Machine Learning Lab 1

Akhilendra Pratap 211112438

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

train_dirs=[]
test_dirs=[]
headers=[7, 7, 9, 7, 8]
for dir in os.listdir("./"):
    if(dir.find("5-fold")!=-1):
        train_dirs.append("./"+dir+"/train/")
        test_dirs.append("./"+dir+"/test/")
```

## Linear Regression

```
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")
```

```
./diabetes-5-fold/train/
        Values
RMSE  0.639498
MSE   0.501970
R2   -0.000552
-----------------------------

./ele-1-5-fold/train/
          Values
RMSE  649.533859
MSE   421.387017
R2      0.682405
-----------------------------

./laser-5-fold/train/
         Values
RMSE  23.300207
MSE   15.579874
R2     0.746418
-----------------------------

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

./quake-5-fold/train/
        Values
RMSE  0.189132
MSE   0.148620
R2    0.002162
-----------------------------
```

## Polynomial Regression of degree 2 and 3

```
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")
```

```
./diabetes-5-fold/train/
        Values
RMSE  0.561297
MSE   0.456880
R2    0.226230
```

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

./ele-1-5-fold/train/
          Values
RMSE  625.020558
MSE   416.170802
R2      0.704994
-----------------------------

./laser-5-fold/train/
         Values
RMSE  10.954232
MSE    6.686850
R2     0.944316
-----------------------------

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

./quake-5-fold/train/
        Values
RMSE  0.189590
MSE   0.148552
R2   -0.002729
-----------------------------
```

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")
```

```
./diabetes-5-fold/train/
        Values
RMSE  0.838511
MSE   0.620331
R2   -0.519498
-----------------------------

./ele-1-5-fold/train/
          Values
RMSE  737.354993
MSE   427.489641
R2      0.588328
-----------------------------

./laser-5-fold/train/
        Values
RMSE  7.379026
MSE   3.270549
R2    0.975378
-----------------------------

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

./quake-5-fold/train/
        Values
RMSE  0.189523
MSE   0.149088
R2   -0.002150
-----------------------------
```

## 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")
```

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

./ele-1-5-fold/train/
                Values
Best RMSE   577.159707
Best Alpha    0.000004
-----------------------------

./laser-5-fold/train/
               Values
Best RMSE   23.499185
Best Alpha   0.000004
-----------------------------

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

```
Best RMSE   1.510579
Best Alpha  0.000004
-----------------------------

./quake-5-fold/train/
                   Values
Best RMSE        0.192593
Best Alpha  262144.000000
-----------------------------
```