PROJECT REPORT

PREDICTING HIGH POTENTIAL EMPLOYEES IN A CORPORATE

Name : Prabhanjan Kumar

Project:Predicting High Potential Employees in a Corporate

Domain: Machine Learning

CONTENTS:

1.INTRODUCTION

1.1 OVERVIEW

Employee turnover has been identified as a key issue for organizations because of its adverse impact on work place productivity and long term growth strategies. To solve this problem, organizations use machine learning techniques to predict employee turnover. Accurate predictions enable organizations to take action for retention or succession planning of employees.

2.LITERATURE SURVEY:

2.1EXISTING PROBLEM:

Employees are the key resources of the organization. The success or failure of an organization depends on the employee. Most of the organizations or companies have a formal performance evaluation system in which employee job performance is graded on a regular basis, usually once or twice a year. A good performance evaluation system can prominently benefit an organization. It helps employee behavior toward organizational aims by permitting employees to know what is expected for them, and it yields information for making employment decisions, such as those regarding pay raises, promotion, or releases.

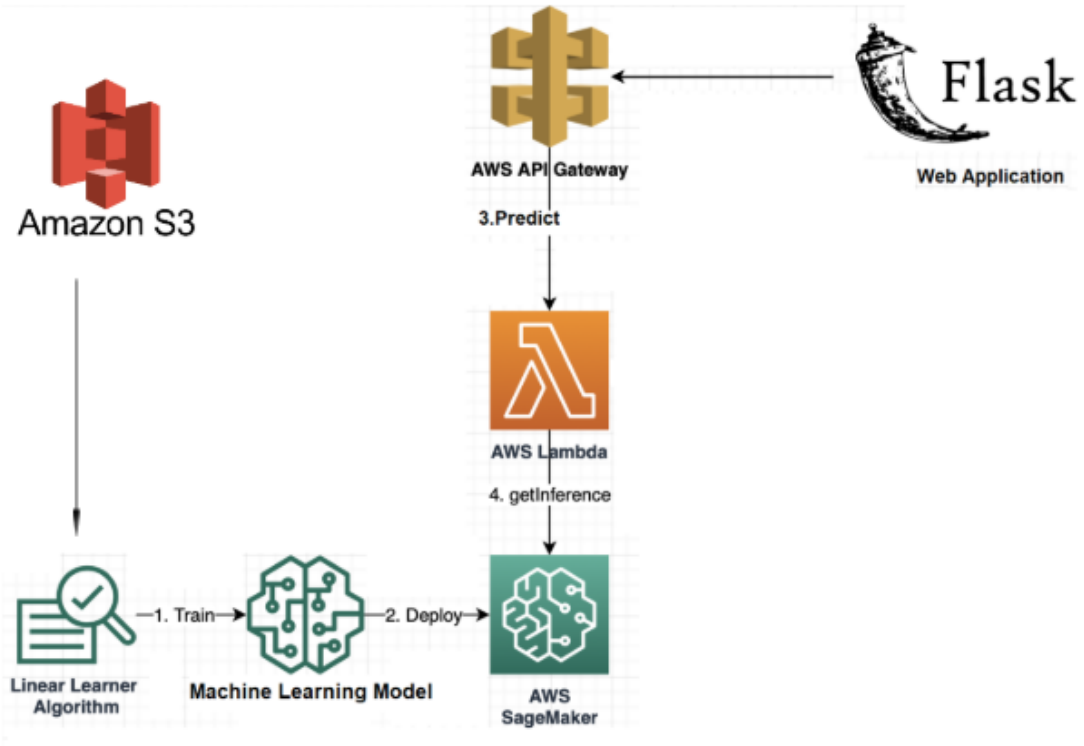
.

2.2PROPOSED SOLUTION:

Build & Deploy a Machine Learning model to rate the employee performance using Amazon SageMaker.Create a python - flask application that interacts with the model deployed on AWS Sagemaker with the help of AWS API Gateway and AWS Lambda Services.

3.THEORITICAL ANALYSIS:

3.1. BLOCK DIAGRAM:



3.2. SOFTWARE DESIGNING:

1. Amazon S3
2. AWS API Gateway
3. AWS Lambda
4. Flask Integration
5. Amazon SageMaker
6. Python 3

4.EXPERIMENTAL INVESTIGATIONS:

Aws Cloud:

Aws Cloud Provides Many Services Such as Sagemaker,lambda and Api Gateway,etc..

Sagemaker:

Amazon SageMaker is a fully managed service that provides every developer and data scientistwith the ability to build, train, and deploy machine learning (ML)

models quickly. SageMakerremoves the heavy lifting from each step of the machine learning process to make it easier todevelop high quality models.

Lambda:

With Lambda, you can run code for virtually any type of application or backend service - all

with zero administration. Just upload your code and Lambda takes care of everything required

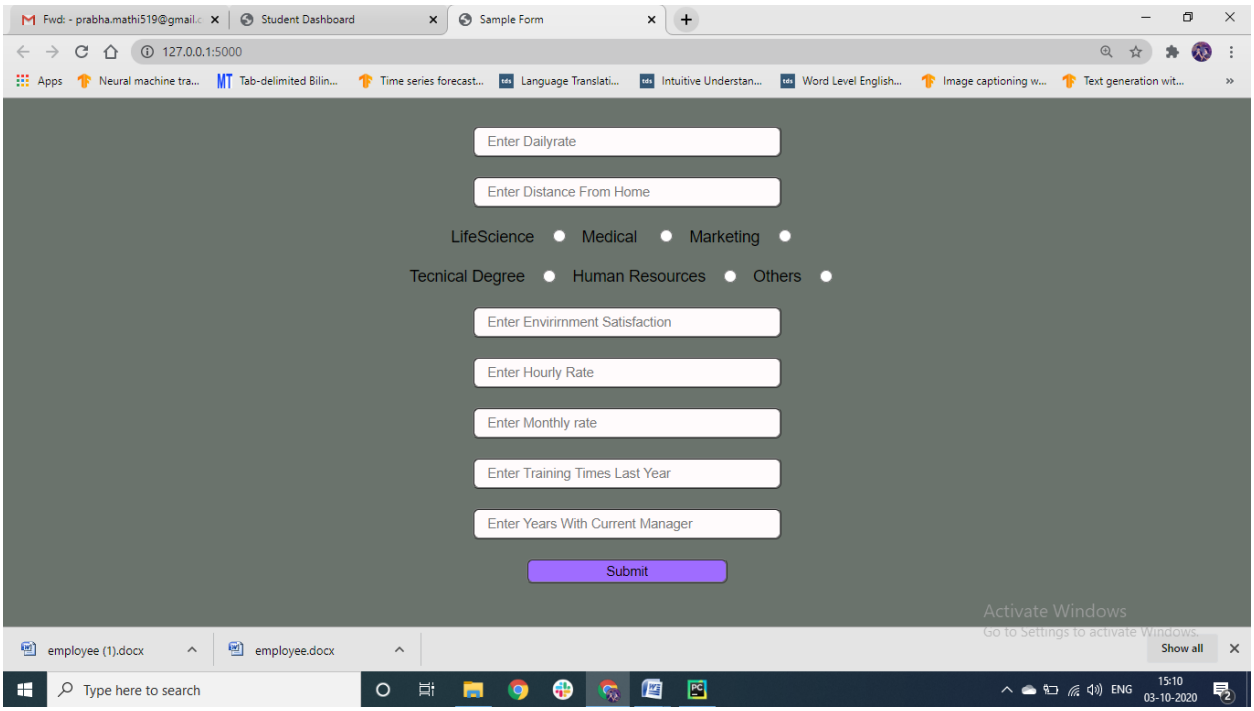
to run and scale your code with high availability. You can set up your code to automatically

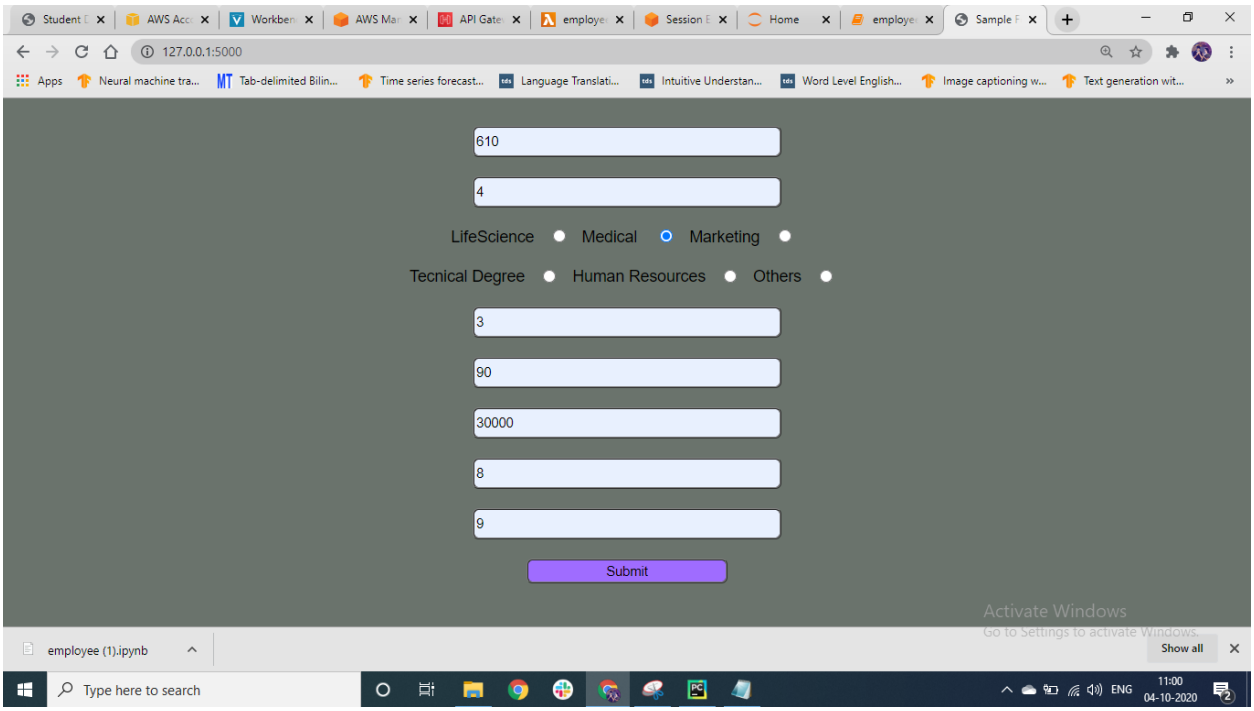
trigger from other AWS services

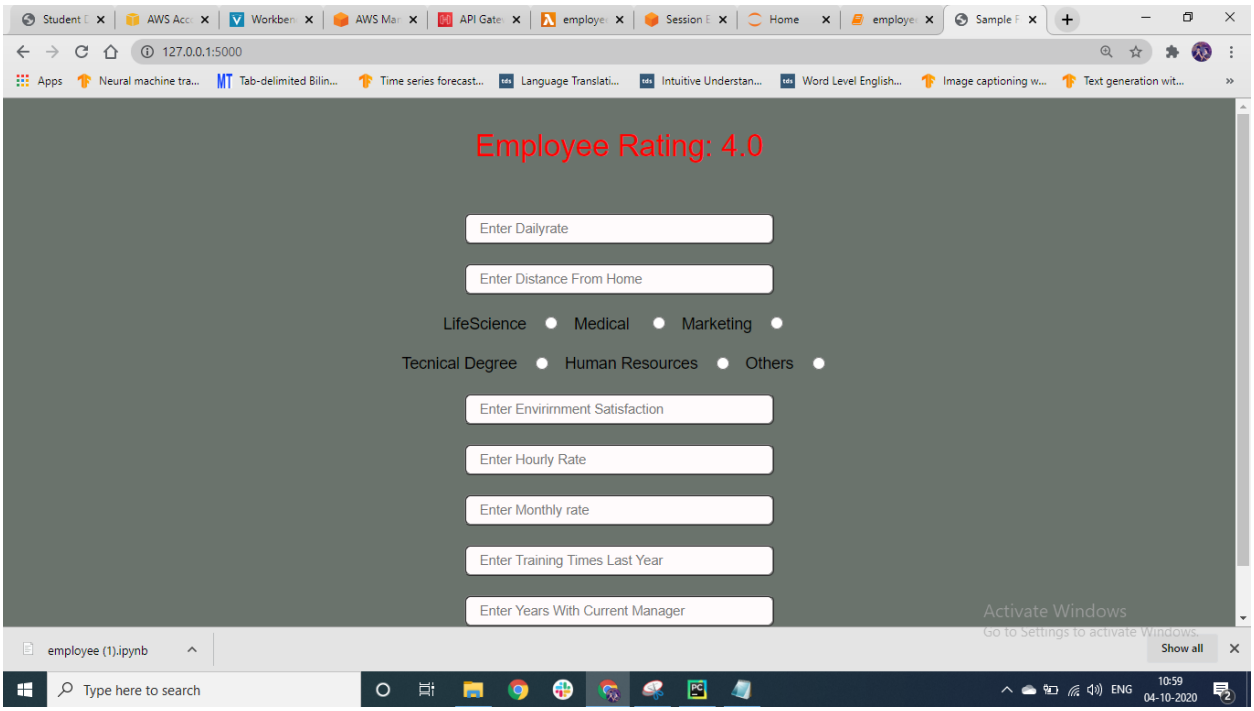
Api Gateway:

Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, andsecuring REST,HTTP, and WebSocket APIs at any scale. API developers can create APIs thataccess AWS or other web services, as well as data stored in the AWS Cloud APIGateway creates RESTful APIs that Are HTTP-based.

5.RESULT:


Fwd: - prabha.mathi519@gmail.c
Student Dashboard
Sample Form
127.0.0.1:5000
Apps
Neural machine tra...
Tab-delimited Bilin...
Time series forecast...
Language Translati...
Intuitive Understan...
Word Level English...
Image captioning w...
Text generation wit...
Enter Dailyrate
Enter Distance From Home
LifeScience
Medical
Marketing
Tecnical Degree
Human Resources
Others
Enter Envirinment Satisfaction
Enter Hourly Rate
Enter Monthly rate
Enter Training Times Last Year
Enter Years With Current Manager
Submit
Activate Windows
Go to Settings to activate Windows.
employee (1).docx
employee.docx
Show all
Type here to search
ENG
15:10
03-10-2020


Student
AWS Acc
Workben
AWS Mar
API Gate
employe
Session
Home
employe
Sample F
127.0.0.1:5000
Apps
Neural machine tra...
Tab-delimited Bilin...
Time series forecast...
Language Translati...
Intuitive Understan...
Word Level English...
Image captioning w...
Text generation wit...
610
4
LifeScience
Medical
Marketing
Tecnical Degree
Human Resources
Others
3
90
30000
8
9
Submit
Activate Windows
Go to Settings to activate Windows.
employee (1).ipynb
Show all
Type here to search
ENG
11:00
04-10-2020



6.ADVANTAGES

1. Easy to understand and efficient training algorithm(xgclassifier algorithm).
2. Always find a “good solution”

7.APPLICATIONS:

1. Used in multinational companies
2. Used in business organizations.

into a prescriptive one, addressing not just the question “Who is at risk?” but also “What can we do?”. It is also recommended to study the application of deep learning models for predicting turnover. A well-designed network with sufficient hidden layers might improve the accuracy, however the scalability and practical implementation aspect has to be

FUTURE SCOPE:

The importance of predicting employee turnover in organizations and the

application of machine learning in building turnover models was done in this project. the bestthing is the capture of data around interventions done by the organization for at-risk at employees and its outcome. This will transform the model

CONCLUSION

The results demonstrate that the XGBoost classifier is a superior algorithm in terms of significantly higher accuracy, relatively low runtimes and efficient memory utilization for predicting turnover. The formulation of its regularization makes it a robust technique capable of handling the noise in the data from HRIS, as compared to the other classifiers, thus overcoming the key challenge in this domain. Because of these reasons itis recommended to use XGBoost for accurately predicting employee turnover, thus enabling organizations to take actions for retention or succession of employees.

9.BIBILOGRAPHY:

- J. L. Cotton and J. M. Tuttle, “Employee turnover: A meta-analysis and review with implications for research”, Academy of management Review, 11(1), 55-70, 1986

Code:

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
```

```
1 dataset=pd.read_csv('employee.csv')
2 dataset.head()
```

	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 rows × 35 columns

```
1 dataset.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
```

```
1 from sklearn.preprocessing import LabelEncoder
2 le=LabelEncoder()
3 enc=[1,2,4,7,11,15,17,21,22]
4 for i in enc:
5     dataset.iloc[:,i]=le.fit_transform(dataset.iloc[:,i])
6 dataset.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSa
0	41	1	2	1102	2	1	2	1	1	1	...	
1	49	0	1	279	1	8	1	1	1	2	...	
2	37	1	2	1373	1	2	2	4	1	4	...	
3	33	0	1	1392	1	3	4	1	1	5	...	
4	27	0	2	591	1	2	1	3	1	7	...	

5 rows × 35 columns

```
1 keys=dataset.keys()
2 from scipy import stats
3 for i in range(35):
```

```
    if i==25:
        continue
    else:

person_cof,p_value=stats.pearsonr(dataset.iloc[:,i],dataset.iloc[:,25])
        print('{0}   person cofficent {1}     p value {2}'.format(keys[i],person_cof,p_value))
```

```
Age   person cofficent 0.05353471967122897     p value 0.04014266700714043
Attrition   person cofficent -0.04587227888112659     p value 0.0787136304846609
BusinessTravel   person cofficent -0.035985692635001225     p value 0.16789930522543173
DailyRate   person cofficent 0.007846030957248371     p value 0.7637423378954085
Department   person cofficent -0.02241442536337588     p value 0.390473559704688
DistanceFromHome   person cofficent 0.006557474646578776     p value 0.8016546900801887
Education   person cofficent -0.009118376696381542     p value 0.726853810425483
EducationField   person cofficent -0.004377711027772298     p value 0.8668177236474405
EmployeeCount   person cofficent nan     p value nan
EmployeeNumber   person cofficent -0.06986141146763689     p value 0.007372720416811787
EnvironmentSatisfaction   person cofficent 0.007665383541074459     p value 0.7690257097593178
Gender   person cofficent 0.022868369968027498     p value 0.38094563473972304
HourlyRate   person cofficent 0.0013304527859505748     p value 0.9593518487150708
JobInvolvement   person cofficent 0.034296820611197654     p value 0.18876949288307246
JobLevel   person cofficent 0.02164151053259153     p value 0.4070255864853394
JobRole   person cofficent -0.020217654202924953     p value 0.4385900481142587
JobSatisfaction   person cofficent -0.012453593161926891     p value 0.6332978249787774
MaritalStatus   person cofficent 0.022549070679790662     p value 0.3876324866668781
MonthlyIncome   person cofficent 0.025873436137557573     p value 0.3215271037687779
MonthlyRate   person cofficent -0.004085329337519513     p value 0.8756378937290534
NumCompaniesWorked   person cofficent 0.05273304856488603     p value 0.04322779773883382
Over18   person cofficent nan     p value nan
OverTime   person cofficent 0.04849280287013848     p value 0.06306200332195713
PercentSalaryHike   person cofficent -0.040490081057077354     p value 0.12072710669196073
PerformanceRating   person cofficent -0.03135145544245528     p value 0.22963263459995237
StandardHours   person cofficent nan     p value nan
StockOptionLevel   person cofficent -0.045952490716561795     p value 0.07819157058428752
TotalWorkingYears   person cofficent 0.024054291821341434     p value 0.3567350247415313
TrainingTimesLastYear   person cofficent 0.002496526392117085     p value 0.9238088848665346
WorkLifeBalance   person cofficent 0.019604405703968677     p value 0.452606292244362
YearsAtCompany   person cofficent 0.01936678687745539     p value 0.4581044493788707
YearsInCurrentRole   person cofficent -0.015122914881937722     p value 0.562345475819942
YearsSinceLastPromotion   person cofficent 0.03349250206935415     p value 0.19935561471772828
YearsWithCurrManager   person cofficent -0.0008674968446256374     p value 0.9734895448807728
```

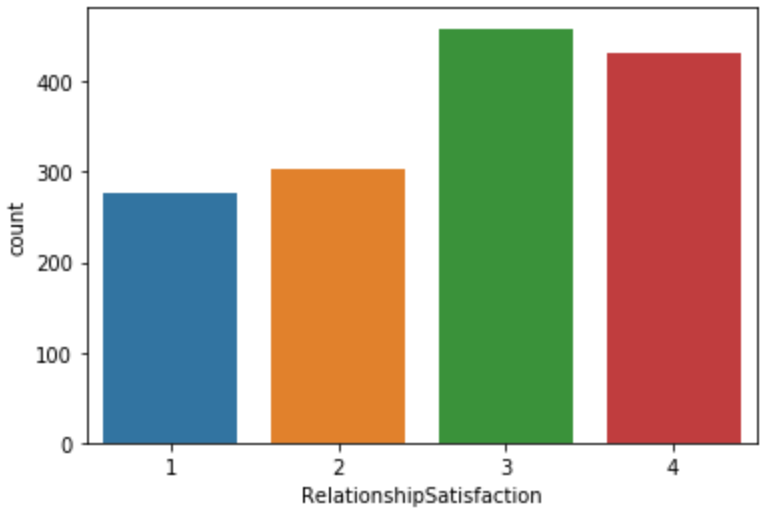
```
data=dataset[['DailyRate','DistanceFromHome','EducationField','EnvironmentSatisfaction','HourlyRate','MonthlyRate','TrainingTimesLastYear','YearsWithCurrManager','RelationshipSatisfaction']]
data.head()
```

ilyRate	DistanceFromHome	EducationField	EnvironmentSatisfaction	HourlyRate	MonthlyRate	TrainingTimesLastYear	YearsWithCurrManager	RelationshipSatisfa
1102	1	1	2	94	19479	0	5	
279	8	1	3	61	24907	3	7	
1373	2	4	4	92	2396	3	0	
1392	3	1	4	56	23159	3	0	
591	2	3	1	40	16632	3	2	

```
1 from sklearn.preprocessing import MinMaxScaler
2 from sklearn.model_selection import train_test_split
```

```
1 sns.countplot(x='RelationshipSatisfaction',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20fa02a860>
```

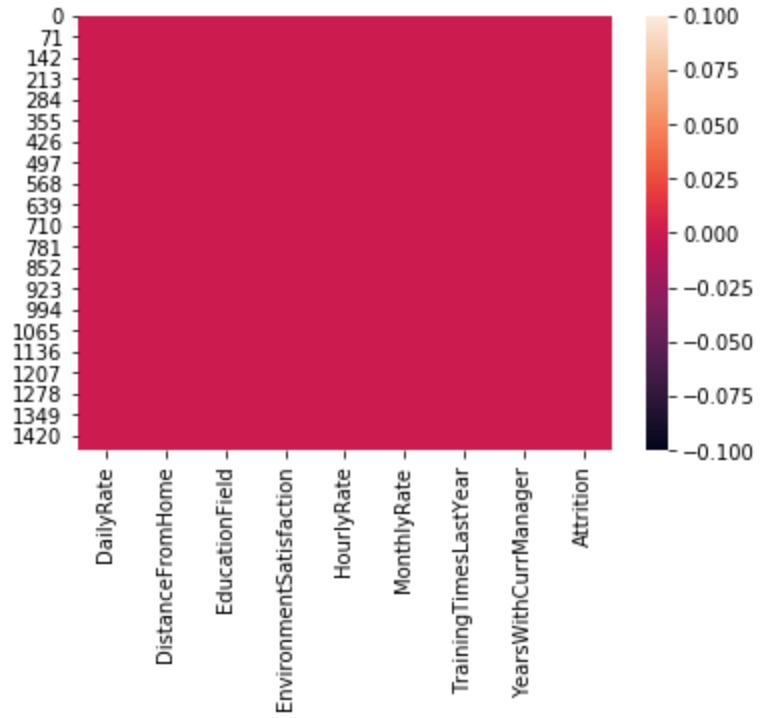



```
1 data.isnull().sum()
```

```
DailyRate                 0
DistanceFromHome          0
EducationField            0
EnvironmentSatisfaction   0
HourlyRate                0
MonthlyRate               0
TrainingTimesLastYear     0
YearsWithCurrManager      0
Attrition                 0
dtype: int64
```

```
1 sns.heatmap(data.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb22a791668>
```




```
1 data_in=data.iloc[:,:-1]
2 data_out=data.iloc[:,-1]
3 sc=MinMaxScaler(feature_range=(0,1))
4 data_in=sc.fit_transform(data_in)
5 keys=data.keys()[:-1]
6 dici={}
7 for i in range(len(keys)):
8     dici.update({keys[i]:data_in[:,i]})
```

```
dataset=pd.DataFrame(dici)
dataset.head()
```

	DailyRate	DistanceFromHome	EducationField	EnvironmentSatisfaction	HourlyRate	MonthlyRate	TrainingTimesLastYear	YearsWithCurrManager
0	0.715820	0.000000	0.2	0.333333	0.914286	0.698053	0.0	0.294118
1	0.126700	0.250000	0.2	0.666667	0.442857	0.916001	0.5	0.411765
2	0.909807	0.035714	0.8	1.000000	0.885714	0.012126	0.5	0.000000
3	0.923407	0.071429	0.2	1.000000	0.371429	0.845814	0.5	0.000000
4	0.350036	0.035714	0.6	0.000000	0.142857	0.583738	0.5	0.117647

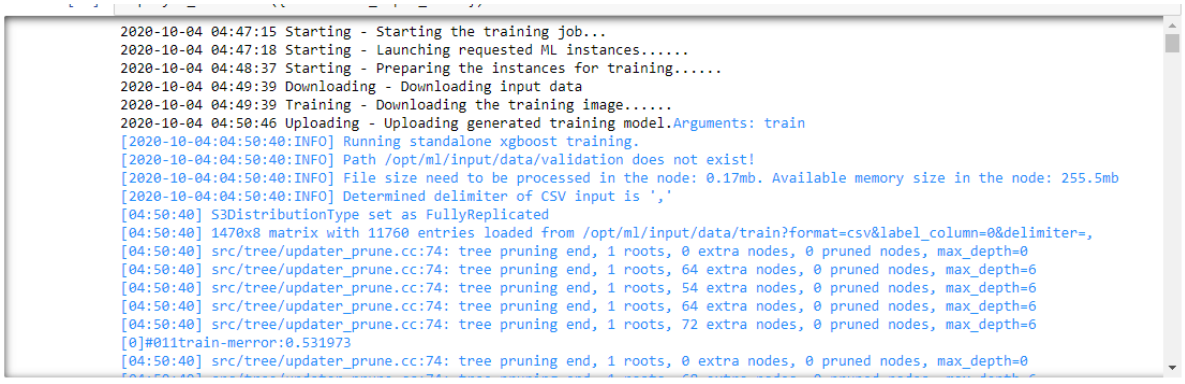
```
final_data=pd.concat([data.iloc[:,-1],dataset],axis=1)
final_data.head()
```

	RelationshipSatisfaction	DailyRate	DistanceFromHome	EducationField	EnvironmentSatisfaction	HourlyRate	MonthlyRate	TrainingTimesLastYear	YearsWith
0	1	0.715820	0.000000	0.2	0.333333	0.914286	0.698053	0.0	
1	4	0.126700	0.250000	0.2	0.666667	0.442857	0.916001	0.5	
2	2	0.909807	0.035714	0.8	1.000000	0.885714	0.012126	0.5	
3	3	0.923407	0.071429	0.2	1.000000	0.371429	0.845814	0.5	
4	4	0.350036	0.035714	0.6	0.000000	0.142857	0.583738	0.5	

```
train,test=train_test_split(final_data,test_size=0.2)
```

```
import boto3,re,os,json,sagemaker
from sagemaker import get_execution_role
role=get_execution_role()
my_region=boto3.session.Session().region_name
containers = {'us-west-2':
'433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1':
'811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2':
'825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1':
'685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'}
```

```
prefix='sagemaker/Employe'
bucket_name='buildathonproject1'
final_data.to_csv('train.csv',index=False,header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix,'train/train.csv')).upload_file('train.csv')
s3_input_train=sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name, prefix),content_type='csv')
sess=sagemaker.Session()
employee_model=sagemaker.estimator.Estimator(containers[my_region],role,train_instance_count=1,train_instance_type='ml.m5.large',output_path='s3://{}/{}/output'.format(bucket_name,prefix),sagemaker_session=sess)
employee_model.set_hyperparameters(objective='multi:softmax',num_round=100,num_class=5)
employee_model.fit({'train':s3_input_train})
```




```
detector=employee_model.deploy(initial_instance_count=1,instance_type='ml.m5.large')
```

```
Parameter image will be renamed to image_uri in SageMaker Python SDK v2.
```

```
-------------!
```

```
detector.endpoint
```

```
'xgboost-2020-10-04-04-47-15-006'
```

```
from sagemaker.predictor import csv_serializer
test_data_array=test.drop('RelationshipSatisfaction',axis=1).values #load the data into an array
detector.content_type = 'text/csv' # set the data type for an inference
detector.serializer = csv_serializer # set the serializer type
predictions=detector.predict(test_data_array).decode('utf-8') # predict!
predictions_array = np.fromstring(predictions[1:], sep=',')
print(predictions)
```

```
3.0,1.0,1.0,3.0,2.0,1.0,4.0,1.0,1.0,4.0,3.0,1.0,2.0,1.0,1.0,4.0,3.0,2.0,3.0,1.0,2.0,3.0,2.0,1.0,4.0,4.0,4.0,2.0,2.0,2.0,1.0,4.0,3.0,2.0,2.0,2.0,3.0,3.0,4.0,3.0,3.0,4.0,2.0,3.0,4.0,1.0,3.0,4.0,4.0,1.0,3.0,3.0,2.0,3.0,3.0,3.0,2.0,4.0,2.0,4.0,3.0,4.0,4.0,2.0,4.0,3.0,3.0,4.0,2.0,3.0,1.0,4.0,2.0,2.0,2.0,4.0,4.0,2.0,2.0,1.0,4.0,4.0,3.0,4.0,4.0,2.0,3.0,2.0,2.0,1.0,4.0,3.0,2.0,3.0,4.0,4.0,4.0,1.0,1.0,1.0,4.0,1.0,2.0,3.0,2.0,2.0,4.0,2.0,2.0,3.0,4.0,4.0,2.0,1.0,4.0,3.0,3.0,3.0,4.0,2.0,3.0,4.0,1.0,1.0,1.0,1.0,4.0,1.0,4.0,2.0,2.0,4.0,4.0,3.0,4.0,2.0,3.0,4.0,4.0,2.0,3.0,3.0,2.0,1.0,4.0,1.0,4.0,2.0,4.0,3.0,2.0,4.0,2.0,4.0,2.0,3.0,1.0,4.0,4.0,2.0,4.0,4.0,3.0,3.0,1.0,3.0,4.0,4.0,2.0,1.0,3.0,2.0,1.0,4.0,3.0,3.0,4.0,4.0,3.0,4.0,4.0,1.0,4.0,1.0,2.0,3.0,3.0,4.0,2.0,3.0,3.0,2.0,3.0,3.0,3.0,3.0,4.0,4.0,3.0,3.0,1.0,4.0,1.0,3.0,2.0,2.0,1.0,1.0,4.0,3.0,3.0,3.0,1.0,3.0,2.0,3.0,1.0,3.0,4.0,4.0,3.0,4.0,4.0,3.0,4.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,4.0,3.0,4.0,4.0,3.0,4.0,3.0,3.0,3.0,3.0,3.0,3.0,3.0,4.0,3.0,4.0,4.0,1.0,1.0,2.0,2.0,2.0,3.0,2.0,4.0,3.0,1.0,4.0,2.0,3.0,3.0,4.0,4.0,4.0,4.0,4.0,3.0,4.0,1.0,3.0,3.0,1.0,3.0,3.0,3.0,3.0,2.0,4.0,2.0,4.0,4.0,3.0,1.0,4.0,4.0,4.0,3.0,4.0,4.0,3.0,4.0,3.0,4.0,1.0,3.0
```

LambdaFunction:

```
import os
import io
import boto3
import json
import csv
def lambda_handler(event, context):
    ENDPOINT_NAME = os.environ['environment_variable']
    runtime= boto3.client('runtime.sagemaker')
    print(ENDPOINT_NAME)
    print("Received event: " , json.dumps(event, indent=2))
    data = json.loads(json.dumps(event))
    print("Data:",data)
    payload = data['data']
```

```
    print("Payload:",payload)
    response =
  runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,

  ContentType='text/csv',
                                          Body=payload)
    print(response)
    result = json.loads(response['Body'].read().decode())
    print(result)
    return result
```

```
Response:
4

Request ID:
"92c5037d-e607-41c1-846c-c556a9694830"

Function logs:
START RequestId: 92c5037d-e607-41c1-846c-c556a9694830 Version: $LATEST
xgboost-2020-10-04-04-47-15-006
Received event:  {
  "data": "0.126700,0.250000,0.2,0.666667,0.442857,0.916001,0.5,0.411765"
}
Data: {'data': '0.126700,0.250000,0.2,0.666667,0.442857,0.916001,0.5,0.411765'}
Payload: 0.126700,0.250000,0.2,0.666667,0.442857,0.916001,0.5,0.411765
{'ResponseMetadata': {'RequestId': 'da273dac-f15b-4e67-8c6b-56a1d58a5c96', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'da273d
```

UI:

hello.py

```
from flask import Flask,render_template,request,url_for
import requests
app=Flask(__name__)
@app.route('/',methods=['POST','GET'])
def hello():
    if request.method=='POST':
        dailyrate=request.form['a']
        dfm=request.form['b']
        ef=request.form['r1']
        es=request.form['d']
        hr=request.form['e']
        mr=request.form['f']
        ttlr=request.form['g']
```

```
14          ycm=request.form['h']
15          print(ef)
16          try:
17              dailyrate=int(dailyrate)
18              dfm=int(dfm)
19              ef=int(ef)
20              es=int(es)
21              hr=int(hr)
22              mr=int(mr)
23              ttlr=int(ttlr)
24              ycm=int(ycm)
25          except:
26              return
    render_template('data.html',err_msg='Enter Valid Data')
27          url =
    "https://7b4168uo26.execute-api.us-east-1.amazonaws.com/emp
    loyee/"
28          payload = " {\"data\":\"" + str(dailyrate) + ',' +
    str(dfm) + ',' + str(ef) + ',' + str(es) + ',' + str(hr) +
    ',' + str(mr) + ',' + str(ttlr) + ',' + str(ycm) + "\"" +
    "}"
29
30          headers = {
31              'X-Amz-Content-Sha256':
    'beaead3198f7da1e70d03ab969765e0821b24fc913697e929e726aeaeb
    f0eba3',
32              'X-Amz-Date': '20200930T095337Z',
33              'Authorization': 'AWS4-HMAC-SHA256
    Credential=ASIA4KDESJFDUSSQKJSG/20200930/us-east-1/execute-
    api/aws4_request,
    SignedHeaders=host;x-amz-content-sha256;x-amz-date,
    Signature=b81935cc533d5efb8db465da9c12f4a3ed76ca80089dfc3ed
    ebdb39df4fe5f7c',
34              'Content-Type': 'text/plain'
35          }
36
37          response = requests.request("POST", url,
```

```
  headers=headers, data=payload)
38          response=response.text.encode('utf8')
39          response=str(response)
40          print(response)
41          result=response[2:-1]
42          print(result)
43          return
  render_template('data.html',result=str(result))
44      else:
45          return render_template('data.html')
46
47 if __name__ == '__main__':
48      app.run(debug=True)
49
```

Data.html:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4     <title>Sample Form</title>
5      <style>
6          .emp
7          {
8              width:300px;
9              height:25px;
10             background-color:#fffbfc;
11             border-style: ridge;
12             border-color:gray;
13             border-radius:6px;
14         }
15         body
16         {
17             font-family:sans-serif;
18         }
19         #sub
```

```
        {
            width:200px;
            height:25px;
            background-color:#9f6cff;
            border-style: ridge;
            border-color:gray;
            border-radius:6px;
        }
    </style>
</head>
<body style="background-color:#6a736c;">
    <center>
        {% if result %}
        <p style="color:red;font-size:30px;"> Employee Rating: {{result}}</p>
        {% endif %}
        {% if err_msg %}
        <p style="color:red;font-size:15px;">{{err_msg}}</p>
        {% endif %}
        <br/>
        <form method="post" action="/">
            <input type="text" name="a" class="emp" placeholder="   Enter Dailyrate" required><br/><br/>
            <input type="text" name="b" class="emp" placeholder="   Enter Distance From Home" required><br/><br/>
            LifeScience   <input type="radio" value="1" name="r1" required>  
            Medical    <input type="radio" value="3" name="r1" required>  
            Marketing   <input type="radio" value="2" name="r1" required>  
            <br/><br/>
            Tecnical Degree   <input type="radio" value="5" name="r1" required>  
            Human Resources   <input type="radio"
```

```
   value="0" name="r1" required>  
48             Others   <input type="radio"
   value="4" name="r1" required>  
49             <br/><br/>
50             <input type="text" name="d" class="emp"
   placeholder="   Enter Envirirnment Satisfaction"
   required><br/><br/>
51             <input type="text" name="e" class="emp"
   placeholder="   Enter Hourly Rate" required><br/><br/>
52             <input type="text" name="f" class="emp"
   placeholder="   Enter Monthly rate" required><br/><br/>
53             <input type="text" name="g" class="emp"
   placeholder="   Enter Training Times Last Year"
   required><br/><br/>
54             <input type="text" name="h" class="emp"
   placeholder="   Enter Years With Current Manager"
   required><br/><br/>
55             <input type="submit" value="Submit" id="sub">
56         </form>
57     </center>
58 </body>
59 </html>
```