Name: Vardhana Sharma

Roll No.: SP22003

Firstly we are importing all the necessary libraries for the Analysis

In [3]:

```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import copy
```

In [4]:

```
# Importing dataset
data = pd.read_csv("roo_data.csv")
data.head()
```

Out[4]:

	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
0	69	63	78	87	94	94	87	84	61
1	78	62	73	60	71	70	73	84	91
2	71	86	91	87	61	81	72	72	94
3	76	87	60	84	89	73	62	88	69
4	92	62	90	67	71	89	73	71	73

5 rows × 39 columns

Modification and Analysis

Now below is the general analysis of the dataset

In [5]:

```
data.shape
```

Out[5]:

```
(20000, 39)
```

In [6]:

```
data.describe()
```

Out[6]:

	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	P Co
count	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	2
mean	77.002300	76.948200	77.017550	77.094500	76.958200	77.015550	77.069850	76.913100	
std	10.085697	10.101733	10.134815	10.087837	10.020088	10.168888	10.069059	10.138555	
min	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	
25%	68.000000	68.000000	68.000000	68.000000	68.000000	68.000000	68.000000	68.000000	
50%	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	77.000000	
75%	86.000000	86.000000	86.000000	86.000000	85.000000	86.000000	86.000000	86.000000	
max	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	94.000000	

In [7]:

```
for i in range(0,data.shape[1]):
    print("Unique values for " + data.columns[i])
    print("Number of Unique Values: ",len(np.unique(data.iloc[:,i])))
    print(data.iloc[:,i].unique())
```

```
Unique values for Acedamic percentage in Operating Systems
Number of Unique Values:  35
[69 78 71 76 92 88 93 84 73 62 63 68 90 94 60 82 67 65 74 75 83 89 80 70
 66 85 61 81 79 86 64 91 72 77 87]
Unique values for percentage in Algorithms
Number of Unique Values:  35
[63 62 86 87 77 72 66 76 80 64 93 83 71 92 91 73 61 89 67 74 82 60 68 88
 70 85 81 78 84 69 94 75 65 79 90]
Unique values for Percentage in Programming Concepts
Number of Unique Values:  35
[78 73 91 60 90 62 69 88 66 85 70 81 61 77 63 94 68 76 75 93 64 65 84 72
 80 86 74 83 67 79 71 87 92 82 89]
Unique values for Percentage in Software Engineering
Number of Unique Values:  35
[87 60 84 67 79 62 81 91 83 90 71 74 63 86 70 75 92 93 72 78 85 64 82 65
 69 94 73 66 80 68 61 88 77 76 89]
Unique values for Percentage in Computer Networks
Number of Unique Values:  35
[94 71 61 89 93 90 66 81 82 70 77 65 62 64 78 63 67 86 69 92 84 85 87 68
 83 60 88 74 75 80 91 72 76 73 79]
Unique values for Percentage in Electronics Subjects
Number of Unique Values:  35
[94 70 81 73 89 84 93 63 69 82 72 67 65 61 88 91 74 90 80 79 75 62 76 77
 83 92 60 71 68 66 87 64 86 85 78]
Unique values for Percentage in Computer Architecture
Number of Unique Values:  35
[87 73 72 62 69 78 61 63 75 86 65 67 92 91 88 82 80 83 90 71 89 81 79 93
 70 84 85 76 74 64 77 94 60 68 66]
Unique values for Percentage in Mathematics
Number of Unique Values:  35
[84 72 88 71 63 94 87 89 64 81 62 73 65 82 60 61 80 77 78 68 76 83 92 93
 70 79 75 85 91 74 67 66 69 90 86]
Unique values for Percentage in Communication skills
Number of Unique Values:  35
[61 91 94 69 73 82 77 60 90 81 89 85 79 62 68 70 84 65 66 67 75 87 76 80
 78 63 64 88 92 71 93 86 83 74 72]
Unique values for Hours working per day
Number of Unique Values:  9
[ 9 12 11  7  4  6 10  8  5]
Unique values for Logical quotient rating
Number of Unique Values:  9
[4 7 1 5 3 2 9 6 8]
Unique values for hackathons
Number of Unique Values:  7
[0 1 4 3 2 6 5]
Unique values for coding skills rating
```

```
Number of Unique Values:  9
[4 2 1 6 8 3 5 9 7]
Unique values for public speaking points
Number of Unique Values:  9
[8 3 5 1 6 4 9 7 2]
Unique values for can work long time before system?
Number of Unique Values:  2
['yes' 'no']
Unique values for self-learning capability?
Number of Unique Values:  2
['yes' 'no']
Unique values for Extra-courses did
Number of Unique Values:  2
['yes' 'no']
Unique values for certifications
Number of Unique Values:  9
['shell programming' 'machine learning' 'app development' 'python'
 'r programming' 'information security' 'hadoop' 'distro making'
 'full stack']
Unique values for workshops
Number of Unique Values:  8
['cloud computing' 'database security' 'web technologies' 'data science'
 'testing' 'hacking' 'game development' 'system designing']
Unique values for talenttests taken?
Number of Unique Values:  2
['no' 'yes']
Unique values for olympiads
Number of Unique Values:  2
['yes' 'no']
Unique values for reading and writing skills
Number of Unique Values:  3
['excellent' 'poor' 'medium']
Unique values for memory capability score
Number of Unique Values:  3
['excellent' 'medium' 'poor']
Unique values for Interested subjects
Number of Unique Values:  10
['cloud computing' 'networks' 'hacking' 'Computer Architecture'
 'programming' 'parallel computing' 'IOT' 'data engineering'
 'Software Engineering' 'Management']
Unique values for interested career area
Number of Unique Values:  6
['system developer' 'Business process analyst' 'developer' 'testing'
 'security' 'cloud computing']
Unique values for Job/Higher Studies?
Number of Unique Values:  2
['higherstudies' 'job']
Unique values for Type of company want to settle in?
Number of Unique Values:  10
['Web Services' 'SAaS services' 'Sales and Marketing'
 'Testing and Maintainance Services' 'product development' 'BPA'
 'Service Based' 'Product based' 'Cloud Services' 'Finance']
Unique values for Taken inputs from seniors or elders
Number of Unique Values:  2
['no' 'yes']
Unique values for interested in games
Number of Unique Values:  2
['no' 'yes']
Unique values for Interested Type of Books
Number of Unique Values:  31
['Prayer books' 'Childrens' 'Travel' 'Romance' 'Cookbooks' 'Self help'
 'Drama' 'Math' 'Religion-Spirituality' 'Anthology' 'Trilogy'
 'Autobiographies' 'Mystery' 'Diaries' 'Journals' 'History' 'Art'
 'Dictionaries' 'Horror' 'Encyclopedias' 'Action and Adventure' 'Fantasy'
 'Comics' 'Science fiction' 'Series' 'Guide' 'Biographies' 'Health'
 'Satire' 'Science' 'Poetry']
Unique values for Salary Range Expected
Number of Unique Values:  2
['salary' 'Work']
Unique values for In a Realtionship?
Number of Unique Values:  2
['no' 'yes']
```

```
Unique values for Gentle or Tuff behaviour?
Number of Unique Values:  2
['stubborn' 'gentle']
Unique values for Management or Technical
Number of Unique Values:  2
['Management' 'Technical']
Unique values for Salary/work
Number of Unique Values:  2
['salary' 'work']
Unique values for hard/smart worker
Number of Unique Values:  2
['hard worker' 'smart worker']
Unique values for worked in teams ever?
Number of Unique Values:  2
['yes' 'no']
Unique values for Introvert
Number of Unique Values:  2
['no' 'yes']
Unique values for Suggested Job Role
Number of Unique Values:  34
['Database Developer' 'Portal Administrator'
 'Systems Security Administrator' 'Business Systems Analyst'
 'Software Systems Engineer' 'Business Intelligence Analyst'
 'CRM Technical Developer' 'Mobile Applications Developer' 'UX Designer'
 'Quality Assurance Associate' 'Web Developer'
 'Information Security Analyst' 'CRM Business Analyst' 'Technical Support'
 'Project Manager' 'Information Technology Manager' 'Programmer Analyst'
 'Design & UX' 'Solutions Architect' 'Systems Analyst'
 'Network Security Administrator' 'Data Architect' 'Software Developer'
 'E-Commerce Analyst' 'Technical Services/Help Desk/Tech Support'
 'Information Technology Auditor' 'Database Manager'
 'Applications Developer' 'Database Administrator' 'Network Engineer'
 'Software Engineer' 'Technical Engineer' 'Network Security Engineer'
 'Software Quality Assurance (QA) / Testing']
```

In [51]:

```
from collections import Counter
Counter(data['Suggested Job Role'])
```

Out[51]:

```
Counter({'Database Developer': 581,
         'Portal Administrator': 593,
         'Systems Security Administrator': 562,
         'Business Systems Analyst': 582,
         'Software Systems Engineer': 575,
         'Business Intelligence Analyst': 540,
         'CRM Technical Developer': 567,
         'Mobile Applications Developer': 538,
         'UX Designer': 589,
         'Quality Assurance Associate': 565,
         'Web Developer': 570,
         'Information Security Analyst': 543,
         'CRM Business Analyst': 584,
         'Technical Support': 565,
         'Project Manager': 602,
         'Information Technology Manager': 591,
         'Programmer Analyst': 529,
         'Design & UX': 588,
         'Solutions Architect': 578,
         'Systems Analyst': 550,
         'Network Security Administrator': 1112,
         'Data Architect': 564,
         'Software Developer': 587,
         'E-Commerce Analyst': 546,
         'Technical Services/Help Desk/Tech Support': 558,
         'Information Technology Auditor': 558,
         'Database Manager': 570,
         'Applications Developer': 551,
         'Database Administrator': 593,
         'Network Engineer': 621,
```

```
 'Software Engineer': 590,
 'Technical Engineer': 557,
 'Network Security Engineer': 630,
 'Software Quality Assurance (QA) / Testing': 571})
```

In [52]:

```
print('Total Classes: {}'.format(len(np.unique(data['Suggested Job Role']))))
```

```
Total Classes: 34
```

Count plot before clubbing classes

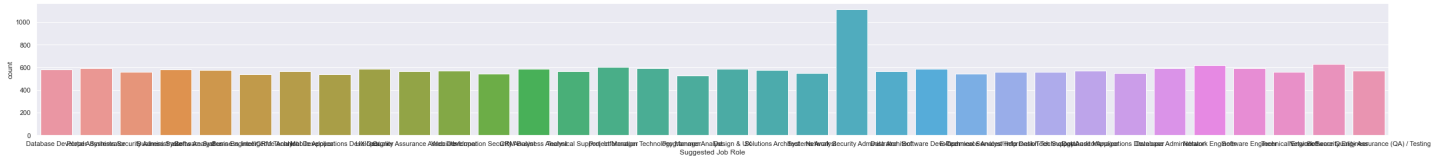
In [97]:

```
print("Count plot before clubbing classes")
sns.set_style(style="darkgrid")
sns.set(rc={"figure.figsize":(40, 4)})
sns.countplot(x='Suggested Job Role',data=data)
```

```
Count plot before clubbing classes
```

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x271c4bfd970>
```



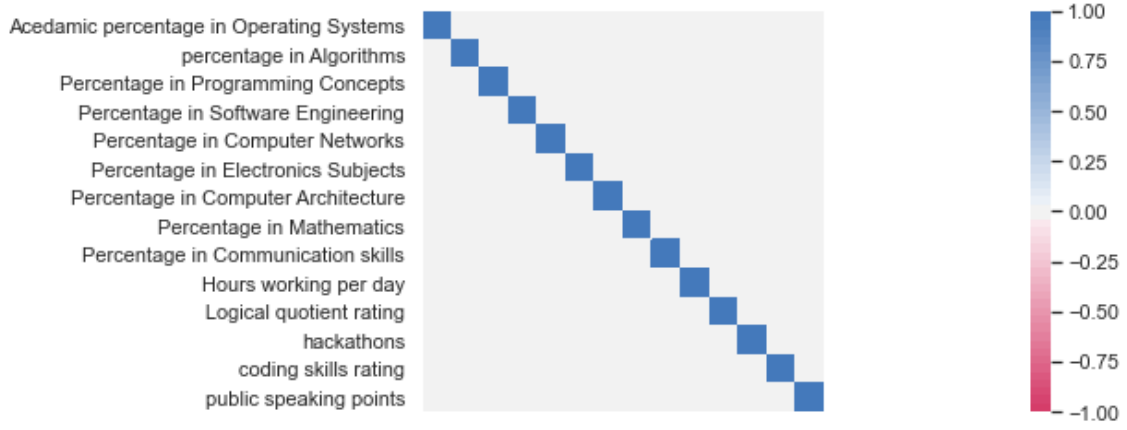
Correlation Heatmap of data

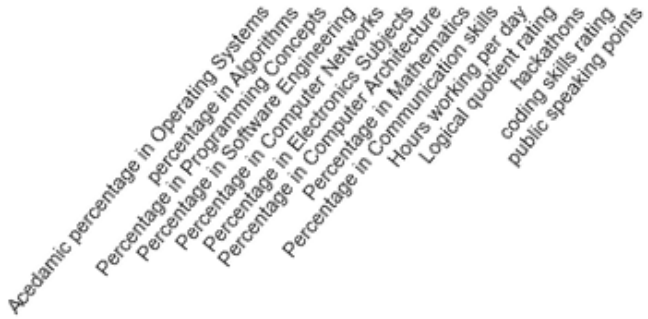
In [94]:

```
correlationMatrix = data.corr(method='pearson')
ax = sns.heatmap(correlationMatrix,vmin=-1, vmax=1, center=0,cmap=sns.diverging_palette(
0,250,n=50),square=True)
ax.set_xticklabels(ax.get_xticklabels(),rotation = 50, horizontalalignment='right')
```

Out[94]:

```
[Text(0.5, 0, 'Acedamic percentage in Operating Systems'),
 Text(1.5, 0, 'percentage in Algorithms'),
 Text(2.5, 0, 'Percentage in Programming Concepts'),
 Text(3.5, 0, 'Percentage in Software Engineering'),
 Text(4.5, 0, 'Percentage in Computer Networks'),
 Text(5.5, 0, 'Percentage in Electronics Subjects'),
 Text(6.5, 0, 'Percentage in Computer Architecture'),
 Text(7.5, 0, 'Percentage in Mathematics'),
 Text(8.5, 0, 'Percentage in Communication skills'),
 Text(9.5, 0, 'Hours working per day'),
 Text(10.5, 0, 'Logical quotient rating'),
 Text(11.5, 0, 'hackathons'),
 Text(12.5, 0, 'coding skills rating'),
 Text(13.5, 0, 'public speaking points')]
```





So Initially there are total 34 classes in the dataset now we are going to reduce it by clubbing some classes

In [10]:

```
# Now replacing and clubbing some labels to decrease the total no. of classes
data = copy.deepcopy(data)
data['Suggested Job Role'].replace({'CRM Technical Developer':'Technical Support', 'Techn
ical Engineer':'Technical Support','Technical Services/Help Desk/Tech Support':'Technical
Support', 'Technical Support':'Technical Support'}, inplace=True)
data['Suggested Job Role'].replace({'Data Architect':'Database Engineer', 'Database Admin
istrator':'Database Engineer', 'Database Manager':'Database Engineer'}, inplace=True)
data['Suggested Job Role'].replace({'Information Technology Auditor':'Software Engineer',
'Information Technology Manager':'Software Engineer','Software Engineer':'Software Engine
er', 'Software Systems Engineer':'Software Engineer','Solutions Architect':'Software Engi
neer'}, inplace=True)
data['Suggested Job Role'].replace({'Developer':'Developer/UX Designer',  'UX Designer':'
Developer/UX Designer'}, inplace=True)
data['Suggested Job Role'].replace({'Business Intelligence Analyst':'Analyst','Business S
ystems Analyst':'Analyst','CRM Business Analyst':'Analyst', 'E-Commerce Analyst':'Analyst
', 'Information Security Analyst':'Analyst', 'Systems Analyst':'Analyst', 'Programmer Ana
lyst':'Analyst' }, inplace=True)
data['Suggested Job Role'].replace({'Applications Developer':'Developer','Database Develo
per':'Developer','Mobile Applications Developer':'Developer','Web Developer':'Developer',
'Software Developer':'Developer'}, inplace = True)
data['Suggested Job Role'].replace({'Quality Assurance Associate':'Quality Assurance','So
ftware Quality Assurance (QA) / Testing':'Quality Assurance'}, inplace=True)
data['Suggested Job Role'].replace({'Network Engineer':'Network Security', 'Network Secur
ity Administrator':'Network Security', 'Network Security Engineer':'Network Security','Po
rtal Administrator':'Network Security', 'Systems Security Administrator':'Network Securit
y'}, inplace=True)
data['Suggested Job Role'].replace({'UX Designer':'UX Designer','Design & UX':'UX Designe
r'}, inplace=True)
data['Suggested Job Role'].replace({'Software Engineer':'Software Engineer/Quality Assura
nce',  'Quality Assurance':'Software Engineer/Quality Assurance'}, inplace=True)
data['Suggested Job Role'].replace({'Technical Support':'Technical Support/Database Engin
eer',  'Database Engineer':'Technical Support/Database Engineer'}, inplace=True)
data['Suggested Job Role'].replace({'Project Manager':'Network Security/Project Manager',
'Network Security':'Network Security/Project Manager'}, inplace=True)
```

In [56]:

```
print('Total Classes after clubing: {}'.format(len(np.unique(data['Suggested Job Role']))
))
```

```
Total Classes after clubing: 5
```

In [70]:

```
Counter(data['Suggested Job Role'])
```

Out[70]:

```
Counter({'Developer/UX Designer': 4004,
         'Network Security/Project Manager': 4120,
         'Analyst': 3874,
         'Software Engineer/Quality Assurance': 4028,
         'Technical Support/Database Engineer': 3974})
```

Now we have clubbed the classes and reduced it to total 5 classes. Now classification would be done on these

basis.

In [102]:

```
print("Count plot after clubbing classes")
sns.set_style(style="darkgrid")
sns.set(rc={"figure.figsize":(15, 4)})
sns.countplot(x='Suggested Job Role',data=data)
```

```
Count plot after clubbing classes
```

Out[102]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x271c59d07c0>
```

Correlation:

In [105]:

```
data.corr()
```

Out[105]:

	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	P Co
Acedamic percentage in Operating Systems	1.000000	0.001781	-0.004693	0.010691	-0.001003	-0.010402	0.011958	-0.003203	
percentage in Algorithms	0.001781	1.000000	0.000914	0.004178	-0.000961	-0.004914	-0.003793	-0.007968	
Percentage in Programming Concepts	-0.004693	0.000914	1.000000	0.006810	0.001120	0.003585	-0.001093	-0.008326	
Percentage in Software Engineering	0.010691	0.004178	0.006810	1.000000	-0.009601	0.005680	0.000722	0.001932	
Percentage in Computer Networks	-0.001003	-0.000961	0.001120	-0.009601	1.000000	-0.002255	0.008840	0.003276	
Percentage in Electronics Subjects	-0.010402	-0.004914	0.003585	0.005680	-0.002255	1.000000	0.002406	-0.003306	
Percentage in Computer Architecture	0.011958	-0.003793	-0.001093	0.000722	0.008840	0.002406	1.000000	-0.002009	
Percentage in Mathematics	-0.003203	-0.007968	-0.008326	0.001932	0.003276	-0.003306	-0.002009	1.000000	
Percentage in Communication	-0.001770	-0.001485	-0.002105	0.015085	-0.008065	0.000635	-0.000076	0.003569	

	skills	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	P Co
Hours working per day	0.009920	0.011561	-0.008532	-0.007267	-0.001204	0.007186	0.000484	0.014630		
Logical quotient rating	0.004602	-0.013894	-0.010540	0.007988	-0.000585	0.015727	-0.000559	-0.005173		
hackathons	-0.006075	0.006679	-0.008081	-0.005637	0.003816	0.002428	0.005101	-0.007806		
coding skills rating	-0.001214	0.013466	-0.009414	-0.000148	0.004241	0.002663	-0.013613	0.001658		
public speaking points	0.003999	0.003912	-0.008243	-0.003006	0.009924	-0.008268	0.011289	-0.001312		

