OBJECTIVE:

- It is often difficult for an HR department to identify which employees are most likely to leave the company and for what reasons; this Employee Attrition Analysis Project aims to give you a way to leverage machine learning to better understand your employees.
- Employees are leaving the workforce faster than they are hired, and it is often outside the employer's control.
- Some of reason of attrition including the **lack of professional growth**, a **hostile work environment**, or **declining confidence in the company's market value**. **Weak leadership** is another factor that often drives attrition among employees.
- Attrition measures how many people left a company/office/department compared to the average number of people employed in that year. This takes into account fresh hires as well.

INTRODUCTION

- We have moved within the past month from an environment of historically tight global labor markets to one of widespread **unemployment** due to the **COVID-19** pandemic.
- Employee attrition analytics will remain important to organizations seeking to retain top talent; even in times of high unemployment, top performers are always in demand and there is competition for their talents.
- Employee attrition analytics is specifically focused on identifying why employees voluntarily leave, what might have prevented them from leaving, and how we can use data to predict attrition risk.

REQUIREMENT SPECIFICATION

To be used efficiently, all computer software needs certain hardware component or the other software resource to be present on a computer.

Types Of Requirements:

1. HARDWARE REQUIREMENTS
2. SOFTWARE REQUIREMENTS

1. HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware.

HARDWARE REQUIREMENTS FOR PRESENT PROJECT:

PROCESSOR : Intel Core i3

RAM : 4 GB

HARD DISK : 1 TB

2. SOFTWARE REQUIREMENTS:

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application.

SOFTWARE REQUIREMENTS FOR PRESENT PROJECT:

OPERATING SYSTEM : Windows 10

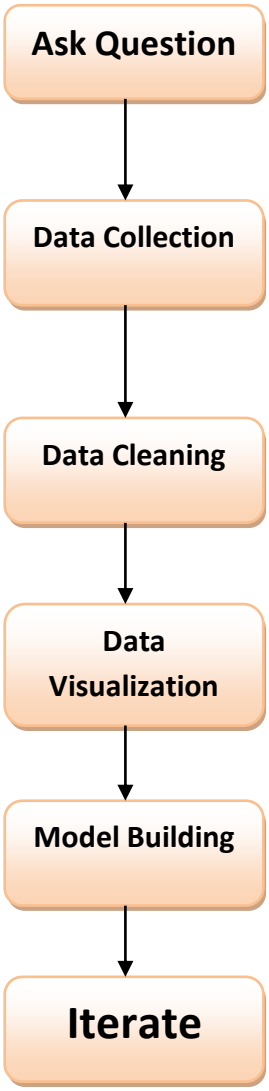
SOFTWARE : Anaconda(Jupyter notebook)

LANGUAGES : Python (Machine Learning)

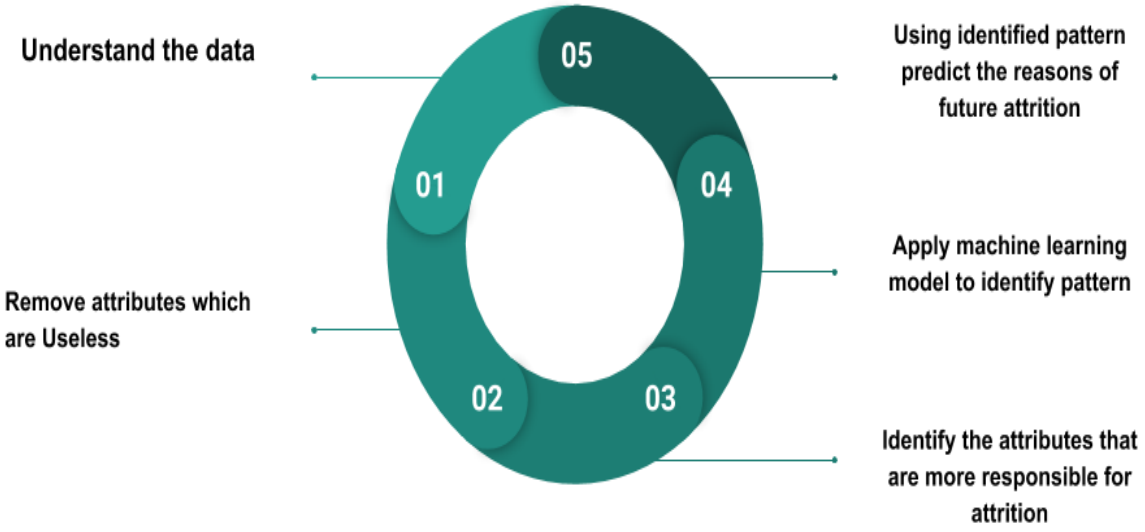
PROJECT DESIGN DETAILS

1. Defining Goal
2. Get the Data
3. Clean the Data
4. Enrich the Data
5. Deploy Machine Learning
6. Iterate

Class Diagram:



Flow of Employee Attrition Analysis:



ADVANTAGES:

- Most of the work we do in the field of people analytics is oriented to helping organizations understand what is most important to their employees, with the goal of making improvements to increase employee engagement and productivity, and reduce unwanted attrition.
- Most importantly, this type of employee predictive analytics can be used to help organizations understand and design the interventions that will be most effective in reducing unwanted attrition.
- It brings to fore the cause of employee disengagement.
- Enables HR managers develop long-term strategies to reduce attrition.
- Competitive measures to enhance company brand image.
- Develops and shapes drills that benefit both the management and the employees.
- Enhanced work culture.

