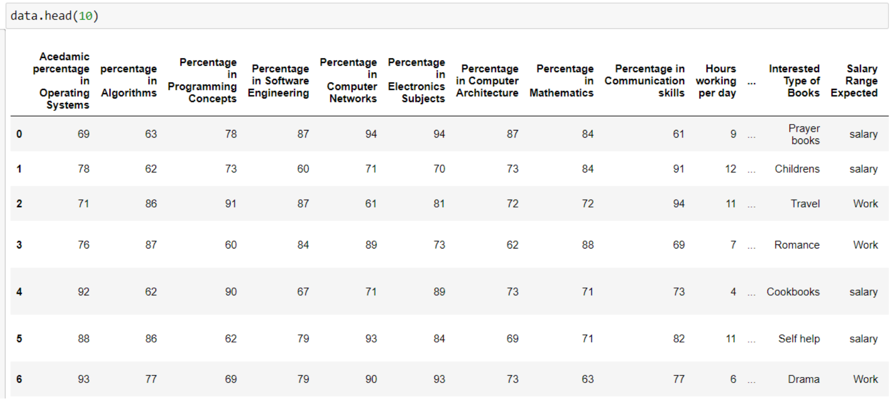
AI Assignment 4 Report

Name: Bhavya Narang

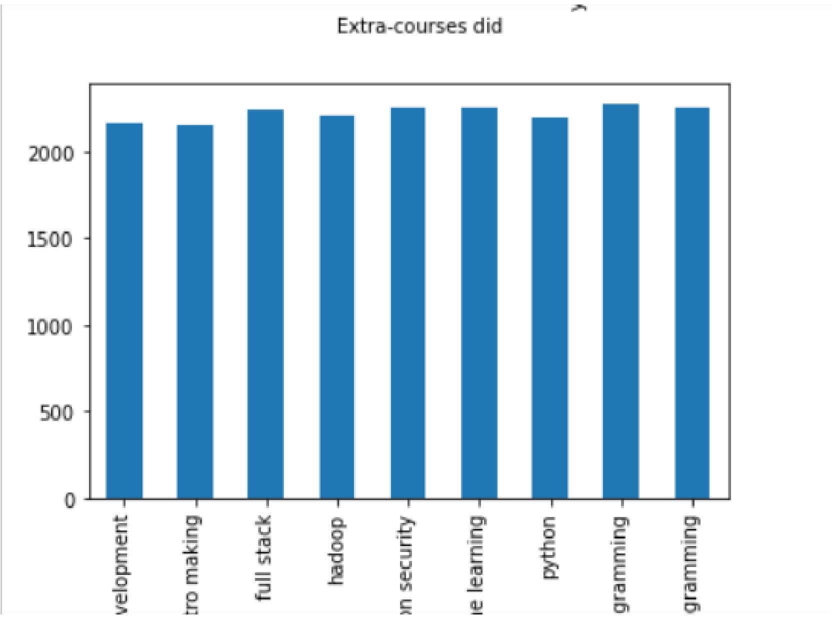
Roll No: 2019462

Data Visualization:



```
data.head(10)
```

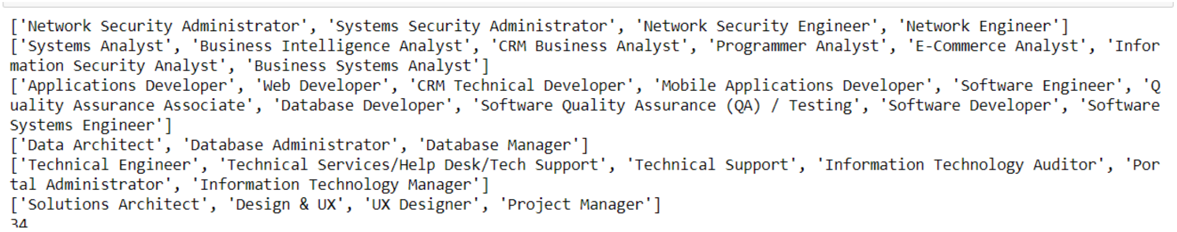
	Acedamic percentage in Operating Systems	percentage in Algorithms	Percentage in Programming Concepts	Percentage in Software Engineering	Percentage in Computer Networks	Percentage in Electronics Subjects	Percentage in Computer Architecture	Percentage in Mathematics	Percentage in Communication skills	Hours working per day	...	Interested Type of Books	Salary Range Expected