Preprocessing

In [13]:

```
# Setting the class label coloumn in y and remaining data to x
X = data.iloc[:,:-1]
y = data.iloc[:,-1]
```

In [14]:

```
# One hot encoding values so that they can be fed into ANN
from sklearn.preprocessing import OneHotEncoder
X1 = OneHotEncoder().fit_transform(X)
```

In [15]:

```
y1 = y.copy(deep=True)
```

In [57]:

```
#Checking for null values
data.isnull().sum()
```

Out[57]:

```
Acedamic percentage in Operating Systems    0
percentage in Algorithms                    0
Percentage in Programming Concepts          0
Percentage in Software Engineering          0
Percentage in Computer Networks             0
Percentage in Electronics Subjects          0
Percentage in Computer Architecture         0
Percentage in Mathematics                   0
Percentage in Communication skills          0
Hours working per day                       0
Logical quotient rating                     0
hackathons                                  0
coding skills rating                        0
public speaking points                      0
can work long time before system?           0
self-learning capability?                   0
Extra-courses did                           0
certifications                              0
workshops                                   0
talenttests taken?                          0
olympiads                                   0
reading and writing skills                  0
memory capability score                     0
Interested subjects                         0
interested career area                      0
Job/Higher Studies?                         0
Type of company want to settle in?          0
Taken inputs from seniors or elders         0
interested in games                         0
```

```
Interested Type of Books                0
Salary Range Expected                   0
In a Realtionship?                      0
Gentle or Tuff behaviour?               0
Management or Technical                 0
Salary/work                             0
hard/smart worker                       0
worked in teams ever?                   0
Introvert                               0
Suggested Job Role                      0
dtype: int64
```

In [58]:

```
#checking for null values in data set/ preprocessing
data.isnull().sum()
data.isnull()
```

Out[58]:

	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 Communicati ski
0	False	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	...	
19995	False	False	False	False	False	False	False	False	Fal
19996	False	False	False	False	False	False	False	False	Fal
19997	False	False	False	False	False	False	False	False	Fal
19998	False	False	False	False	False	False	False	False	Fal
19999	False	False	False	False	False	False	False	False	Fal

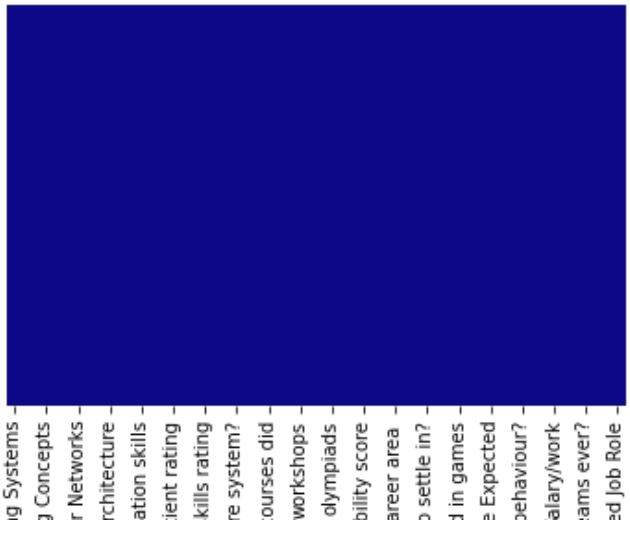
20000 rows × 39 columns

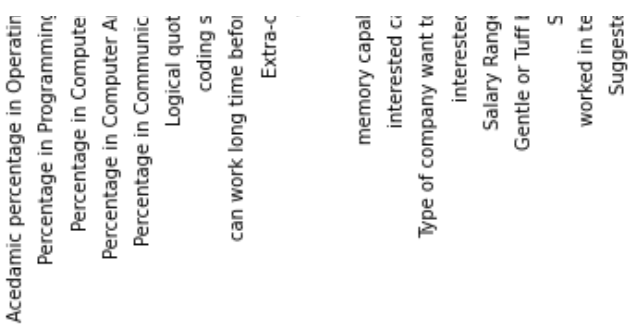
In [59]:

```
sns.heatmap(data.isnull(),yticklabels=False,cbar=False,cmap='plasma')
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x271bf42e1c0>
```





In [60]:

```
data = data.dropna(axis=0)  #data clean
[c for c in data.columns if data[c].isnull().sum()>0]
```

Out[60]:

```
[]
```

Now all the data cleaning, removal of null values, data scaling preprocessing is done.

Model Experiment

Experiment:1

In this we are making different models according to default parameters, parameters by grid search and with different activation functions having same train:test split. Then comparing their accuracy.

Firstly we are splitting the train test dataset in ratio 80:20

In [18]:

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1,y1,test_size=0.2)
```

Now we are applying the grid search so that we can select the best parameters

In [16]:

```
parameters = [{'random_state': [1],'max_iter': [5,10,20],'alpha': [0.001, 0.01,0.1],'act
ivation' : ['identity','logistic', 'tanh', 'relu'],'solver' : ['lbfgs', 'sgd', 'adam'],
            'hidden_layer_sizes': [(6,2),(2,2),(8,),]} ]
```

In [17]:

```
model = GridSearchCV(MLPClassifier(), parameters, scoring='accuracy',n_jobs=-1)
model.fit(X_train1,y_train1)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (20) reached and the
optimization hasn't converged yet.
  warnings.warn(
```

Out[17]:

```
GridSearchCV(estimator=MLPClassifier(), n_jobs=-1,
             param_grid=[{'activation': ['identity', 'logistic', 'tanh',
                                         'relu'],
                          'alpha': [0.001, 0.01, 0.1],
                          'hidden_layer_sizes': [(6, 2), (2, 2), (8,)],
                          'max_iter': [5, 10, 20], 'random_state': [1],
                          'solver': ['lbfgs', 'sgd', 'adam']}],
             scoring='accuracy')
```

```
            scoring='accuracy')
```

In [19]:

```
print("Best Parameters are: ")
model.best_params_
```

```
Best Parameters are:
```

Out[19]:

```
{'activation': 'relu',
 'alpha': 0.001,
 'hidden_layer_sizes': (6, 2),
 'max_iter': 20,
 'random_state': 1,
 'solver': 'adam'}
```

In [51]:

```
model0 = MLPClassifier(random_state=1, max_iter=10).fit(X_train1,y_train1)

model1 = MLPClassifier(activation='relu', hidden_layer_sizes = (6,2), solver = 'adam', r
andom_state=1, alpha= 0.001, max_iter= 20)
model1.fit(X_train1,y_train1)

model2 = MLPClassifier(activation='tanh', hidden_layer_sizes = (30,30), solver = 'sgd',
random_state=1, alpha= 0.0001, max_iter = 70)
model2.fit(X_train1,y_train1)

model3 = MLPClassifier(activation='relu', hidden_layer_sizes = (25,25), solver = 'lbfgs'
, random_state=1, alpha= 0.01, max_iter = 70)
model3.fit(X_train1,y_train1)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (10) reached and the
optimization hasn't converged yet.
  warnings.warn(
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (20) reached and the
optimization hasn't converged yet.
  warnings.warn(
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (70) reached and the
optimization hasn't converged yet.
  warnings.warn(
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:471: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