PROJECT IMPLEMENTATION :

1. Ask Question: Who many employees are likely to leave Company and what are factors responsible for that?

Step 1:- Import all libraries which are required.

- **Numpy**:- NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

- **Pandas**:- Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

- **Seaborn:-** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- **Matplotlib:- Matplotlib** is a plotting library for the **Python** programming language and its numerical mathematics extension NumPy.

- **Sklearn:-Scikit-learn** (formerly scikits. learn and also known as **sklearn**) is a free software machine learning library for the **Python** programming language.

Code:-

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as plt
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
```

2. Data Collection:- Get the Data Set.

Step 2:- Fetch the data.

Code:-

```
df = pd.read_csv("F:\Datasets\EmployeeAttrition.csv")
df
```

output:

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Relations
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061	...	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062	...	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064	...	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065	...	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068	...	

1470 rows × 35 columns

Step3:- for understanding data print first 7 rows.

Code:-

```
df.head(7)
```

output:

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Relationship
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	
5	32	No	Travel_Frequently	1005	Research & Development	2	2	Life Sciences	1	8	...	
6	59	No	Travel_Rarely	1324	Research & Development	3	3	Medical	1	10	...	

7 rows × 35 columns

Step 4:-

Code:-

```
df.shape
```

output:

```
(1470, 35)
```

Step 5:- data type of each attribute.

Code:-

```
df.dtypes
```

output:

```
In [5]: df.dtypes

Out[5]: Age                         int64
        Attrition                  object
        BusinessTravel             object
        DailyRate                   int64
        Department                 object
        DistanceFromHome            int64
        Education                   int64
        EducationField             object
        EmployeeCount               int64
        EmployeeNumber              int64
        EnvironmentSatisfaction     int64
        Gender                     object
        HourlyRate                  int64
        JobInvolvement              int64
        JobLevel                    int64
        JobRole                    object
        JobSatisfaction             int64
        MaritalStatus              object
        MonthlyIncome               int64
        MonthlyRate                 int64
        NumCompaniesWorked          int64
        Over18                     object
        OverTime                   object
        PercentSalaryHike           int64
        PerformanceRating           int64
        RelationshipSatisfaction    int64
        StandardHours               int64
        StockOptionLevel            int64
        TotalWorkingYears           int64
        TrainingTimesLastYear       int64
        WorkLifeBalance             int64
        YearsAtCompany              int64
        YearsInCurrentRole          int64
```

3. Data Cleaning:-

Step 6:- check of any null value

Code:-

```
df.isnull().values.any()
```

output:

```
False
```

Step7:- view some statistics

Code:-

```
df.describe()
```

output:

Out[11]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	

8 rows × 26 columns

Step8:- get the count of no of employees that stayed and left the company.

Code:-

```
df['Attrition'].value_counts()
```

output:

```
Out[12]: No     1233
         Yes     237
         Name: Attrition, dtype: int64
```

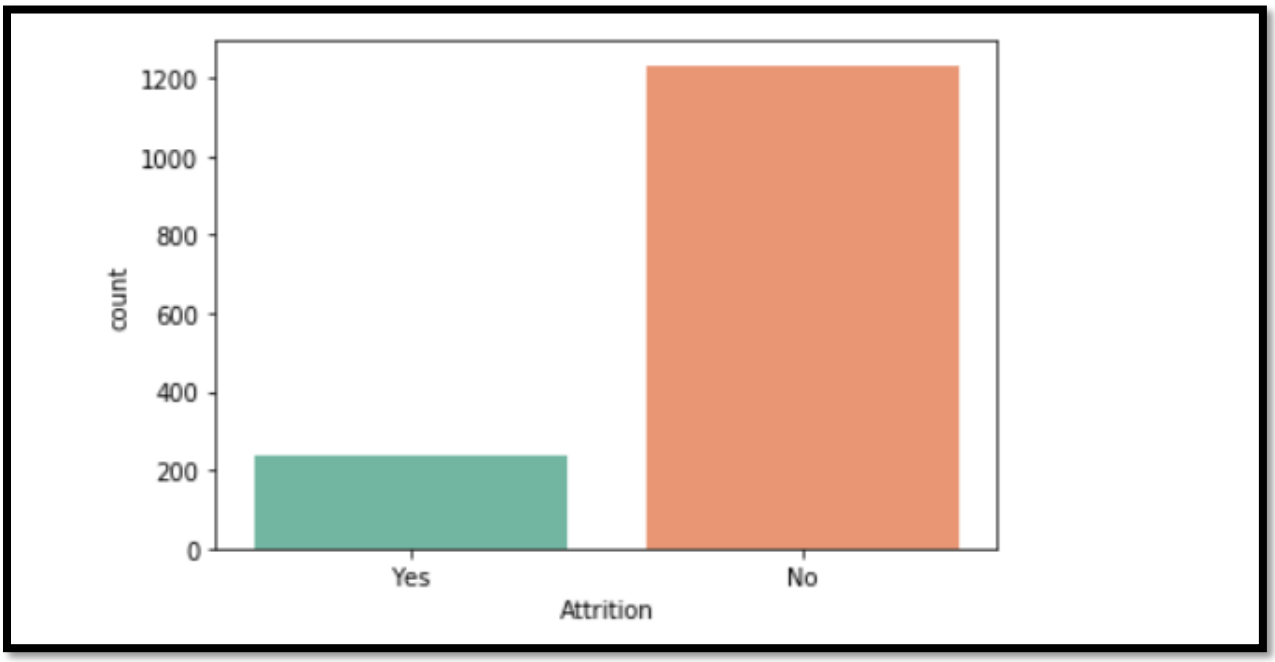
4. Visualization:-

Step9:- visualize

Code:-

```
gp = sns.countplot(x="Attrition", data=df, palette="Set2")
```

output:



Step10:-visualize attributes which are more responsible for attrition.

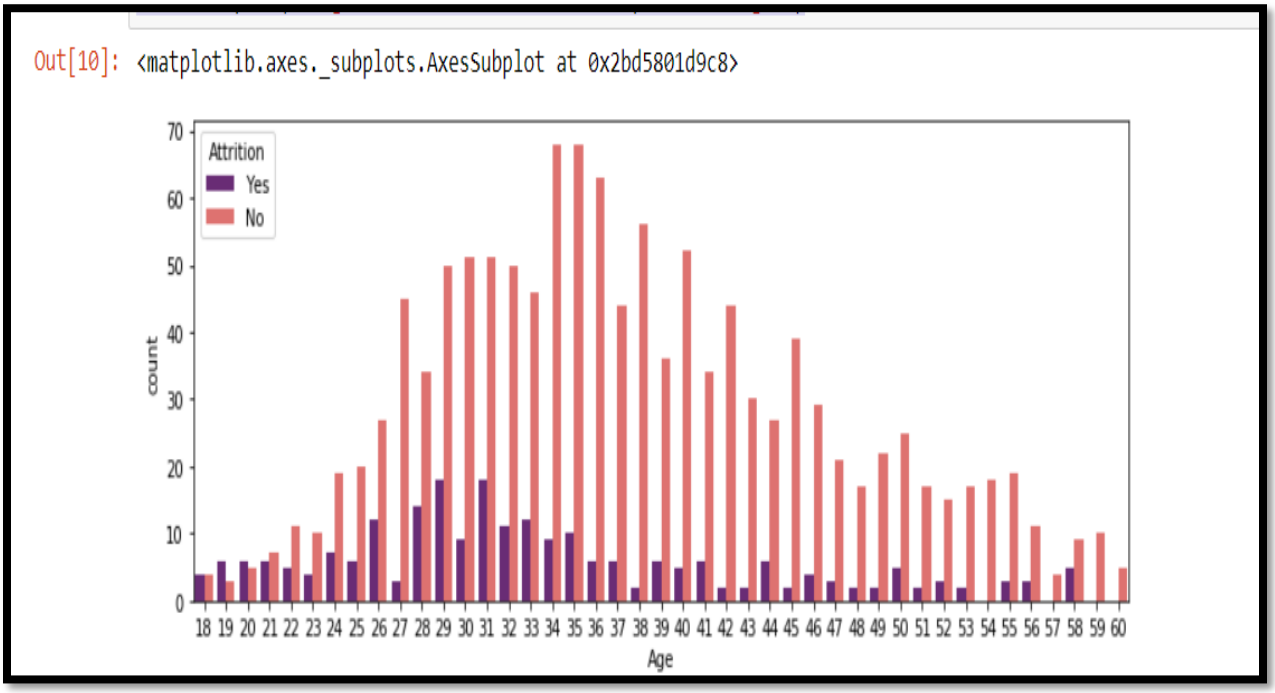
By Age:

Code:-

```
plt.subplots(figsize=(12,4))
```

```
sns.countplot(x='Age',hue='Attrition',data=df,palette='magma')
```

output:



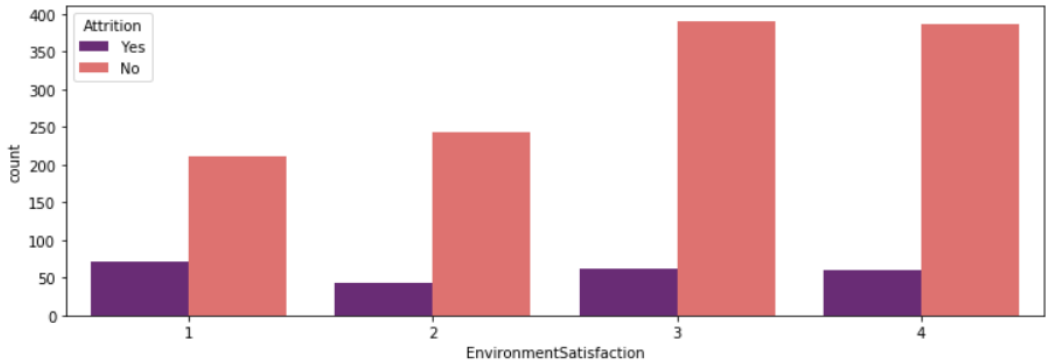
By EnvironmentSatisfaction:

Code:-

```
plt.subplots(figsize=(12,4))

sns.countplot(x='EnvironmentSatisfaction',hue='Attrition',data=df,
palette='magma')
```

output:-

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x2bd582719c8>
```

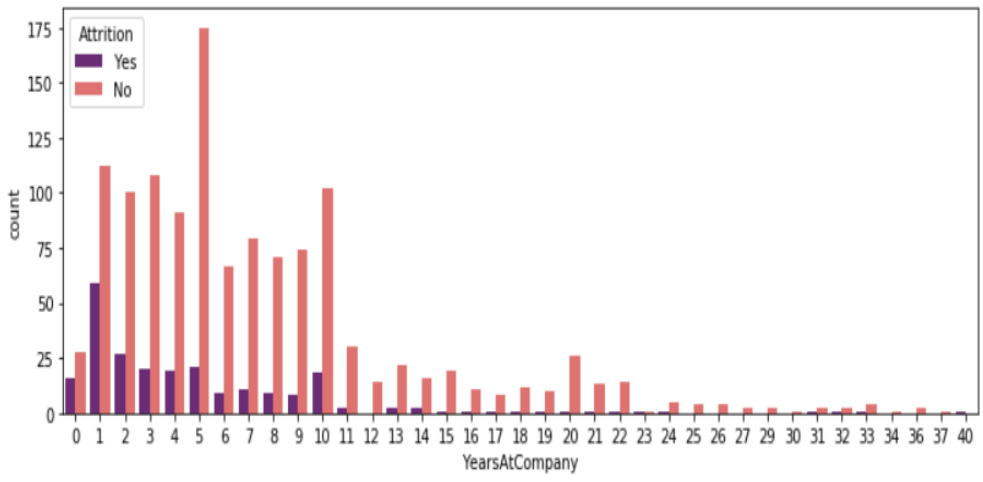


By YearsAtCompany:

```
plt.subplots(figsize=(12,4))

sns.countplot(x='YearsAtCompany',hue='Attrition',data=df,palette='magma')
```

output:-

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x2bd59d0b9c8>
```



Step11:- print all the data types and their unique values

Code:-

```
for column in df.columns:
   if df[column].dtype == object:
      print(str(column) + ' : ' + str(df[column].unique()))
      print(df[column].value_counts())

print('_____
_____')
```

output:-

```
Attrition : ['Yes' 'No']
No     1233
Yes     237
Name: Attrition, dtype: int64
_____
_____
BusinessTravel : ['Travel_Rarely'
'Travel_Frequently' 'Non-Travel']
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
_____
_____
Department : ['Sales' 'Research & Development'
'Human Resources']
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

```
_____________
EducationField : ['Life Sciences' 'Other'
'Medical' 'Marketing' 'Technical Degree'
 'Human Resources']
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
_____________
Gender : ['Female' 'Male']
Male      882
Female    588
Name: Gender, dtype: int64
_____________
JobRole : ['Sales Executive' 'Research
Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare
Representative' 'Manager'
 'Sales Representative' 'Research Director'
'Human Resources']
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
```

```
Name: JobRole, dtype: int64
```

```
MaritalStatus : ['Single' 'Married' 'Divorced']
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

```
Over18 : ['Y']
Y    1470
Name: Over18, dtype: int64
```

```
OverTime : ['Yes' 'No']
No      1054
Yes      416
Name: OverTime, dtype: int64
```

Step12:-drop all useless attributes.

Code:-

```
df.drop('Over18' , axis = 1 , inplace=True)

df.drop('EmployeeCount' , axis = 1 , inplace=True)

df.drop('EmployeeNumber' , axis = 1 , inplace=True)

df.drop('StandardHours' , axis = 1 , inplace=True)
```

df

output:

Out[14]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	Performanc
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	2	Female	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	3	Male	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	4	Male	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	4	Female	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	Male	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	3	Male	...	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	4	Male	...	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	2	Male	...	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	4	Male	...	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	2	Male	...	

1470 rows × 31 columns

Step13:- get the ecorrelation

Code:-

df.corr()

output:

Out[15]:

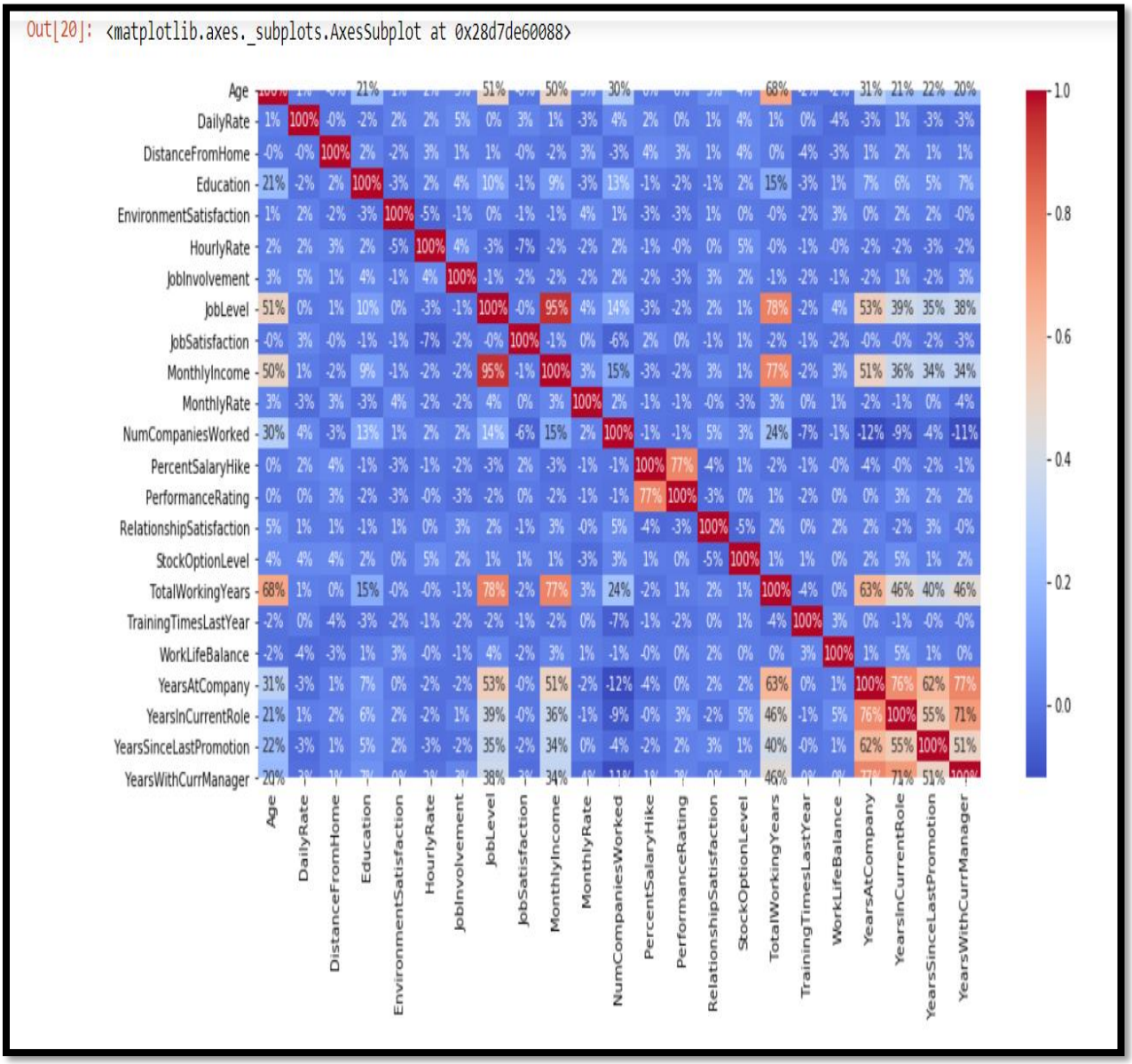
	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction
Age	1.000000	0.010661	-0.001686	0.208034	0.010146	0.024287	0.029820	0.509604	-0.004892
DailyRate	0.010661	1.000000	-0.004985	-0.016806	0.018355	0.023381	0.046135	0.002966	0.030571
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	-0.016075	0.031131	0.008783	0.005303	-0.003669
Education	0.208034	-0.016806	0.021042	1.000000	-0.027128	0.016775	0.042438	0.101589	-0.011296
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	1.000000	-0.049857	-0.008278	0.001212	-0.006784
HourlyRate	0.024287	0.023381	0.031131	0.016775	-0.049857	1.000000	0.042861	-0.027853	-0.071335
JobInvolvement	0.029820	0.046135	0.008783	0.042438	-0.008278	0.042861	1.000000	-0.012630	-0.021476
JobLevel	0.509604	0.002966	0.005303	0.101589	0.001212	-0.027853	-0.012630	1.000000	-0.001944
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	-0.006784	-0.071335	-0.021476	-0.001944	1.000000
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	-0.006259	-0.015794	-0.015271	0.950300	-0.007157
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	0.037600	-0.015297	-0.016322	0.039563	0.000644
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	0.012594	0.022157	0.015012	0.142501	-0.055699
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	-0.031701	-0.009062	-0.017205	-0.034730	0.020002
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	-0.029548	-0.002172	-0.029071	-0.021222	0.002297
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	0.007665	0.001330	0.034297	0.021642	-0.012454
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	0.003432	0.050263	0.021523	0.013984	0.010690
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	-0.002693	-0.002334	-0.005533	0.782208	-0.020185
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	-0.019359	-0.008548	-0.015338	-0.018191	-0.005779
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	0.027627	-0.004607	-0.014617	0.037818	-0.019459
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	0.001458	-0.019582	-0.021355	0.534739	-0.003803
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	0.018007	-0.024106	0.008717	0.389447	-0.002305
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	0.016194	-0.026716	-0.024184	0.353885	-0.018214
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	-0.004999	-0.020123	0.025976	0.375281	-0.027656

Step14:- visualize the correlation

Code:-

```
plt.figure(figsize=(14,14))

sns.heatmap(df.corr(),annot=True,fmt='.0%',cmap='coolwarm')
```

output:-



Step15:- transform non-numerical into numerical

Code:-

from sklearn.preprocessing import LabelEncoder

```
for column in df.columns:
    if df[column].dtype == np.number:
        continue
    df[column]=LabelEncoder().fit_transform(df[column])
df
```

output:-

Out[21]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	PerformanceF
0	23	1	2	624	2	0	1	1	1	0	...	
1	31	0	1	113	1	7	0	1	2	1	...	
2	19	1	2	805	1	1	1	4	3	1	...	
3	15	0	1	820	1	2	3	1	3	0	...	
4	9	0	2	312	1	1	0	3	0	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
1465	18	0	1	494	1	22	1	3	2	1	...	
1466	21	0	2	327	1	5	0	3	3	1	...	
1467	9	0	2	39	1	3	2	1	1	1	...	
1468	31	0	1	579	2	1	2	3	3	1	...	
1469	16	0	2	336	1	7	2	3	1	1	...	

1470 rows × 31 columns

Step16:- create a new column

Code:-

```
df['Age_year']=df['Age']

df
```

output:-

Out[23]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	RelationshipS
0	23	1	2	624	2	0	1	1	1	0	...	
1	31	0	1	113	1	7	0	1	2	1	...	
2	19	1	2	805	1	1	1	4	3	1	...	
3	15	0	1	820	1	2	3	1	3	0	...	
4	9	0	2	312	1	1	0	3	0	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
1465	18	0	1	494	1	22	1	3	2	1	...	
1466	21	0	2	327	1	5	0	3	3	1	...	
1467	9	0	2	39	1	3	2	1	1	1	...	
1468	31	0	1	579	2	1	2	3	3	1	...	
1469	16	0	2	336	1	7	2	3	1	1	...	

1470 rows × 32 columns

Step17:- drop age column

Code:-

```
df.drop('Age',axis=1,inplace=True)
df
```

output:

Out[25]:

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate	...	Relati
0	1	2	624	2	0	1	1	1	0	64	...	
1	0	1	113	1	7	0	1	2	1	31	...	
2	1	2	805	1	1	1	4	3	1	62	...	
3	0	1	820	1	2	3	1	3	0	26	...	
4	0	2	312	1	1	0	3	0	1	10	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1465	0	1	494	1	22	1	3	2	1	11	...	
1466	0	2	327	1	5	0	3	3	1	12	...	
1467	0	2	39	1	3	2	1	1	1	57	...	
1468	0	1	579	2	1	2	3	3	1	33	...	
1469	0	2	336	1	7	2	3	1	1	52	...	

1470 rows × 31 columns

Step18:- Split the data

Code:-

```
X=df.iloc[:,1:df.shape[1]].values   #remaining datapoints
Y=df.iloc[:,0].values #attrition datapoints
df
```

Output:-

Out[27]:

	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate	...	Relati
0	1	2	624	2	0	1	1	1	0	64	...	
1	0	1	113	1	7	0	1	2	1	31	...	
2	1	2	805	1	1	1	4	3	1	62	...	
3	0	1	820	1	2	3	1	3	0	26	...	
4	0	2	312	1	1	0	3	0	1	10	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1465	0	1	494	1	22	1	3	2	1	11	...	
1466	0	2	327	1	5	0	3	3	1	12	...	
1467	0	2	39	1	3	2	1	1	1	57	...	
1468	0	1	579	2	1	2	3	3	1	33	...	
1469	0	2	336	1	7	2	3	1	1	52	...	

1470 rows × 31 columns

Step19:-

Code:-

X

Output:

```
Out[28]: array([[  2, 624,   2, ...,   0,   5,  23],
                [  1, 113,   1, ...,   1,   7,  31],
                [  2, 805,   1, ...,   0,   0,  19],
                ...,
                [  2,  39,   1, ...,   0,   3,   9],
                [  1, 579,   2, ...,   0,   8,  31],
                [  2, 336,   1, ...,   1,   2,  16]], dtype=int64)
```

Code:-

Y

Output:

```
Out[29]: array([1, 0, 1, ..., 0, 0, 0])
```

Step20:- split the data into 75% training & 25% testing

Code:

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,random_state=0)

X_train,X_test,Y_train,Y_test

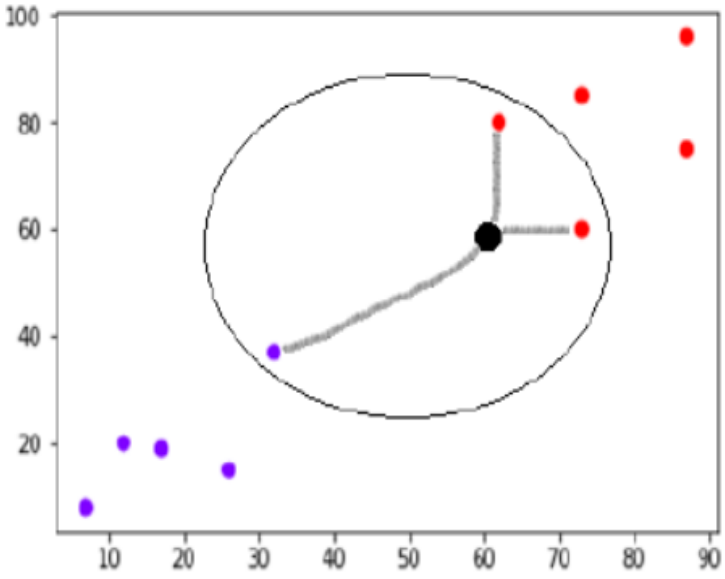
Output:-

```
Out[31]: (array([[  0, 874,   1, ...,   7,   5,  10],
                 [  2, 739,   1, ...,   0,   0,  13],
                 [  2, 880,   1, ...,   0,   7,  31],
                 ...,
                 [  2, 671,   2, ...,   8,   8,  25],
                 [  2, 109,   1, ...,   0,   1,  20],
                 [  2, 358,   2, ...,   0,   0,  22]], dtype=int64),
          array([[  0, 341,   2, ...,   9,   7,  18],
                 [  2, 300,   1, ...,   0,   2,  15],
                 [  1, 361,   2, ...,   3,   2,  17],
                 ...,
                 [  2, 104,   1, ...,   2,  17,  23],
                 [  2, 478,   1, ...,   0,   8,  30],
                 [  1, 419,   2, ...,   0,   8,  13]], dtype=int64),
          array([0, 0, 0, ..., 0, 0, 0]),
          array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
                 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
                 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
                 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
                 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1]))
```

5. Model Building:-

KNN Classifier:-

- K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry.
- K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.



Step21:-Use KNN Classifier

Code:-

```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier()

kn.fit(X_train,Y_train)
```

output:-

```
Out[63]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                              weights='uniform')
```

Step22:-Confusion Matrixs

A confusion matrix is a matrix (table) that can be used to measure the performance of an machine learning algorithm, usually a supervised learning one. Each row of the confusion matrix represents the instances of an actual class and each column represents the instances of a predicted class.

Code:-

```
from sklearn.metrics import confusion_matrix

y_predict=kn.predict(X_test)

confusion_matrix(Y_test,y_predict)
```

output:-

```
Out[64]: array([[298,  12],
                [ 55,   3]], dtype=int64)
```

6. Get the prediction accuracy

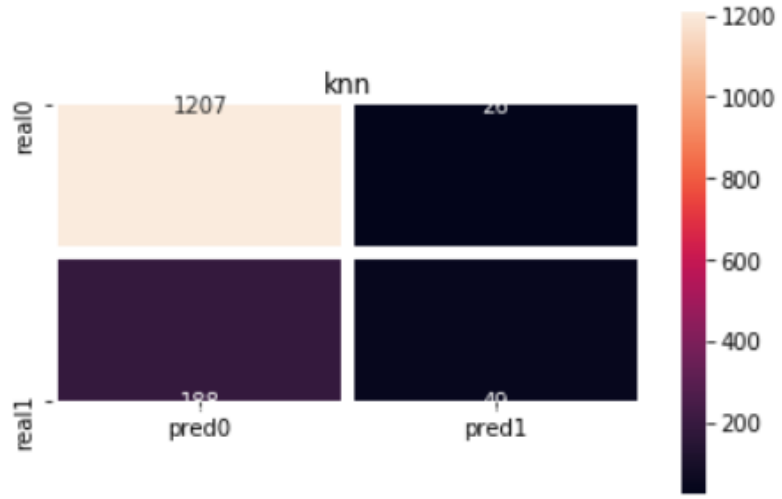
Step23:-printing accuracy and number of employees which are likely to leave.

Code:-

```
knn_prediction=kn.predict(X)

ax = plt.axes()

ax.set_title("knn")

accuracy=kn.score(X_test,Y_test)

print("Accuracy:",accuracy)

print("Total number of employees which are likely to leave.
",sum(knn_prediction))


df_cm = pd.DataFrame(confusion_matrix(Y,knn_prediction),
index=['real0','real1'],columns=['pred0','pred1'])

df_cm


plt.figure(figsize=(14,14))

sns.heatmap(df_cm,annot=True,ax=ax,square=True,fmt='d',linewi
dth=5)

plt.show()
```

output:-

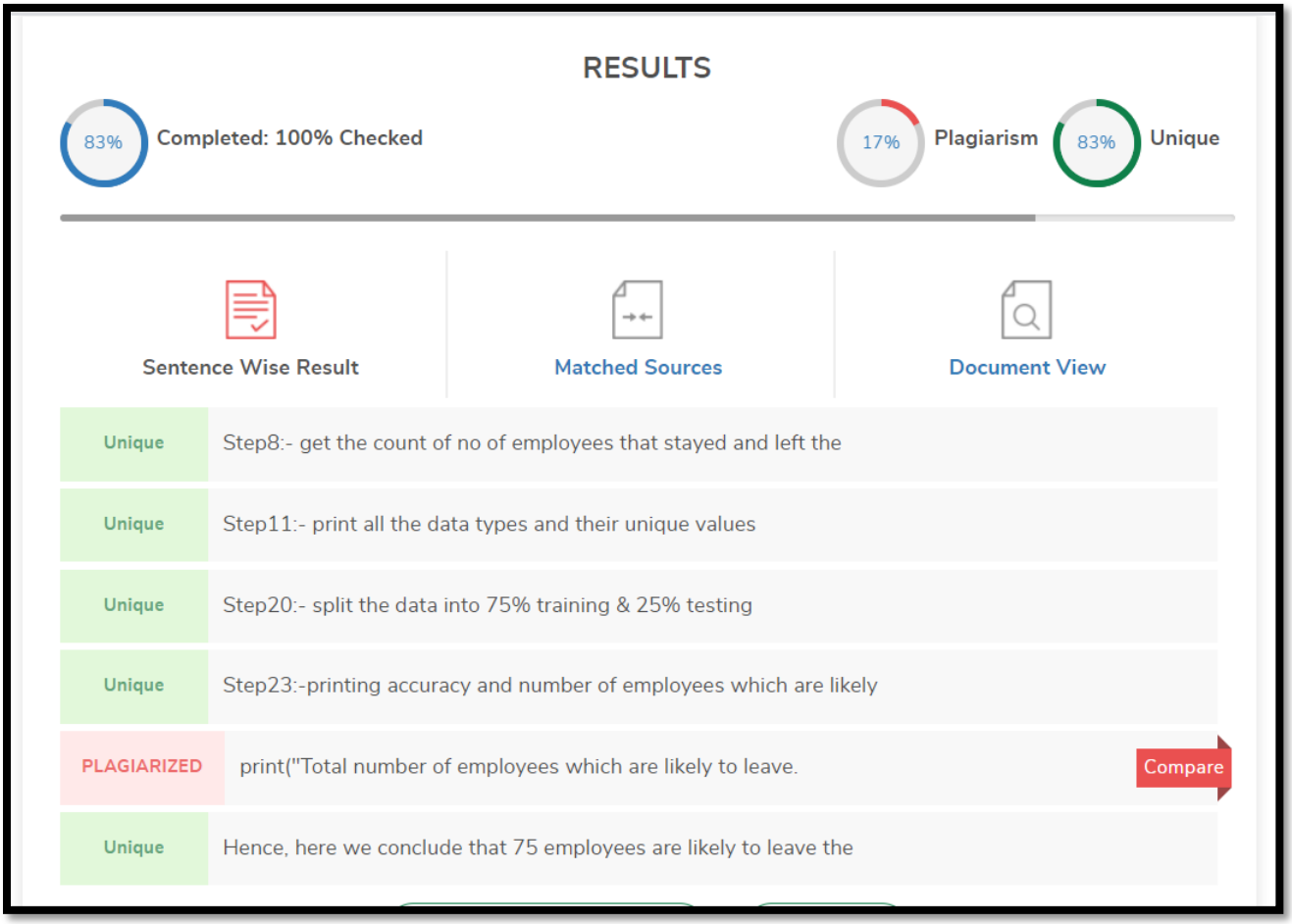
```
Accuracy: 0.8179347826086957
Total number of employees which are likely to leave.  75
```




```
<Figure size 1008x1008 with 0 Axes>
```

Hence, here we conclude that 75 employees are likely to leave the job.

PLAGIARISM REPORT :



CONCLUSION :

Possible reasons for people leaving:-

- Experienced:- Experienced people may be not finding any challenges at work.
- Very low satisfaction level:- A lot of good talent can be lost if the employees feel trapped in dead-end positions. Often talented individuals are forced to job-hop from one company to another in order to grow in status and compensation.
- Spent more time at work.

REFERENCES :

- https://www.w3schools.com/python/numpy_intro.asp#:~:text=NumPy%20is%20a%20python%20library,NumPy%20stands%20for%20Numerical%20Python.

- https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm

- https://www.python-course.eu/confusion_matrix.php