0	69	63	78	87	94	94	87	84	61	9	...	Prayer books	salary
1	78	62	73	60	71	70	73	84	91	12	...	Childrens	salary
2	71	86	91	87	61	81	72	72	94	11	...	Travel	Work
3	76	87	60	84	89	73	62	88	69	7	...	Romance	Work
4	92	62	90	67	71	89	73	71	73	4	...	Cookbooks	salary
5	88	86	62	79	93	84	69	71	82	11	...	Self help	salary
6	93	77	69	79	90	93	73	63	77	6	...	Drama	Work



(plotting bar graphs for every column)

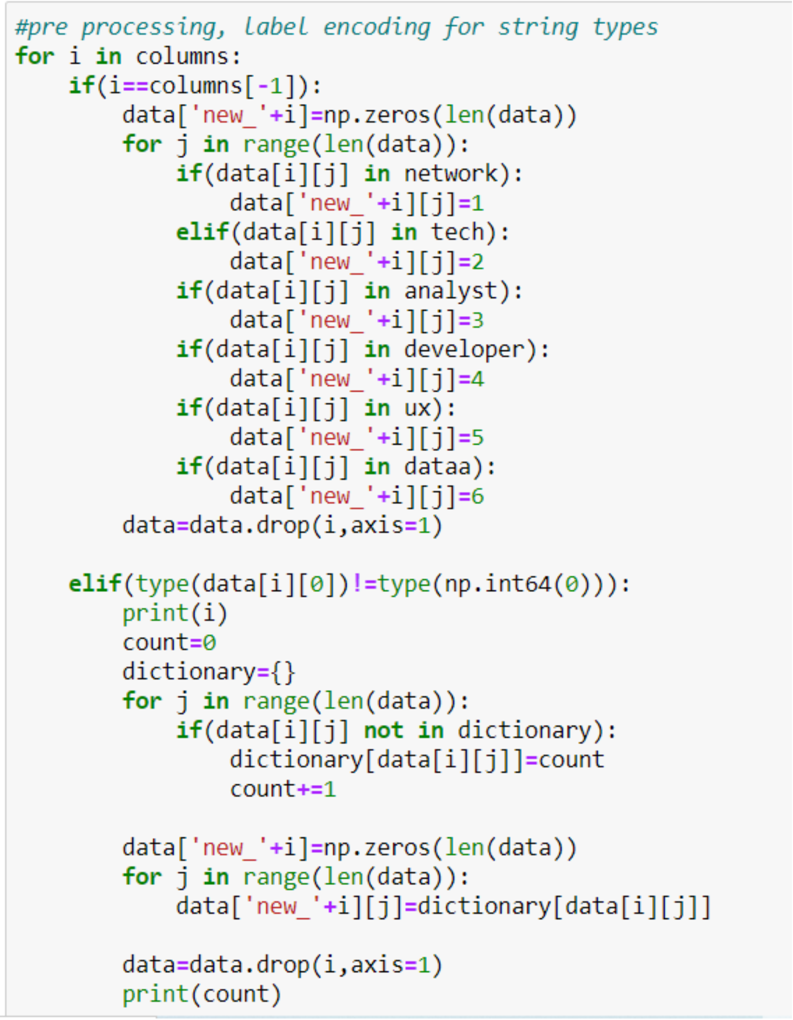
(rest are present in ipynb file)

Grouping the final 34 classes to a cluster of 6 classes based on certain heuristics manually:



```
['Network Security Administrator', 'Systems Security Administrator', 'Network Security Engineer', 'Network Engineer']
['Systems Analyst', 'Business Intelligence Analyst', 'CRM Business Analyst', 'Programmer Analyst', 'E-Commerce Analyst', 'Infor
mation Security Analyst', 'Business Systems Analyst']
['Applications Developer', 'Web Developer', 'CRM Technical Developer', 'Mobile Applications Developer', 'Software Engineer', 'Q
uality Assurance Associate', 'Database Developer', 'Software Quality Assurance (QA) / Testing', 'Software Developer', 'Software
Systems Engineer']
['Data Architect', 'Database Administrator', 'Database Manager']
['Technical Engineer', 'Technical Services/Help Desk/Tech Support', 'Technical Support', 'Information Technology Auditor', 'Por
tal Administrator', 'Information Technology Manager']
['Solutions Architect', 'Design & UX', 'UX Designer', 'Project Manager']
34
```

Label encoding for string types



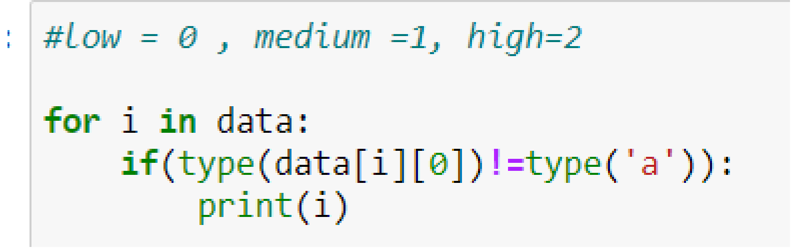
```
#pre processing, label encoding for string types
for i in columns:
    if(i==columns[-1]):
        data['new_'+i]=np.zeros(len(data))
        for j in range(len(data)):
            if(data[i][j] in network):
                data['new_'+i][j]=1
            elif(data[i][j] in tech):
                data['new_'+i][j]=2
            if(data[i][j] in analyst):
                data['new_'+i][j]=3
            if(data[i][j] in developer):
                data['new_'+i][j]=4
            if(data[i][j] in ux):
                data['new_'+i][j]=5
            if(data[i][j] in dataa):
                data['new_'+i][j]=6
        data=data.drop(i,axis=1)

    elif(type(data[i][0])!=type(np.int64(0))):
        print(i)
        count=0
        dictionary={}
        for j in range(len(data)):
            if(data[i][j] not in dictionary):
                dictionary[data[i][j]]=count
                count+=1

        data['new_'+i]=np.zeros(len(data))
        for j in range(len(data)):
            data['new_'+i][j]=dictionary[data[i][j]]

        data=data.drop(i,axis=1)
        print(count)
```

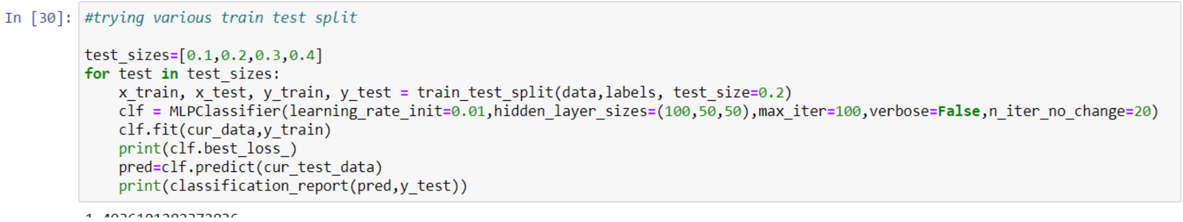
Making buckets for numerical data.



```
#low = 0 , medium =1, high=2

for i in data:
    if(type(data[i][0])!=type('a')):
        print(i)
```