Out[51]:

```
MLPClassifier(alpha=0.01, hidden_layer_sizes=(25, 25), max_iter=70,
              random_state=1, solver='lbfgs')
```

Model0 : default parameters

Model1 : grid search parameters

Model2 : different parameters

Model3 : different parameters

Comparision Between training & testing accuracy of differnt models

In [50]:

```
output=pd.DataFrame(['Model0'],columns=['Model Name'])
output.loc[0,'Test Accuracy']=accuracy_score(model0.predict(X_train1),y_train1)
output.loc[0,'Train Accuracy']=accuracy_score(model0.predict(X_test1),y_test1)
output.loc[1,'Model Name']='Model1'
output.loc[1,'Test Accuracy']=accuracy_score(model1.predict(X_train1),y_train1)
output.loc[1,'Train Accuracy']=accuracy_score(model1.predict(X_test1),y_test1)
output.loc[2,'Model Name']='Model2'
output.loc[2,'Test Accuracy']=accuracy_score(model2.predict(X_train1),y_train1)
output.loc[2,'Train Accuracy']=accuracy_score(model2.predict(X_test1),y_test1)
output.loc[3,'Model Name']='Model3'
output.loc[3,'Test Accuracy']=accuracy_score(model3.predict(X_train1),y_train1)
output.loc[3,'Train Accuracy']=accuracy_score(model3.predict(X_test1),y_test1)
output
```

Out[50]:

	Model Name	Test Accuracy	Train Accuracy
0	Model0	0.612533	0.1960
1	Model1	0.235933	0.2038
2	Model2	0.236467	0.2036
3	Model3	0.458400	0.2016

Confusion Matrix and Classification Report

In [23]:

```
from sklearn.metrics import confusion_matrix, classification_report
print("For model0")
print("---------------------------------------")
print("Train Confusin Matrix")
m0ctm=confusion_matrix(model0.predict(X_train1),y_train1)
print(m0ctm)
print("Test Confusin Matrix")
m0ctem=confusion_matrix(model0.predict(X_test1),y_test1)
print(m0ctem)
print("---------------------------------------")
print("For model1")
print("---------------------------------------")
print("Train Confusin Matrix")
m1ctm=confusion_matrix(model1.predict(X_train1),y_train1)
print(m1ctm)
print("Test Confusin Matrix")
m1ctem=confusion_matrix(model1.predict(X_test1),y_test1)
print(m1ctem)
print("---------------------------------------")
print("For model2")
print("---------------------------------------")
print("Train Confusin Matrix")
m2ctm=confusion_matrix(model2.predict(X_train1),y_train1)
print(m2ctm)
print("Test Confusin Matrix")
m2ctem=confusion_matrix(model2.predict(X_test1),y_test1)
print(m2ctem)
print("---------------------------------------")
print("For model3")
print("---------------------------------------")
print("Train Confusin Matrix")
m3ctm=confusion_matrix(model3.predict(X_train1),y_train1)
print(m3ctm)
print("Test Confusin Matrix")
m3ctem=confusion_matrix(model3.predict(X_test1),y_test1)
print(m3ctem)
print("---------------------------------------")
```

```
For model0
```

```
--------------------------------------
Train Confusin Matrix
[[1764  293  235  277  298]
 [ 263 1846  276  284  241]
 [ 365  437 2018  433  451]
 [ 172  184  207 1745  172]
 [ 350  271  331  272 1815]]
Test Confusin Matrix
[[178 196 229 188 200]
 [195 204 220 211 187]
 [261 243 234 247 251]
 [153 129 157 149 144]
 [173 201 213 222 215]]
--------------------------------------
For model1
--------------------------------------
Train Confusin Matrix
[[   0    0    0    0    0]
 [ 105   66  137  129   27]
 [2335 2343 2684 2717 2161]
 [   0    0    0    0    0]
 [ 474  622  246  165  789]]
Test Confusin Matrix
[[  0   0   0   0   0]
 [ 34  39  43  36  24]
 [791 760 820 794 813]
 [  0   0   0   0   0]
 [135 174 190 187 160]]
--------------------------------------
For model2
--------------------------------------
Train Confusin Matrix
[[ 456  479  490  443  494]
 [ 853  903  834  882  895]
 [ 507  536  593  550  502]
 [1007 1025 1033 1060  980]
 [  91   88  117   76  106]]
Test Confusin Matrix
[[139 164 181 169 154]
 [286 268 288 254 289]
 [171 166 202 209 185]
 [321 347 345 347 342]
 [ 43  28  37  38  27]]
--------------------------------------
For model3
--------------------------------------
Train Confusin Matrix
[[1428 1192  986 1247 1175]
 [ 354  526  369  294  391]
 [ 407  601  805  558  504]
 [ 373  345  385  628  303]
 [ 352  367  522  284  604]]
Test Confusin Matrix
[[423 393 422 411 402]
 [134 132 143 114 138]
 [160 192 200 191 198]
 [126 116 141 136 121]
 [117 140 147 165 138]]
--------------------------------------
```

