# PRML06

April 21, 2022

## 1 Observations -

1. We have written code for scratch coded Logistic Regression model and then compared the results with built-in library results.
2. Accuracy of scratch-coded model is lesser because dataset is skewed towards one class.
3. Given sample.csv dataset has 200 data-points, but only 49 data-points belong to class '1', rest all 151 data-points belongs to class '0'.
4. We have taken 5 unique learning_rates and calculated accuracy for each over all folds.
5. If we increase the value of learning rate, time to reach to minima will decrease but till a certain point. After that, function will start skipping the minima and model may not converge in the given iterations.

## 2 Code

[ ]:
```
!pwd
```

```
/Users/ambuj/Downloads/Document/College Semester/2nd Sem/Pattern Recognition and
ML/Assignments/202116003_PRML_assignment 6
```

[ ]:
```
DATASET = '/Users/ambuj/Downloads/Document/College Semester/2nd Sem/Pattern␣
↪Recognition and ML/Assignments/202116003_PRML_assignment 6/sample.csv'
```

### 2.1 Part-1 : Logistic Regression Model using Gradient Descent

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

[ ]:
```
##### function to calculate accuracies using self-coded logistic regression␣
↪model
```

```
def Logistic_Regression(X, y, lr, iterations, test_X, test_y):
    w = np.array([0]*X.shape[1])
    b = 0
    for it in range(iterations):
        model = np.dot(X, w) + b

        y_est = 1/(1 + np.exp(-model))

        dw = (1 / X.shape[0]) * np.dot(X.T, (y_est - y))
        db = (1 / X.shape[0]) * np.sum(y_est - y)

        w = w - lr * dw
        b = b - lr * db

    model = np.dot(test_X, w) + b
    test_y_est = 1/(1 + np.exp(-model))
    test_est_class = [1 if i > 0.5 else 0 for i in test_y_est]
    accuracy = np.sum(test_y == test_est_class) / len(test_y)
    return accuracy
```

### 2.2 Part-2 : Finding accuracy with 5-fold cross-validation

[ ]:
```
df = pd.read_csv(DATASET)
df.head()
```

[ ]:

| | female | read | write | math | hon | femalexmath |
|---|---|---|---|---|---|---|
| 0 | 0 | 57 | 52 | 41 | 0 | 0 |
| 1 | 1 | 68 | 59 | 53 | 0 | 53 |
| 2 | 0 | 44 | 33 | 54 | 0 | 0 |
| 3 | 0 | 63 | 44 | 47 | 0 | 0 |
| 4 | 0 | 47 | 52 | 57 | 0 | 0 |

[ ]:
```
print(df.shape)
print("unique labels = " + str( df['hon'].unique() ))
```

```
(200, 6)
unique labels = [0 1]
```

[ ]:
```
y = df['hon']
df.drop(['hon'], axis = 1, inplace = True)
X = df
```

[ ]:
```
X.head()
```

[ ]:

| | female | read | write | math | femalexmath |
|---|---|---|---|---|---|
| 0 | 0 | 57 | 52 | 41 | 0 |

```
1       1    68     59     53           53
2       0    44     33     54            0
3       0    63     44     47            0
4       0    47     52     57            0
```

[ ]:
```
y.head()
```

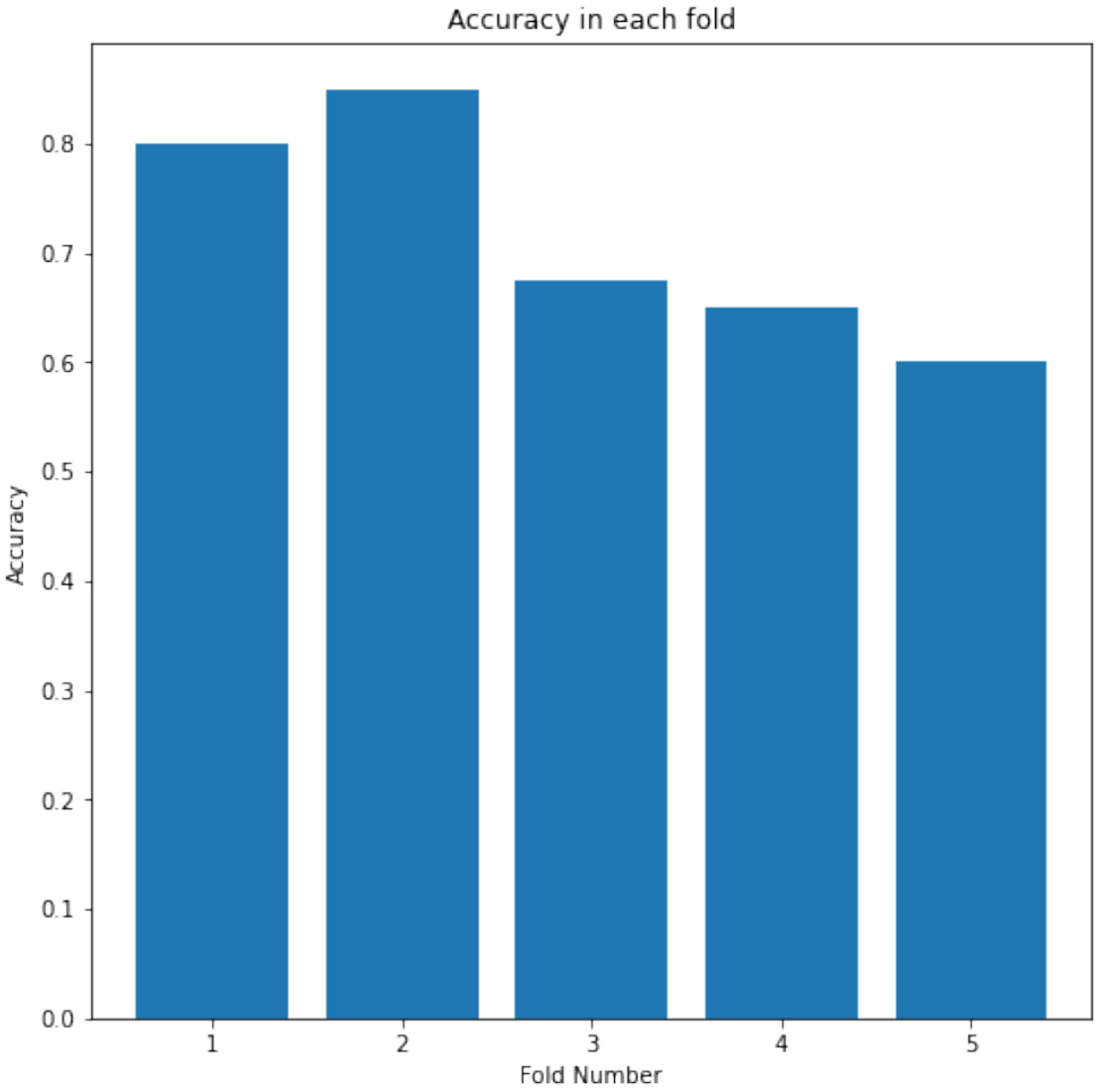
[ ]:
```
0    0
1    0
2    0
3    0
4    0
Name: hon, dtype: int64
```

[ ]:
```
#### Applying 5-fold cross validation and calculating accuracies for each fold

Fold_num = 1
learning_rate = 0.001
iterations = 10000
ACCURACY = []

for i in range(0, 200, 40):
    test_X = X[i:i+40]
    test_y = y[i:i+40]

    train_X = pd.concat([X[0:i],X[i+40:200]])
    train_y = pd.concat([y[0:i],y[i+40:200]])

    result_acc = Logistic_Regression(train_X, train_y, learning_rate,␣
 ↪iterations, test_X, test_y)
    print('Accuracy in Fold number {} is : {}'.format(Fold_num, result_acc))

    ACCURACY.append(result_acc)

    Fold_num = Fold_num+1
```

```
Accuracy in Fold number 1 is : 0.8
Accuracy in Fold number 2 is : 0.85
Accuracy in Fold number 3 is : 0.675
Accuracy in Fold number 4 is : 0.65
Accuracy in Fold number 5 is : 0.6
```

[ ]:
```
#### Plotting accuracy chart for each fold

plt.figure(figsize = (8,8))
plt.bar(list(range(1,6,1)), ACCURACY)
plt.title('Accuracy in each fold')
```

```
plt.xlabel('Fold Number')
plt.ylabel('Accuracy')
plt.show()
```



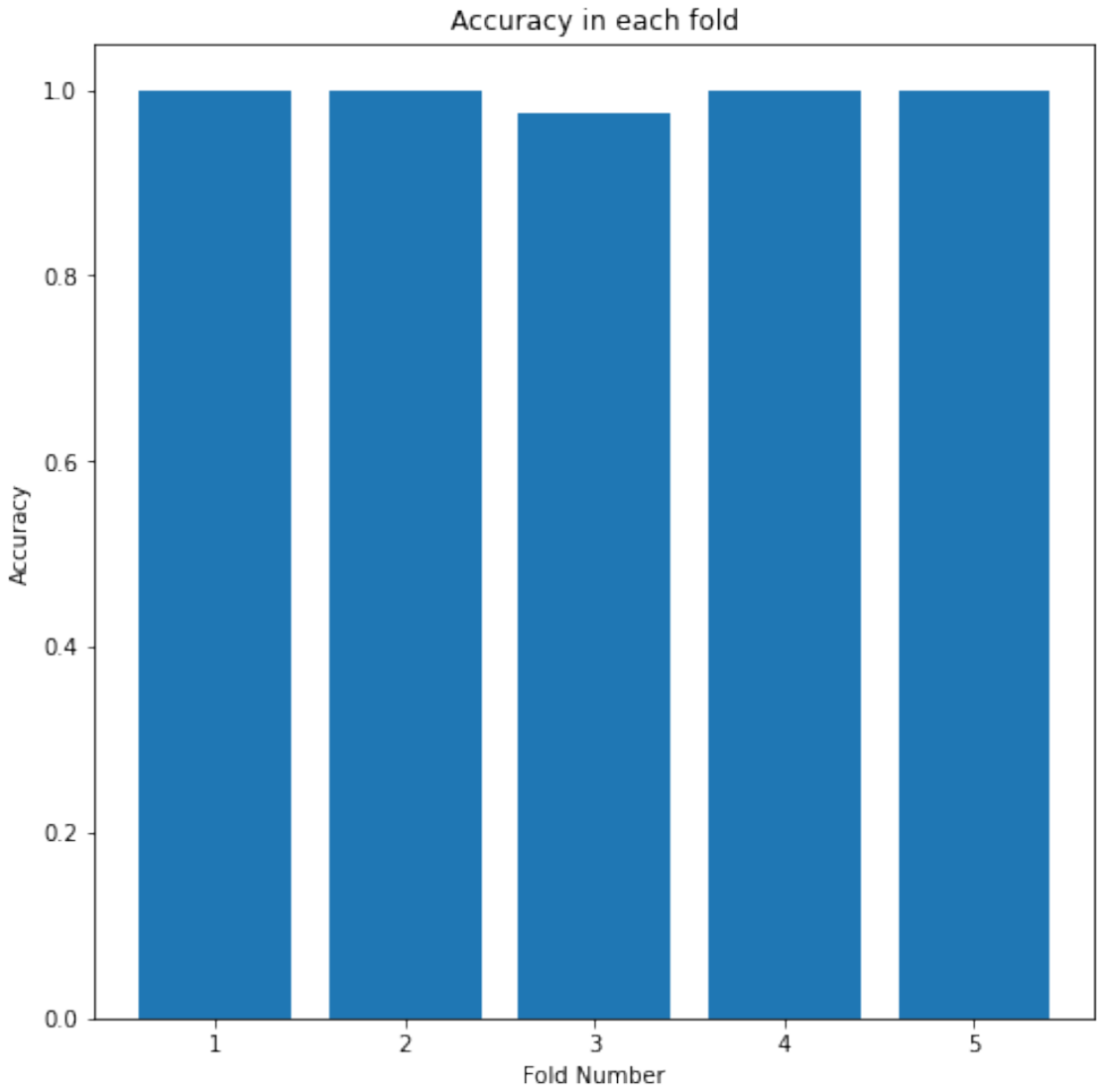

### 2.3 Part-3 : Calculating accuracies using build-in functions

[ ]:
```
#### creating built-in Logistic Regression model and calculating accuracies for␣
 ↪each fold

from sklearn.linear_model import LogisticRegression
```

```
Fold_num = 1
learning_rate = 0.001
iterations = 1000
ACCURACY_Q3 = []

for i in range(0, 200, 40):
    test_X = X[i:i+40]
    test_y = y[i:i+40]

    train_X = pd.concat([X[0:i],X[i+40:200]])
    train_y = pd.concat([y[0:i],y[i+40:200]])

    model = LogisticRegression(random_state=0).fit(train_X, train_y)

    result_acc = model.score(test_X, test_y)
    print('Accuracy in Fold number {} is : {}'.format(Fold_num, result_acc))

    ACCURACY_Q3.append(result_acc)

    Fold_num = Fold_num+1
```

```
Accuracy in Fold number 1 is : 1.0
Accuracy in Fold number 2 is : 1.0
Accuracy in Fold number 3 is : 0.975
Accuracy in Fold number 4 is : 1.0
Accuracy in Fold number 5 is : 1.0
```

[ ]:
```
#### Plotting accuracy chart for each fold

plt.figure(figsize = (8,8))
plt.bar(list(range(1,6,1)), ACCURACY_Q3)
plt.title('Accuracy in each fold')
plt.xlabel('Fold Number')
plt.ylabel('Accuracy')
plt.show()
```



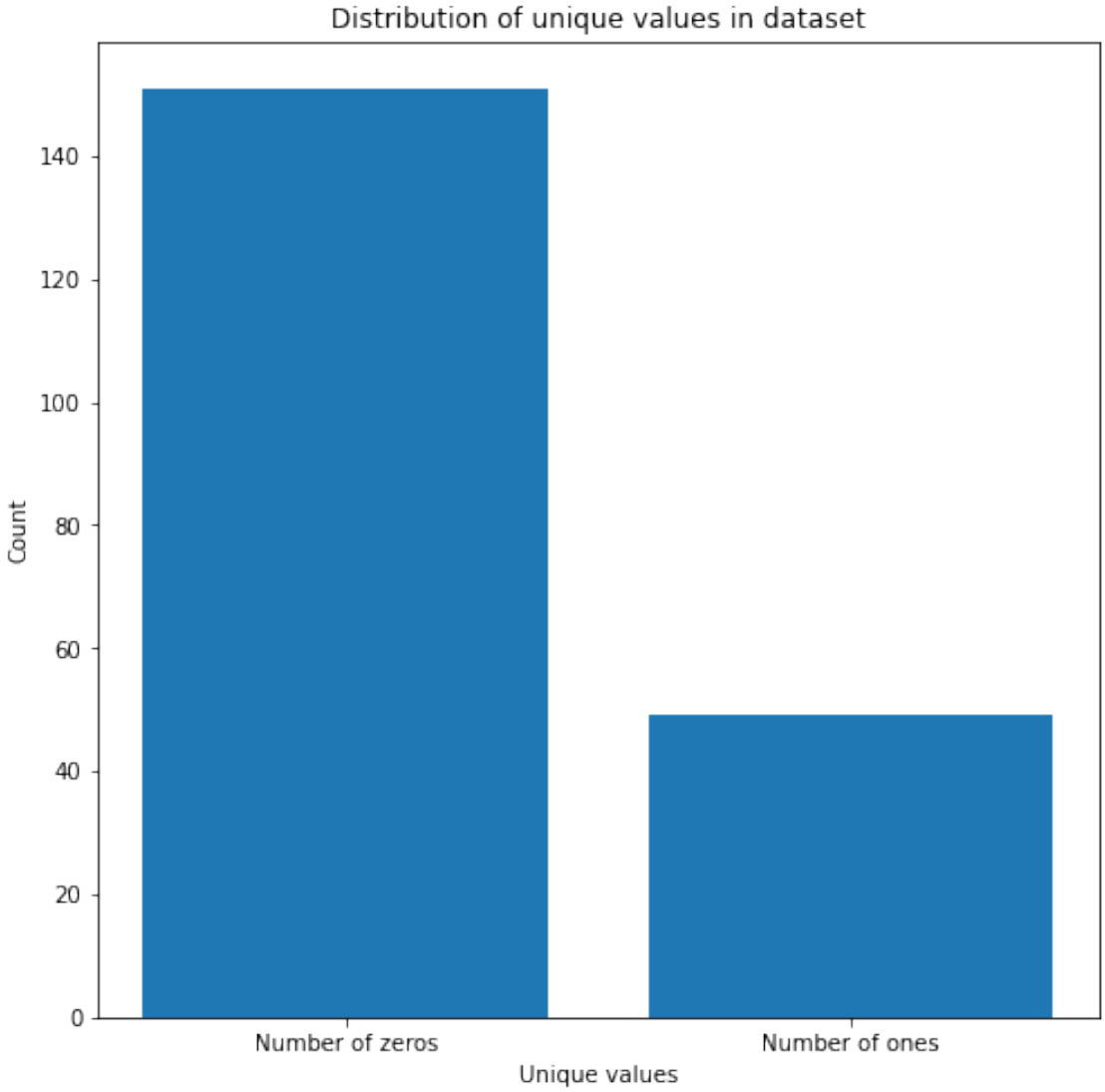

## 3 Observation

Accuracy in our case is low because dataset given, is skewed towards one class. Given sample.csv dataset has 200 data-points, but only 49 data-points belong to class '1', rest all 151 data-points belongs to class '0'.

[ ]:
```
number_of_zeros = len(y[y == 0])
number_of_ones = len(y[y == 1])

print('Number of zeros in dataset : ' + str(number_of_zeros))
print('Number of ones in dataset : ' + str(number_of_ones))
```

```
plt.figure(figsize = (8,8))
plt.bar(['Number of zeros', 'Number of ones'], [number_of_zeros,␣
 ↪number_of_ones])
plt.title('Distribution of unique values in dataset')
plt.xlabel('Unique values')
plt.ylabel('Count')
plt.show()
```

```
Number of zeros in dataset : 151
Number of ones in dataset : 49
```



### 3.1 Part-4 : Varying Step length

[ ]:
```
#### Taking 5 different learning rates and calculating accuracy score for each␣
 ↪on all folds

learning_rates = [0.0001, 0.001, 0.01, 0.1, 0.5]
ACCURACY_lr = []
for lr in learning_rates:
    Fold_num = 1
    iterations = 10000
    ACCURACY_temp = []

    for i in range(0, 200, 40):
        test_X = X[i:i+40]
        test_y = y[i:i+40]

        train_X = pd.concat([X[0:i],X[i+40:200]])
        train_y = pd.concat([y[0:i],y[i+40:200]])

        result_acc = Logistic_Regression(train_X, train_y, lr, iterations,␣
 ↪test_X, test_y)
        print('Accuracy in Fold number {} is : {}'.format(Fold_num, result_acc))

        ACCURACY_temp.append(result_acc)

        Fold_num = Fold_num+1

    ACCURACY_lr.append(ACCURACY_temp)

ACCURACY_lr
```

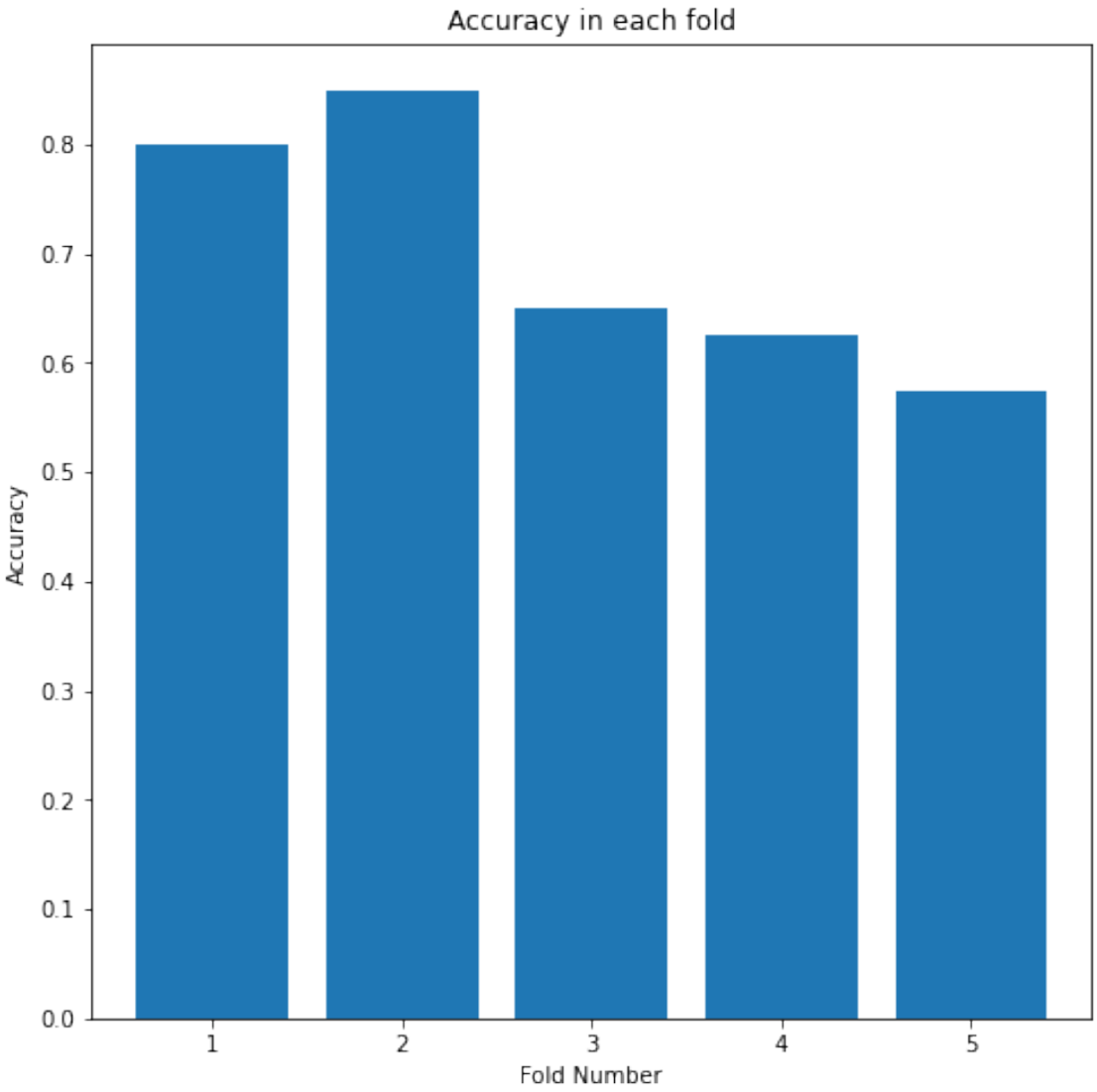
```
Accuracy in Fold number 1 is : 0.8
Accuracy in Fold number 2 is : 0.85
Accuracy in Fold number 3 is : 0.65
Accuracy in Fold number 4 is : 0.625
Accuracy in Fold number 5 is : 0.575
Accuracy in Fold number 1 is : 0.8
Accuracy in Fold number 2 is : 0.85
Accuracy in Fold number 3 is : 0.675
Accuracy in Fold number 4 is : 0.65
Accuracy in Fold number 5 is : 0.6
Accuracy in Fold number 1 is : 0.8
Accuracy in Fold number 2 is : 0.85
Accuracy in Fold number 3 is : 0.675
Accuracy in Fold number 4 is : 0.825
Accuracy in Fold number 5 is : 0.625
```

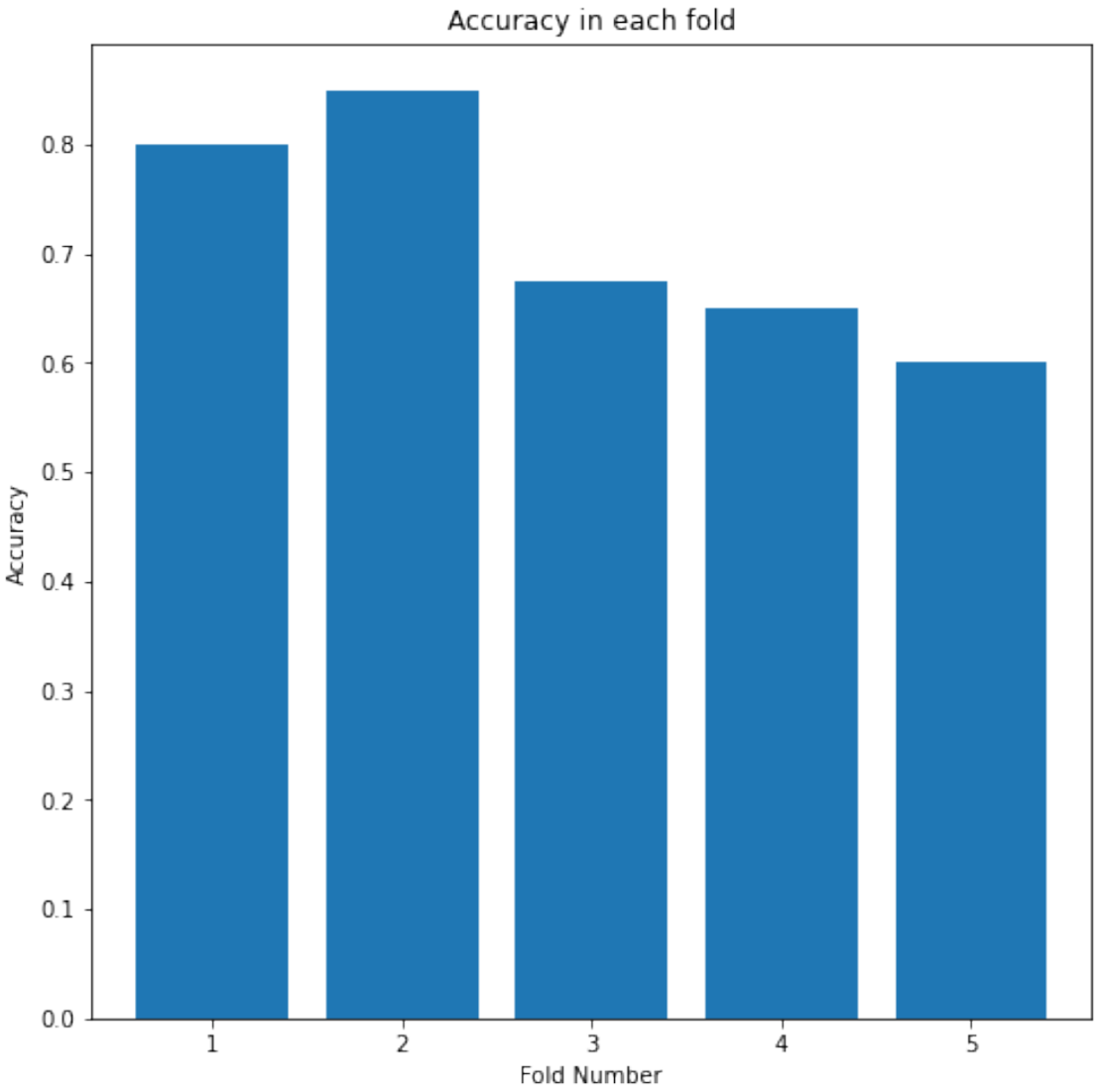
```
/var/folders/kl/55tfkryd5vs0fjm2fnv6vd140000gn/T/ipykernel_36339/3330011543.py:7
```

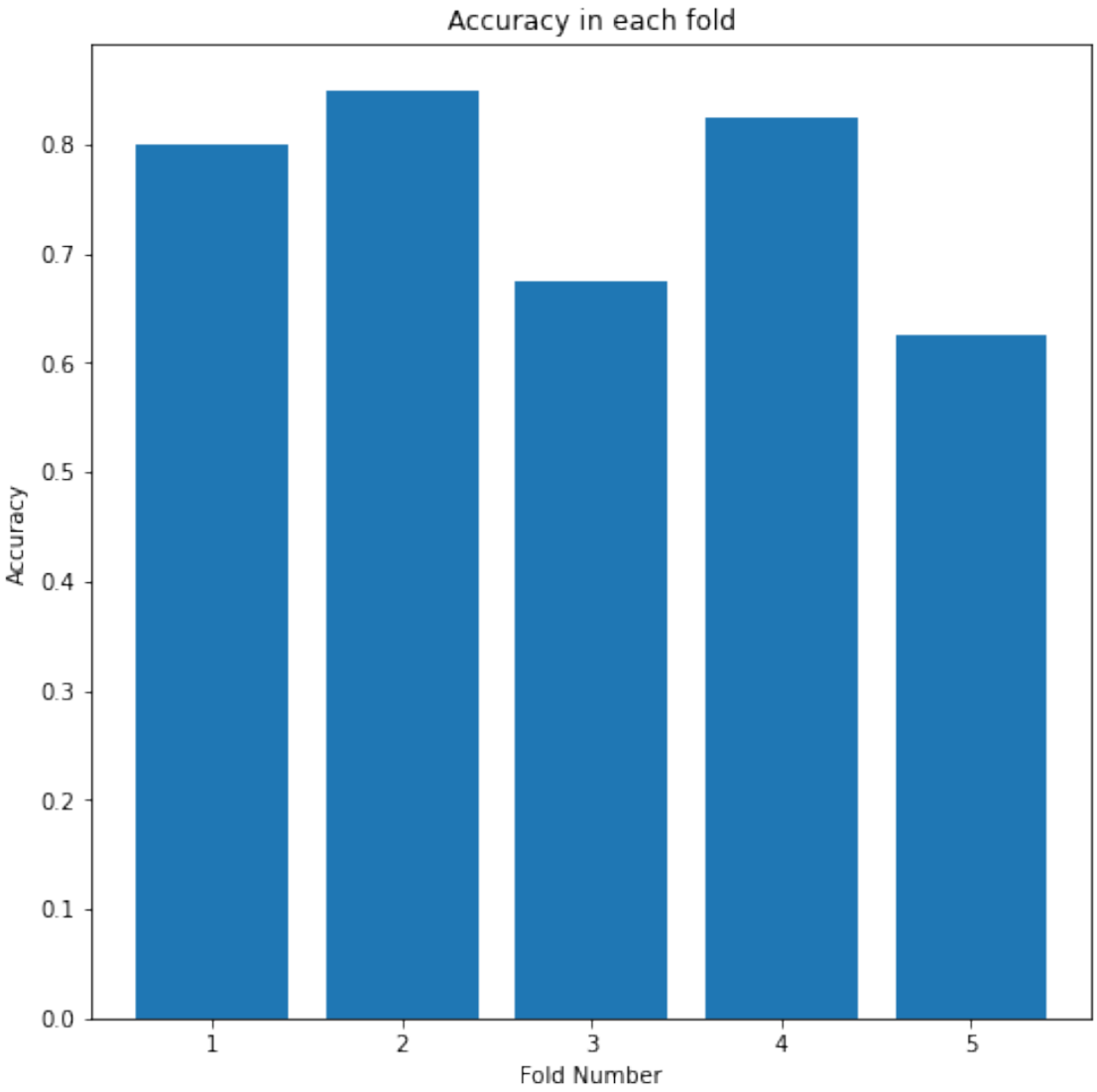
```
: RuntimeWarning: overflow encountered in exp
  y_est = 1/(1 + np.exp(-model))

Accuracy in Fold number 1 is : 0.6
Accuracy in Fold number 2 is : 0.85
Accuracy in Fold number 3 is : 0.675
Accuracy in Fold number 4 is : 0.175
Accuracy in Fold number 5 is : 0.625

/var/folders/kl/55tfkryd5vs0fjm2fnv6vd140000gn/T/ipykernel_36339/3330011543.py:1
6: RuntimeWarning: overflow encountered in exp
  test_y_est = 1/(1 + np.exp(-model))

Accuracy in Fold number 1 is : 0.8
Accuracy in Fold number 2 is : 0.85
Accuracy in Fold number 3 is : 0.675
Accuracy in Fold number 4 is : 0.825
Accuracy in Fold number 5 is : 0.625
```

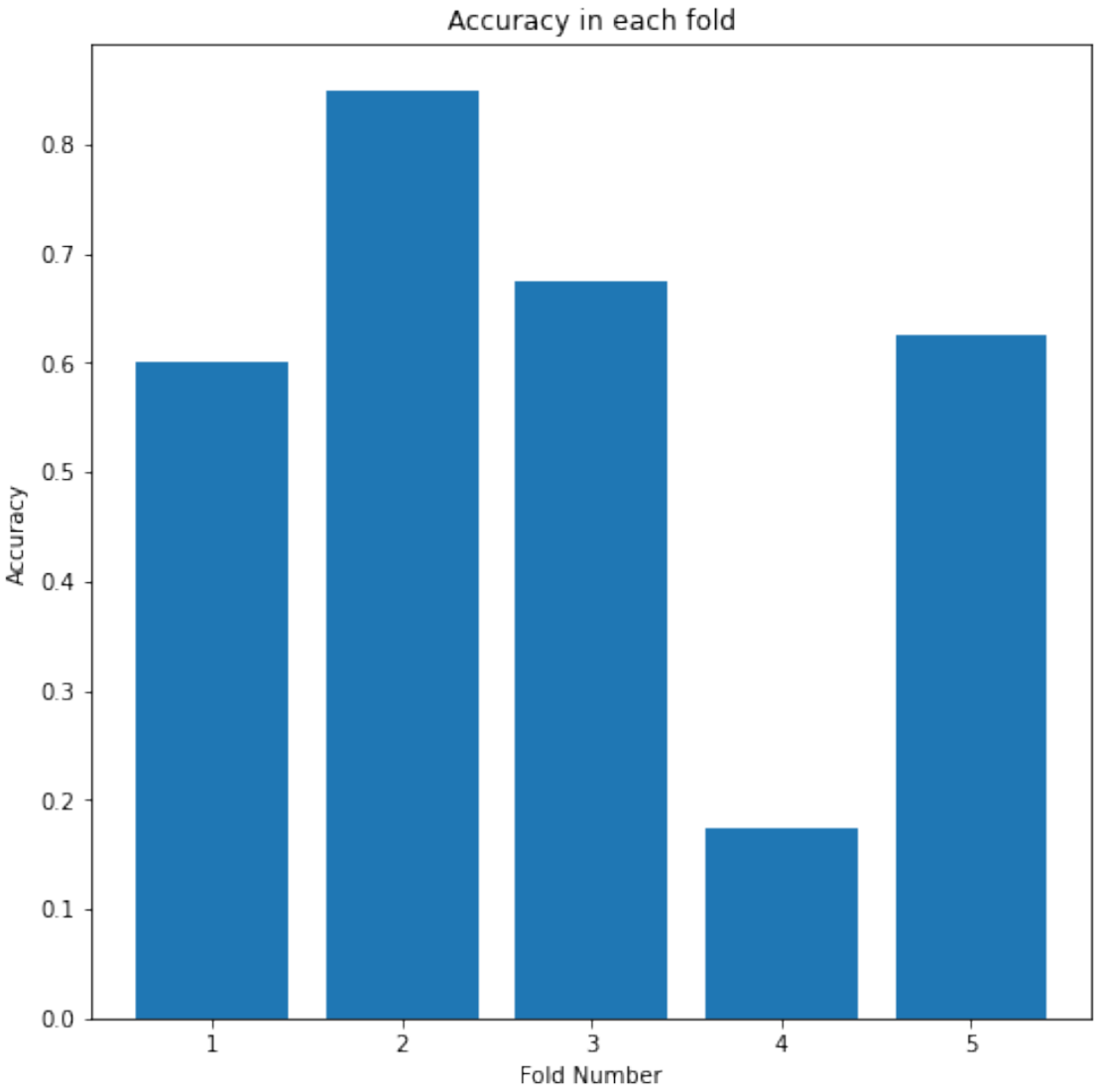
[ ]:
```
[[0.8, 0.85, 0.65, 0.625, 0.575],
 [0.8, 0.85, 0.675, 0.65, 0.6],
 [0.8, 0.85, 0.675, 0.825, 0.625],
 [0.6, 0.85, 0.675, 0.175, 0.625],
 [0.8, 0.85, 0.675, 0.825, 0.625]]
```

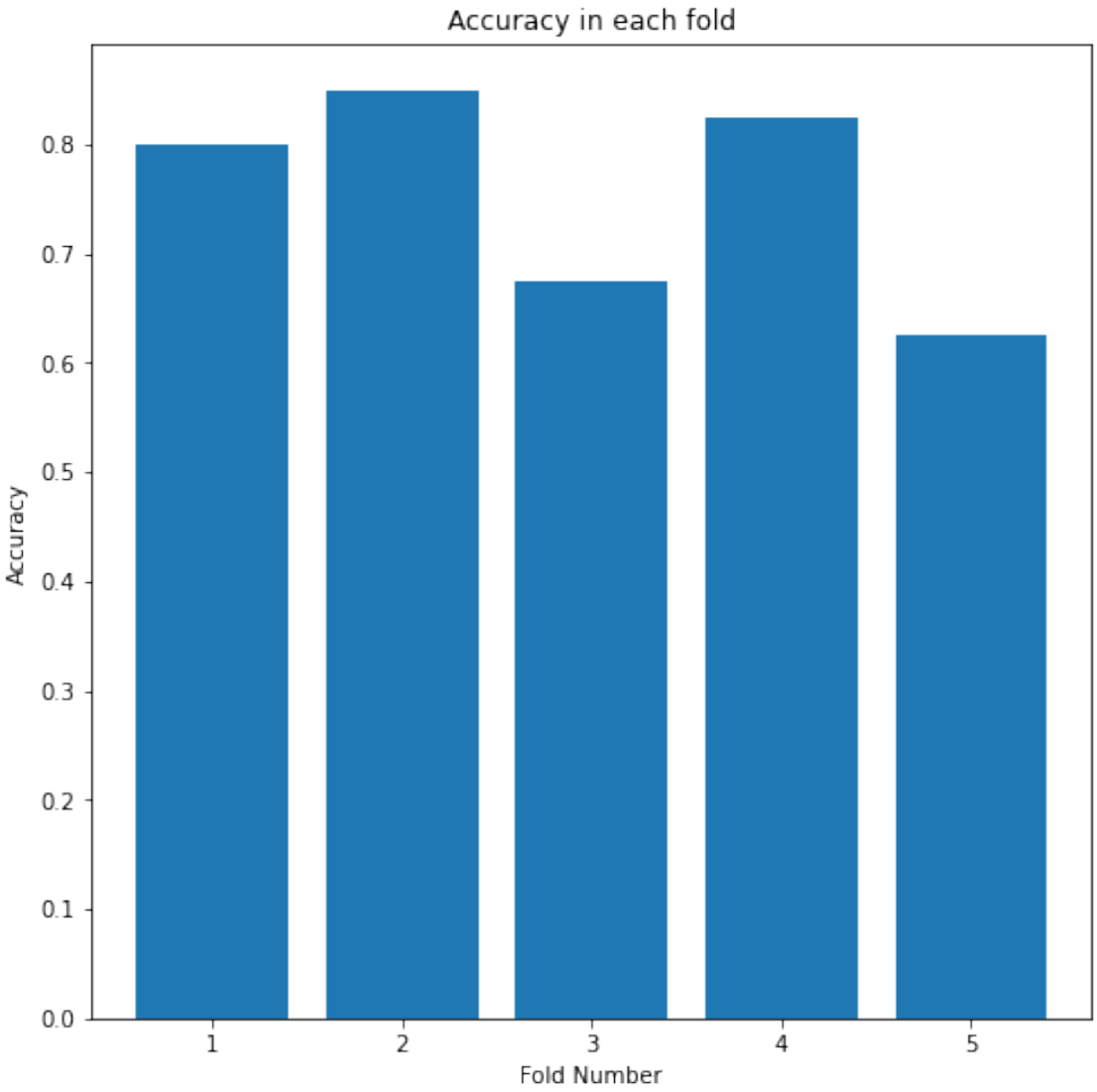
[ ]:
```
for i in range(5):
    lr = learning_rates[i]
    acc_vec = ACCURACY_lr[i]
    fig1 = plt.figure(figsize = (8,8))
    plt.bar(list(range(1,6,1)), acc_vec)
    plt.title('Accuracy in each fold')
    plt.xlabel('Fold Number')
    plt.ylabel('Accuracy')
    plt.show()
    plt.close(fig1)
```


Accuracy in each fold
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1
2
3
4
5
Fold Number


Accuracy in each fold
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1
2
3
4
5
Fold Number


Accuracy in each fold
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1
2
3
4
5
Fold Number


Accuracy in each fold
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1
2
3
4
5
Fold Number




### 3.2 Thank you!

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('PRML06.ipynb')
```

```
--2022-04-21 07:42:26--  https://raw.githubusercontent.com/brpy/colab-
```

```
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.111.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: ‘colab_pdf.py’

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-04-21 07:42:26 (20.9 MB/s) - ‘colab_pdf.py’ saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```

[ ]: