In [108]:

```
import pandas as pd
import numpy as np
from sklearn import decomposition
import matplotlib.pyplot as plt
```

In [109]:

```
data = pd.read_csv("roo1_data.csv")
```

In [110]:

```
y= data.iloc[:,-1].values
```

In [74]:

```


data.head()
```

Out[74]:

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

5 rows × 39 columns

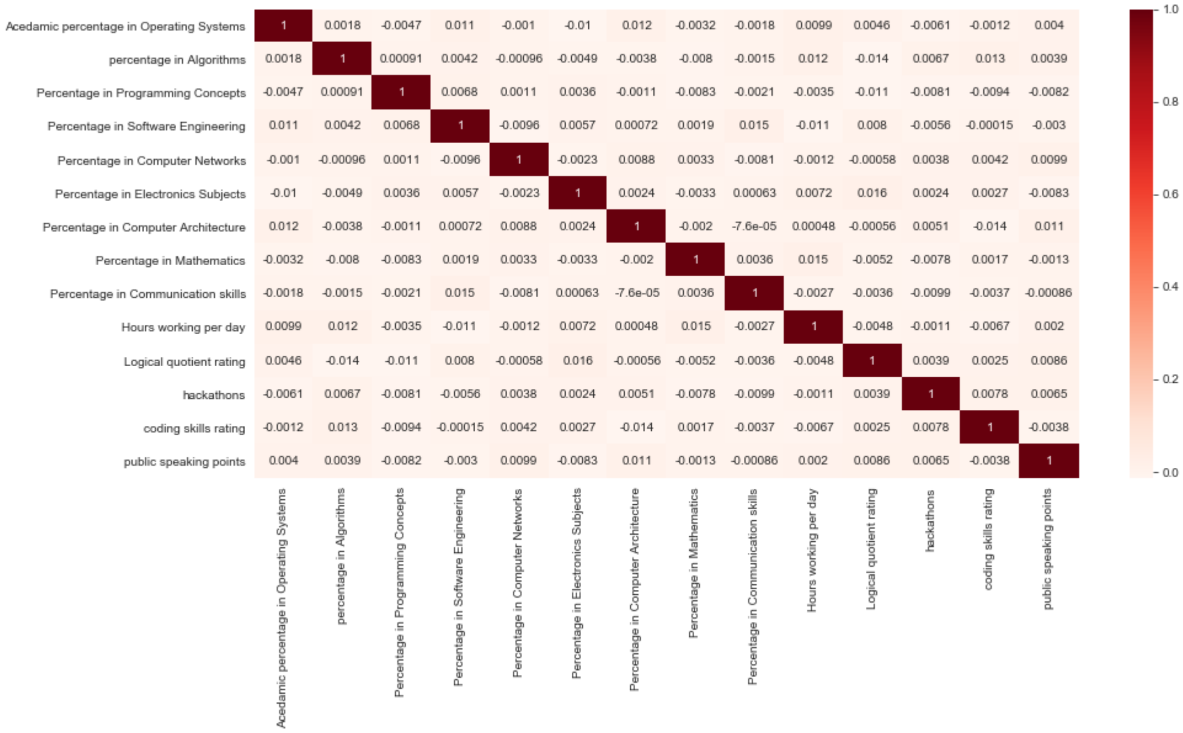
In [75]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style('whitegrid')
pd.options.display.float_format='{:.3f}'.format
```

Co-relation heat Map

In [76]:

```
corrMatrix=data.corr()
plt.figure(figsize=(15,7))
sns.heatmap(corrMatrix,annot=True,cmap='Reds')
plt.show()
```



PCA

In [77]:

```
from sklearn.decomposition import PCA
```

In [78]:

```
pca_set = PCA(n_components=2)
```

In [88]:

```
Xdata = data.iloc[:,:-1].values
```

In [89]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [90]:

```
labelencoder = LabelEncoder()
```

In [91]:

```
for i in range(14,38):
    Xdata[:,i] = labelencoder.fit_transform(Xdata[:,i])
```

In [92]:

```
component_set = pca_set.fit_transform(Xdata)
```

In [93]:

```
component_set
```

Out[93]:

```
array([[  6.92455735,  15.51628793],
       [ -9.64836184,  -1.08584709],
       [ 22.83910281, -12.37537294],
       ...,
       [ 11.28384035, -15.06111604],
       [  6.34199002,   2.98779088],
       [ -1.00708864, -15.33193353]])
```

In [ ]:

```

```

In [94]:

```
X1=data[['Acedamic percentage in Operating Systems','percentage in Algorithms','Perc
```

In [95]:

```
component_set = pca_set.fit_transform(X1)
```

In [96]:

```
principal_set = pd.DataFrame(data = component_set
             , columns = ['p1', 'p2'])
```

In [97]:

```
principal_set.head()
```

Out[97]:

	p1	p2
0	6.465	15.272
1	-9.834	-1.440
2	23.364	-11.777
3	-13.178	-2.327
4	8.394	8.728

In [98]:

```
final_pca = pd.concat([principal_set, data[['Suggested Job Role']]], axis = 1)
```

In [99]:

```
final_pca.head()
```

Out[99]:

	p1	p2	Suggested Job Role
0	6.465	15.272	Database Developer
1	-9.834	-1.440	Portal Administrator
2	23.364	-11.777	Portal Administrator
3	-13.178	-2.327	Systems Security Administrator
4	8.394	8.728	Business Systems Analyst

PCA visualization 2D projection

In [100]:

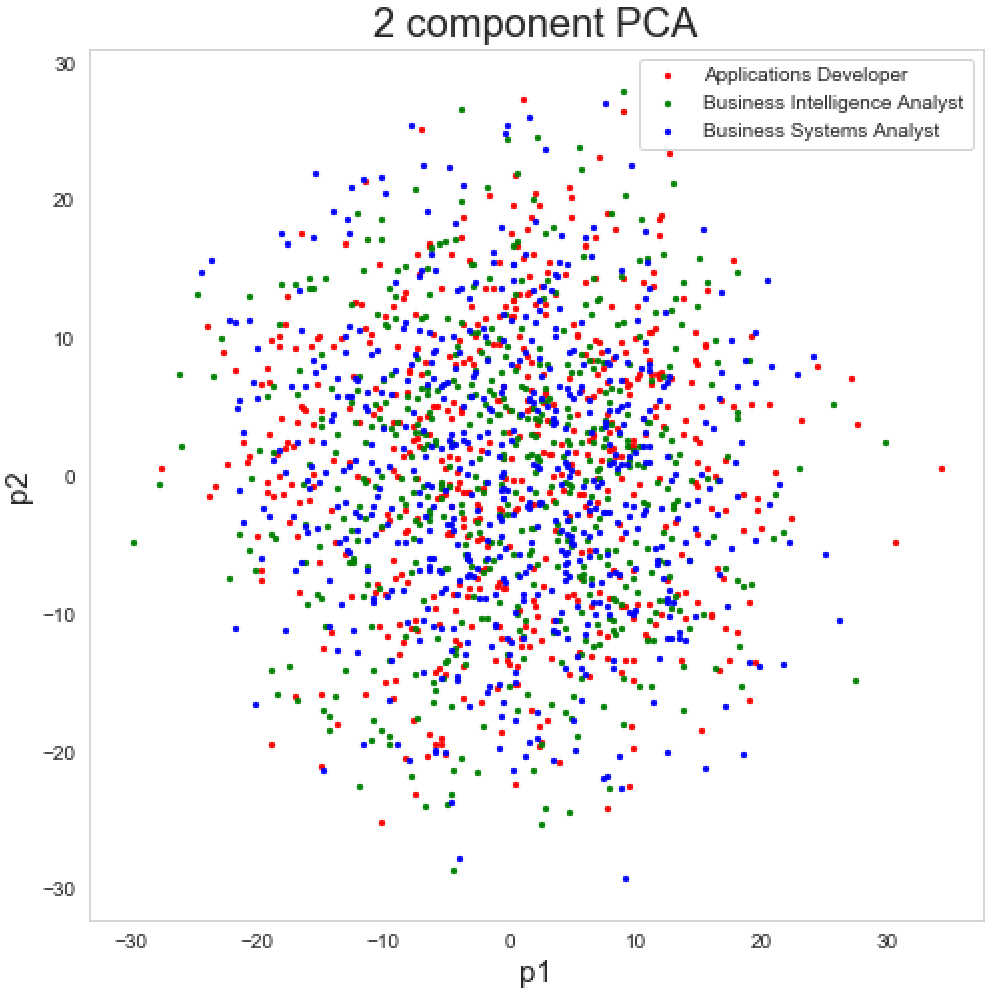
```
fig = plt.figure(figsize = (8,8))
```

```
<Figure size 576x576 with 0 Axes>
```

In [101]:

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('p1', fontsize = 15)
ax.set_ylabel('p2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)


targets = ['Applications Developer', 'Business Intelligence Analyst', 'Business Syst
colors = ['r', 'g', 'b','rb','br']
for y, color in zip(targets,colors):
    indicesToKeep = final_pca['Suggested Job Role'] == y
    ax.scatter(final_pca.loc[indicesToKeep, 'p1']
               , final_pca.loc[indicesToKeep, 'p2']
               , c = color
               , s = 5)
ax.legend(targets)
ax.grid()
```



In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```

```

In [ ]:

```

```

Dummies Variable

In [59]:

```
dummiesC = pd.get_dummies(data['certifications'])
```

In [60]:

```
dummiesC.head()
```

Out[60]:

	app development	distro making	full stack	hadoop	information security	machine learning	python	r programming	s programr
0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	1	0	0	
2	1	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	1	0	
4	1	0	0	0	0	0	0	0	

In [61]:

```
X = pd.DataFrame(data)
```

In [62]:

```
X1=data[['Acedamic percentage in Operating Systems','percentage in Algorithms','Perc
```

In [63]:

```
X1.head()
```

Out[63]:

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

In [64]:

```
df_dummies= pd.concat([X1,dummiesC],axis='columns')
```

In [65]:

```
df_dummies.head()
```

Out[65]:

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

5 rows × 23 columns

In [111]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [112]:

```
labelencoder_X = LabelEncoder()
```

In [116]:

```
y1 = labelencoder_X.fit_transform(y)
```

In [117]:

```
y1 = pd.DataFrame(y1)
```

In [ ]:

```

```

Model Fitting

In [118]:

```
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
```

In [119]:

```
X_train,X_test,y_train,y_test=train_test_split(df_dummies,y1,test_size=0.2,random_st
```

Decision tree classifier

In [120]:

```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
```

In [121]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [122]:

```
y_pred = clf.predict(X_test)
```

In [123]:

```
y_pred
```

Out[123]:

```
array([11,  2,  8, ..., 29, 15, 21])
```

In [124]:

```
DA=clf.score(df_dummies,y1)
DA
```

Out[124]:

```
0.8049
```

XGBClassifier

In [125]:

```
from xgboost import XGBClassifier
```

In [126]:

```
model = XGBClassifier()
model.fit(X_train, y_train)
```

```
C:\Users\Ashok\Anaconda3\lib\site-packages\sklearn\preprocessing\labe
l.py:219: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), f
or example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Ashok\Anaconda3\lib\site-packages\sklearn\preprocessing\labe
l.py:252: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), f
or example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[126]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=
0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=10
0,
       n_jobs=1, nthread=None, objective='multi:softprob', random_stat
e=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

In [127]:

```
y_pred = model.predict(X_test)
```

In [128]:

```
DXgbA=model.score(df_dummies,y1)
DXgbA
```

Out[128]:

```
0.1495
```

SVM

In [129]:

```
from sklearn import svm
```

In [130]:

```
clf = svm.SVC()
clf.fit(X_train, y_train)
```

```
C:\Users\Ashok\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:761: DataConversionWarning: A column-vector y was passed when a 1d a
rray was expected. Please change the shape of y to (n_samples, ), for
example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Ashok\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: Fu
tureWarning: The default value of gamma will change from 'auto' to 'sc
ale' in version 0.22 to account better for unscaled features. Set gamm
a explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[130]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [131]:

```
y_pred = clf.predict(X_test)
```

In [132]:

```
SA=clf.score(df_dummies,y1)
SA
```

Out[132]:

```
0.8112
```

RANDOM forestRegreesor

In [133]:

```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=20, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

```
C:\Users\Ashok\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: Da
taConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example us
ing ravel().
  after removing the cwd from sys.path.
```

In [134]:

```
RFRA=regressor.score(df_dummies,y1)
RFRA
```

Out[134]:

```
0.6528025688186774
```

RANDOM FOREST CLASSIFIER

In [135]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [136]:

```
 from sklearn.datasets import make_classification
```

In [137]:

```
clf = RandomForestClassifier()
```

In [138]:

```
clf.fit(X_train, y_train)
```

```
C:\Users\Ashok\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:
246: FutureWarning: The default value of n_estimators will change from
10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Ashok\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Da
taConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example us
ing ravel().
  """Entry point for launching an IPython kernel.
```

Out[138]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='g
ini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=Non
e,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [139]:

```
y_pred = clf.predict(X_test)
```

In [140]:

```
RFACA=clf.score(df_dummies,y1)
```

In [141]:

```
 X1, y1 = make_classification(n_samples=1000, n_features=10,
                            n_informative=2, n_redundant=0,
                           random_state=0, shuffle=False)
```

In [142]:

```
clf = RandomForestClassifier(n_estimators=100, max_depth=2,
                             random_state=0)
clf.fit(X1, y1)
```

Out[142]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='g
ini',
            max_depth=2, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=Non
e,
            oob_score=False, random_state=0, verbose=0, warm_start=Fal
se)
```

In [143]:

```
print(clf.feature_importances_)
```

```
[0.10709928 0.5484963  0.00847003 0.06325648 0.03568049 0.04204675
 0.03366965 0.0590075  0.05193304 0.05034048]
```

In [ ]:

In [ ]:

In [144]:

```
X1[0]
```

Out[144]:

```
array([-1.66853167, -1.29901346,  0.2746472 , -0.60362044,  0.7088595
8,
        0.42281857, -3.11685659,  0.64445203, -1.91374267,  0.6635615
8])
```

In [145]:

```
y1[0]
```

Out[145]:

```
0
```

In [146]:

```
clf.predict([[-1.66853167, -1.29901346,  0.2746472 , -0.60362044,  0.70885958,
        0.42281857, -3.11685659,  0.64445203, -1.91374267,  0.66356158]])
```

Out[146]:

```
array([0])
```

In [147]:

```
X2=pd.DataFrame(X1)
```

In [148]:

```
X2.head()
```

Out[148]:

	0	1	2	3	4	5	6	7	8	9
0	-1.669	-1.299	0.275	-0.604	0.709	0.423	-3.117	0.644	-1.914	0.664
1	-2.973	-1.089	-0.154	1.194	-0.098	-0.887	-0.147	1.060	0.026	-0.114
2	-0.596	-1.370	0.744	0.210	-0.006	1.366	1.555	0.613	-0.286	1.497
3	-1.069	-1.175	1.183	0.719	-1.216	0.141	-0.744	-0.159	0.240	0.100
4	-1.305	-0.966	-0.475	1.273	-1.696	0.730	-1.857	0.383	-0.887	0.878

Final Accuracy

Y_pred=clf.predict(X2)

In [150]:

```

Y_pred=clf.predict(X2)
```

In [151]:

```
finalA=clf.score(X2,Y_pred)
```

In [152]:

```
finalA
```

Out[152]:

```
1.0
```

Accuracy Comparision

all acuracies

In [155]:

```
print(DA)
print(SA)
print(RFRA)
print(RFACA)
print(DXgbA)
print(finalA)
DecisionTree_A=DA
SVM_A=SA
Regressor_A=RFRA
RandomForestClassifier_A=RFACA
XgbClassifier=DXgbA
```

```
0.8049
0.8112
0.6528025688186774
0.8032
0.1495
1.0
```

In [ ]:

```
```