In [26]:

```
print("For model0")
print("--------------------------------------")
print("Classification Report (train)")
c1=classification_report(model0.predict(X_train1),y_train1)
print(c1)
print("--------------------------------------")
print("For model1")
print("--------------------------------------")
print("Classification Report (train)")
c2=classification_report(model1.predict(X_train1),y_train1)
```

```
print(c2)
print("--------------------------------------")
print("For model2")
print("--------------------------------------")
print("Classification Report (train)")
c3=classification_report(model2.predict(X_train1),y_train1)
print(c3)
print("--------------------------------------")
print("For model3")
print("--------------------------------------")
print("Classification Report (train)")
c4=classification_report(model3.predict(X_train1),y_train1)
print(c4)
print("--------------------------------------")
```

```
For model0
--------------------------------------
Classification Report (train)
                                    precision    recall  f1-score   support

                           Analyst       0.61      0.62      0.61      2867
             Developer/UX Designer       0.61      0.63      0.62      2910
   Network Security/Project Manager       0.66      0.54      0.60      3704
Software Engineer/Quality Assurance       0.58      0.70      0.64      2480
Technical Support/Database Engineer       0.61      0.60      0.60      3039

                          accuracy                           0.61     15000
                         macro avg       0.61      0.62      0.61     15000
                      weighted avg       0.62      0.61      0.61     15000

--------------------------------------
For model1
--------------------------------------
Classification Report (train)
                                    precision    recall  f1-score   support

                           Analyst       0.00      0.00      0.00         0
             Developer/UX Designer       0.02      0.14      0.04       464
   Network Security/Project Manager       0.88      0.22      0.35     12240
Software Engineer/Quality Assurance       0.00      0.00      0.00         0
Technical Support/Database Engineer       0.27      0.34      0.30      2296

                          accuracy                           0.24     15000
                         macro avg       0.23      0.14      0.14     15000
                      weighted avg       0.76      0.24      0.33     15000

--------------------------------------
For model2
--------------------------------------
Classification Report (train)
                                    precision    recall  f1-score   support

                           Analyst       0.16      0.19      0.17      2362
             Developer/UX Designer       0.30      0.21      0.24      4367
   Network Security/Project Manager       0.19      0.22      0.21      2688
Software Engineer/Quality Assurance       0.35      0.21      0.26      5105
Technical Support/Database Engineer       0.04      0.22      0.06       478

                          accuracy                           0.21     15000
                         macro avg       0.21      0.21      0.19     15000
                      weighted avg       0.27      0.21      0.23     15000

--------------------------------------
For model3
--------------------------------------
Classification Report (train)
                                    precision    recall  f1-score   support

                           Analyst       0.49      0.24      0.32      6028
             Developer/UX Designer       0.17      0.27      0.21      1934
   Network Security/Project Manager       0.26      0.28      0.27      2875
Software Engineer/Quality Assurance       0.21      0.31      0.25      2034
```

```
Technical Support/Database Engineer       0.20      0.28      0.24      2129

                           accuracy                           0.27     15000
                          macro avg       0.27      0.28      0.26     15000
                       weighted avg       0.33      0.27      0.27     15000

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

Classwise Accuracy

In [27]:

```
print("For model0 testing accuracy according to class")
print("--------------------------------------")
print(m0ctem.diagonal()/m0ctem.sum(axis=1))
print("--------------------------------------")
print("For model1 testing accuracy according to class")
print("--------------------------------------")
print(m1ctem.diagonal()/m1ctem.sum(axis=1))
print("--------------------------------------")
print("For model2 testing accuracy according to class")
print("--------------------------------------")
print(m2ctem.diagonal()/m2ctem.sum(axis=1))
print("--------------------------------------")
print("For model3 testing accuracy according to class")
print("--------------------------------------")
print(m3ctem.diagonal()/m3ctem.sum(axis=1))
print("--------------------------------------")
```

```
For model0 testing accuracy according to class
--------------------------------------
[0.17961655 0.20058997 0.18932039 0.20355191 0.20996094]
--------------------------------------
For model1 testing accuracy according to class
--------------------------------------
[       nan 0.22159091 0.20613374        nan 0.1891253 ]
--------------------------------------
For model2 testing accuracy according to class
--------------------------------------
[0.17224287 0.19350181 0.21650589 0.20387779 0.15606936]
--------------------------------------
For model3 testing accuracy according to class
--------------------------------------
[0.20624086 0.19969743 0.21253985 0.2125     0.19519095]
--------------------------------------
```

```
<ipython-input-27-dd3e5f6e0a3a>:7: RuntimeWarning: invalid value encountered in true_divi
de
  print(m1ctem.diagonal()/m1ctem.sum(axis=1))
```

In [28]:

```
print("For model0 training accuracy according to class")
print("--------------------------------------")
print(m0ctm.diagonal()/m0ctm.sum(axis=1))
print("--------------------------------------")
print("For model1 training accuracy according to class")
print("--------------------------------------")
print(m1ctm.diagonal()/m1ctm.sum(axis=1))
print("--------------------------------------")
print("For model2 training accuracy according to class")
print("--------------------------------------")
print(m2ctm.diagonal()/m2ctm.sum(axis=1))
print("--------------------------------------")
print("For model3 training accuracy according to class")
print("--------------------------------------")
print(m3ctm.diagonal()/m3ctm.sum(axis=1))
print("--------------------------------------")
```

```
For model0 training accuracy according to class
--------------------------------------
```

```
[0.61527729 0.63436426 0.54481641 0.70362903 0.59723593]
--------------------------------------
For model1 training accuracy according to class
--------------------------------------
[       nan 0.14224138 0.21928105        nan 0.34364111]
--------------------------------------
For model2 training accuracy according to class
--------------------------------------
[0.19305673 0.20677811 0.22061012 0.20763957 0.22175732]
--------------------------------------
For model3 training accuracy according to class
--------------------------------------
[0.23689449 0.27197518 0.28       0.30875123 0.28370127]
--------------------------------------
```

```
<ipython-input-28-684f3676d5af>:7: RuntimeWarning: invalid value encountered in true_divi
de
  print(m1ctm.diagonal()/m1ctm.sum(axis=1))
```

Experiment:2

In this we are making different models according to same parameters by grid search and with different train:test split. Then comparing their accuracy.

Standarizing dataset

In [19]:

```
from sklearn.preprocessing import StandardScaler
x1 = StandardScaler(with_mean=False).fit_transform(X1)
y1 = y.copy(deep=True)
```

In [71]:

```
parameters = [{'random_state': [1],'max_iter': [100,50,110],'alpha': [0.001, 0.01],'acti
vation' : ['identity','logistic', 'tanh', 'relu'],
              'solver' : ['lbfgs', 'sgd', 'adam'],'hidden_layer_sizes': [(60,60),(60,60,60
),(80,80)]}]
```

Application of Grid Search

In [72]:

```
modell = GridSearchCV(MLPClassifier(), parameters, scoring='accuracy',n_jobs=-1)
modell.fit(X_train1,y_train1)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (110) reached and the
optimization hasn't converged yet.
  warnings.warn(
```

Out[72]:

```
GridSearchCV(estimator=MLPClassifier(), n_jobs=-1,
             param_grid=[{'activation': ['identity', 'logistic', 'tanh',
                                         'relu'],
                          'alpha': [0.001, 0.01],
                          'hidden_layer_sizes': [(60, 60), (60, 60, 60),
                                                 (80, 80)],
                          'max_iter': [100, 50, 110], 'random_state': [1],
                          'solver': ['lbfgs', 'sgd', 'adam']}],
             scoring='accuracy')
```

In [74]:

```
print("Best Parameters are: ")
modell.best_params_
```

```
Best Parameters are:
```

Best Parameters are:

Out[74]:

```
{'activation': 'tanh',
 'alpha': 0.01,
 'hidden_layer_sizes': (60, 60),
 'max_iter': 110,
 'random_state': 1,
 'solver': 'adam'}
```

Model11: Train-Test split = 70:30

Model12: Train-Test split = 85:15

Model13: Train-Test split = 90:10

In [76]:

```
# For train-test split-> 70:30
X_train11, X_test11, y_train11, y_test11 = train_test_split(x1,y1,test_size=0.3)
```

In [77]:

```
# For train-test split-> 85:15
X_train12, X_test12, y_train12, y_test12 = train_test_split(x1,y1,test_size=0.15)
```

In [78]:

```
# For train-test split-> 90:10
X_train13, X_test13, y_train13, y_test13 = train_test_split(x1,y1,test_size=0.10)
```

In [79]:

```
model11 = MLPClassifier(activation='tanh', hidden_layer_sizes = (60,60), solver = 'adam'
, random_state=1, alpha= 0.01, max_iter= 110)
model11.fit(X_train11,y_train11)

model13 = MLPClassifier(activation='tanh', hidden_layer_sizes = (60,60), solver = 'adam'
, random_state=1, alpha= 0.01, max_iter = 110)
model13.fit(X_train12,y_train12)

model13 = MLPClassifier(activation='tanh', hidden_layer_sizes = (60,60), solver = 'adam'
, random_state=1, alpha= 0.001, max_iter = 110)
model13.fit(X_train13,y_train13)
```

Out[79]:

```
MLPClassifier(activation='tanh', alpha=0.001, hidden_layer_sizes=(60, 60),
              max_iter=110, random_state=1)
```

Comparision Between training & testing accuracy of differnt models

In [42]:

```
output1=pd.DataFrame(['Model0'],columns=['Model Name'])
output1.loc[1,'Model Name']='Model11'
output1.loc[1,'Test Accuracy']=accuracy_score(model11.predict(X_train11),y_train11)
output1.loc[1,'Train Accuracy']=accuracy_score(model11.predict(X_test11),y_test11)
output1.loc[2,'Model Name']='Model12'
output1.loc[2,'Test Accuracy']=accuracy_score(model12.predict(X_train12),y_train12)
output1.loc[2,'Train Accuracy']=accuracy_score(model12.predict(X_test12),y_test12)
output1.loc[3,'Model Name']='Model13'
output1.loc[3,'Test Accuracy']=accuracy_score(model13.predict(X_train13),y_train13)
output1.loc[3,'Train Accuracy']=accuracy_score(model13.predict(X_test13),y_test13)
output1
```

Out[42]:

	Model Name	Test Accuracy	Train Accuracy
0	Model0	NaN	NaN
1	Model11	0.302714	0.1950
2	Model12	0.879765	0.8840
3	Model13	0.480500	0.1995

Confusion Matrix and Classification Report

In [45]:

```
from sklearn.metrics import confusion_matrix, classification_report
print("-------------------------------------")
print("For model11")
print("-------------------------------------")
print("Train Confusin Matrix")
mm1ctm=confusion_matrix(model11.predict(X_train11),y_train11)
print(mm1ctm)
print("Test Confusin Matrix")
mm1ctem=confusion_matrix(model11.predict(X_test11),y_test11)
print(mm1ctem)
print("-------------------------------------")
print("For model12")
print("-------------------------------------")
print("Train Confusin Matrix")
mm2ctm=confusion_matrix(model12.predict(X_train12),y_train12)
print(mm2ctm)
print("Test Confusin Matrix")
mm2ctem=confusion_matrix(model12.predict(X_test12),y_test12)
print(mm2ctem)
print("-------------------------------------")
print("For model13")
print("-------------------------------------")
print("Train Confusin Matrix")
mm3ctm=confusion_matrix(model13.predict(X_train13),y_train13)
print(mm3ctm)
print("Test Confusin Matrix")
mm3ctem=confusion_matrix(model13.predict(X_test13),y_test13)
print(mm3ctem)
print("-------------------------------------")
```

```
-------------------------------------
For model11
-------------------------------------
Train Confusin Matrix
[[793 394 491 466 443]
 [497 887 522 519 495]
 [496 521 884 480 492]
 [518 506 543 908 568]
 [450 442 462 457 766]]
Test Confusin Matrix
[[213 262 240 247 224]
 [204 262 233 224 262]
 [229 221 221 242 256]
 [260 269 289 254 248]
 [214 240 235 231 220]]
-------------------------------------
For model12
-------------------------------------
Train Confusin Matrix
[[2908  102  105   92   97]
 [  84 3010  108   99   93]
 [  95  113 3074  108  106]
 [  99   95  112 3028  110]
 [ 116   96  105  109 2936]]
Test Confusin Matrix
[[501  11  21  12  19]
 [ 15 508  18  13  16]
```

```
 [ 19  28 547  27  14]
 [ 18  17  16 527  14]
 [ 19  24  14  13 569]]
--------------------------------------
For model13
--------------------------------------
Train Confusin Matrix
[[1483  405  497  372  430]
 [ 426 1582  495  392  404]
 [ 580  563 1636  528  393]
 [ 466  573  615 1981  364]
 [ 530  499  489  330 1967]]
Test Confusin Matrix
[[ 73  63  65  83  77]
 [ 76  67  78  85  72]
 [ 77  79  73  76  80]
 [ 75 100  83 100 101]
 [ 88  73  89  81  86]]
--------------------------------------
```

In [46]:

```
print("--------------------------------------")
print("For model11")
print("--------------------------------------")
print("Classification Report (train)")
cc2=classification_report(model11.predict(X_train11),y_train11)
print(cc2)
print("--------------------------------------")
print("For model12")
print("--------------------------------------")
print("Classification Report (train)")
cc3=classification_report(model12.predict(X_train12),y_train12)
print(cc3)
print("--------------------------------------")
print("For model13")
print("--------------------------------------")
print("Classification Report (train)")
cc4=classification_report(model13.predict(X_train13),y_train13)
print(cc4)
print("--------------------------------------")
```

```
--------------------------------------
For model11
--------------------------------------
Classification Report (train)
                                    precision    recall  f1-score   support

                           Analyst       0.29      0.31      0.30      2587
             Developer/UX Designer       0.32      0.30      0.31      2920
   Network Security/Project Manager       0.30      0.31      0.31      2873
Software Engineer/Quality Assurance       0.32      0.30      0.31      3043
Technical Support/Database Engineer       0.28      0.30      0.29      2577

                          accuracy                           0.30     14000
                         macro avg       0.30      0.30      0.30     14000
                      weighted avg       0.30      0.30      0.30     14000

--------------------------------------
For model12
--------------------------------------
Classification Report (train)
                                    precision    recall  f1-score   support

                           Analyst       0.88      0.88      0.88      3304
             Developer/UX Designer       0.88      0.89      0.88      3394
   Network Security/Project Manager       0.88      0.88      0.88      3496
Software Engineer/Quality Assurance       0.88      0.88      0.88      3444
Technical Support/Database Engineer       0.88      0.87      0.88      3362

                          accuracy                           0.88     17000
                         macro avg       0.88      0.88      0.88     17000
```

```
                    weighted avg       0.88      0.88      0.88     17000

-------------------------------------
For model13
-------------------------------------
Classification Report (train)
                                    precision    recall  f1-score   support

                           Analyst       0.43      0.47      0.44      3187
             Developer/UX Designer       0.44      0.48      0.46      3299
   Network Security/Project Manager       0.44      0.44      0.44      3700
Software Engineer/Quality Assurance       0.55      0.50      0.52      3999
Technical Support/Database Engineer       0.55      0.52      0.53      3815

                          accuracy                           0.48     18000
                         macro avg       0.48      0.48      0.48     18000
                      weighted avg       0.48      0.48      0.48     18000

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

Classwise Accuracy

In [47]:

```
print("-------------------------------------")
print("For model11 testing accuracy according to class")
print("-------------------------------------")
print(mm1ctem.diagonal()/mm1ctem.sum(axis=1))
print("-------------------------------------")
print("For model12 testing accuracy according to class")
print("-------------------------------------")
print(mm2ctem.diagonal()/mm2ctem.sum(axis=1))
print("-------------------------------------")
print("For model13 testing accuracy according to class")
print("-------------------------------------")
print(mm3ctem.diagonal()/mm3ctem.sum(axis=1))
print("-------------------------------------")
```

```
-------------------------------------
For model11 testing accuracy according to class
-------------------------------------
[0.17959528 0.22109705 0.18905047 0.19242424 0.19298246]
-------------------------------------
For model12 testing accuracy according to class
-------------------------------------
[0.88829787 0.89122807 0.86141732 0.8902027  0.89045383]
-------------------------------------
For model13 testing accuracy according to class
-------------------------------------
[0.20221607 0.17724868 0.18961039 0.21786492 0.20623501]
-------------------------------------
```

In [48]:

```
print("-------------------------------------")
print("For model11 training accuracy according to class")
print("-------------------------------------")
print(mm1ctem.diagonal()/mm1ctem.sum(axis=1))
print("-------------------------------------")
print("For model12 training accuracy according to class")
print("-------------------------------------")
print(mm2ctem.diagonal()/mm2ctem.sum(axis=1))
print("-------------------------------------")
print("For model13 training accuracy according to class")
print("-------------------------------------")
print(mm3ctem.diagonal()/mm3ctem.sum(axis=1))
print("-------------------------------------")
```

```
-------------------------------------
For model11 training accuracy according to class
```

```
--------------------------------------
[0.17959528 0.22109705 0.18905047 0.19242424 0.19298246]
--------------------------------------
For model12 training accuracy according to class
--------------------------------------
[0.88829787 0.89122807 0.86141732 0.8902027  0.89045383]
--------------------------------------
For model13 training accuracy according to class
--------------------------------------
[0.20221607 0.17724868 0.18961039 0.21786492 0.20623501]
--------------------------------------
```

Experiment:3

In this we are making different models according to the activation function, iterations, alpha and solver which gave better accuracy along with selecting test-train split 85:15 which performed well in previous experiment. So by shuffeling these parameters we are trying to find better model. Then comparing their accuracy.

In [82]:

```
# For train-test split-> 85:15
X_train, X_test, y_train, y_test = train_test_split(x1,y1,test_size=0.15)
```

In [66]:

```
model111 = MLPClassifier(activation='relu', hidden_layer_sizes = (60,60,60), solver = 'adam', alpha= 0.001, max_iter = 100)
model111.fit(X_train,y_train)
model222 = MLPClassifier(activation='relu', hidden_layer_sizes = (80,80,60), solver = 'adam', alpha= 0.001, max_iter = 100)
model222.fit(X_train,y_train)
model333 = MLPClassifier(activation='relu', hidden_layer_sizes = (60,30,80), solver = 'adam', alpha= 0.001, max_iter = 100)
model333.fit(X_train,y_train)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[66]:

```
MLPClassifier(alpha=0.001, hidden_layer_sizes=(60, 30, 80), max_iter=100)
```

In [73]:

```
output2=pd.DataFrame(['Model0'],columns=['Model Name'])
output2.loc[0,'Model Name']='Model111'
output2.loc[0,'Test Accuracy']=accuracy_score(model111.predict(X_train),y_train)
output2.loc[0,'Train Accuracy']=accuracy_score(model111.predict(X_test),y_test)
output2.loc[1,'Model Name']='Model222'
output2.loc[1,'Test Accuracy']=accuracy_score(model222.predict(X_train),y_train)
output2.loc[1,'Train Accuracy']=accuracy_score(model222.predict(X_test),y_test)
output2.loc[2,'Model Name']='Model333'
output2.loc[2,'Test Accuracy']=accuracy_score(model333.predict(X_train),y_train)
output2.loc[2,'Train Accuracy']=accuracy_score(model333.predict(X_test),y_test)
output2
```

Out[73]:

	Model Name	Test Accuracy	Train Accuracy
0	Model111	0.876832	0.826833
1	Model222	0.879983	0.853240
2	Model333	0.792696	0.758734

So from here we can see that model performs better with the parameters:

activation='relu'

hidden_layer_sizes = (80,80,60)

solver = 'adam'

alpha= 0.001

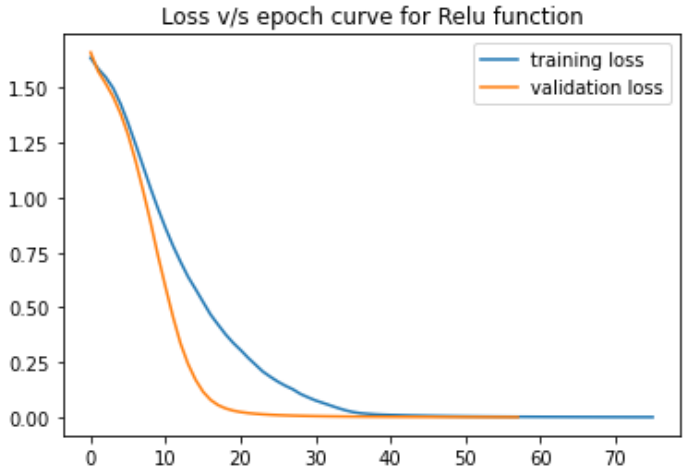
max_iter = 100

Train-Test Split = 85:15

Plotting curve

In [84]:

```
#Relu function
model = MLPClassifier(hidden_layer_sizes=(60, 60, 60),activation='relu',max_iter=100)
model1=model.fit(X_train, y_train)
loss_values1 = model1.loss_curve_
model2=model.fit(X_test, y_test)
loss_values2 = model2.loss_curve_
plt.title("Loss v/s epoch curve for Relu function")
plt.plot(loss_values1)
plt.plot(loss_values2)
plt.legend(['training loss','validation loss'])
plt.show()
```



From the above plot we can see that the training and validation loss are decreasing gradually with the number of epoches in the case of ReLu activation function. Training and validation Losses are minimum in the case of ReLu as compared to other activation function therefore we can say that Relu is the best activation function for the case of given dataset.

In [ ]: