211112238-lab1

February 2, 2024

0.0.1 Name : Siddhant Patil

0.0.2 Scholar Number : 211112238

Machine Learning Lab 1

Topics : Linear Regression, Polynomial Regression (Regularization and degree) and Gradient Descent

0.0.3 Q1>1. Linear Regression

[2]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
def LinearRegressionFunc(x_train, y_train, x_test, y_test):
    # Create a Linear Regression model
    model = LinearRegression()

    # Fit the model on the training data
    model.fit(x_train, y_train)

    # Make predictions on the test set
    y_pred = model.predict(x_test)

    # Calculate RMSE, MAE, and R-squared
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    return rmse, mae, r2
    # # Print the results
    # print(f'RMSE: {rmse:.4f}')
    # print(f'MAE: {mae:.4f}')
    # print(f'R-squared: {r2:.4f}')
```

[3]:
```
import glob
import pandas as pd
```

```
folders =␣
 ↪["diabetes-5-fold","machineCPU-5-fold","mortgage-5-fold","plastic-5-fold","stock-5-fold"]
for folder in folders:
    print(folder)
    # Define the file pattern
    file_pattern = "*tra.dat"  # Matches files with the pattern diabetes-5-*tra.
 ↪dat
    # Use glob to find files that match the pattern
    training_files = glob.glob("./"+folder+"/"+file_pattern)
    file_pattern = "*tst.dat"  # Matches files with the pattern␣
 ↪diabetes-5-*-tra.dat
    testing_files = glob.glob("./"+folder+"/"+file_pattern)



    trmse = 0
    tmae = 0
    tr2 = 0
    for train_file,test_file in zip(training_files, testing_files):
#         print(train_file)
        df = pd.read_csv(train_file,delimiter=',', header=None, comment='@')
        df_test = pd.read_csv(test_file, delimiter=',', header=None,␣
 ↪comment='@')
#         print(df)
        x_train = df.iloc[:,:-1]
        y_train = df.iloc[:,-1]
        x_test = df_test.iloc[:,:-1]
        y_test = df_test.iloc[:,-1]
        rmse, mae, r2 = LinearRegressionFunc(x_train,y_train,x_test,y_test)
        # print(r2)
        trmse+=rmse
        tmae+=mae
        tr2+=r2
    trmse/=5
    tmae/=5
    tr2/=5
    # gl_map[(1,)]
    print(f"RMSE : {trmse}\nMAE : {tmae}\nr2_score : {tr2}\n")
```

```
diabetes-5-fold
RMSE : 0.5726339848919423
MAE : 0.45959514037717064
r2_score : 0.18247065187852046

machineCPU-5-fold
RMSE : 61.17915669734837
MAE : 39.657831008292426
```

```
r2_score : 0.8409107965438972

mortgage-5-fold
RMSE : 0.11682526015990233
MAE : 0.08166348189953801
r2_score : 0.9985159926028266

plastic-5-fold
RMSE : 1.5294552267824453
MAE : 1.231564052891787
r2_score : 0.7985929281992478

stock-5-fold
RMSE : 2.324541515018755
MAE : 1.8222017952952094
r2_score : 0.8727356604473936
```

0.0.4 Q1>2. Polynomial Regression

[4]:
```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

def PolynomialRegressionFunc(x_train, y_train, x_test, y_test, degree):
    # Create polynomial features
    poly = PolynomialFeatures(degree=degree)
    x_train_poly = poly.fit_transform(x_train)
    x_test_poly = poly.transform(x_test)

    # Create a Linear Regression model
    model = LinearRegression()

    # Fit the model on the polynomial features
    model.fit(x_train_poly, y_train)

    # Make predictions on the test set
    y_pred = model.predict(x_test_poly)

    # Calculate metrics
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    return mae, rmse, r2
```

```
# Example usage:
# mae, rmse, r2 = PolynomialRegressionMetrics(x_train, y_train, x_test, y_test,␣
 ↪degree)
```

[5]:
```
import glob
import pandas as pd

folders =␣
 ↪["diabetes-5-fold","machineCPU-5-fold","mortgage-5-fold","plastic-5-fold","stock-5-fold"]
for folder in folders:
    print(folder)
    # Define the file pattern
    file_pattern = "*tra.dat"  # Matches files with the pattern diabetes-5-*tra.
 ↪dat
    # Use glob to find files that match the pattern
    training_files = glob.glob("./"+folder+"/"+file_pattern)
    file_pattern = "*tst.dat"  # Matches files with the pattern␣
 ↪diabetes-5-*-tra.dat
    testing_files = glob.glob("./"+folder+"/"+file_pattern)

    degree = [1,2,3]
    for d in degree:
        trmse = 0
        tmae = 0
        tr2 = 0
        for train_file,test_file in zip(training_files, testing_files):
            df = pd.read_csv(train_file,delimiter=',', header=None, comment='@')
            df_test = pd.read_csv(test_file, delimiter=',', header=None,␣
 ↪comment='@')
            # print(df)
            x_train = df.iloc[:,:-1]
            y_train = df.iloc[:,-1]
            x_test = df_test.iloc[:,:-1]
            y_test = df_test.iloc[:,-1]
            rae, mae, r2 =␣
 ↪PolynomialRegressionFunc(x_train,y_train,x_test,y_test,d)
            # print(r2)
            trmse+=rmse
            tmae+=mae
            tr2+=r2
        trmse/=5
        tmae/=5
        tr2/=5
        print(f"degree : {d}")
        # print(f"{folder}:\n")
        print(f"RMSE : {trmse}\nMAE : {tmae}\nr2_score : {tr2}\n")
    print("++++++++++++++++++++++++++++++++++++++++++++++\n")
```

```
diabetes-5-fold
degree : 1
RMSE : 2.2228481913487035
MAE : 0.5726339848919423
r2_score : 0.18247065187852032

degree : 2
RMSE : 2.2228481913487035
MAE : 0.482598485058156
r2_score : 0.4223777375427928

degree : 3
RMSE : 2.2228481913487035
MAE : 0.52785516575866
r2_score : 0.35074927782410514

++++++++++++++++++++++++++++++++++++++++++++++

machineCPU-5-fold
degree : 1
RMSE : 2.2228481913487035
MAE : 61.1791566973483
r2_score : 0.8409107965438973

degree : 2
RMSE : 2.2228481913487035
MAE : 75.34025414143346
r2_score : 0.7229638820162693

degree : 3
RMSE : 2.2228481913487035
MAE : 92.3564360572407
r2_score : 0.2927740289983438

++++++++++++++++++++++++++++++++++++++++++++++

mortgage-5-fold
degree : 1
RMSE : 2.2228481913487035
MAE : 0.11682526015990254
r2_score : 0.9985159926028266

degree : 2
RMSE : 2.2228481913487035
MAE : 0.05585357002398559
r2_score : 0.9996630874522321

degree : 3
```

```
RMSE : 2.2228481913487035
MAE : 0.1839569329552293
r2_score : 0.983503970969036

++++++++++++++++++++++++++++++++++++++++++++++++

plastic-5-fold
degree : 1
RMSE : 2.2228481913487035
MAE : 1.5294552267824453
r2_score : 0.7985929281992478

degree : 2
RMSE : 2.2228481913487035
MAE : 1.523866658319739
r2_score : 0.8000488831509648

degree : 3
RMSE : 2.2228481913487035
MAE : 1.466470528999766
r2_score : 0.8146760964535

++++++++++++++++++++++++++++++++++++++++++++++++

stock-5-fold
degree : 1
RMSE : 2.2228481913487035
MAE : 2.324541515018756
r2_score : 0.8727356604473935

degree : 2
RMSE : 2.2228481913487035
MAE : 1.0642904226522785
r2_score : 0.973318988505007

degree : 3
RMSE : 2.2228481913487035
MAE : 0.6567431107669582
r2_score : 0.9895467202190007

++++++++++++++++++++++++++++++++++++++++++++++++
```

0.0.5 Q1>3. Ridge Regularization

Alpha very high => Underfitting

medium => Perfect

Very low => Overfitting

[2]:
```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

def PolynomialRidgeRegressionMetrics(x_train, y_train, x_test, y_test, degree,␣
 ↪alpha):
    # Create polynomial features
    poly = PolynomialFeatures(degree=degree)
    x_train_poly = poly.fit_transform(x_train)
    x_test_poly = poly.transform(x_test)


    # Create a Ridge Regression model
    model = Ridge(alpha=alpha)

    # Fit the model on the polynomial features
    model.fit(x_train_poly, y_train)

    # Make predictions on the test set
    y_pred = model.predict(x_test_poly)

    # Calculate metrics
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)


    return rmse, mae, r2
```

[8]:
```
import glob
import pandas as pd
import math
import warnings
warnings.filterwarnings("ignore")

folders =␣
 ↪["diabetes-5-fold","machineCPU-5-fold","mortgage-5-fold","plastic-5-fold","stock-5-fold"]
for folder in folders:
    gmae = (10**10)
    grmse = 10**10
    gr2 = -1000
    galpha1 = -1 #global mae
    gd1 = -1
    galpha2 = -1 #global rmse
```

```
    gd2 = -1
    galpha3 = -1 #global r2
    gd3 = -1
#    gl_map = {}
#     print(folder)
    # Define the file pattern
    file_pattern = "*tra.dat"  # Matches files with the pattern diabetes-5-*tra.
 ↪dat
    # Use glob to find files that match the pattern
    training_files = glob.glob("./"+folder+"/"+file_pattern)
    file_pattern = "*tst.dat"  # Matches files with the pattern␣
 ↪diabetes-5-*-tra.dat
    testing_files = glob.glob("./"+folder+"/"+file_pattern)

    # ans_map = {}
    alpha_values = [2**i for i in range(-18,51,2)]
    degree = [1,2,3]
    for d in degree:
#         gl_map = {}
        # max_mae = -1, max_rmse
        for alpha in alpha_values:
            trmse = 0
            tmae = 0
            tr2 = 0
            for train_file,test_file in zip(training_files, testing_files):
                df = pd.read_csv(train_file,delimiter=',', header=None,␣
 ↪comment='@')
                df_test = pd.read_csv(test_file, delimiter=',', header=None,␣
 ↪comment='@')
                # print(df)
                x_train = df.iloc[:,:-1]
                y_train = df.iloc[:,-1]
                x_test = df_test.iloc[:,:-1]
                y_test = df_test.iloc[:,-1]
                rmse, mae, r2 =␣
 ↪PolynomialRidgeRegressionMetrics(x_train,y_train,x_test,y_test,d,alpha)
                # print(r2)
                trmse+=rmse
                tmae+=mae
                tr2+=r2
            trmse/=5
            tmae/=5
            tr2/=5
            if gmae>tmae :
                galpha1 = alpha
                gd1 = d
                gmae = tmae
```

```
            if(grmse>trmse):
                galpha2 = alpha
                gd2 = d
                grmse = trmse

            if(gr2<tr2):
                galpha3 = alpha
                gd3 = d
                gr2 = tr2

    print(f"{folder} :-")
    print(f"MAE : {grmse}, {math.log(galpha1,2)}, {gd1}")
    print(f"RMSE: {gmae}, {math.log(galpha2,2)}, {gd2}")
    print(f"R2  :{gr2}, {math.log(galpha3,2)}, {gd3}")
        # for k,v in gl_map:
        #     if()
            # key = (d, alpha)
            # ans_map[key] = (trmse, tmae, tr2)
            # print(f"{folder}:\n")
            # print(f"RMSE : {trmse}\nMAE : {tmae}\nr2_score : {tr2}\n")
        # print("++++++++++++++++++++++++++++++++++++++++++++++++\n")
```

```
diabetes-5-fold :-
MAE : 0.4805647936305403, 0.0, 3
RMSE: 0.39135367183979675, 0.0, 2
R2  :0.4241074180868695, 0.0, 2
machineCPU-5-fold :-
MAE : 35.70422438238954, 46.0, 3
RMSE: 21.638653313432137, 46.0, 3
R2  :0.9407311653588574, 46.0, 3
mortgage-5-fold :-
MAE : 0.037571464685945855, 0.0, 3
RMSE: 0.02091624923583358, 2.0, 3
R2  :0.999812463356664, 20.0, 3
plastic-5-fold :-
MAE : 1.4664705309159882, -2.0, 3
RMSE: 1.1587330338252702, -18.0, 3
R2  :0.8146760959794485, -18.0, 3
stock-5-fold :-
MAE : 0.638771457305997, -18.0, 3
RMSE: 0.49348504660666775, -18.0, 3
R2  :0.990012197101929, -18.0, 3
```

[68]:
```
# Assuming ans_map is your dictionary
for key, values in ans_map.items():
    degree, alpha = key
```

```
    rmse, mae, r2 = values
    print(f"Degree: {degree:.4f}, Alpha: {alpha:.8f}, RMSE: {rmse:.4f}, MAE:␣
↪{mae:.4f}, R2: {r2:.4f}")
```

```
Degree: 1.0000, Alpha: 0.00000381, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00000763, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00001526, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00003052, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00006104, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00012207, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00024414, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00048828, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00097656, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00195312, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00390625, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.00781250, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.01562500, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.03125000, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.06250000, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.12500000, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.25000000, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 0.50000000, RMSE: 2.3477, MAE: 1.8381, R2: 0.8701
Degree: 1.0000, Alpha: 1.00000000, RMSE: 2.3477, MAE: 1.8380, R2: 0.8701
Degree: 1.0000, Alpha: 2.00000000, RMSE: 2.3476, MAE: 1.8380, R2: 0.8701
Degree: 1.0000, Alpha: 4.00000000, RMSE: 2.3476, MAE: 1.8379, R2: 0.8701
Degree: 1.0000, Alpha: 8.00000000, RMSE: 2.3476, MAE: 1.8376, R2: 0.8701
Degree: 1.0000, Alpha: 16.00000000, RMSE: 2.3475, MAE: 1.8372, R2: 0.8702
Degree: 1.0000, Alpha: 32.00000000, RMSE: 2.3473, MAE: 1.8364, R2: 0.8702
Degree: 1.0000, Alpha: 64.00000000, RMSE: 2.3472, MAE: 1.8350, R2: 0.8702
Degree: 1.0000, Alpha: 128.00000000, RMSE: 2.3473, MAE: 1.8329, R2: 0.8702
Degree: 1.0000, Alpha: 256.00000000, RMSE: 2.3485, MAE: 1.8310, R2: 0.8700
Degree: 1.0000, Alpha: 512.00000000, RMSE: 2.3533, MAE: 1.8316, R2: 0.8695
Degree: 1.0000, Alpha: 1024.00000000, RMSE: 2.3671, MAE: 1.8429, R2: 0.8680
Degree: 1.0000, Alpha: 2048.00000000, RMSE: 2.4020, MAE: 1.8787, R2: 0.8640
Degree: 1.0000, Alpha: 4096.00000000, RMSE: 2.4868, MAE: 1.9741, R2: 0.8542
Degree: 1.0000, Alpha: 8192.00000000, RMSE: 2.6860, MAE: 2.1928, R2: 0.8298
Degree: 1.0000, Alpha: 16384.00000000, RMSE: 3.0868, MAE: 2.5962, R2: 0.7754
Degree: 1.0000, Alpha: 32768.00000000, RMSE: 3.7052, MAE: 3.1745, R2: 0.6768
Degree: 1.0000, Alpha: 65536.00000000, RMSE: 4.4184, MAE: 3.8176, R2: 0.5408
Degree: 1.0000, Alpha: 131072.00000000, RMSE: 5.0636, MAE: 4.3699, R2: 0.3973
Degree: 1.0000, Alpha: 262144.00000000, RMSE: 5.5646, MAE: 4.7638, R2: 0.2725
Degree: 1.0000, Alpha: 524288.00000000, RMSE: 5.9285, MAE: 5.0283, R2: 0.1744
Degree: 1.0000, Alpha: 1048576.00000000, RMSE: 6.1800, MAE: 5.2079, R2: 0.1030
Degree: 1.0000, Alpha: 2097152.00000000, RMSE: 6.3405, MAE: 5.3257, R2: 0.0559
Degree: 1.0000, Alpha: 4194304.00000000, RMSE: 6.4343, MAE: 5.3970, R2: 0.0278
Degree: 1.0000, Alpha: 8388608.00000000, RMSE: 6.4855, MAE: 5.4367, R2: 0.0122
Degree: 1.0000, Alpha: 16777216.00000000, RMSE: 6.5124, MAE: 5.4578, R2: 0.0040
```

```
Degree: 1.0000, Alpha: 33554432.00000000, RMSE: 6.5262, MAE: 5.4686, R2: -0.0002
Degree: 1.0000, Alpha: 67108864.00000000, RMSE: 6.5332, MAE: 5.4740, R2: -0.0024
Degree: 1.0000, Alpha: 134217728.00000000, RMSE: 6.5367, MAE: 5.4768, R2:
-0.0034
Degree: 1.0000, Alpha: 268435456.00000000, RMSE: 6.5385, MAE: 5.4781, R2:
-0.0040
Degree: 1.0000, Alpha: 536870912.00000000, RMSE: 6.5394, MAE: 5.4788, R2:
-0.0042
Degree: 1.0000, Alpha: 1073741824.00000000, RMSE: 6.5398, MAE: 5.4792, R2:
-0.0044
Degree: 1.0000, Alpha: 2147483648.00000000, RMSE: 6.5400, MAE: 5.4793, R2:
-0.0045
Degree: 1.0000, Alpha: 4294967296.00000000, RMSE: 6.5401, MAE: 5.4794, R2:
-0.0045
Degree: 1.0000, Alpha: 8589934592.00000000, RMSE: 6.5402, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 17179869184.00000000, RMSE: 6.5402, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 34359738368.00000000, RMSE: 6.5402, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 68719476736.00000000, RMSE: 6.5402, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 137438953472.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 274877906944.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 549755813888.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 1099511627776.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 2199023255552.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 4398046511104.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 8796093022208.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 17592186044416.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 35184372088832.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 70368744177664.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 140737488355328.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 281474976710656.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 1.0000, Alpha: 562949953421312.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
```

```
Degree: 1.0000, Alpha: 1125899906842624.00000000, RMSE: 6.5403, MAE: 5.4795, R2:
-0.0045
Degree: 2.0000, Alpha: 0.00000381, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00000763, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00001526, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00003052, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00006104, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00012207, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00024414, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00048828, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00097656, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00195312, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00390625, RMSE: 1.1249, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.00781250, RMSE: 1.1248, MAE: 0.9008, R2: 0.9702
Degree: 2.0000, Alpha: 0.01562500, RMSE: 1.1247, MAE: 0.9007, R2: 0.9702
Degree: 2.0000, Alpha: 0.03125000, RMSE: 1.1244, MAE: 0.9006, R2: 0.9702
Degree: 2.0000, Alpha: 0.06250000, RMSE: 1.1240, MAE: 0.9004, R2: 0.9703
Degree: 2.0000, Alpha: 0.12500000, RMSE: 1.1232, MAE: 0.9001, R2: 0.9703
Degree: 2.0000, Alpha: 0.25000000, RMSE: 1.1219, MAE: 0.8999, R2: 0.9704
Degree: 2.0000, Alpha: 0.50000000, RMSE: 1.1204, MAE: 0.8996, R2: 0.9704
Degree: 2.0000, Alpha: 1.00000000, RMSE: 1.1190, MAE: 0.8998, R2: 0.9705
Degree: 2.0000, Alpha: 2.00000000, RMSE: 1.1184, MAE: 0.9004, R2: 0.9705
Degree: 2.0000, Alpha: 4.00000000, RMSE: 1.1192, MAE: 0.9018, R2: 0.9705
Degree: 2.0000, Alpha: 8.00000000, RMSE: 1.1215, MAE: 0.9041, R2: 0.9704
Degree: 2.0000, Alpha: 16.00000000, RMSE: 1.1258, MAE: 0.9075, R2: 0.9701
Degree: 2.0000, Alpha: 32.00000000, RMSE: 1.1330, MAE: 0.9123, R2: 0.9697
Degree: 2.0000, Alpha: 64.00000000, RMSE: 1.1466, MAE: 0.9222, R2: 0.9690
Degree: 2.0000, Alpha: 128.00000000, RMSE: 1.1717, MAE: 0.9411, R2: 0.9676
Degree: 2.0000, Alpha: 256.00000000, RMSE: 1.2100, MAE: 0.9701, R2: 0.9655
Degree: 2.0000, Alpha: 512.00000000, RMSE: 1.2538, MAE: 1.0034, R2: 0.9630
Degree: 2.0000, Alpha: 1024.00000000, RMSE: 1.2932, MAE: 1.0329, R2: 0.9606
Degree: 2.0000, Alpha: 2048.00000000, RMSE: 1.3254, MAE: 1.0582, R2: 0.9586
Degree: 2.0000, Alpha: 4096.00000000, RMSE: 1.3546, MAE: 1.0814, R2: 0.9568
Degree: 2.0000, Alpha: 8192.00000000, RMSE: 1.3865, MAE: 1.1068, R2: 0.9547
Degree: 2.0000, Alpha: 16384.00000000, RMSE: 1.4246, MAE: 1.1372, R2: 0.9522
Degree: 2.0000, Alpha: 32768.00000000, RMSE: 1.4689, MAE: 1.1713, R2: 0.9492
Degree: 2.0000, Alpha: 65536.00000000, RMSE: 1.5184, MAE: 1.2094, R2: 0.9458
Degree: 2.0000, Alpha: 131072.00000000, RMSE: 1.5726, MAE: 1.2511, R2: 0.9418
Degree: 2.0000, Alpha: 262144.00000000, RMSE: 1.6306, MAE: 1.2943, R2: 0.9375
Degree: 2.0000, Alpha: 524288.00000000, RMSE: 1.6931, MAE: 1.3402, R2: 0.9326
Degree: 2.0000, Alpha: 1048576.00000000, RMSE: 1.7636, MAE: 1.3922, R2: 0.9268
Degree: 2.0000, Alpha: 2097152.00000000, RMSE: 1.8463, MAE: 1.4561, R2: 0.9198
Degree: 2.0000, Alpha: 4194304.00000000, RMSE: 1.9424, MAE: 1.5321, R2: 0.9111
Degree: 2.0000, Alpha: 8388608.00000000, RMSE: 2.0481, MAE: 1.6110, R2: 0.9011
Degree: 2.0000, Alpha: 16777216.00000000, RMSE: 2.1580, MAE: 1.6889, R2: 0.8902
Degree: 2.0000, Alpha: 33554432.00000000, RMSE: 2.2704, MAE: 1.7713, R2: 0.8784
Degree: 2.0000, Alpha: 67108864.00000000, RMSE: 2.3976, MAE: 1.8848, R2: 0.8644
Degree: 2.0000, Alpha: 134217728.00000000, RMSE: 2.5896, MAE: 2.0849, R2: 0.8418
```

```
Degree: 2.0000, Alpha: 268435456.00000000, RMSE: 2.9366, MAE: 2.4482, R2: 0.7967
Degree: 2.0000, Alpha: 536870912.00000000, RMSE: 3.4974, MAE: 3.0033, R2: 0.7120
Degree: 2.0000, Alpha: 1073741824.00000000, RMSE: 4.2027, MAE: 3.6449, R2:
0.5846
Degree: 2.0000, Alpha: 2147483648.00000000, RMSE: 4.8953, MAE: 4.2443, R2:
0.4368
Degree: 2.0000, Alpha: 4294967296.00000000, RMSE: 5.4592, MAE: 4.6922, R2:
0.2998
Degree: 2.0000, Alpha: 8589934592.00000000, RMSE: 5.8675, MAE: 4.9887, R2:
0.1913
Degree: 2.0000, Alpha: 17179869184.00000000, RMSE: 6.1432, MAE: 5.1837, R2:
0.1137
Degree: 2.0000, Alpha: 34359738368.00000000, RMSE: 6.3180, MAE: 5.3099, R2:
0.0626
Degree: 2.0000, Alpha: 68719476736.00000000, RMSE: 6.4212, MAE: 5.3869, R2:
0.0317
Degree: 2.0000, Alpha: 137438953472.00000000, RMSE: 6.4784, MAE: 5.4310, R2:
0.0144
Degree: 2.0000, Alpha: 274877906944.00000000, RMSE: 6.5087, MAE: 5.4547, R2:
0.0051
Degree: 2.0000, Alpha: 549755813888.00000000, RMSE: 6.5243, MAE: 5.4670, R2:
0.0004
Degree: 2.0000, Alpha: 1099511627776.00000000, RMSE: 6.5322, MAE: 5.4732, R2:
-0.0021
Degree: 2.0000, Alpha: 2199023255552.00000000, RMSE: 6.5362, MAE: 5.4764, R2:
-0.0033
Degree: 2.0000, Alpha: 4398046511104.00000000, RMSE: 6.5382, MAE: 5.4779, R2:
-0.0039
Degree: 2.0000, Alpha: 8796093022208.00000000, RMSE: 6.5392, MAE: 5.4787, R2:
-0.0042
Degree: 2.0000, Alpha: 17592186044416.00000000, RMSE: 6.5398, MAE: 5.4791, R2:
-0.0044
Degree: 2.0000, Alpha: 35184372088832.00000000, RMSE: 6.5400, MAE: 5.4793, R2:
-0.0044
Degree: 2.0000, Alpha: 70368744177664.00000000, RMSE: 6.5401, MAE: 5.4794, R2:
-0.0045
Degree: 2.0000, Alpha: 140737488355328.00000000, RMSE: 6.5402, MAE: 5.4795, R2:
-0.0045
Degree: 2.0000, Alpha: 281474976710656.00000000, RMSE: 6.5402, MAE: 5.4795, R2:
-0.0045
Degree: 2.0000, Alpha: 562949953421312.00000000, RMSE: 6.5402, MAE: 5.4795, R2:
-0.0045
Degree: 2.0000, Alpha: 1125899906842624.00000000, RMSE: 6.5402, MAE: 5.4795, R2:
-0.0045
```

```
The Kernel crashed while executing code in the the current cell or a previous␣
↪cell. Please review the code in the cell(s) to identify a possible cause of␣
↪the failure. Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a>␣
↪for more info. View Jupyter <a href='command:jupyter.viewOutput'>log</a> for␣
↪further details.
```

0.0.6 Q2 Gradient Decent

[19]:
```python
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
Y = np.array([3, 4, 2, 5, 7])

# Gradient Descent Function
def gradient_descent(X, Y, learning_rate, epochs, alpha=None):
    m, n = X.shape
    theta = np.zeros((n, 1))
    cost_history = []

    for _ in range(epochs):
        # Hypothesis
        h = X.dot(theta)

        # Error
        error = h - Y

        # Regularization term for ridge regression
        regularization_term = 0 if alpha is None else alpha * np.sum(theta[1:
↪]**2)

        # Update rule
        theta = theta - (learning_rate / m) * (X.T.dot(error) +␣
↪regularization_term * np.vstack([0, theta[1:]]))

        # Cost function (mean squared error)
        cost = np.sum(error**2) / (2 * m) + regularization_term
        cost_history.append(cost)

    return theta, cost_history

# Function to calculate metrics
def calculate_metrics(X, Y, theta):
    h = X.dot(theta)
    rmse = np.sqrt(mean_squared_error(Y, h))
```

```
    mae = mean_absolute_error(Y, h)
    r2 = r2_score(Y, h)
    return rmse, mae, r2

# Add a column of ones to X for the bias term
X_bias = np.c_[np.ones((X.shape[0], 1)), X]

# Set hyperparameters
learning_rate = 0.01
epochs = 1000
alpha_values = [2**i for i in range(-18,10)]

# Perform gradient descent and calculate metrics for different alpha values
for alpha in alpha_values:
    theta, _ = gradient_descent(X_bias, Y.reshape(-1, 1), learning_rate,␣
 ↪epochs, alpha)
    rmse, mae, r2 = calculate_metrics(X_bias, Y, theta)
    print(f"Alpha: {alpha}, RMSE: {rmse:.4f}, MAE: {mae:.4f}, R2: {r2:.4f}")
```

```
Alpha: 3.814697265625e-06, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 7.62939453125e-06, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 1.52587890625e-05, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 3.0517578125e-05, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 6.103515625e-05, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 0.0001220703125, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 0.000244140625, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 0.00048828125, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 0.0009765625, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 0.001953125, RMSE: 1.1612, MAE: 0.9509, R2: 0.5445
Alpha: 0.00390625, RMSE: 1.1611, MAE: 0.9508, R2: 0.5445
Alpha: 0.0078125, RMSE: 1.1611, MAE: 0.9507, R2: 0.5445
Alpha: 0.015625, RMSE: 1.1610, MAE: 0.9504, R2: 0.5446
Alpha: 0.03125, RMSE: 1.1609, MAE: 0.9498, R2: 0.5447
Alpha: 0.0625, RMSE: 1.1607, MAE: 0.9488, R2: 0.5449
Alpha: 0.125, RMSE: 1.1602, MAE: 0.9467, R2: 0.5453
Alpha: 0.25, RMSE: 1.1594, MAE: 0.9427, R2: 0.5459
Alpha: 0.5, RMSE: 1.1585, MAE: 0.9352, R2: 0.5466
Alpha: 1, RMSE: 1.1583, MAE: 0.9222, R2: 0.5468
Alpha: 2, RMSE: 1.1615, MAE: 0.9016, R2: 0.5443
Alpha: 4, RMSE: 1.1730, MAE: 0.8978, R2: 0.5351
Alpha: 8, RMSE: 1.1977, MAE: 0.8944, R2: 0.5154
Alpha: 16, RMSE: 1.2361, MAE: 0.8906, R2: 0.4838
Alpha: 32, RMSE: 1.2850, MAE: 0.9579, R2: 0.4422
Alpha: 64, RMSE: 1.3390, MAE: 1.0223, R2: 0.3943
Alpha: 128, RMSE: 1.3934, MAE: 1.0797, R2: 0.3441
Alpha: 256, RMSE: 1.4447, MAE: 1.1296, R2: 0.2948
Alpha: 512, RMSE: 1.4911, MAE: 1.1721, R2: 0.2489
```

[ ]: