**INTELLIGENT CAREER PREDICTION SYSTEM**

**ABSTRACT:**

* **Intelligent Career Prediction System** will enable a more effective way to choose the career and to know the profession that will perfectly suit a person based on the skill and interest.
* System will rank the CGPA, no. of mini projects, no. of projects, core-sub skill, aptitude skill, problem solving skill, programming skill, abstract thinking ability, ds\_coding, technology used, symposium attended,

symposium won, extracurricular activities, college performance, college skill and other key skills and suggest the profession that match with the skills of the candidate ,the job opportunities in that profession

* This system will help especially the students/candidates to select right job that matches the skill which helps to excel in the Career.
* This project can also be applied in a company to short list the candidates.
* This system will focus not only in qualification and experience but also focuses on other important aspects which are required for particular job position.
* This system will help the HR department to easily shortlist the candidate.

**Problem Statement:**

1. Finding the profession in which a person possesses talent, interest is a challenging task in deciding Career. Due to lack of self realization and misguidance (others cannot correctly understand the talent, interest of a person) even talented person fail to choose a right Career.
2. Many Multinational Companies spend a lot of time and Human efforts in short listing Candidates for a particular job position.

**Existing Solution:**

* Students get general guidance from experts which may or may not suit an individual.
* Interview panel spend a lot of time to shortlist Candidates manually.
* There is no proper existing system.

**Objective:**

* To Develop a system which helps an individual to choose a right profession based on the person’s interest and talent required for that profession and make aware of job opportunities and average salary in that profession
* To make Recruitment process easy by making use of the Intelligent Career Prediction system to Shortlist Candidates.

**Proposed Methodology:**

* In Intelligent Career Prediction System we get the candidates details.
* We train our System with data
* Now the system is ready to predict the career of the Candidates.
* From the evaluation of the Candidate details,the system identifies in which field the Candidate is interested and in which field the Candidate is really talented and skilled.
* Now, Our Trained system will predict the Candidate to choose the right profession that suits the best.
* It also gives information about available job opportunities and average salary in that profession and the scope in that profession.

|  |
| --- |
|  |

**Technology stack:**

**Build Phase**

* Creating dataset
* Handling missing values
* Splitting data into train and test datasets
* Training random forest classifier with Python scikit learn
* Random Forest Classifier is the most accurate learning algorithms available.
* For many data sets, it produces a highly accurate **classifier**
* It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

**Operational Phase**

* Perform predictions
* Accuracy calculations
* Train Accuracy
* Test Accuracy
* Use Web Browser API to suggest the available jobs in that Profession.

**Features and Scope of the Project:**

* Candidate choose right profession
* Candidate can be updated with available job opportunities
* Identify the key Skill and strength of the Candidate
* Help Interview panel to save time, effort.
* This can be employed in IT industries and other industries to Shortlist right Candidates, in Colleges for Student’s Career guidance.

**Career predict**

|  |  |
| --- | --- |
| **Skills and Talents** | **Rate of interest for candidates** |
| CGPA | 3 |
| No.of mini projects | 1 |
| No.of projects | 1 |
| Core-sub\_skill | 2 |
| Aptitude skill | 4 |
| Problem solving skill | 2 |
| Programming skill | 3 |
| abstract thinking | 3 |
| ds\_coding | 3 |
| technology used | 1 |
| symposium attended | 3 |
| symposium won | 1 |
| extracurricular activities | 2 |
| college performance | 4 |
| college skills | 3 |

**Conclusion and future work:**

* This project is very useful to the Student community and MNC’s.
* In future this project can be developed as a mobile application to make it easy to access.
* In future, the app automatically gives information, once new vacancies are found in that profession and with the permission of the user it applies for the jobs.
* MNC can use this software to shortlist right Candidates without much Effort thereby saving time and money.
* In future, we can upload the details of all the candidates in a folder and give folder as input and get the shortlisted Candidates rather than giving each one by bye in Recruitment process.

**OUTPUT OBTAINED:**

Anaconda Prompt-python

(base) C:\Users\user>python

Python 3.6.6 |Anaconda 4.3.1 (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import pandas as pd

>>> from sklearn.model\_selection import train\_test\_split

>>> from sklearn.ensemble import RandomForestClassifier

>>> from sklearn.metrics import accuracy\_score

>>> from sklearn.metrics import confusion\_matrix

>>> INPUT\_PATH = "C:/Users/user/Desktop/hcl.data"

>>> OUTPUT\_PATH = "C:/Users/user/Desktop/hcl.csv"

>>> def data\_file\_to\_csv():

... """

... :return:

... """

... headers=['sslc', 'hsc', 'cgpa', 'school\_type', 'no\_of\_miniprojects',

... 'problemsolving\_skill', 'programming\_skill', 'abstractthink\_skill',

... 'design\_skill', 'first\_computer', 'first\_program', 'lab\_programs',

... 'ds\_coding', 'technology\_used', 'sympos\_attend', 'sympos\_won',

... 'extracurricular', 'learning\_style', 'college\_bench',

... 'clg\_eachers\_know', 'college\_performence ', 'college\_skills', 'ROLE']

... dataset = read\_data(INPUT\_PATH)

... dataset = add\_headers(dataset, headers)

... dataset.to\_csv(OUTPUT\_PATH, index=False)

... print ("File saved ...!")

...

>>> def add\_headers(dataset, headers):

... """

... Add the headers to the dataset

... :param dataset:

... :param headers:

... :return:

... """

... dataset.columns = headers

... return dataset

...

>>> def dataset\_statistics(dataset):

... """

... Basic statistics of the dataset

... :param dataset: Pandas dataframe

... :return: None, print the basic statistics of the dataset

... """

... print (dataset.describe())

...

>>> def main():

... """

... Main function

... :return:

... """

... dataset = pd.read\_csv(OUTPUT\_PATH)

... dataset\_statistics(dataset)

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

sslc hsc cgpa school\_type no\_of\_miniprojects \

count 7525.000000 7525.000000 7525.000000 7525.000000 7525.000000

mean 2.907243 3.043056 2.138738 1.799070 1.537276

std 0.939084 0.900414 0.803465 0.711248 0.735916

min 1.000000 1.000000 1.000000 1.000000 1.000000

25% 2.000000 3.000000 2.000000 1.000000 1.000000

50% 3.000000 3.000000 2.000000 2.000000 1.000000

75% 4.000000 4.000000 3.000000 2.000000 2.000000

max 4.000000 4.000000 4.000000 4.000000 4.000000

no\_of\_projects coresub\_skill aptitude\_skill problemsolving\_skill \

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 1.188439 1.513223 2.041196 2.863256

std 0.545728 0.499858 1.085477 0.728428

min 1.000000 1.000000 1.000000 1.000000

25% 1.000000 1.000000 1.000000 2.000000

50% 1.000000 2.000000 2.000000 3.000000

75% 1.000000 2.000000 3.000000 3.000000

max 4.000000 2.000000 4.000000 4.000000

programming\_skill ... lab\_programs ds\_coding \

count 7525.000000 ... 7525.000000 7525.000000

mean 2.887442 ... 3.584585 2.050498

std 0.754894 ... 0.790142 0.748276

min 1.000000 ... 1.000000 1.000000

25% 2.000000 ... 4.000000 2.000000

50% 3.000000 ... 4.000000 2.000000

75% 3.000000 ... 4.000000 2.000000

max 5.000000 ... 4.000000 4.000000

technology\_used sympos\_attend sympos\_won extracurricular \

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 1.204917 2.856877 1.972359 2.182990

std 0.574120 1.027328 1.002539 1.094488

min 1.000000 1.000000 1.000000 1.000000

25% 1.000000 2.000000 1.000000 1.000000

50% 1.000000 3.000000 2.000000 2.000000

75% 1.000000 4.000000 3.000000 3.000000

max 4.000000 4.000000 5.000000 4.000000

college\_bench clg\_teachers\_know college\_performence college\_skills

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 3.988040 3.597874 3.350299 2.995748

std 0.778578 0.565860 0.522819 1.409500

min 1.000000 1.000000 1.000000 1.000000

25% 4.000000 3.000000 3.000000 2.000000

50% 4.000000 4.000000 3.000000 3.000000

75% 4.000000 4.000000 4.000000 4.000000

max 5.000000 4.000000 4.000000 5.000000

[8 rows x 24 columns]

>>> def handel\_missing\_values(dataset, missing\_values\_header, missing\_label):

... """

... Filter missing values from the dataset

... :param dataset:

... :param missing\_values\_header:

... :param missing\_label:

... :return:

... """

... return dataset[dataset[missing\_values\_header] != missing\_label]

...

>>> def main():

... """

... Main function

... :return:

... """

... headers=['sslc', 'hsc', 'cgpa', 'school\_type', 'no\_of\_miniprojects',

... 'problemsolving\_skill', 'programming\_skill', 'abstractthink\_skill',

... 'design\_skill', 'first\_computer', 'first\_program', 'lab\_programs',

... 'ds\_coding', 'technology\_used', 'sympos\_attend', 'sympos\_won',

... 'extracurricular', 'learning\_style', 'college\_bench',

... 'clg\_eachers\_know', 'college\_performence ', 'college\_skills', 'ROLE']

... dataset = pd.read\_csv(OUTPUT\_PATH)

... dataset\_statistics(dataset)

... dataset = handel\_missing\_values(dataset, headers[22], '?')

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

sslc hsc cgpa school\_type no\_of\_miniprojects \

count 7525.000000 7525.000000 7525.000000 7525.000000 7525.000000

mean 2.907243 3.043056 2.138738 1.799070 1.537276

std 0.939084 0.900414 0.803465 0.711248 0.735916

min 1.000000 1.000000 1.000000 1.000000 1.000000

25% 2.000000 3.000000 2.000000 1.000000 1.000000

50% 3.000000 3.000000 2.000000 2.000000 1.000000

75% 4.000000 4.000000 3.000000 2.000000 2.000000

max 4.000000 4.000000 4.000000 4.000000 4.000000

no\_of\_projects coresub\_skill aptitude\_skill problemsolving\_skill \

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 1.188439 1.513223 2.041196 2.863256

std 0.545728 0.499858 1.085477 0.728428

min 1.000000 1.000000 1.000000 1.000000

25% 1.000000 1.000000 1.000000 2.000000

50% 1.000000 2.000000 2.000000 3.000000

75% 1.000000 2.000000 3.000000 3.000000

max 4.000000 2.000000 4.000000 4.000000

programming\_skill ... lab\_programs ds\_coding \

count 7525.000000 ... 7525.000000 7525.000000

mean 2.887442 ... 3.584585 2.050498

std 0.754894 ... 0.790142 0.748276

min 1.000000 ... 1.000000 1.000000

25% 2.000000 ... 4.000000 2.000000

50% 3.000000 ... 4.000000 2.000000

75% 3.000000 ... 4.000000 2.000000

max 5.000000 ... 4.000000 4.000000

technology\_used sympos\_attend sympos\_won extracurricular \

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 1.204917 2.856877 1.972359 2.182990

std 0.574120 1.027328 1.002539 1.094488

min 1.000000 1.000000 1.000000 1.000000

25% 1.000000 2.000000 1.000000 1.000000

50% 1.000000 3.000000 2.000000 2.000000

75% 1.000000 4.000000 3.000000 3.000000

max 4.000000 4.000000 5.000000 4.000000

college\_bench clg\_teachers\_know college\_performence college\_skills

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 3.988040 3.597874 3.350299 2.995748

std 0.778578 0.565860 0.522819 1.409500

min 1.000000 1.000000 1.000000 1.000000

25% 4.000000 3.000000 3.000000 2.000000

50% 4.000000 4.000000 3.000000 3.000000

75% 4.000000 4.000000 4.000000 4.000000

max 5.000000 4.000000 4.000000 5.000000

[8 rows x 24 columns]

>>> def split\_dataset(dataset, train\_percentage, feature\_headers, target\_header):

... """

... Split the dataset with train\_percentage

... :param dataset:

... :param train\_percentage:

... :param feature\_headers:

... :param target\_headers:

... :return: train\_x, test\_x, train\_y, test\_y

... """

... train\_x, test\_x, train\_y, test\_y = train\_test\_split(dataset[feature\_headers], dataset[target\_header],train\_size=train\_percentage)

... return train\_x, test\_x, train\_y, test\_y

...

>>> def main():

... """

... Main function

... :return:

... """

... headers=['sslc', 'hsc', 'cgpa', 'school\_type', 'no\_of\_miniprojects',

... 'problemsolving\_skill', 'programming\_skill', 'abstractthink\_skill',

... 'design\_skill', 'first\_computer', 'first\_program', 'lab\_programs',

... 'ds\_coding', 'technology\_used', 'sympos\_attend', 'sympos\_won',

... 'extracurricular', 'learning\_style', 'college\_bench',

... 'clg\_teachers\_know', 'college\_performence ', 'college\_skills', 'ROLE']

... dataset = pd.read\_csv(OUTPUT\_PATH)

... dataset\_statistics(dataset)

... dataset = handel\_missing\_values(dataset, headers[22], '?')

... train\_x, test\_x, train\_y, test\_y = split\_dataset(dataset, 0.7, headers[1:-1], headers[-1])

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

sslc hsc cgpa school\_type no\_of\_miniprojects \

count 7525.000000 7525.000000 7525.000000 7525.000000 7525.000000

mean 2.907243 3.043056 2.138738 1.799070 1.537276

std 0.939084 0.900414 0.803465 0.711248 0.735916

min 1.000000 1.000000 1.000000 1.000000 1.000000

25% 2.000000 3.000000 2.000000 1.000000 1.000000

50% 3.000000 3.000000 2.000000 2.000000 1.000000

75% 4.000000 4.000000 3.000000 2.000000 2.000000

max 4.000000 4.000000 4.000000 4.000000 4.000000

no\_of\_projects coresub\_skill aptitude\_skill problemsolving\_skill \

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 1.188439 1.513223 2.041196 2.863256

std 0.545728 0.499858 1.085477 0.728428

min 1.000000 1.000000 1.000000 1.000000

25% 1.000000 1.000000 1.000000 2.000000

50% 1.000000 2.000000 2.000000 3.000000

75% 1.000000 2.000000 3.000000 3.000000

max 4.000000 2.000000 4.000000 4.000000

programming\_skill ... lab\_programs ds\_coding \

count 7525.000000 ... 7525.000000 7525.000000

mean 2.887442 ... 3.584585 2.050498

std 0.754894 ... 0.790142 0.748276

min 1.000000 ... 1.000000 1.000000

25% 2.000000 ... 4.000000 2.000000

50% 3.000000 ... 4.000000 2.000000

75% 3.000000 ... 4.000000 2.000000

max 5.000000 ... 4.000000 4.000000

technology\_used sympos\_attend sympos\_won extracurricular \

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 1.204917 2.856877 1.972359 2.182990

std 0.574120 1.027328 1.002539 1.094488

min 1.000000 1.000000 1.000000 1.000000

25% 1.000000 2.000000 1.000000 1.000000

50% 1.000000 3.000000 2.000000 2.000000

75% 1.000000 4.000000 3.000000 3.000000

max 4.000000 4.000000 5.000000 4.000000

college\_bench clg\_teachers\_know college\_performence college\_skills

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 3.988040 3.597874 3.350299 2.995748

std 0.778578 0.565860 0.522819 1.409500

min 1.000000 1.000000 1.000000 1.000000

25% 4.000000 3.000000 3.000000 2.000000

50% 4.000000 4.000000 3.000000 3.000000

75% 4.000000 4.000000 4.000000 4.000000

max 5.000000 4.000000 4.000000 5.000000

[8 rows x 24 columns]

>>> def main():

... """

... Main function

... :return:

... """

... dataset = pd.read\_csv(OUTPUT\_PATH)

... dataset\_statistics(dataset)

... dataset = handel\_missing\_values(dataset, headers[22], '?')

... train\_x, test\_x, train\_y, test\_y = split\_dataset(dataset, 0.7, headers[1:-1], headers[-1])

...

>>> def main():

... """

... Main function

... :return:

... """

... headers=['sslc', 'hsc', 'cgpa', 'school\_type', 'no\_of\_miniprojects',

... 'problemsolving\_skill', 'programming\_skill', 'abstractthink\_skill',

... 'design\_skill', 'first\_computer', 'first\_program', 'lab\_programs',

... 'ds\_coding', 'technology\_used', 'sympos\_attend', 'sympos\_won',

... 'extracurricular', 'learning\_style', 'college\_bench',

... 'clg\_teachers\_know', 'college\_performence ', 'college\_skills', 'ROLE']

... dataset = pd.read\_csv(OUTPUT\_PATH)

... dataset\_statistics(dataset)

... dataset = handel\_missing\_values(dataset, headers[22], '?')

... train\_x, test\_x, train\_y, test\_y = split\_dataset(dataset, 0.7, headers[1:-1], headers[-1])

... print("Train\_x Shape :: ",train\_x.shape)

... print("Train\_y Shape :: ",train\_y.shape)

... print("Test\_x Shape :: ",test\_x.shape)

... print("Test\_y Shape :: ",test\_y.shape)

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

sslc hsc cgpa school\_type no\_of\_miniprojects \

count 7525.000000 7525.000000 7525.000000 7525.000000 7525.000000

mean 2.907243 3.043056 2.138738 1.799070 1.537276

std 0.939084 0.900414 0.803465 0.711248 0.735916

min 1.000000 1.000000 1.000000 1.000000 1.000000

25% 2.000000 3.000000 2.000000 1.000000 1.000000

50% 3.000000 3.000000 2.000000 2.000000 1.000000

75% 4.000000 4.000000 3.000000 2.000000 2.000000

max 4.000000 4.000000 4.000000 4.000000 4.000000

no\_of\_projects coresub\_skill aptitude\_skill problemsolving\_skill \

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 1.188439 1.513223 2.041196 2.863256

std 0.545728 0.499858 1.085477 0.728428

min 1.000000 1.000000 1.000000 1.000000

25% 1.000000 1.000000 1.000000 2.000000

50% 1.000000 2.000000 2.000000 3.000000

75% 1.000000 2.000000 3.000000 3.000000

max 4.000000 2.000000 4.000000 4.000000

programming\_skill ... lab\_programs ds\_coding \

count 7525.000000 ... 7525.000000 7525.000000

mean 2.887442 ... 3.584585 2.050498

std 0.754894 ... 0.790142 0.748276

min 1.000000 ... 1.000000 1.000000

25% 2.000000 ... 4.000000 2.000000

50% 3.000000 ... 4.000000 2.000000

75% 3.000000 ... 4.000000 2.000000

max 5.000000 ... 4.000000 4.000000

technology\_used sympos\_attend sympos\_won extracurricular \

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 1.204917 2.856877 1.972359 2.182990

std 0.574120 1.027328 1.002539 1.094488

min 1.000000 1.000000 1.000000 1.000000

25% 1.000000 2.000000 1.000000 1.000000

50% 1.000000 3.000000 2.000000 2.000000

75% 1.000000 4.000000 3.000000 3.000000

max 4.000000 4.000000 5.000000 4.000000

college\_bench clg\_teachers\_know college\_performence college\_skills

count 7525.000000 7525.000000 7525.000000 7525.000000

mean 3.988040 3.597874 3.350299 2.995748

std 0.778578 0.565860 0.522819 1.409500

min 1.000000 1.000000 1.000000 1.000000

25% 4.000000 3.000000 3.000000 2.000000

50% 4.000000 4.000000 3.000000 3.000000

75% 4.000000 4.000000 4.000000 4.000000

max 5.000000 4.000000 4.000000 5.000000

[8 rows x 24 columns]

Train\_x Shape :: (5267, 21)

Train\_y Shape :: (5267,)

Test\_x Shape :: (2258, 21)

Test\_y Shape :: (2258,)

>>> def random\_forest\_classifier(features, target):

... """

... To train the random forest classifier with features and target data

... :param features:

... :param target:

... :return: trained random forest classifier

... """

... rfc = RandomForestClassifier()

... rfc.fit(train\_x,train\_y)

... return rfc

…

>>> def main():

... """

... Main function

... :return:

... """

... headers=['sslc', 'hsc', 'cgpa', 'school\_type', 'no\_of\_miniprojects',

... 'problemsolving\_skill', 'programming\_skill', 'abstractthink\_skill',

... 'design\_skill', 'first\_computer', 'first\_program', 'lab\_programs',

... 'ds\_coding', 'technology\_used', 'sympos\_attend', 'sympos\_won',

... 'extracurricular', 'learning\_style', 'college\_bench',

... 'clg\_teachers\_know', 'college\_performence ', 'college\_skills', 'ROLE']

... dataset = pd.read\_csv(OUTPUT\_PATH)

... dataset = handel\_missing\_values(dataset, headers[22], '?')

... train\_x, test\_x, train\_y, test\_y = split\_dataset(dataset, 0.7, headers[1:-1], headers[-1])

... trained\_model = RandomForestClassifier(train\_x,train\_y)

... print ("Trained model :: ",trained\_model)

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Trained model :: RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion=5598 NaN

3596 NaN

2019 NaN

4132 NaN

7371 NaN

6304 NaN

6775 NaN

6219 NaN

5110 NaN

7433 NaN

4592 NaN

4793 NaN

2140 NaN

798 NaN

1822 NaN

6663 NaN

4322 NaN

2543 NaN

623 NaN

7476 NaN

5736 NaN

276 NaN

5087 NaN

6345 NaN

61...aN

289 NaN

2533 NaN

5752 NaN

2712 NaN

6920 NaN

6905 NaN

Name: ROLE, dtype: object,

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators= hsc cgpa school\_type no\_of\_miniprojects problemsolving\_skill \

5598 2 2 1 1 3

3596 3 2 1 1 3

2019 1 1 2 2 ...5 3 4 4 1

[5267 rows x 21 columns],

n\_jobs=1, oob\_score=False, random\_state=None, verbose=0,

warm\_start=False)

>>> le = preprocessing.LabelEncoder()

>>> le.fit(["Technical Support", " Software Developer", " UI/UX Designer", " Technical Support", " Data Analyst", " Technical Writer", " Web Developer", "Technical Support", " Technical Writer", "SoftwareTester", "Software Developer", "Technical Support", "Web Developer", "Software Developer", " Software Developer", " Web Developer", " Technical Writer", "Software Developer", "Technical Support", " Technical Writer", " Technical Support", " Business Analyst", "Web Developer", "Software Developer", "Technical Writer", "UI/UX Designer", "Software Developer", "Web Developer", "Business Analyst", "Web Developer", "Technical Support", "Software Developer", "Technical Writer", "Business Analyst", "UI/UX Designer", "Software Developer", "Software Tester","Software Developer", "Data Analyst", "Technical Support", "Software Developer", "Software Tester", "Business Analyst", "Web Developer", "Technical Support", "Software Tester", "TechnicalSupport", "Data Analyst", "Software Developer"])

LabelEncoder()

>>> list(le.classes\_)

[' Business Analyst', ' Data Analyst', ' Software Developer', ' Technical Support', ' Technical Writer', ' UI/UX Designer', ' Web Developer', 'Business Analyst', 'Data Analyst', 'Software Developer', 'Software Tester', 'SoftwareTester', 'Technical Support', 'Technical Writer', 'TechnicalSupport', 'UI/UX Designer', 'Web Developer']

>>> le.transform(["Technical Support", " Software Developer", " UI/UX Designer", " Technical Support", " Data Analyst", " Technical Writer", " Web Developer", "Technical Support", " Technical Writer", "Software Tester", "Software Developer", "Technical Support", "Web Developer", "Software Developer", " Software Developer", " Web Developer", " Technical Writer", "Software Developer", "Technical Support", " Technical Writer", " Technical Support", " Business Analyst", "Web Developer", "Software Developer", "Technical Writer", "UI/UX Designer", "Software Developer", "Web Developer", "Business Analyst", "Web Developer", "Technical Support", "Software Developer", "Technical Writer", "Business Analyst", "UI/UX Designer", "Software Developer", "Software Tester", "Software Developer", "Data Analyst", "Technical Support", "Software Developer", "Software Tester", "Business Analyst", "Web Developer", "Technical Support", "Software Tester", "Technical Support", "Data Analyst", "Software Developer"])

array([12, 2, 5, 3, 1, 4, 6, 12, 4, 10, 9, 12, 16, 9, 2, 6, 4,

9, 12, 4, 3, 0, 16, 9, 13, 15, 9, 16, 7, 16, 12, 9, 13, 7,

15, 9, 10, 9, 8, 12, 9, 10, 7, 16, 12, 10, 12, 8, 9], dtype=int64)

>>> list(le.inverse\_transform([12, 2, 5, 3, 1, 4, 6, 12, 4, 10, 9, 12, 16, 9, 2, 6, 4,9, 12, 4, 3, 0, 16, 9, 13, 15, 9, 16, 7, 16, 12, 9, 13, 7,15, 9, 10, 9, 8, 12, 9, 10, 7, 16, 12, 10, 12, 8, 9]))

['Technical Support', ' Software Developer', ' UI/UX Designer', ' Technical Support', ' Data Analyst', ' Technical Writer', ' Web Developer', 'Technical Support', ' Technical Writer', 'Software Tester', 'Software Developer', 'Technical Support', 'Web Developer', 'Software Developer', ' Software Developer', ' Web Developer', ' Technical Writer', 'Software Developer', 'Technical Support', ' Technical Writer', ' Technical Support', ' Business Analyst', 'Web Developer', 'Software Developer', 'Technical Writer', 'UI/UX Designer', 'Software Developer', 'Web Developer', 'Business Analyst', 'Web Developer', 'Technical Support', 'Software Developer', 'Technical Writer', 'Business Analyst', 'UI/UX Designer', 'Software Developer', 'Software Tester', 'Software Developer', 'Data Analyst', 'Technical Support', 'Software Developer', 'Software Tester', 'Business Analyst', 'Web Developer', 'Technical Support', 'Software Tester', 'Technical Support', 'Data Analyst', 'Software Developer']

>>> def main():

... """

... Main function

... :return:

... """

... headers=['sslc', 'hsc', 'cgpa', 'school\_type', 'no\_of\_miniprojects',

... 'problemsolving\_skill', 'programming\_skill', 'abstractthink\_skill',

... 'design\_skill', 'first\_computer', 'first\_program', 'lab\_programs',

... 'ds\_coding', 'technology\_used', 'sympos\_attend', 'sympos\_won',

... 'extracurricular', 'learning\_style', 'college\_bench',

... 'clg\_teachers\_know', 'college\_performence ', 'college\_skills', 'ROLE']

... dataset = pd.read\_csv(OUTPUT\_PATH)

... dataset = handel\_missing\_values(dataset, headers[22], '?')

... dataset.iloc[0:49]

... dataset.ROLE=[12, 2, 5, 3, 1, 4, 6, 12, 4, 10, 9, 12, 16, 9, 2, 6, 4,9, 12, 4, 3, 0, 16, 9, 13, 15, 9, 16, 7, 16, 12, 9, 13, 7,15, 9, 10, 9, 8, 12, 9, 10, 7, 16, 12, 10, 12, 8, 9]

... train\_x, test\_x, train\_y, test\_y = split\_dataset(dataset, 0.7, headers[1:-1], headers[-1])

... trained\_model = RandomForestClassifier(train\_x,train\_y)

... print ("Trained model :: ",trained\_model)

... rfc = RandomForestClassifier()

... rfc.fit(train\_x,train\_y)

... return rfc

... predictions = trained\_model.predict(test\_x)

... for i in xrange (0,5):

... print ("Actual outcome :: {} and Predicted outcome :: {}".format(list(test\_y)[i], predictions[i]))

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Trained model :: RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion=32 13

6 6

20 3

23 9

41 10

8 4

9 10

38 8

10 9

42 7

11 12

48 9

26 9

46 12

28 7

22 16

5 4

39 12

25 15

4 1

34 15

44 12

19 4

37 9

12 16

18 12

14 2

21 0

30 12

31 9

36 10

43 16

27 16

24 13

Name: ROLE, dtype: int64,

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators= hsc cgpa school\_type no\_of\_miniprojects problemsolving\_skill \

32 4 3 3 1 2

6 3 2 2 2 3

20 2 2 2 1 4 ... 5

24 3 3 3

[34 rows x 21 columns],

n\_jobs=1, oob\_score=False, random\_state=None, verbose=0,

warm\_start=False)

RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='gini',

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=10, n\_jobs=1, oob\_score=False, random\_state=None,

verbose=0, warm\_start=False)

Actual outcome :: 2 and Predicted outcome :: 2

Actual outcome :: 2 and Predicted outcome :: 2

Actual outcome :: 2 and Predicted outcome :: 2

Actual outcome :: 2 and Predicted outcome :: 2

Actual outcome :: 4 and Predicted outcome :: 4

>>> def main():

…     """

…     Main function

…     :return:

…     """

…  dataset = pd.read\_csv(OUTPUT\_PATH)

…  dataset = handel\_missing\_values(dataset,headers[22], '?')

…   train\_x, test\_x, train\_y, test\_y = split\_dataset(dataset, 0.7, headers[1:-1], headers[-1])

…   trained\_model = random\_forest\_classifier(train\_x, train\_y)

…   print "Trained model :: ", trained\_model

…   predictions = trained\_model.predict(test\_x)

…   print "Train Accuracy :: ", accuracy\_score(train\_y, trained\_model.predict(train\_x))

…   print "Test Accuracy  :: ", accuracy\_score(test\_y, predictions)

…

>>> if \_\_name\_\_ == "\_\_main\_\_":

…  main()

Train Accuracy ::  0.991614255765

Test Accuracy  ::  0.970731707317

>>> import webbrowser

… webbrowser.open("https://www.google.co.in/search?ei=LA51W\_\_4D9CsrQHIwp\_wDg&q=job+opportunities+for+softw

are+engineers+passed+out&oq=job+opportunities+for+software+engineers+passed+out&gs\_l=psy-

ab.3...9862.17571.0.20361.0.0.0.0.0.0.0.0..0.0....0...1c.1.64.psy-

ab..0.0.0....0.3o079rtVjIc&ibp=htl;jobs&sa=X&ved=2ahUKEwi57L-

K7vDcAhXLeX0KHQOSCi4QiYsCKAF6BAgFEB8#fpstate=tldetail&htidocid=v1YpM8KFvPGgB-

IlAAAAAA%3D%3D&htivrt=jobs")

