

```
1  ## Zahara Kazmi
2  ## Campus Ministry AI
3  ## MLP
4
5  import numpy
6  import sklearn.decomposition
7  import sklearn.neural_network
8
9
10 def read_data():
11     """Read all of the data from disk."""
12     # Read each data file,
13     d1 = numpy.loadtxt('datacleaned.csv', delimiter=',')
14     return d1
15
16 def main():
17     # Change these
18     activation = 'logistic'
19     learning_rate = 'adaptive'
20     hidden_layer_sizes = (10,)
21     early_stopping = False
22     num_runs = 1
23     myArray=[25]
24
25     # Read the data from disk
26     data = read_data()
27     # Normalize the data
28     normalized = sklearn.preprocessing.normalize(data)
29
30     # Run PCA and transform the data
31     pca = sklearn.decomposition.PCA(svd_solver='full',
32                                     n_components='mle')
33     pca.fit(normalized)
34     transformed = pca.transform(normalized)
35
36     # Split the data into training (90%) and testing (10
37     %) data
38     split_point = int(len(data) * .9)
39
40     train = transformed[:split_point, :2]
41     train_target = data[:split_point, 2]
42
43     test = transformed[split_point:, :2]
44     test_target = data[split_point:, 2]
```

```
45     # Used to calculate the average accuracy
46     total_accuracy = 0
47
48     for i in range(num_runs):
49         # Run the MLP and calculate its accuracy
50         mlp = sklearn.neural_network.MLPClassifier(
51             activation=activation,
52             learning_rate=learning_rate,
53             hidden_layer_sizes=hidden_layer_sizes,
54             early_stopping=early_stopping)
55
56         net = mlp.fit(train, train_target)
57         predictions = net.predict(test)
58         score = net.score(test, test_target)
59
60         # Add the accuracy to the total
61         total_accuracy += score
62
63     # Calculate and print the average accuracy
64     average_accuracy = total_accuracy / num_runs
65
66     resultY=mlp.predict(myArray)
67     print('Accuracy: ', average_accuracy)
68     print('Result for Prediction: ', resultY)
69
70
71 if __name__ == '__main__':
72     main()
73
```

```

1  """
2  This script runs PCA on campusmin data.
3  """
4  import numpy
5  import sklearn.decomposition
6  import sklearn.preprocessing
7
8
9  def read_data():
10     """Read all of the data from disk."""
11     # Read each data file, skipping the first 3 weeks of
data
12     d1 = numpy.loadtxt('datacleaned.csv', delimiter=',')
13     return numpy.concatenate(d1)
14
15  def main():
16     # Change these
17     svd_solver = 'randomized'
18     iterated_power = 0
19     n_components = None
20     num_runs = 1
21     print(read_data())
22     # Read the data from disk, ignoring the targets, and
normalize it
23     data = sklearn.preprocessing.normalize(read_data())
24
25     # Used to calculate the average eigenvalues
26     total_eigenvalues = 0
27
28     for i in range(num_runs):
29         # Run PCA and add the eigenvalues to the total
30         pca = sklearn.decomposition.PCA(
31             svd_solver=svd_solver,
32             iterated_power=iterated_power,
33             n_components=n_components)
34
35         pca.fit(data)
36         total_eigenvalues += pca.explained_variance_
37
38     # Calculate and print the average eigenvalues
39     average_eigenvalues = total_eigenvalues / num_runs
40
41     print('Eigenvalues: ', average_eigenvalues)
42
43  if __name__ == '__main__':

```

```
44     main()
```

```
1 Time ,Date ,Attendance
2 10,1,10
3 11,2,10
4 12,2,10
5 13,2,15
6 13,4,15
7 15,3,15
8 15,5,15
9 20,3,3
10 20,4,4
11 20,4,10
12 20,4,20
13 19,3,3
14 19,3,4
15 19,3,10
16 19,3,15
17
```

```
1  #!/usr/bin/python
2
3  import numpy
4  hostname = 'localhost'
5  username = 'root'
6  password = ''
7  database = 'campusmin'
8
9  # Simple routine to run a query on a database and print
   the results:
10 def doQuery( conn ) :
11     cur = conn.cursor()
12
13     cur.execute( "SELECT * FROM program" )
14
15     for pdate,ptime,patten in cur.fetchall() :
16         a=numpy.array(pdate, ptime, patten)
17         numpy.savetxt("rawdata.csv", a, delimiter=',')
18
19
20 print("Using mysql.connector...")
21 import mysql.connector
22 myConnection = mysql.connector.connect( host=hostname,
    user=username, passwd=password, db=database )
23 doQuery( myConnection )
24 myConnection.close()
```

```
1 10,1,10
2
3 11,2,10
4
5 12,2,10
6
7 13,2,15
8
9 13,4,15
10
11 15,3,15
12
13 15,5,15
14
15 20,3,3
16
17 20,4,4
18
19 20,4,10
20
21 20,4,20
22
23 19,3,3
24
25 19,3,4
26
27 19,3,10
28
29 19,3,15
30
31
```

```
1  ## Zahara Kazmi
2  ## Campus Ministry AI
3  ## MLP
4  #
5  "This script first cleans and then feature-engineers the
   data."
6
7
8
9  import collections
10 import csv
11
12
13
14
15 def is_numeric(value):
16     """Check whether a value is numeric."""
17     try:
18         float(value)
19         return True
20     except ValueError:
21         # An exception will be thrown if the value is non-
numeric.
22         return False
23
24 def clean_data(rows):
25     """Clean the data by keeping rows that are numeric."""
26     # Cleaned rows.
27     new_rows = []
28     for row in rows:
29         # Keep the row if it contains valid team names,
valid
30         # statistics, and a valid winning team.
31         if (is_numeric(row[0])
32             and is_numeric(row[1]))\
33             and is_numeric(row[2]):
34             new_rows.append(row)
35     return new_rows
36
37
38
39
40 def write_csv(filename, rows):
41     """Write rows of data to a CSV file."""
42     with open(filename, 'w') as f:
```



```
43         writer = csv.writer(f)
44
45         writer.writerows(rows)
46
47
48     def main():
49         in_filename = 'rawdata.csv'.format()
50
51         out_filename1 = 'datacleaned.csv'.format()
52
53         with open(in_filename, 'r') as f:
54             reader = csv.reader(f)
55
56             # Skip the header.
57             next(reader)
58
59             # Clean the data.
60             cleaned_rows = clean_data(list(reader))
61
62             # Engineer the features
63
64             # Write the cleaned data to a file.
65             write_csv(out_filename1, cleaned_rows)
66
67
68 if __name__ == '__main__':
69     main()
```