```
import pandas as pd
import numpy as np
from datetime import datetime
```

```
df = pd.read_csv("HR-Employee-Attrition.csv")
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	
1	49	No	Travel_Frequently	279	Research & Development	8	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	
4	27	No	Travel_Rarely	591	Research & Development	2	

5 rows × 35 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
df.describe(include='all')
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFro
count	1470.000000	1470	1470	1470.000000	1470	1470.0
unique	NaN	2	3	NaN	3	
top	NaN	No	Travel_Rarely	NaN	Research & Development	
freq	NaN	1233	1043	NaN	961	
mean	36.923810	NaN	NaN	802.485714	NaN	9.1
std	9.135373	NaN	NaN	403.509100	NaN	8.1
min	18.000000	NaN	NaN	102.000000	NaN	1.0
25%	30.000000	NaN	NaN	465.000000	NaN	2.0
50%	36.000000	NaN	NaN	802.000000	NaN	7.0
75%	43.000000	NaN	NaN	1157.000000	NaN	14.0
max	60.000000	NaN	NaN	1499.000000	NaN	29.0

11 rows × 35 columns

```
df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeC
Age	1.000000	0.010661	-0.001686	0.208034	
DailyRate	0.010661	1.000000	-0.004985	-0.016806	
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	
Education	0.208034	-0.016806	0.021042	1.000000	
EmployeeCount	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	
HourlyRate	0.024287	0.023381	0.031131	0.016775	
JobInvolvement	0.029820	0.046135	0.008783	0.042438	
JobLevel	0.509604	0.002966	0.005303	0.101589	
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	
StandardHours	NaN	NaN	NaN	NaN	
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	

```
df.corr()[df.corr()>0.5]
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCc
Age	1.000000	NaN	NaN	NaN	I
DailyRate	NaN	1.0	NaN	NaN	I
DistanceFromHome	NaN	NaN	1.0	NaN	I
Education	NaN	NaN	NaN	1.0	I
EmployeeCount	NaN	NaN	NaN	NaN	I
EmployeeNumber	NaN	NaN	NaN	NaN	I
EnvironmentSatisfaction	NaN	NaN	NaN	NaN	I
HourlyRate	NaN	NaN	NaN	NaN	I
JobInvolvement	NaN	NaN	NaN	NaN	I
JobLevel	0.509604	NaN	NaN	NaN	I
JobSatisfaction	NaN	NaN	NaN	NaN	I
MonthlyIncome	NaN	NaN	NaN	NaN	I
MonthlyRate	NaN	NaN	NaN	NaN	I
NumCompaniesWorked	NaN	NaN	NaN	NaN	I
PercentSalaryHike	NaN	NaN	NaN	NaN	I
PerformanceRating	NaN	NaN	NaN	NaN	I
RelationshipSatisfaction	NaN	NaN	NaN	NaN	I
StandardHours	NaN	NaN	NaN	NaN	I
StockOptionLevel	NaN	NaN	NaN	NaN	I
TotalWorkingYears	0.680381	NaN	NaN	NaN	I
TrainingTimesLastYear	NaN	NaN	NaN	NaN	I
WorkLifeBalance	NaN	NaN	NaN	NaN	I
YearsAtCompany	NaN	NaN	NaN	NaN	I
YearsInCurrentRole	NaN	NaN	NaN	NaN	I
YearsWithCurrManager	NaN	NaN	NaN	NaN	I

```
pd.set_option("display.float_format", "{:.2f}".format)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.offline as py
py.init_notebook_mode(connected = True)
%matplotlib inline
```

```
df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis = 'columr
df.columns
```

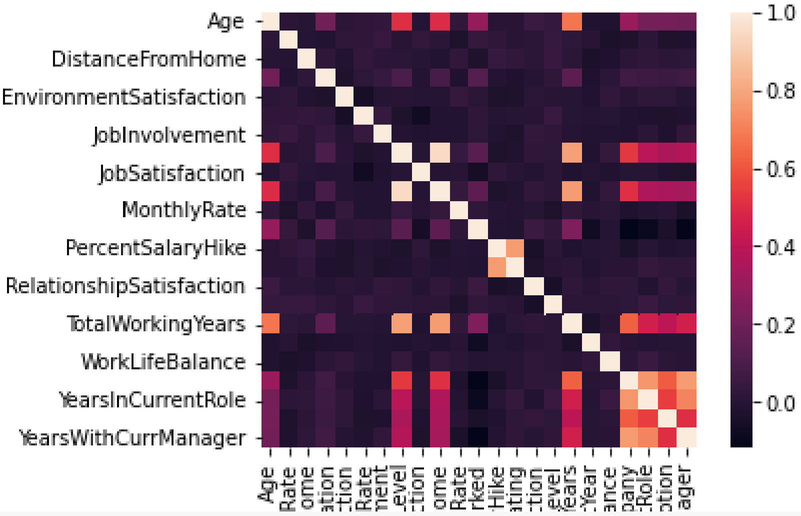
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
       'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'OverTime',
       'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager'],
      dtype='object')
```

```
categorical_col = []
for column in df.columns:
    if df[column].dtype == object:
        categorical_col.append(column)
        print(f"{column}:\n{df[column].unique()}")
        print("                              ")
```

```
Attrition:
['Yes' 'No']

BusinessTravel:
['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']

Department:
['Sales' 'Research & Development' 'Human Resources']

EducationField:
['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'
 'Human Resources']

Gender:
['Female' 'Male']

JobRole:
['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']

MaritalStatus:
['Single' 'Married' 'Divorced']

OverTime:
['Yes' 'No']
```

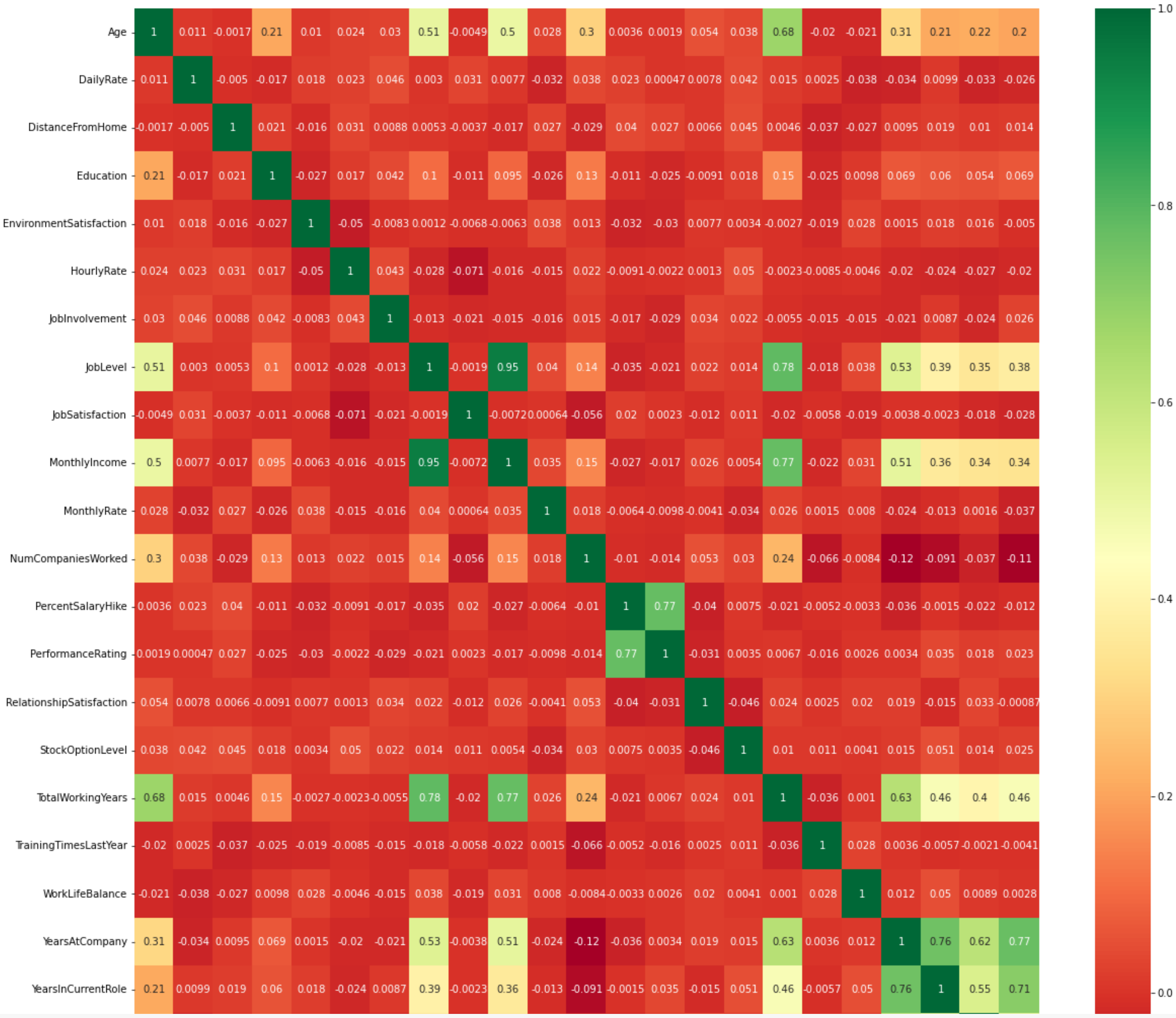
```
  sns.heatmap(df.corr(), vmax = 1, square = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc14aa1bb10>
```



```
plt.figure(figsize =(20,20))
sns.heatmap(df.corr(), annot = True, cmap="RdYlGn", annot_kws = {"size": 10})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc148582d90>
```



```
# Data Processing for ML Algorithm


from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

categorical_col.remove('Attrition')

from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
for column in categorical_col:
    df[column] = label.fit_transform(df[column])
```

```
df
```

1 to 25 of 1470 entries Filter

index	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Edu
0	41	Yes	2	1102	2	1	2	
1	49	No	1	279	1	8	1	
2	37	Yes	2	1373	1	2	2	
3	33	No	1	1392	1	3	4	
4	27	No	2	591	1	2	1	
5	32	No	1	1005	1	2	2	
6	59	No	2	1324	1	3	3	
7	30	No	2	1358	1	24	1	
8	38	No	1	216	1	23	3	
9	36	No	2	1299	1	27	3	
10	35	No	2	809	1	16	3	
11	29	No	2	153	1	15	2	
12	31	No	2	670	1	26	1	
13	34	No	2	1346	1	19	2	
14	28	Yes	2	103	1	24	3	
15	29	No	2	1389	1	21	4	
16	32	No	2	334	1	5	2	
17	22	No	0	1123	1	16	2	
18	53	No	2	1219	2	2	4	
19	38	No	2	371	1	2	3	
20	24	No	0	673	1	11	2	
21	36	Yes	2	1218	2	9	4	

```
# Define a function module to print results of ML Classifier Score

from·sklearn.metrics·import·accuracy_score,·confusion_matrix,·precision_score,·recall

def·print_score(clf,·x_train,·y_train,·x_test,·y_test,·train·=·True):
····if·train:
········pred·=·clf.predict(x_train)
········print("Train·Result:\·=====================================================")
········print(f"accuracy·score:·{accuracy_score(y_train,·pred):.4f}\n")
········print("Classification·Data:")
········print(f"Precision:·{precision_score(y_train,·pred,·average=None,·zero_divisio
········print·(f"Recall·Score:·{recall_score(y_train,·pred,·average=None,·zero_divisi
········print(f"Confusion_matrix:\n·{confusion_matrix(y_train,·clf.predict(x_train))}
····elif·train·==·False:
········pred·=·clf.predict(x_test)
········print("Test·Result:\·=====================================================")
········print(f"accuracy·score:·{accuracy_score(y_test,·pred)}\n")
········print("Classification·Data:")
········print(f"Precision:·{precision_score(y_test,·pred,·average=None,·zero_division
········print·(f"Recall·Score:·{recall_score(y_test,·pred,·average=None,·zero_divisio
········print(f"Confusion_matrix:\n·{confusion_matrix(y_test,·clf.predict(x_test))}\n
```

```
x=df.drop('Attrition', axis = 1)
```

```
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educatio
0	41	2	1102	2	1	2	
1	49	1	279	1	8	1	
2	37	2	1373	1	2	2	
3	33	1	1392	1	3	4	
4	27	2	591	1	2	1	

5 rows × 30 columns

```
# split data into X and Yx = df.drop('Attrition', axis = 1)
y = df.Attrition
y.head()
```

```
0    Yes
1     No
2    Yes
3     No
4     No
Name: Attrition, dtype: object
```

```
# Applying ML Algorithms

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_stat
```

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state = 38)
tree.fit(x_train, y_train)

print_score(tree, x_train, y_train, x_test, y_test, train = True)
print_score(tree, x_train, y_train, x_test, y_test, train = False)
```

```
Train Result:\ ====================================================
accuracy score: 1.0000

Classification Data:
Precision: [1. 1.]

Recall Score: [1. 1.]

Confusion_matrix:
 [[853   0]
 [  0 176]]

Test Result:\ ====================================================
accuracy score: 0.7845804988662132
```

```
Classification Data:
Precision: [0.88828338 0.27027027]

Recall Score: [0.85789474 0.32786885]

Confusion_matrix:
 [[326  54]
 [ 41  20]]
```

```
from sklearn.ensemble import RandomForestClassifier

rand_forest = RandomForestClassifier(n_estimators = 30)
rand_forest.fit(x_train, y_train)

print_score(rand_forest, x_train, y_train, x_test, y_test, train = True)
print_score(rand_forest, x_train, y_train, x_test, y_test, train = False)
```

```
Train Result:\ ====================================================
accuracy score: 0.9990

Classification Data:
Precision: [0.99882904 1.        ]

Recall Score: [1.         0.99431818]

Confusion_matrix:
 [[853   0]
 [  1 175]]

Test Result:\ ====================================================
accuracy score: 0.8616780045351474

Classification Data:
Precision: [0.87006961 0.5       ]

Recall Score: [0.98684211 0.08196721]

Confusion_matrix:
 [[375   5]
 [ 56   5]]
```

Overfit underfit means talk about

ensemblement , decision tree

k-fold classification

bagging

●