Cost: 0.001987948129171215
Iteration: 816 Cost: 0.0019812326328950062
Iteration: 817 Cost: 0.0019745398222661336
Iteration: 818 Cost: 0.001967869620650192
Iteration: 819 Cost: 0.001961221951671484
Iteration: 820 Cost: 0.0019545967392124507
Iteration: 821 Cost: 0.001947993907412625
Iteration: 822 Cost: 0.0019414133806677802
Iteration: 823 Cost: 0.0019348550836292041
Iteration: 824 Cost: 0.0019283189412024788
Iteration: 825 Cost: 0.0019218048785471329
Iteration: 826 Cost: 0.0019153128210753708
Iteration: 827 Cost: 0.001908842694451333
Iteration: 828 Cost: 0.0019023944245903771
Iteration: 829 Cost: 0.0018959679376580386
Iteration: 830 Cost: 0.0018895631600693852
Iteration: 831 Cost: 0.001883180018487891
Iteration: 832 Cost: 0.00187681843982495
```

```
Iteration: 833 Cost: 0.0018704783512387285
Iteration: 834 Cost: 0.0018641596801334845
Iteration: 835 Cost: 0.00185786235415876
Iteration: 836 Cost: 0.0018515863012085174
Iteration: 837 Cost: 0.0018453314494201716
Iteration: 838 Cost: 0.001839097727174055
Iteration: 839 Cost: 0.0018328850630922953
Iteration: 840 Cost: 0.0018266933860382757
Iteration: 841 Cost: 0.0018205226251156478
Iteration: 842 Cost: 0.001814372709667438
Iteration: 843 Cost: 0.001808243569275609
Iteration: 844 Cost: 0.0018021351337596998
Iteration: 845 Cost: 0.0017960473331765884
Iteration: 846 Cost: 0.00178998009781925
Iteration: 847 Cost: 0.001783933358216212
Iteration: 848 Cost: 0.0017779070451307308
Iteration: 849 Cost: 0.0017719010895598318
Iteration: 850 Cost: 0.0017659154227337148
Iteration: 851 Cost: 0.0017599499761149647
Iteration: 852 Cost: 0.0017540046813974911
Iteration: 853 Cost: 0.0017480794705061376
Iteration: 854 Cost: 0.001742174275595659
Iteration: 855 Cost: 0.00173628902904993
Iteration: 856 Cost: 0.0017304236634812938
Iteration: 857 Cost: 0.0017245781117298166
Iteration: 858 Cost: 0.0017187523068622438
Iteration: 859 Cost: 0.0017129461821716
Iteration: 860 Cost: 0.0017071596711761984
Iteration: 861 Cost: 0.0017013927076189358
Iteration: 862 Cost: 0.001695645225466507
Iteration: 863 Cost: 0.0016899171589086714
Iteration: 864 Cost: 0.0016842084423575171
Iteration: 865 Cost: 0.0016785190104467653
Iteration: 866 Cost: 0.0016728487980308636
Iteration: 867 Cost: 0.0016671977401843253
Iteration: 868 Cost: 0.001661565772201046
Iteration: 869 Cost: 0.0016559528295934391
Iteration: 870 Cost: 0.0016503588480918125
Iteration: 871 Cost: 0.0016447837636436424
Iteration: 872 Cost: 0.0016392275124126072
Iteration: 873 Cost: 0.0016336900307781644
Iteration: 874 Cost: 0.0016281712553347007
Iteration: 875 Cost: 0.0016226711228907013
Iteration: 876 Cost: 0.0016171895704681766
Iteration: 877 Cost: 0.0016117265353019078
Iteration: 878 Cost: 0.001606281954838602
Iteration: 879 Cost: 0.0016008557667364087
Iteration: 880 Cost: 0.0015954479088639679
```

```
Iteration: 881 Cost: 0.001590058319299964
Iteration: 882 Cost: 0.0015846869363319801
Iteration: 883 Cost: 0.0015793336984563137
Iteration: 884 Cost: 0.0015739985443769884
Iteration: 885 Cost: 0.0015686814130049325
Iteration: 886 Cost: 0.0015633822434577074
Iteration: 887 Cost: 0.0015581009750583138
Iteration: 888 Cost: 0.0015528375473348025
Iteration: 889 Cost: 0.0015475919000195169
Iteration: 890 Cost: 0.0015423639730484037
Iteration: 891 Cost: 0.0015371537065602553
Iteration: 892 Cost: 0.0015319610408961394
Iteration: 893 Cost: 0.0015267859165986482
Iteration: 894 Cost: 0.0015216282744111716
Iteration: 895 Cost: 0.001516488055277367
Iteration: 896 Cost: 0.0015113652003402914
Iteration: 897 Cost: 0.0015062596509419243
Iteration: 898 Cost: 0.0015011713486223262
Iteration: 899 Cost: 0.0014961002351190232
Iteration: 900 Cost: 0.0014910462523664862
Iteration: 901 Cost: 0.0014860093424952025
Iteration: 902 Cost: 0.0014809894478311993
Iteration: 903 Cost: 0.0014759865108953402
Iteration: 904 Cost: 0.0014710004744026158
Iteration: 905 Cost: 0.0014660312812616047
Iteration: 906 Cost: 0.001461078874573714
Iteration: 907 Cost: 0.0014561431976324595
Iteration: 908 Cost: 0.0014512241939231719
Iteration: 909 Cost: 0.001446321807121793
Iteration: 910 Cost: 0.0014414359810947248
Iteration: 911 Cost: 0.0014365666598979406
Iteration: 912 Cost: 0.0014317137877764244
Iteration: 913 Cost: 0.0014268773091634785
Iteration: 914 Cost: 0.0014220571686800808
Iteration: 915 Cost: 0.0014172533111343377
Iteration: 916 Cost: 0.0014124656815208026
Iteration: 917 Cost: 0.0014076942250198351
Iteration: 918 Cost: 0.0014029388869969408
Iteration: 919 Cost: 0.001398199613002468
Iteration: 920 Cost: 0.0013934763487697972
Iteration: 921 Cost: 0.001388769040216892
Iteration: 922 Cost: 0.0013840776334436238
Iteration: 923 Cost: 0.0013794020747320928
Iteration: 924 Cost: 0.0013747423105459258
Iteration: 925 Cost: 0.0013700982875295272
Iteration: 926 Cost: 0.0013654699525076602
Iteration: 927 Cost: 0.0013608572524845518
Iteration: 928 Cost: 0.001356260134643563
```

```
Iteration: 929 Cost: 0.0013516785463464837
Iteration: 930 Cost: 0.0013471124351329318
Iteration: 931 Cost: 0.0013425617487195984
Iteration: 932 Cost: 0.0013380264349999993
Iteration: 933 Cost: 0.0013335064420434812
Iteration: 934 Cost: 0.0013290017180949952
Iteration: 935 Cost: 0.0013245122115741659
Iteration: 936 Cost: 0.0013200378710750069
Iteration: 937 Cost: 0.0013155786453650515
Iteration: 938 Cost: 0.0013111344833849543
Iteration: 939 Cost: 0.0013067053342479184
Iteration: 940 Cost: 0.0013022911472389564
Iteration: 941 Cost: 0.0012978918718144696
Iteration: 942 Cost: 0.0012935074576015315
Iteration: 943 Cost: 0.0012891378543974494
Iteration: 944 Cost: 0.0012847830121690822
Iteration: 945 Cost: 0.0012804428810522872
Iteration: 946 Cost: 0.0012761174113514434
Iteration: 947 Cost: 0.0012718065535386829
Iteration: 948 Cost: 0.0012675102582535974
Iteration: 949 Cost: 0.0012632284763024171
Iteration: 950 Cost: 0.001258961158657584
Iteration: 951 Cost: 0.0012547082564571615
Iteration: 952 Cost: 0.0012504697210042974
Iteration: 953 Cost: 0.0012462455037665766
Iteration: 954 Cost: 0.001242035556375618
Iteration: 955 Cost: 0.0012378398306263434
Iteration: 956 Cost: 0.001233658278476587
Iteration: 957 Cost: 0.001229490852046447
Iteration: 958 Cost: 0.001225337503617798
Iteration: 959 Cost: 0.0012211981856336306
Iteration: 960 Cost: 0.0012170728506976877
Iteration: 961 Cost: 0.0012129614515737012
Iteration: 962 Cost: 0.0012088639411851347
Iteration: 963 Cost: 0.0012047802726143323
Iteration: 964 Cost: 0.001200710399102179
Iteration: 965 Cost: 0.001196654274047568
Iteration: 966 Cost: 0.0011926118510067838
Iteration: 967 Cost: 0.0011885830836929434
Iteration: 968 Cost: 0.0011845679259755873
Iteration: 969 Cost: 0.0011805663318801072
Iteration: 970 Cost: 0.0011765782555871705
Iteration: 971 Cost: 0.0011726036514321861
Iteration: 972 Cost: 0.0011686424739049044
Iteration: 973 Cost: 0.0011646946776487665
Iteration: 974 Cost: 0.0011607602174604323
Iteration: 975 Cost: 0.0011568390482892736
Iteration: 976 Cost: 0.0011529311252368771
```

```
Iteration: 977 Cost: 0.0011490364035564634
Iteration: 978 Cost: 0.00114515483865236
Iteration: 979 Cost: 0.001141286386079666
Iteration: 980 Cost: 0.0011374310015435348
Iteration: 981 Cost: 0.0011335886408988204
Iteration: 982 Cost: 0.0011297592601493811
Iteration: 983 Cost: 0.0011259428154478206
Iteration: 984 Cost: 0.0011221392630947871
Iteration: 985 Cost: 0.0011183485595386588
Iteration: 986 Cost: 0.0011145706613747265
Iteration: 987 Cost: 0.0011108055253451704
Iteration: 988 Cost: 0.001107053108338123
Iteration: 989 Cost: 0.0011033133673873613
Iteration: 990 Cost: 0.0010995862596719086
Iteration: 991 Cost: 0.001095871742515324
Iteration: 992 Cost: 0.0010921697733853799
Iteration: 993 Cost: 0.0010884803098935919
Iteration: 994 Cost: 0.0010848033097945857
Iteration: 995 Cost: 0.0010811387309856595
Iteration: 996 Cost: 0.0010774865315064798
Iteration: 997 Cost: 0.0010738466695383326
Iteration: 998 Cost: 0.0010702191034038455
Iteration: 999 Cost: 0.0010666037915663834
Iteration: 1000 Cost: 0.0010630006926296643
```

[116]:

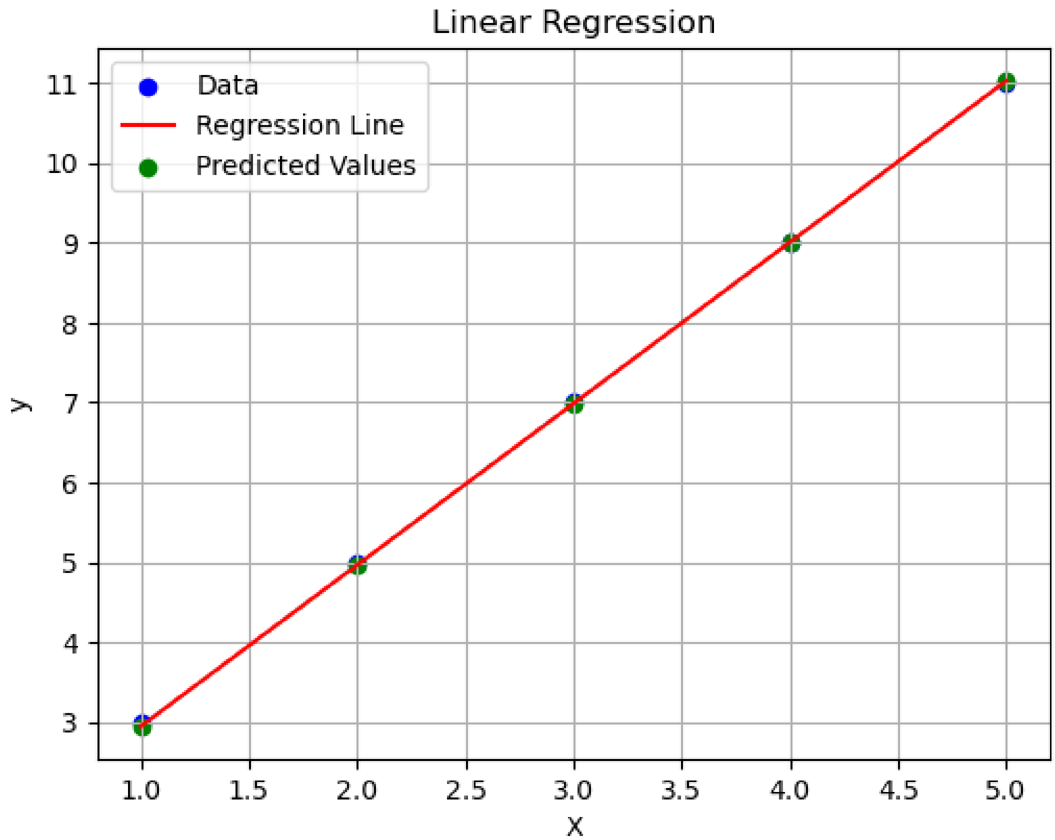
```python
# Step 3: Model Evaluation and Visualization
# Plot the data points
plt.scatter(X, y, color='blue', label='Data')

# Plot the regression line
y_pred = np.dot(X_with_intercept, theta_optimized)
plt.plot(X, y_pred, color='red', label='Regression Line')

# Plot the predicted values
plt.scatter(X, y_pred, color='green', label='Predicted Values')

# Add labels and title
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression')
plt.legend()

# Show plot
plt.grid(True)
plt.show()
```



[117]:
```
def root_mean_squared_error(y_true, y_pred):
    return np.sqrt(np.mean((y_true - y_pred) ** 2))

def mean_absolute_error(y_true, y_pred):
    return np.mean(np.abs(y_true - y_pred))

def r_squared(y_true, y_pred):
    tss = np.sum((y_true - np.mean(y_true)) ** 2)
    rss = np.sum((y_true - y_pred) ** 2)
    r_squared = 1 - (rss / tss)
    return r_squared

rmse = root_mean_squared_error(y, y_pred)
mae = mean_absolute_error(y, y_pred)
r2 = r_squared(y, y_pred)

print("Root Mean Squared Error (RMSE):", rmse)
print("Mean Absolute Error (MAE):", mae)
print("R-squared (R²) coefficient:", r2)
```

```
Root Mean Squared Error (RMSE): 0.032548575473241075
Mean Absolute Error (MAE): 0.027937166008357116
R-squared (R²) coefficient: 0.9998675737793329
```

0.0.3 Answer the following

What is learning rate? Ans: The learning rate is a hyperparameter that determines the step size at each iteration during the gradient descent optimization process. What will happen if the learning rate is too large? Ans: Overshooting: A large learning rate can cause the optimization algorithm to take excessively large steps during each iteration. As a result, the algorithm may overshoot the optimal solution and fail to converge. This overshooting phenomenon leads to oscillations or instability in the optimization process.

Divergence: In extreme cases, a very large learning rate can cause the optimization algorithm to diverge completely. Instead of converging to the optimal solution, the objective function may increase infinitely or fluctuate wildly, making it impossible to obtain meaningful results.

Unstable Training: Large learning rates can make the optimization process highly sensitive to small changes in the input data or model parameters. This sensitivity can lead to erratic behavior, making the training process unstable and unpredictable.

Poor Generalization: When the optimization process is unstable due to a large learning rate, the resulting model may generalize poorly to unseen data. The model may overfit to the training data, capturing noise and irrelevant patterns rather than the underlying structure of the data.

What will happen if the learning rate is too small? Ans: Slow Convergence: With a very small learning rate, the optimization algorithm takes tiny steps towards the optimal solution at each iteration. As a result, the convergence of the algorithm becomes very slow. It may require a large number of iterations to reach the optimal solution, making the training process computationally expensive and time-consuming.

Getting Getting Stuck in Local Minima: A small learning rate may cause the optimization algorithm to get stuck in local minima or saddle points, especially in complex and high-dimensional optimization problems. In such cases, the algorithm may struggle to escape from these suboptimal points and converge to the global minimum of the objective function.

Susceptibility to Noise: When the learning rate is too small, the optimization process becomes highly sensitive to noise in the training data or gradients. Small fluctuations or perturbations in the gradients may have a significant impact on the optimization trajectory, leading to erratic behavior and poor convergence.

Difficulty in Escaping Plateaus: In regions of the objective function landscape with very flat or plateau-like surfaces, a small learning rate may prevent the optimization algorithm from making meaningful progress. The algorithm may take tiny steps along the flat surface, making it challenging to escape these regions and reach more favorable leas of th e

landscape. If you what to change the second order (quadratic) model to third order model what all things will c ange in th.

Ans: Feature Transformation: Transform the input feature (Level) into third-order polynomial features. Model Training: Train the linear regression model using the transformed polynomial features. Prediction: Predict the target values using the trained model and the polynomial feature. Your answer

[99]:
```
import numpy as np

def polynomial_features(X, degree=3):
    X_poly = np.ones((len(X), 1))  # Initialize polynomial features with bias␣
 ↪term

    for d in range(1, degree + 1):
        X_poly = np.concatenate((X_poly, np.power(X, d)), axis=1)  # Add␣
 ↪polynomial features up to the specified degree

    return X_poly

X_poly = polynomial_features(X, degree=3)

def train_linear_regression(X, y):
    X_with_bias = np.concatenate((np.ones((len(X), 1)), X), axis=1)

    coefficients = np.linalg.inv(X_with_bias.T.dot(X_with_bias)).
 ↪dot(X_with_bias.T).dot(y)

    return coefficients

coefficients = train_linear_regression(X_poly, y)

def predict(X, coefficients):
    X_with_bias = np.concatenate((np.ones((len(X), 1)), X), axis=1)

    y_pred = X_with_bias.dot(coefficients)

    return y_pred
y_pred = predict(X_poly, coefficients)

def mean_squared_error(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

mse = mean_squared_error(y, y_pred)
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 5818.646578265527
```