211112262
VAIBHAV PATEL

ML LAB1

[1]:
```
import numpy as
np import pandas
as pd
import matplotlib.pyplot as plt
import math
```

[5]:
```
train_dirs=[]
test_dirs=[]
headers=[0, 0, 0, 0, 0]
for dir in os.listdir("./"):
    if(dir.find("5-fold")!=-1):
        train_dirs.append("./"+dir+"/train/")
        test_dirs.append("./"+dir+"/test/")
```

0.0.1 Linear Regression

[1]:
```
def LinearRegression(train_file, test_file, header):
    train_df = pd.read_csv(train_file, header=header, delimiter=",")
    test_df = pd.read_csv(test_file, header=header, delimiter=",")

    X_train = train_df.iloc[:, :-1].values
    y_train = train_df.iloc[:, -1].values

    X_test = test_df.iloc[:, :-1].values
    y_test = test_df.iloc[:, -1].values

    from sklearn.linear_model import LinearRegression
    regressor = LinearRegression()

    regressor.fit(X_train, y_train)

    y_pred =

    regressor.predict(X_test)

    from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
 r2_score
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2score = r2_score(y_test, y_pred)
    return np.array([mse, mae, r2score])
```

```
[7]: for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
         train_files=os.listdir(train_dir)
         test_files=os.listdir(test_dir)
         val=np.zeros(3)
         for train, test in zip(train_files, test_files):
             val+=LinearRegression(train_dir+train, test_dir+test, header)
         print(train_dir)
         val/=len(train_files)
         val[0]=math.sqrt(val[0])
         val=pd.DataFrame(val, index=["RMSE", "MSE", "R2"], columns=["Values"])
         print(val)
         print("                                \n")
                --------------------------------
```

```
./concrete-5-fold/train/
         Values
RMSE  10.441372
MSE    8.272734
R2     0.603040
--------------------------------

./diabetes-5-fold/train/
        Values
RMSE
      0.63949
8
MSE   0.501970
R2   -0.000552
--------------------------------

./mortgage-5-fold/train/
        Values
RMSE
      0.12182
4
MSE   0.083536
R2    0.998404
--------------------------------

./plastic-5-fold/train/
        Values
RMSE  1.531465
MSE   1.232442
R2    0.798437
--------------------------------

./stock-5-fold/train/
        Values
```

```
RMSE  2.346750
MSE   1.832574
R2    0.870620
--------------------------------
```

0.0.2 Polynomial Regression of degree 2 and 3

[8]:

```python
def PolynomialRegression(train_file, test_file, header, degree):
    train_df = pd.read_csv(train_file, header=header, delimiter=",")
    test_df = pd.read_csv(test_file, header=header, delimiter=",")

    X_train = train_df.iloc[:, :-1].values
    y_train = train_df.iloc[:, -1].values

    X_test = test_df.iloc[:, :-1].values
    y_test = test_df.iloc[:, -1].values

    from sklearn.linear_model import LinearRegression
    from sklearn.preprocessing import PolynomialFeatures

    poly_reg=PolynomialFeatures(degree=degree)
    X_poly=poly_reg.fit_transform(X_train)

    regressor = LinearRegression()
    regressor.fit(X_poly,  y_train)

    y_pred  =  regressor.predict(poly_reg.transform(X_test))

    from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
 r2_score

    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2score = r2_score(y_test, y_pred)
    return np.array([mse, mae, r2score])
```

[9]:

Polynomial Regression of degree 2

```python
for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
    train_files  =  os.listdir(train_dir)
    test_files  =  os.listdir(test_dir)

    val = np.zeros(3)
    for train, test in zip(train_files, test_files):
        val += PolynomialRegression(train_dir + train, test_dir + test, header,␣
 2)
```

```
    print(train_dir)
    val /= len(train_files)
    val[0] =
    math.sqrt(val[0])
    val = pd.DataFrame(val, index=["RMSE", "MSE", "R2"], columns=["Values"])
    print(val)
    print("--------------------------------\n")
```

```
./concrete-5-fold/train/
          Values
RMSE
       7.66514
3
MSE    5.892674
R2     0.786115
--------------------------------

./diabetes-5-fold/train/
          Values
RMSE
       0.56129
7
MSE    0.456880
R2     0.226230
--------------------------------

./mortgage-5-fold/train/
          Values
RMSE
       0.12061
6
MSE    0.055462
R2     0.998544
--------------------------------

./plastic-5-fold/train/
          Values
RMSE   1.528545
MSE    1.226209
R2     0.799254
--------------------------------

./stock-5-fold/train/
          Values
RMSE   1.127098
MSE    0.901557
R2     0.970154
--------------------------------
```

Polynomial Regression of degree 3

[10]:
```
for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
    train_files = os.listdir(train_dir)
    test_files = os.listdir(test_dir)
    val = np.zeros(3)
    for train, test in zip(train_files, test_files):
        val += PolynomialRegression(train_dir + train, test_dir + test, header,␣
 3)
    print(train_dir)
    val /= len(train_files)
    val[0] =
    math.sqrt(val[0])
    val = pd.DataFrame(val, index=["RMSE", "MSE", "R2"], columns=["Values"])
    print(val)
    print("--------------------------------\n")
```

```
./concrete-5-fold/train/
          Values
RMSE
        6.72537
9
MSE   4.674355
R2    0.834883
--------------------------------

./diabetes-5-fold/train/
          Values
RMSE
        0.83851
1
MSE   0.620331
R2   -0.519498
--------------------------------

./mortgage-5-fold/train/
          Values
RMSE
        2.59757
3
MSE   0.512271
R2    0.338858
--------------------------------

./plastic-5-fold/train/
          Values
RMSE  1.473863
MSE   1.166224
R2    0.813267
--------------------------------

./stock-5-fold/train/
          Values
RMSE  0.880364
MSE   0.661540
R2    0.981784
```

0.0.3 Regularization in Linear Regression

```
[24]: def Regularization(train_file, test_file, header):
          train_df = pd.read_csv(train_file, header=header, delimiter=",")
          test_df = pd.read_csv(test_file, header=header, delimiter=",")

          X_train = train_df.iloc[:, :-1].values
          y_train = train_df.iloc[:, -1].values

          X_test = test_df.iloc[:, :-1].values
          y_test = test_df.iloc[:, -1].values

          from sklearn.linear_model import Ridge
          alphas = np.array([2**i for i in range(-18, 51, 2)])

          best_mse, best_alpha=float('inf'), None

          from sklearn.metrics import mean_squared_error
          for alpha in alphas: regressor=Ridge(alpha=alpha)
              regressor.fit(X_train, y_train)
              y_pred=regressor.predict(X_test)
              mse=mean_squared_error(y_test, y_pred)

              if mse<best_mse:
                  best_mse, best_alpha=mse, alpha
        return np.array([best_mse, best_alpha])
```

```
[25]: for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
          train_files = os.listdir(train_dir)
          test_files = os.listdir(test_dir)

          val = np.zeros(2)
          for train, test in zip(train_files, test_files):
              val = Regularization(train_dir + train, test_dir + test, header)
          print(train_dir)
          val[0]=math.sqrt(val[0])
          val = pd.DataFrame(val, index=["Best RMSE", "Best Alpha"],␣
       columns=["Values"])
          print(val)
          print("--------------------------------\n")
```

```
./concrete-5-fold/train/
                  Values
 Best RMSE     10.222187
Best Alpha   1024.000000
  --------------------------------
```

```
./diabetes-5-fold/train/
                    Values
Best  RMSE
                 0.63437
1
Best  Alpha   0.000004
--------------------------------

./mortgage-5-fold/train/
                    Values
Best  RMSE
                 0.11068
4
Best  Alpha   1.000000
--------------------------------

./plastic-5-fold/train/
                    Values
Best  RMSE
                 1.51057
9
Best  Alpha   0.000004
--------------------------------

./stock-5-fold/train/
                    Values
Best  RMSE
                 2.31511
7
Best  Alpha   0.000004
--------------------------------
```

0.0.4 Ridge on Linear Reg for Best MAE

[26]:
```python
def Regularization(train_file, test_file, header):
    train_df = pd.read_csv(train_file, header=header, delimiter=",")
    test_df = pd.read_csv(test_file, header=header, delimiter=",")

    X_train = train_df.iloc[:, :-1].values
    y_train = train_df.iloc[:, -1].values

    X_test = test_df.iloc[:, :-1].values
    y_test = test_df.iloc[:, -1].values

    from sklearn.linear_model import Ridge
    alphas = np.array([2**i for i in range(-18, 51, 2)])

    best_mse, best_alpha=float('inf'), None

    from sklearn.metrics import mean_absolute_error
    for alpha in alphas:
        regressor=Ridge(alpha=alpha)
        regressor.fit(X_train, y_train)
        y_pred=regressor.predict(X_test)
        mse=mean_absolute_error(y_test, y_pred)

        if  mse<best_mse:
            best_mse, best_alpha=mse, alpha
    return  np.array([best_mse,  best_alpha])
```

```
[27]: for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
          train_files  =  os.listdir(train_dir)
          test_files  =  os.listdir(test_dir)

          val  =  np.zeros(2)
          for train, test in zip(train_files, test_files):
              val  =  Regularization(train_dir  +  train,  test_dir  +  test,  header)
          print(train_dir)
          val  =  pd.DataFrame(val,  index=["Best  MAE",  "Best  Alpha"],
      columns=["Values"])
          print(val)
          print("--------------------------------\n")
```

```
./concrete-5-fold/train/
                   Values
 Best MAE        8.084741
Best Alpha    4096.000000
--------------------------------

./diabetes-5-fold/train/
                 Values
Best MAE    0.47644
Best Alpha  4.00000
--------------------------------

./mortgage-5-fold/train/
                 Values
Best MAE
              0.07921
9
Best Alpha   1.000000
--------------------------------

./plastic-5-fold/train/
                 Values
 Best MAE     1.234517
Best Alpha   64.000000
--------------------------------

./stock-5-fold/train/
                  Values
 Best MAE      1.816376
Best Alpha   256.000000
```

```
--------------------------------
```

0.0.5 Ridge for Linear Reg, Best R2 score

```
[28]: def Regularization(train_file, test_file, header):
          train_df = pd.read_csv(train_file, header=header, delimiter=",")
          test_df = pd.read_csv(test_file, header=header, delimiter=",")

          X_train = train_df.iloc[:, :-1].values
          y_train = train_df.iloc[:, -1].values

          X_test = test_df.iloc[:, :-1].values
          y_test = test_df.iloc[:, -1].values

          from sklearn.linear_model import Ridge
          alphas = np.array([2**i for i in range(-18, 51, 2)])

          best_mse, best_alpha=-float('inf'), None

          from sklearn.metrics import r2_score
          for alpha in alphas:
              regressor=Ridge(alpha=alpha)
              regressor.fit(X_train, y_train)
              y_pred=regressor.predict(X_test)
              mse=r2_score(y_test, y_pred)

              if mse>best_mse:
                  best_mse, best_alpha=mse, alpha
```

```
[29]: for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
          train_files = os.listdir(train_dir)
          test_files = os.listdir(test_dir)

          val = np.zeros(2)
          for train, test in zip(train_files, test_files):
              val = Regularization(train_dir + train, test_dir + test, header)
          print(train_dir)
          val = pd.DataFrame(val, index=["Best R2 Score", "Best Alpha"],␣
      ↪columns=["Values"])
          print(val)
          print("--------------------------------\n")
```

```
./concrete-5-fold/train/
                      Values
Best R2 Score   0.674538
Best                   Alpha
1024.000000
```

```
--------------------------------

./diabetes-5-fold/train/
                     Values
Best        R2        Score
0.319099   Best    Alpha
0.000004
--------------------------------

./mortgage-5-fold/train/
                     Values
Best        R2        Score
0.998849   Best    Alpha
1.000000
--------------------------------

./plastic-5-fold/train/
                     Values
Best        R2        Score
0.790441   Best    Alpha
0.000004
--------------------------------

./stock-5-fold/train/
                     Values
Best        R2        Score
0.866915   Best    Alpha
0.000004
--------------------------------
```

[17]:

```
import warnings
warnings.filterwarnings('ignore')
```

0.0.6 Ridge for All using MSE

[30]:

```
def PolynomialRidge(train_file, test_file, header, degree):
    train_df = pd.read_csv(train_file, header=header, delimiter=",")
    test_df = pd.read_csv(test_file, header=header, delimiter=",")

    X_train = train_df.iloc[:, :-1].values
    y_train = train_df.iloc[:, -1].values

    X_test = test_df.iloc[:, :-1].values
    y_test = test_df.iloc[:, -1].values

    from sklearn.linear_model import Ridge
    from sklearn.preprocessing import PolynomialFeatures

    poly_reg=PolynomialFeatures(degree=degree)
    X_poly=poly_reg.fit_transform(X_train)
    alphas=np.array([2**i for i in range(-18, 30)])

    best_alpha, best_mse=None, float('inf')
    for alpha in alphas:
        regressor = Ridge(alpha=alpha)
        regressor.fit(X_poly, y_train)
        y_pred = regressor.predict(poly_reg.transform(X_test))

        from sklearn.metrics import mean_squared_error,
        mean_absolute_error,␣
r2_score
        mse = mean_squared_error(y_test, y_pred)
        if(mse<best_mse):
            best_mse, best_alpha=mse, alpha

    val = np.array([best_mse, best_alpha])
```

```
[31]: for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
    train_files = os.listdir(train_dir)
    test_files = os.listdir(test_dir)

    val = np.zeros(2)
    for train, test in zip(train_files, test_files):
        val += PolynomialRidge(train_dir + train, test_dir + test, header, 2)
    print(train_dir)
    val /= len(train_files)
    val[0] =
    math.sqrt(val[0])
    val = pd.DataFrame(val, index=["RMSE", "Alpha"], columns=["Values"])
    print(val)
    print("------------------------------\n")
```

```
./concrete-5-fold/train/
                Values
RMSE
7.646834e+00
Alpha   3.355445e+06
------------------------------

./diabetes-5-fold/train/
                Values
RMSE    5.275634e-
01              Alpha
1.677722e+06
------------------------------

./mortgage-5-fold/train/
              Values
RMSE      0.075369
Alpha   153.900001
```

```
--------------------------------

./plastic-5-fold/train/
                   Values
RMSE          1.527185
Alpha   26265.600002
--------------------------------

./stock-5-fold/train/
             Values
RMSE
          1.11370
6
Alpha   5.200002
--------------------------------
```

```
[21]: for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
          train_files  =  os.listdir(train_dir)
          test_files  =  os.listdir(test_dir)

          val = np.zeros(2)
          for train, test in zip(train_files, test_files):
              val += PolynomialRidge(train_dir + train, test_dir + test, header, 3)
          print(train_dir)
          val /= len(train_files)
          val[0] =
          math.sqrt(val[0])
          val = pd.DataFrame(val, index=["RMSE", "Alpha"], columns=["Values"])
          print(val)
          print("--------------------------------\n")
```

```
./concrete-5-fold/train/
                   Values
RMSE
6.089751e+00
Alpha   1.074266e+08
--------------------------------

./diabetes-5-fold/train/
                   Values
RMSE     6.234885e-
01                Alpha
1.090521e+08
--------------------------------

./mortgage-5-fold/train/
                   Values
RMSE     6.824711e-
02                Alpha
1.250165e+08
--------------------------------

./plastic-5-fold/train/
```

```
            Values
RMSE     1.472243
Alpha   25.625002
--------------------------------

./stock-5-fold/train/
                Values
RMSE          0.836642
Alpha   16896.000001
--------------------------------
```

0.0.7 Ridge for all using MAE

```
[22]: def PolynomialRidge(train_file, test_file, header, degree):
          train_df = pd.read_csv(train_file, header=header, delimiter=",")
          test_df = pd.read_csv(test_file, header=header, delimiter=",")

          X_train = train_df.iloc[:, :-1].values
          y_train = train_df.iloc[:, -1].values

          X_test = test_df.iloc[:, :-1].values
          y_test = test_df.iloc[:, -1].values

          from sklearn.linear_model import Ridge
          from sklearn.preprocessing import PolynomialFeatures

          poly_reg=PolynomialFeatures(degree=degree)
          X_poly=poly_reg.fit_transform(X_train)

          alphas=np.array([2**i for i in range(-18, 30)])

          best_alpha, best_mse=None, float('inf')
          for alpha in alphas:
              regressor = Ridge(alpha=alpha)
              regressor.fit(X_poly, y_train)
              y_pred = regressor.predict(poly_reg.transform(X_test))

              from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
       r2_score
              mse = mean_absolute_error(y_test, y_pred)
              if(mse<best_mse):
                  best_mse, best_alpha=mse, alpha

          return np.array([best_mse, best_alpha])
```

[23]:
```
for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
    train_files  =  os.listdir(train_dir)
    test_files  =  os.listdir(test_dir)

    val = np.zeros(2)
    for train, test in zip(train_files, test_files):
        val  += PolynomialRidge(train_dir + train, test_dir + test, header, 3)
    print(train_dir)
    val /= len(train_files)
    val = pd.DataFrame(val, index=["MAE", "Alpha"], columns=["Values"])
    print(val)
    print("--------------------------------\n")
```

```
./concrete-5-fold/train/
                    Values
MAE
           4.500617e+0
0 Alpha
           1.074791e+0
8
--------------------------------

./diabetes-5-fold/train/
                    Values
MAE      4.660749e-01
Alpha
           1.078460e+0
8
--------------------------------

./mortgage-5-fold/train/
                    Values
MAE      4.360656e-02
Alpha
           1.183318e+0
8
--------------------------------

./plastic-5-fold/train/
              Values
MAE      1.164341
Alpha   0.362501
--------------------------------

./stock-5-fold/train/
                  Values
MAE         0.647212
Alpha   7782.400196
--------------------------------
```

[32]:
```
for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
    train_files  =  os.listdir(train_dir)
    test_files  =  os.listdir(test_dir)
    val = np.zeros(2)
    for train, test in zip(train_files, test_files):
        val += PolynomialRidge(train_dir + train, test_dir + test, header, 2)
    print(train_dir)
    val /= len(train_files)
    val = pd.DataFrame(val, index=["MAE", "Alpha"], columns=["Values"])
    print(val)
    print("________________________________\n")
```

```
./concrete-5-fold/train/
                    Values
MAE
          5.847408e+0
1 Alpha
          3.355445e+0
6
--------------------------------

./diabetes-5-fold/train/
                    Values
MAE       2.783232e-01
Alpha
          1.677722e+0
6
--------------------------------

./mortgage-5-fold/train/
                 Values
MAE          0.005681
Alpha    153.900001
--------------------------------

./plastic-5-fold/train/
                    Values
MAE            2.332295
Alpha    26265.600002
--------------------------------

./stock-5-fold/train/
              Values
MAE       1.240341
Alpha     5.200002
--------------------------------
```

0.0.8 Ridge for all using R2 score

[33]:

```
def PolynomialRidge(train_file, test_file, header, degree):
    train_df = pd.read_csv(train_file, header=header, delimiter=",")
    test_df = pd.read_csv(test_file, header=header, delimiter=",")
    X_train = train_df.iloc[:, :-1].values
    y_train = train_df.iloc[:, -1].values

    X_test = test_df.iloc[:, :-1].values
    y_test = test_df.iloc[:, -1].values

    from sklearn.linear_model import Ridge
    from sklearn.preprocessing import PolynomialFeatures

    poly_reg=PolynomialFeatures(degree=degree)
    X_poly=poly_reg.fit_transform(X_train)

    alphas=np.array([2**i for i in range(-18, 30)])

    best_alpha, best_mse=None, -float('inf')
    for alpha in alphas:
        regressor = Ridge(alpha=alpha)
        regressor.fit(X_poly, y_train)
        y_pred = regressor.predict(poly_reg.transform(X_test))

        from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
↪r2_score
        mse = r2_score(y_test, y_pred)
        if(mse>best_mse):
            best_mse, best_alpha=mse, alpha
```

```
[34]: for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
          train_files = os.listdir(train_dir)
          test_files = os.listdir(test_dir)

          val = np.zeros(2)
          for train, test in zip(train_files, test_files):
              val += PolynomialRidge(train_dir + train, test_dir + test, header, 2)
          print(train_dir)
          val /= len(train_files)
          val = pd.DataFrame(val, index=["R2", "Alpha"], columns=["Values"])
          print(val)
          print("--------------------------------\n")
```

```
./concrete-5-fold/train/
             Values
R2     7.871333e-01
Alpha
       3.355445e+0
6
--------------------------------
```

```
./diabetes-5-fold/train/
                Values
R2        3.330702e-01
Alpha
          1.677722e+0
6
--------------------------------

./mortgage-5-fold/train/
              Values
R2          0.999395
Alpha   153.900001
--------------------------------

./plastic-5-fold/train/
                Values
R2              0.799613
Alpha   26265.600002
--------------------------------

./stock-5-fold/train/
             Values
R2        0.970870
Alpha   5.200002
--------------------------------
```

```
[35]: for train_dir, test_dir, header in zip(train_dirs, test_dirs, headers):
          train_files = os.listdir(train_dir)
          test_files = os.listdir(test_dir)

          val = np.zeros(2)
          for train, test in zip(train_files, test_files):
              val += PolynomialRidge(train_dir + train, test_dir + test, header, 3)
          print(train_dir)
          val /= len(train_files)
          val = pd.DataFrame(val, index=["R2 score", "Alpha"], columns=["Values"])
          print(val)
          print("--------------------------------\n")
```

```
./concrete-5-fold/train/
                   Values
R2  score  8.650418e-
01                 Alpha
1.074266e+08
--------------------------------

./diabetes-5-fold/train/
                   Values
R2  score  1.826710e-
01                 Alpha
1.090521e+08
```

```
--------------------------------

./mortgage-5-fold/train/
                    Values
R2   score   9.995106e-
01                  Alpha
1.250165e+08
--------------------------------

./plastic-5-fold/train/
                Values
R2 score
             0.81367
1
Alpha       25.625002
--------------------------------

./stock-5-fold/train/
                    Values
R2 score          0.983591
Alpha    16896.000001
--------------------------------
```