Trying various test-train ratios

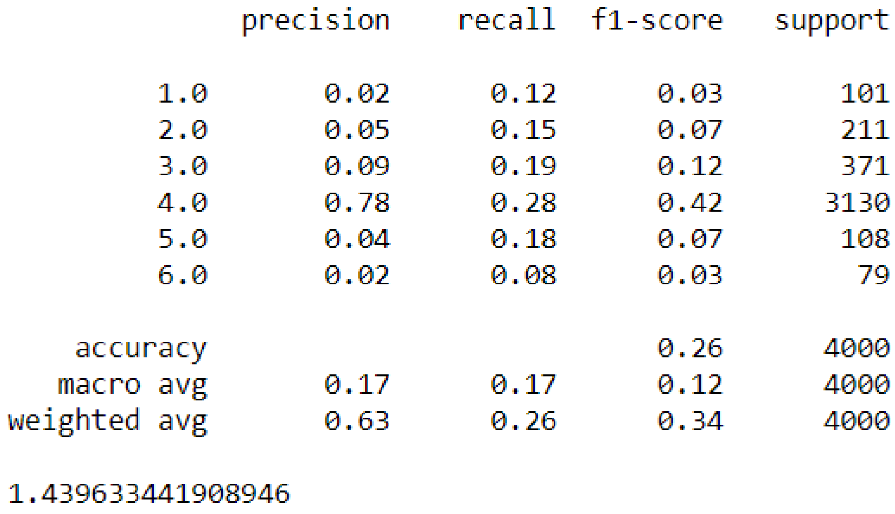


```
In [30]: #trying various train test split

         test_sizes=[0.1,0.2,0.3,0.4]
         for test in test_sizes:
             x_train, x_test, y_train, y_test = train_test_split(data,labels, test_size=0.2)
             clf = MLPClassifier(learning_rate_init=0.01,hidden_layer_sizes=(100,50,50),max_iter=100,verbose=False,n_iter_no_change=20)
             clf.fit(cur_data,y_train)
             print(clf.best_loss_)
             pred=clf.predict(cur_test_data)
             print(classification_report(pred,y_test))

         1.4036101382373836
```

Ratio with 0.8:0.2 gives best accuracy on model.



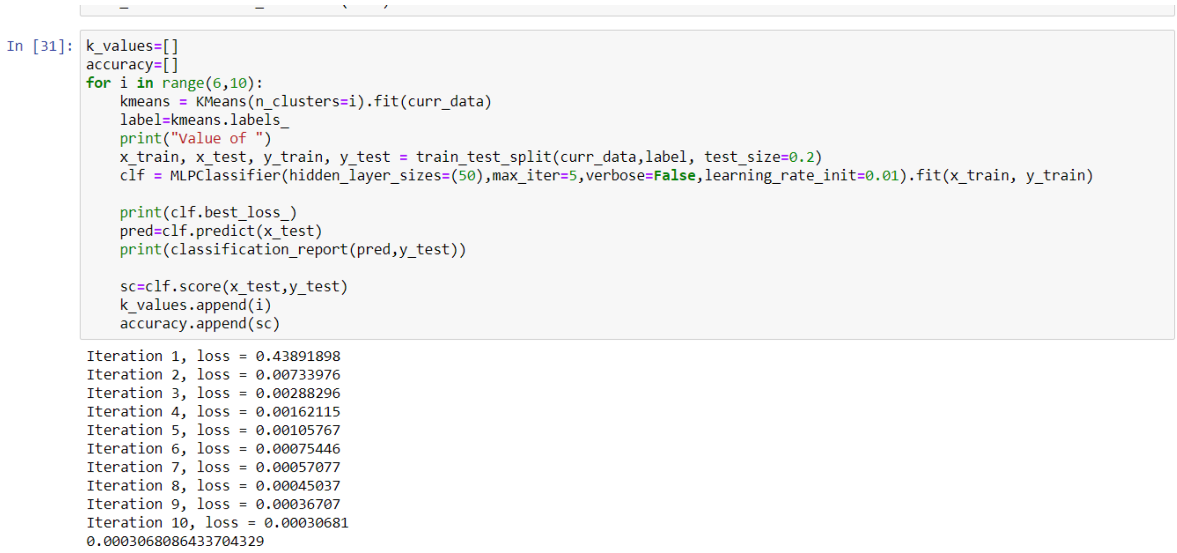
```
              precision    recall  f1-score   support

         1.0       0.02      0.12      0.03       101
         2.0       0.05      0.15      0.07       211
         3.0       0.09      0.19      0.12       371
         4.0       0.78      0.28      0.42      3130
         5.0       0.04      0.18      0.07       108
         6.0       0.02      0.08      0.03        79

    accuracy                           0.26      4000
   macro avg       0.17      0.17      0.12      4000
weighted avg       0.63      0.26      0.34      4000

1.4396334419089 46
```

Grouping data non manually (applying clustering and then applying ANN)



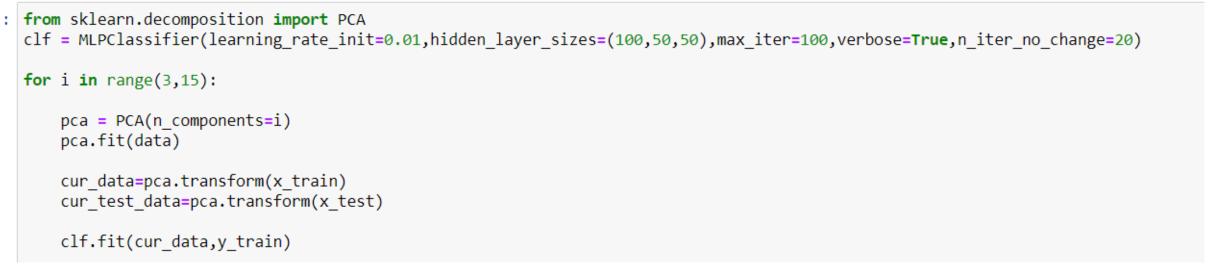
```
In [31]: k_values=[]
         accuracy=[]
         for i in range(6,10):
             kmeans = KMeans(n_clusters=i).fit(curr_data)
             label=kmeans.labels_
             print("Value of ")
             x_train, x_test, y_train, y_test = train_test_split(curr_data,label, test_size=0.2)
             clf = MLPClassifier(hidden_layer_sizes=(50),max_iter=5,verbose=False,learning_rate_init=0.01).fit(x_train, y_train)

             print(clf.best_loss_)
             pred=clf.predict(x_test)
             print(classification_report(pred,y_test))

             sc=clf.score(x_test,y_test)
             k_values.append(i)
             accuracy.append(sc)

         Iteration 1, loss = 0.43891898
         Iteration 2, loss = 0.00733976
         Iteration 3, loss = 0.00288296
         Iteration 4, loss = 0.00162115
         Iteration 5, loss = 0.00105767
         Iteration 6, loss = 0.00075446
         Iteration 7, loss = 0.00057077
         Iteration 8, loss = 0.00045037
         Iteration 9, loss = 0.00036707
         Iteration 10, loss = 0.00030681
         0.0003068086433704329
```

Applying PCA to reduce dimensions and then applying ANN model as simple ANN gives poor accuracy.



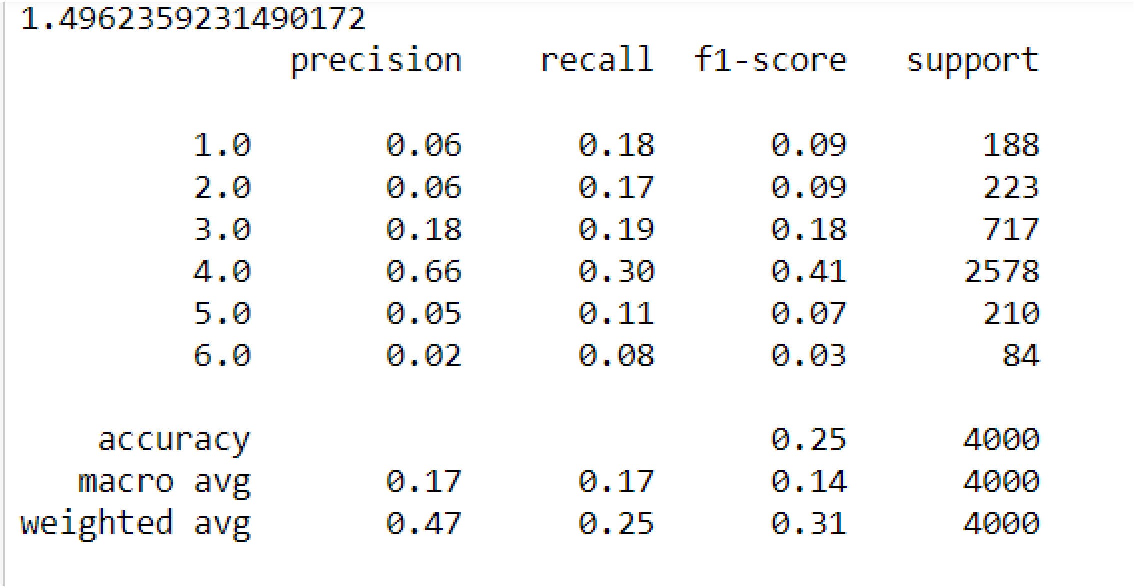
```
: from sklearn.decomposition import PCA
  clf = MLPClassifier(learning_rate_init=0.01,hidden_layer_sizes=(100,50,50),max_iter=100,verbose=True,n_iter_no_change=20)

  for i in range(3,15):

      pca = PCA(n_components=i)
      pca.fit(data)

      cur_data=pca.transform(x_train)
      cur_test_data=pca.transform(x_test)

      clf.fit(cur_data,y_train)
```

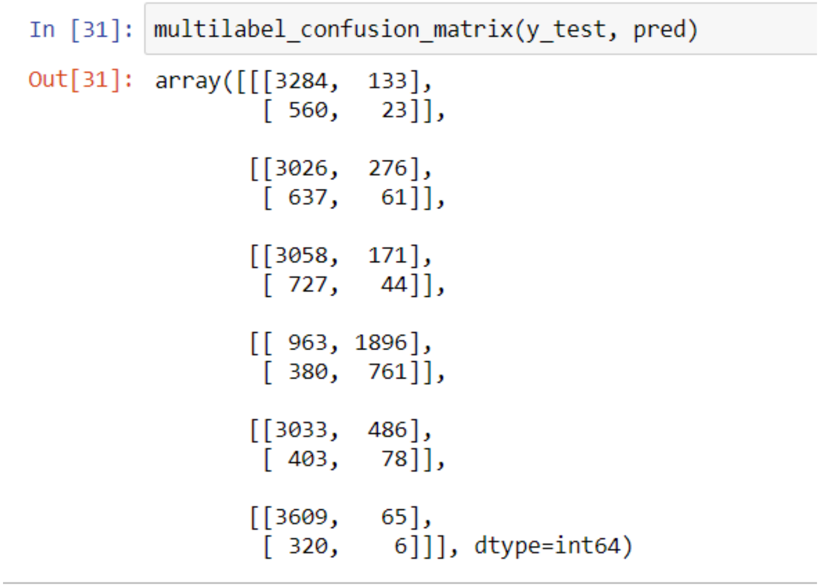


```
1.4962359231490172
              precision    recall  f1-score   support

         1.0       0.06      0.18      0.09       188
         2.0       0.06      0.17      0.09       223
         3.0       0.18      0.19      0.18       717
         4.0       0.66      0.30      0.41      2578
         5.0       0.05      0.11      0.07       210
         6.0       0.02      0.08      0.03        84

    accuracy                           0.25      4000
   macro avg       0.17      0.17      0.14      4000
weighted avg       0.47      0.25      0.31      4000
```

Finally confusion matrix for labels:



```
In [31]: multilabel_confusion_matrix(y_test, pred)

Out[31]: array([[[3284,  133],
                 [ 560,   23]],

                [[3026,  276],
                 [ 637,   61]],

                [[3058,  171],
                 [ 727,   44]],

                [[ 963, 1896],
                 [ 380,  761]],

                [[3033,  486],
                 [ 403,   78]],

                [[3609,   65],
                 [ 320,    6]]], dtype=int64)
```