Machine Learning Lab 1

Akhilendra Pratap 211112438

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import os
```

```
/tmp/ipykernel_5139/4209716058.py:2: DeprecationWarning: 
Pyarrow will become a required dependency of pandas in the next major
release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type,
and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at
https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

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

def cal_header_val(file_path):
    with open(file_path, "r") as file:
        lines=file.readlines()
    return lines.index('@data\n')+1

headers=[]
for dir in train_dirs:
    file_path=dir+os.listdir (dir)[0]
    headers.append(cal_header_val(file_path))

headers=np.array(headers)
```

Linear Regression

```python
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])


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")
```

```
/usr/lib/python3/dist-packages/scipy/__init__.py:146: UserWarning: A
NumPy version >=1.17.3 and <1.25.0 is required for this version of
SciPy (detected version 1.26.3
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"

./ele-1-5-fold/train/
          Values
RMSE  633.808233
MSE   419.216579
R2      0.697299
------------------------------

./diabetes-5-fold/train/
        Values
RMSE  0.589029
MSE   0.465676
R2    0.144824
```

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

./quake-5-fold/train/
        Values
RMSE  0.188958
MSE   0.148494
R2    0.003954
-----------------------------

./laser-5-fold/train/
         Values
RMSE  23.022278
MSE   15.511976
R2     0.752009
-----------------------------

./plastic-5-fold/train/
        Values
RMSE  1.529883
MSE   1.231589
R2    0.798854
-----------------------------
```

Polynomial Regression of degree 2 and 3

```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])
```

Polynomial Regression of degree 2

```
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")
```

```
./ele-1-5-fold/train/
          Values
RMSE  593.068697
MSE   404.025764
R2      0.732678
-----------------------------

./diabetes-5-fold/train/
        Values
RMSE  0.507230
MSE   0.414338
R2    0.355809
-----------------------------

./quake-5-fold/train/
        Values
RMSE  0.188900
MSE   0.148034
R2    0.004362
-----------------------------

./laser-5-fold/train/
         Values
RMSE  10.382359
MSE    6.491844
R2     0.949910
```

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

./plastic-5-fold/train/
        Values
RMSE  1.524322
MSE   1.222752
R2    0.800348
-----------------------------
```

Polynomial Regression of degree 3

```
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")
```

```
./ele-1-5-fold/train/
          Values
RMSE  584.772891
MSE   397.998001
R2      0.739070
-----------------------------

./diabetes-5-fold/train/
        Values
RMSE  0.479272
MSE   0.364019
R2    0.463297
-----------------------------

./quake-5-fold/train/
        Values
RMSE  0.188188
MSE   0.148102
R2    0.011537
-----------------------------
```

```
./laser-5-fold/train/
        Values
RMSE  5.376591
MSE   2.819652
R2    0.986938
-----------------------------

./plastic-5-fold/train/
        Values
RMSE  1.468539
MSE   1.161601
R2    0.814599
-----------------------------
```

Regularization in Linear Regression

```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_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])

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")
```

```
./ele-1-5-fold/train/
                Values
Best RMSE   575.377949
Best Alpha    0.000004
-----------------------------

./diabetes-5-fold/train/
              Values
Best RMSE   0.634371
Best Alpha  0.000004
-----------------------------

./quake-5-fold/train/
              Values
Best RMSE   0.178702
Best Alpha  0.000004
-----------------------------

./laser-5-fold/train/
                   Values
Best RMSE       24.206672
Best Alpha  262144.000000
-----------------------------

./plastic-5-fold/train/
              Values
Best RMSE   1.510579
Best Alpha  0.000004
-----------------------------
```

Ridge on Linear Reg for Best MAE

```
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])

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")
```

```
./ele-1-5-fold/train/
                  Values
Best MAE      397.714181
Best Alpha  65536.000000
-----------------------------

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

./quake-5-fold/train/
                  Values
Best MAE        0.139492
Best Alpha  65536.000000
-----------------------------
```

```
./laser-5-fold/train/
                   Values
Best MAE        15.894284
Best Alpha  262144.000000
-----------------------------

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

Ridge for Linear Reg, Best R2 score

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

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")
```

```
./ele-1-5-fold/train/
                 Values
Best R2 Score  0.738707
Best Alpha     0.000004
-----------------------------

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

./quake-5-fold/train/
                 Values
Best R2 Score  0.005949
Best Alpha     0.000004
-----------------------------

./laser-5-fold/train/
                      Values
Best R2 Score       0.662978
Best Alpha     262144.000000
-----------------------------

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

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

Ridge for All using MSE

```
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
```

```python
    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

    return np.array([best_mse, best_alpha])

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")
```

```
./ele-1-5-fold/train/
             Values
RMSE     592.073943
Alpha  56524.800002
-----------------------------

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

```
RMSE         0.505328
Alpha  419533.200002
-----------------------------

./quake-5-fold/train/
             Values
RMSE   1.887830e-01
Alpha  1.385431e+07
-----------------------------

./laser-5-fold/train/
           Values
RMSE    10.373138
Alpha  217.600002
-----------------------------

./plastic-5-fold/train/
             Values
RMSE       1.523200
Alpha  26252.800002
-----------------------------
```

```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(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")
```

```
./ele-1-5-fold/train/
             Values
RMSE   5.845551e+02
Alpha  1.074135e+08
-----------------------------

./diabetes-5-fold/train/
           Values
RMSE     0.455948
Alpha  218.403126
```

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

./quake-5-fold/train/
             Values
RMSE   1.879962e-01
Alpha  1.109393e+08
-----------------------------

./laser-5-fold/train/
          Values
RMSE    5.236308
Alpha  52.800002
-----------------------------

./plastic-5-fold/train/
          Values
RMSE    1.468035
Alpha  25.625002
-----------------------------
```

Ridge for all using MAE

```python
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])

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")
```

```
./ele-1-5-fold/train/
              Values
MAE       397.745854
Alpha  420352.000002
-----------------------------

./diabetes-5-fold/train/
             Values
MAE        0.353695
Alpha  52441.600002
-----------------------------

./quake-5-fold/train/
             Values
MAE    1.480013e-01
Alpha  3.460301e+06
-----------------------------

./laser-5-fold/train/
          Values
MAE     2.816814
Alpha  12.800003
-----------------------------

./plastic-5-fold/train/
         Values
MAE    1.160367
Alpha  0.262501
```

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


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")
```

```
./ele-1-5-fold/train/
             Values
MAE      401.655189
Alpha  59801.600002
-----------------------------

./diabetes-5-fold/train/
              Values
MAE         0.401606
Alpha  419841.600002
-----------------------------

./quake-5-fold/train/
             Values
MAE    1.479610e-01
Alpha  6.710886e+06
-----------------------------

./laser-5-fold/train/
            Values
MAE       6.439135
Alpha  1356.800000
-----------------------------

./plastic-5-fold/train/
             Values
MAE        1.219512
Alpha  26368.000001
-----------------------------
```

Ridge for all using R2 score

```
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

    return np.array([best_mse, best_alpha])

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")
```

```
./ele-1-5-fold/train/
             Values
```

```
R2         0.733629
Alpha  56524.800002
-----------------------------

./diabetes-5-fold/train/
              Values
R2          0.361546
Alpha  419533.200002
-----------------------------

./quake-5-fold/train/
             Values
R2     5.585534e-03
Alpha  1.385431e+07
-----------------------------

./laser-5-fold/train/
           Values
R2       0.949986
Alpha  217.600002
-----------------------------

./plastic-5-fold/train/
             Values
R2         0.800643
Alpha  26252.800002
-----------------------------
```

```
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")
```

```
./ele-1-5-fold/train/
                Values
R2 score  7.392552e-01
Alpha     1.074135e+08
-----------------------------
```

```
./diabetes-5-fold/train/
              Values
R2 score    0.502298
Alpha     218.403126
-----------------------------

./quake-5-fold/train/
                Values
R2 score  1.351850e-02
Alpha     1.109393e+08
-----------------------------

./laser-5-fold/train/
             Values
R2 score   0.987530
Alpha     52.800002
-----------------------------

./plastic-5-fold/train/
             Values
R2 score   0.814724
Alpha     25.625002
-----------------------------
```