PROJECT REPORT

PREDICTING HIGH POTENTIAL EMPLOYEES IN A CORPORATE

BY

KRISHNA SHARMA S

1.INTRODUCTION

1.1 OVERVIEW

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 behaviour 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.

1.2 PURPOSE

This project aims to let users rate the employees using a machine learning model trained and deployed using AWS.

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

Predicting the employee performance is crucial for a corporate. It is based on the employee performance that corporates can make crucial decisions regarding pay raises, promotions and releases. Also, when there are many employees, it becomes more difficult and time consuming to

appraise them. Hence, the process becomes time consuming and error prone.

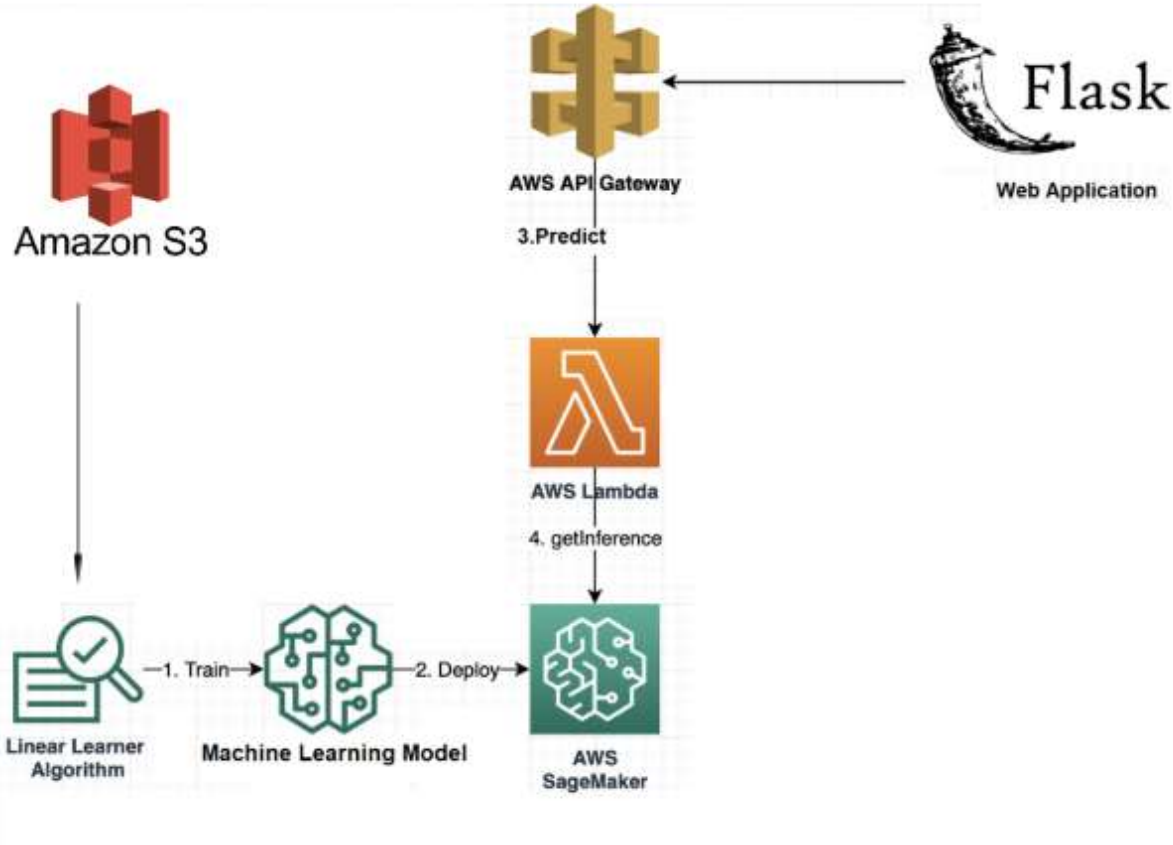
2.2 PROPOSED SOLUTION:

Build & Deploy a Machine Learning model to rate the employee performance using Amazon SageMaker.

Then, 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. Thus, the user will be able to predict the rating of the employee.

3. THEORITICAL ANALYSIS

3.1 FLOWCHART



3.2 SOFTWARE DESIGNING

The software designing involved the following steps:

1. PROJECT PLANNING AND KICKOFF:

- Understanding the project description and analyze the data and attributes in the given dataset.
- Plan how to proceed with project.

2. Exploring AWS Platform

3. Use Amazon S3 to store the Dataset

4. Train and Deploy the model using Amazon Sagemaker and create an Endpoint for the model deployed.

5. Use AWS Lambda to Invoke Endpoint.

6. Create an API Gateway to trigger the Lambda function.

7. Create a flask app using python to serve as the frontend.

4. EXPERIMENTAL INVESTIGATIONS

The Dataset:

The dataset has various information about employees like age, daily rate, monthly rate, job involvement, overtime etc. which can be used to predict the performance.

AWS Cloud:

Amazon Web Services offers reliable, scalable, and inexpensive cloud computing services.

Amazon S3:

Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services that provides object storage through a web service interface.

Amazon Sagemaker:

Amazon SageMaker enables developers to create, train, and deploy machine-learning models in the cloud. SageMaker also enables developers to deploy ML models on embedded systems and edge-devices.

AWS Lambda:

AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services.

Amazon API Gateway:

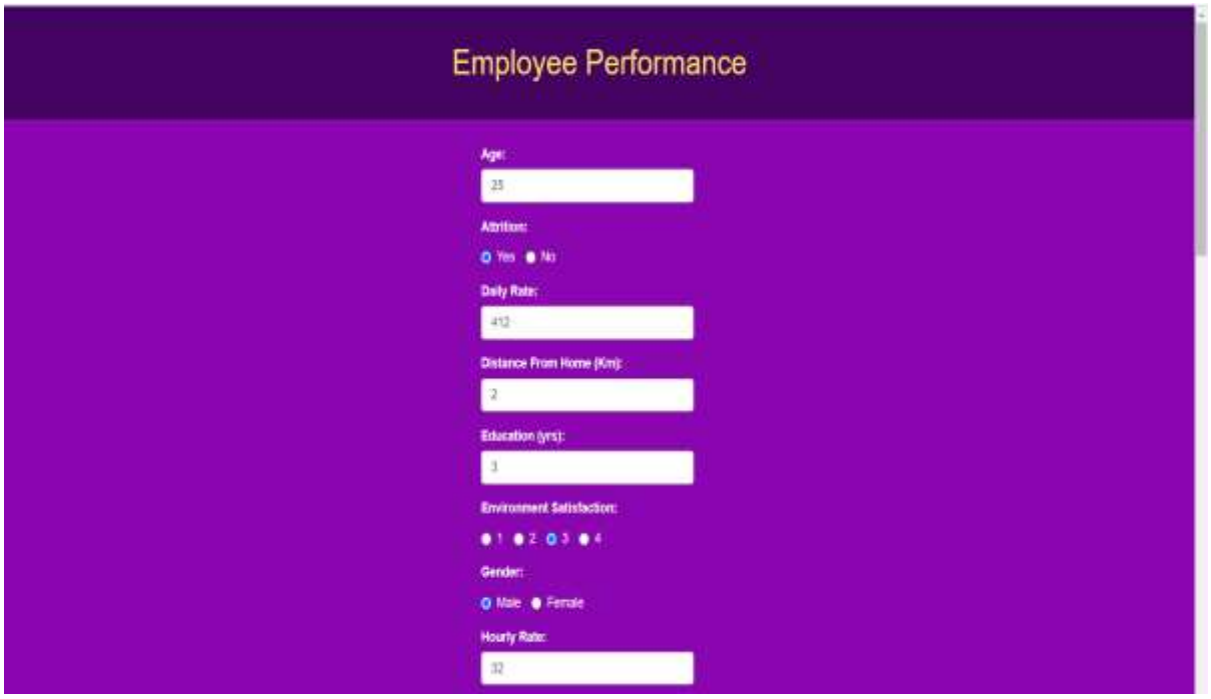
Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.

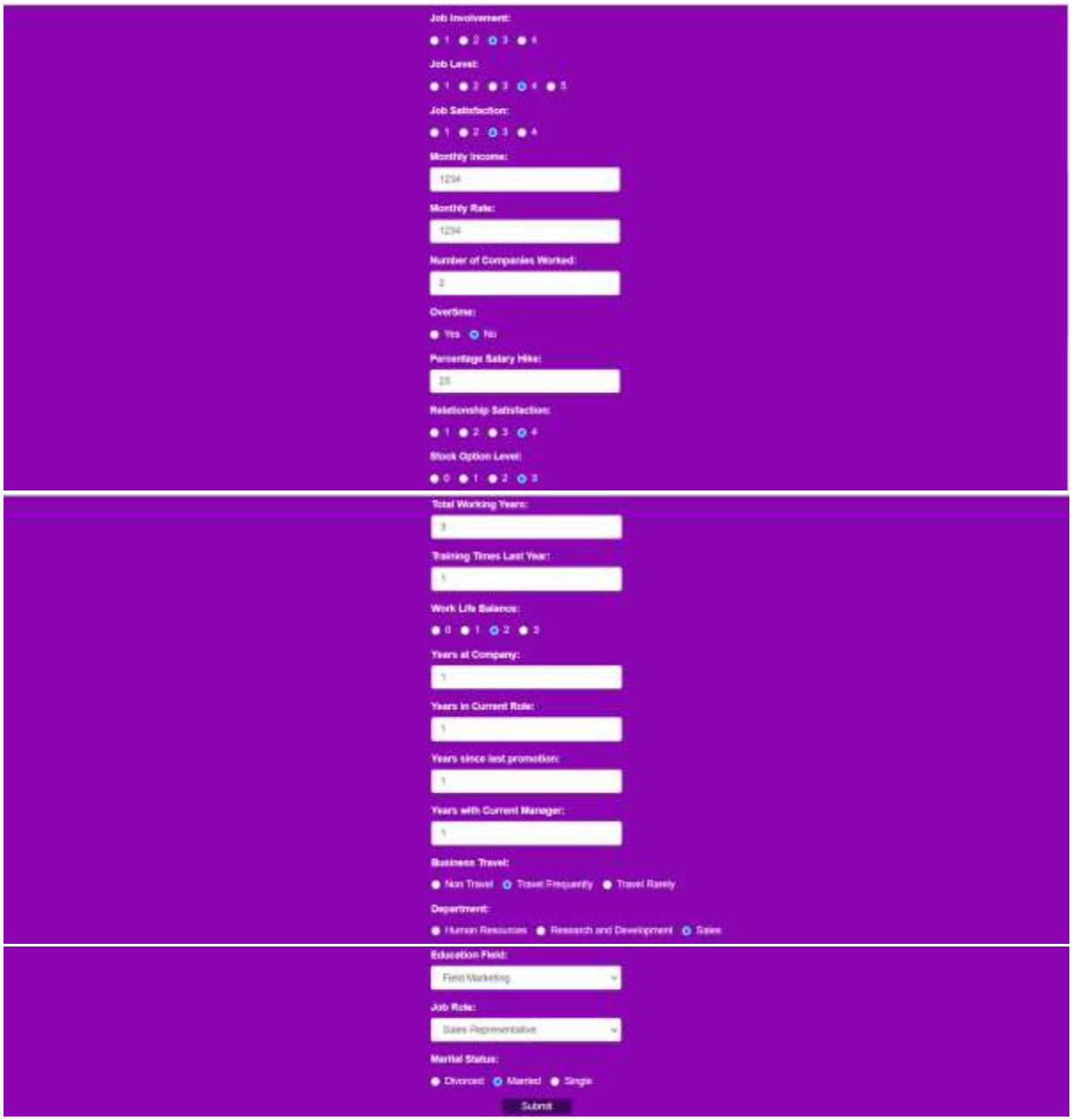
Flask:

Flask is a web framework. Flask provides users with tools, libraries and technologies that allows users to build a web application.

5. RESULT

Flask UI




Job Involvement:
1 2 3 4
Job Level:
1 2 3 4 5
Job Satisfaction:
1 2 3 4
Monthly Income:
1234
Monthly Rate:
1234
Number of Companies Worked:
2
OverSime:
Yes No
Percentage Salary Hike:
23
Relationship Satisfaction:
1 2 3 4
Stock Option Level:
0 1 2 3
Total Working Years:
3
Training Times Last Year:
1
Work Life Balance:
0 1 2 3
Years at Company:
1
Years in Current Role:
1
Years since last promotion:
1
Years with Current Manager:
1
Business Travel:
Non Travel Travel Frequently Travel Rarely
Department:
Human Resources Research and Development Sales
Education Field:
Field Marketing
Job Role:
Sales Representative
Marital Status:
Divorced Married Single
Submit

OUTPUT


Employee Performance
Rating : Great! Outstanding Performance!
Age:
Attrition:
Yes No

6. ADVANTAGES AND DISADVANTAGES

Advantages:

Machine learning technology typically improves efficiency and accuracy thanks to the ever-increasing amounts of data that are processed. The application learns the patterns and trends hidden within the data without human intervention which makes predicting much simpler and easier. The more data is fed to the algorithm, the higher the accuracy of the algorithm.

Deploying it in cloud makes the process simple and removes the need to worry about resources and availability.

Disadvantages:

Using machine learning interface comes with its own problems. Since the whole point of it is minimize human involvement, it also makes error detection and fixing much more problematic. It takes a lot of time to identify the root cause for the problem.

Machine learning can also be very resource consuming especially while dealing with large amount of data.

7. APPLICATIONS

- It is useful for large corporates with many employees.
- Individuals may also use this to predict their performance rating.

8. CONCLUSION

- This model takes into account many factors like job involvement, overtime, work life balance etc. to predict the performance rating of employees.
- AWS was used to train and deploy the machine learning model and this reduced the complexity of the task.
- A user-friendly UI built with flask was used to interact with the deployed model.

9. FUTURE SCOPE

- The UI can be made to look more attractive.
- More details about the employee could be obtained and displayed.

10. BIBLIOGRAPHY

1. Dataset reference https://www.kaggle.com/patelprashant/employee-attrition
2. Sagemaker https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/
3. AWS Lambda and API Gateway https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html

APPENDIX

1. Source Code

Employee Notebook

Import libraries

In [1]:
```
import boto3, re, sys, math, json, os, sagemaker, urllib.request
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer
```

Define IAM role

In [2]:
```
role = get_execution_role()

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'} # each region has its XGBoost containe
my_region = boto3.session.Session().region_name # set the region of the instance
print("Success - the MySageMakerInstance is in the " + my_region + " region. You will use the " + containers[my_region] + " cont
```

```
Success - the MySageMakerInstance is in the us-east-1 region. You will use the 811284229777.dkr.ecr.us-east-1.amazonaws.com/xgb
oost:latest container for your SageMaker endpoint.
```

In [3]:
```
bucket_name = 'sagemaker-employee' # <--- CHANGE THIS VARIABLE TO A UNIQUE NAME FOR YOUR BUCKET
s3 = boto3.resource('s3')
```

In [4]:
```
data= pd.read_csv('s3://sagemaker-employee/HR-Employee-Attrition.csv')
data.head()
```

Out[4]:

	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

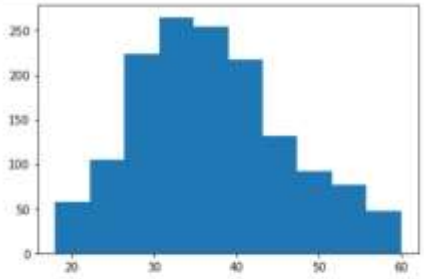
In [5]:
```
data.isnull().sum()
```

Out[5]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

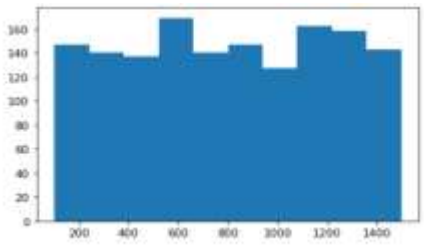
There are no NULL values

Data Visualization

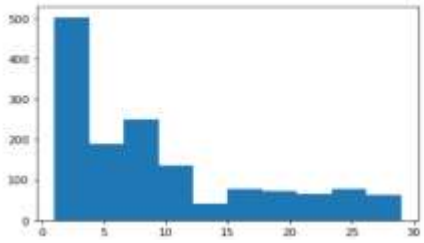
In [6]:
```
plt.hist(data['Age'])
plt.show()
```



In [7]:
```
plt.hist(data['DailyRate'])
plt.show()
```



In [8]:
```
plt.hist(data['DistanceFromHome'])
plt.show()
```



In [9]:
```
data.shape
```

Out[9]:
```
(1470, 35)
```

In [10]:
```
data['Over18'].value_counts()
```

Out[10]:
```
Y    1470
Name: Over18, dtype: int64
```

In [11]:
```
data['StandardHours'].value_counts()
```

Out[11]:
```
80    1470
Name: StandardHours, dtype: int64
```

In [12]:
```
data['EmployeeCount'].value_counts()
```

Out[12]:
```
1    1470
Name: EmployeeCount, dtype: int64
```

these Over18, StandardHours, EmployeeCount have constant values and can be dropped.

In [13]:
```
data = data.drop(['Over18', 'StandardHours', 'EmployeeCount', 'EmployeeNumber'], axis = 1)
```

In [14]:
```
data.shape
```

Out[14]:
```
(1470, 31)
```

Perform One Hot Encoding

In [15]:
```
data.loc[data['Attrition']=='No','Attrition'] = 0
data.loc[data['Attrition']=='Yes','Attrition'] = 1
```

In [16]:
```
data.loc[data['OverTime']=='No','OverTime'] = 0
data.loc[data['OverTime']=='Yes','OverTime'] = 1
```

In [17]:
```
data.loc[data['Gender']=='Male','Gender'] = 1
data.loc[data['Gender']=='Female','Gender'] = 0
```

In [18]:
```
data['Business_Travel_Rarely']=0
data['Business_Travel_Frequently']=0
data['Business_Non-Travel']=0

data.loc[data['BusinessTravel']=='Travel_Rarely','Business_Travel_Rarely'] = 1
data.loc[data['BusinessTravel']=='Travel_Frequently','Business_Travel_Frequently'] = 1
data.loc[data['BusinessTravel']=='Non-Travel','Business_Non-Travel'] = 1
```

In [19]:
```
data['Department_Sales']=0
data['Department_R&D']=0
data['Department_Dept_Human Resources'] =0

data.loc[data['Department']=='Sales','Department_Sales'] = 1
data.loc[data['Department']=='Research & Development','Department_R&D'] = 1
data.loc[data['Department']=='Human Resources','Department_Dept_Human Resources'] = 1
```

In [20]:
```
data['EducationField_Life Sciences']=0
data['EducationField_Medical']=0
data['EducationField_Marketing']=0
data['EducationField_Technical Degree']=0
data['EducationField_Education Human Resources']=0
data['EducationField_Education_Other']=0

data.loc[data['EducationField']=='Life Sciences','EducationField_Life Sciences'] = 1
data.loc[data['EducationField']=='Medical','EducationField_Medical'] = 1
data.loc[data['EducationField']=='Other','EducationField_Education_Other'] = 1
data.loc[data['EducationField']=='Technical Degree','EducationField_Technical Degree'] = 1
data.loc[data['EducationField']=='Human Resources','EducationField_Education Human Resources'] = 1
data.loc[data['EducationField']=='Marketing','EducationField_Marketing'] = 1
```

In [21]:
```
data['JobRole_Research Scientist']=0
data['JobRole_Laboratory Technician']=0
data['JobRole_Sales Executive']=0
data['JobRole_Manufacturing Director']=0
data['JobRole_Healthcare Representative']=0
data['JobRole_Sales Representative']=0
data['JobRole_Research Director']=0
data['JobRole_Manager'] = 0
data['JobRole_Job_Human_Resources'] = 0

data.loc[data['JobRole']=='Research Scientist','JobRole_Research Scientist'] = 1
data.loc[data['JobRole']=='Laboratory Technician','JobRole_Laboratory Technician'] = 1
data.loc[data['JobRole']=='Sales Executive','JobRole_Sales Executive'] = 1
data.loc[data['JobRole']=='Sales Representative','JobRole_Sales Representative'] = 1
data.loc[data['JobRole']=='Manufacturing Director','JobRole_Manufacturing Director'] = 1
data.loc[data['JobRole']=='Healthcare Representative','JobRole_Healthcare Representative'] = 1
data.loc[data['JobRole']=='Research Director','JobRole_Research Director'] = 1
data.loc[data['JobRole']=='Manager','JobRole_Manager'] = 1
data.loc[data['JobRole']=='Human Resources','JobRole_Job_Human_Resources'] = 1
```

In [22]:
```
data['MaritalStatus_Single']=0
data['MaritalStatus_Married']=0
data['MaritalStatus_Divorced']=0

data.loc[data['MaritalStatus']=='Married','MaritalStatus_Married'] = 1
data.loc[data['MaritalStatus']=='Single','MaritalStatus_Single'] = 1
data.loc[data['MaritalStatus']=='Divorced','MaritalStatus_Divorced'] = 1
```

In [23]:
```
data = data.drop(['BusinessTravel','EducationField',
                  'Department','JobRole','MaritalStatus'],axis=1)
data.head()
```

Out[23]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	HourlyRate	JobInvolvement	JobLevel	...	JobRole_Sales Executive	JobR
0	41	1	1102	1	2	2	0	94	3	2	...	1	
1	49	0	279	8	1	3	1	61	2	2	...	0	
2	37	1	1373	2	2	4	1	92	2	1	...	0	
3	33	0	1392	3	4	4	0	56	3	1	...	0	
4	27	0	591	2	1	1	1	40	3	1	...	0	

5 rows × 50 columns

In [24]:
```
data.columns
```

Out[24]:
```
Index(['Age', 'Attrition', 'DailyRate', 'DistanceFromHome', 'Education',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate',
       'NumCompaniesWorked', 'OverTime', 'PercentSalaryHike',
       'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager', 'Business_Travel_Rarely',
       'Business_Travel_Frequently', 'Business_Non-Travel', 'Department_Sales',
       'Department_R&D', 'Department_Dept_Human Resources',
       'EducationField_Life Sciences', 'EducationField_Medical',
       'EducationField_Marketing', 'EducationField_Technical Degree',
       'EducationField_Education Human Resources',
       'EducationField_Education_Other', 'JobRole_Research Scientist',
       'JobRole_Laboratory Technician', 'JobRole_Sales Executive',
       'JobRole_Manufacturing Director', 'JobRole_Healthcare Representative',
       'JobRole_Sales Representative', 'JobRole_Research Director',
       'JobRole_Manager', 'JobRole_Job_Human_Resources',
       'MaritalStatus_Single', 'MaritalStatus_Married',
       'MaritalStatus_Divorced'],
      dtype='object')
```

Building the model

In [25]:
```
from sklearn.model_selection import train_test_split

train_x = data.drop(['PerformanceRating'],axis=1)
train_y = data['PerformanceRating']
train_x.insert(0, 'PerformanceRating', train_y)

X,test_x,Y,test_y = train_test_split(train_x, train_y, test_size=0.2,random_state=42)
```

In [26]:
```
train_x.head()
```

Out[26]:

	PerformanceRating	Age	Attrition	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	HourlyRate	JobInvolvement	...	JobRole_Sal Executi
0	3	41	1	1102	1	2	2	0	94	3	...	
1	4	49	0	279	8	1	3	1	61	2	...	
2	3	37	1	1373	2	2	4	1	92	2	...	
3	3	33	0	1392	3	4	4	0	56	3	...	
4	3	27	0	591	2	1	1	1	40	3	...	

5 rows × 50 columns

In [27]:
```
train_x.to_csv('train.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join('train/train.csv')).upload_file('train.csv')
s3_input_train = sagemaker.s3_input(s3_data='s3://{}/train'.format(bucket_name), content_type='csv')
```

```
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python SDK v2.
```

In [28]:
```
sess = sagemaker.Session()
xgb = sagemaker.estimator.Estimator(containers[my_region],role, train_instance_count=1, train_instance_type='ml.m5.large',output_
```

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

In [29]:
```
xgb.set_hyperparameters(objective="multi:softmax",colsample_bytree= 0.5 ,learning_rate= 0.075,
                        max_depth=5, alpha= 8, num_round=100,num_class=5)
```

```
In [30]: xgb.fit({'train': s3_input_train})
```

```
2020-10-03 17:09:55 Starting - Starting the training job...
2020-10-03 17:09:57 Starting - Launching requested ML instances......
2020-10-03 17:11:06 Starting - Preparing the instances for training...
2020-10-03 17:11:51 Downloading - Downloading input data
2020-10-03 17:11:51 Training - Downloading the training image...
2020-10-03 17:12:25 Training - Training image download completed. Training in progress..Arguments: train
[2020-10-03:17:12:25:INFO] Running standalone xgboost training.
[2020-10-03:17:12:25:INFO] Path /opt/ml/input/data/validation does not exist!
[2020-10-03:17:12:25:INFO] File size need to be processed in the node: 0.16mb. Available memory size in the node: 171.71mb
[2020-10-03:17:12:25:INFO] Determined delimiter of CSV input is ','
[17:12:25] S3DistributionType set as FullyReplicated
[17:12:25] 1470x49 matrix with 72030 entries loaded from /opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
[17:12:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[17:12:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[17:12:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
[17:12:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned nodes, max_depth=1
[17:12:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=5
[0]#011train-merror:0
[17:12:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 0 pruned nodes, max_depth=0
```

Deploy model

```
In [31]: xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type='ml.m5.large')
```

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

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

```
In [32]: test_data_array = test_x.drop(['PerformanceRating'],axis=1).values
         xgb_predictor.content_type = 'text/csv'
         xgb_predictor.serializer = csv_serializer # set the serializer type
         predictions = xgb_predictor.predict(test_data_array).decode('utf-8') # predict!
         predictions_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array
         print(predictions_array.shape)
```

```
(294,)
```

Evaluate Model

```
In [33]: from sklearn.metrics import mean_squared_error
         from math import sqrt
         rmse = sqrt(mean_squared_error(test_y,predictions_array))
         rmse
```

```
Out[33]: 0.1749635530559413
```

Delete Endpoint

```
In [ ]: #sagemaker.Session().delete_endpoint(xgb_predictor.endpoint)
```

Lambda Function

```
import os

import io

import boto3

import json

import csv


# grab environment variables

ENDPOINT_NAME = os.environ['ENDPOINT_NAME']
```

```
runtime= boto3.client('runtime.sagemaker')


def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))
    data = json.loads(json.dumps(event))
    payload = data['body']
    print('data',data)
    response                                                    =
runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                ContentType='text/csv',
                                Body=payload)
    print('res',response)
    result = json.loads(response['Body'].read().decode())
    print('result',type(result))
    if(result == 0.0 ):
        prediction = 'Bad.'
    elif(result == 0.1):
        prediction = 'Not Satisfactory.'
    elif(result == 0.2):
        prediction = 'Satisfactory'
    elif(result==3.0):
        prediction = 'Good!'
    elif(result == 4.0):
        prediction = 'Great!'
```

```
    return prediction
```

Flask

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import json
import requests

app = Flask(__name__)
URL = "https://jjyhrpmbd5.execute-api.us-east-
1.amazonaws.com/test/employee"

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/y_predict', methods = ['POST'])
def y_predict():
    req = request.form
    age = req.get('age')
    att = req.get('attradio')
    dRate = req.get('dRate')
    dist = req.get('dist')
    edu = req.get('edu')
```

```
envradio = req.get('envradio')
Genderradio = req.get('Genderradio')
hrr = req.get('hrr')
JIradio = req.get('JIradio')
JLradio = req.get('JLradio')
JSradio = req.get('JSradio')
MI = req.get('MI')
Mrate = req.get('Mrate')
NumComp = req.get('NumComp')
Overtimeradio = req.get('Overtimeradio')
PercentHike = req.get('PercentHike')
PerfRateradio = req.get('PerfRateradio')
RSradio = req.get('RSradio')
StockOLradio = req.get('StockOLradio')
WorkingYrs = req.get('WorkingYrs')
TTLY = req.get('TTLY')
WLbalanceradio = req.get('WLbalanceradio')
YrsComp = req.get('YrsComp')
YrsCurrent = req.get('YrsCurrent')
YrsPromotion = req.get('YrsPromotion')
YrsCurrManager = req.get('YrsCurrManager')
Bradio = req.get('Bradio')
if(int(Bradio) == 1):
    Non_travel = 1
```

```
    travel_freq = 0
    travel_r = 0
elif(int(Bradio) == 2):
    Non_travel = 0
    travel_freq = 1
    travel_r = 0
else:
    Non_travel = 0
    travel_freq = 0
    travel_r = 1
Deptradio = req.get('Deptradio')
if(int(Deptradio) == 1):
    hRes = 1
    RnD = 0
    sales = 0
elif(int(Deptradio) == 2):
    hRes = 0
    RnD = 1
    sales = 0
else:
    hRes = 0
    RnD = 0
    sales = 1
EduField = req.get('EduField')
```

```
if(int(EduField) == 1):
    hr_field = 1
    ls_field = 0
    fm_field = 0
    med_field = 0
    td_field = 0
    o_field =0
elif(int(EduField) == 2):
    hr_field = 0
    ls_field = 1
    fm_field = 0
    med_field = 0
    td_field = 0
    o_field =0
elif(int(EduField) == 3):
    hr_field = 0
    ls_field = 0
    fm_field = 1
    med_field = 0
    td_field = 0
    o_field =0
elif(int(EduField) == 4):
    hr_field = 0
    ls_field = 0
```

```
    fm_field = 0
    med_field = 1
    td_field = 0
    o_field =0
elif(int(EduField) == 5):
    hr_field = 0
    ls_field = 0
    fm_field = 0
    med_field = 0
    td_field = 1
    o_field =0
else:
    hr_field = 0
    ls_field = 0
    fm_field = 0
    med_field = 0
    td_field = 0
    o_field = 1
JobRole = req.get('JobRole')
if(int(JobRole) == 1):
    health_jr = 1
    hr_jr = 0
    lt_jr =0
    man_jr = 0
```

```
    md_jr = 0
    rd_jr = 0
    rs_jr = 0
    se_jr = 0
    sr_jr = 0
elif(int(JobRole) == 2):
    health_jr = 0
    hr_jr = 1
    lt_jr =0
    man_jr = 0
    md_jr = 0
    rd_jr = 0
    rs_jr = 0
    se_jr = 0
    sr_jr = 0
elif(int(JobRole) == 3):
    health_jr = 0
    hr_jr = 0
    lt_jr =1
    man_jr = 0
    md_jr = 0
    rd_jr = 0
    rs_jr = 0
    se_jr = 0
```

```
    sr_jr = 0
elif(int(JobRole) == 4):
    health_jr = 0
    hr_jr = 0
    lt_jr =0
    man_jr = 1
    md_jr = 0
    rd_jr = 0
    rs_jr = 0
    se_jr = 0
    sr_jr = 0
elif(int(JobRole) == 5):
    health_jr = 0
    hr_jr = 0
    lt_jr =0
    man_jr = 0
    md_jr = 1
    rd_jr = 0
    rs_jr = 0
    se_jr = 0
    sr_jr = 0
elif(int(JobRole) == 6):
    health_jr = 0
    hr_jr = 0
```

```
    lt_jr =0
    man_jr = 0
    md_jr = 0
    rd_jr = 1
    rs_jr = 0
    se_jr = 0
    sr_jr = 0
elif(int(JobRole) == 7):
    health_jr = 0
    hr_jr = 0
    lt_jr =0
    man_jr = 0
    md_jr = 0
    rd_jr = 0
    rs_jr = 1
    se_jr = 0
    sr_jr = 0
elif(int(JobRole) == 8):
    health_jr = 0
    hr_jr = 0
    lt_jr =0
    man_jr = 0
    md_jr = 0
    rd_jr = 0
```

```
    rs_jr = 0
    se_jr = 1
    sr_jr = 0
else:
    health_jr = 0
    hr_jr = 0
    lt_jr =0
    man_jr = 0
    md_jr = 0
    rd_jr = 0
    rs_jr = 0
    se_jr = 0
    sr_jr = 1
MaritalStatusradio = req.get('MaritalStatusradio')
if(int(MaritalStatusradio) == 1):
    div = 1
    mar = 0
    sing = 0
elif(int(MaritalStatusradio) ==2):
    div = 0
    mar = 1
    sing = 0
else:
    div = 0
```

```
    mar = 0
    sing = 1

print(age,att,dRate,dist,edu,envradio,Genderradio,hrr,JIradio,JLradio
,JSradio,

MI,Mrate,NumComp,Overtimeradio,PercentHike,RSradio,StockOLrad
io,

WorkingYrs,TTLY,WLbalanceradio,YrsComp,YrsCurrent,YrsPromotion,
YrsCurrManager,

travel_r,travel_freq,Non_travel,sales,RnD,hRes,ls_field,med_field,fm
_field,

td_field,hr_field,o_field,rs_jr,lt_jr,se_jr,md_jr,health_jr,sr_jr,rd_jr,
      man_jr,hr_jr,sing,mar,div)
  body_list =
[age,att,dRate,dist,edu,envradio,Genderradio,hrr,JIradio,JLradio,JSra
dio,

MI,Mrate,NumComp,Overtimeradio,PercentHike,RSradio,StockOLrad
io,

WorkingYrs,TTLY,WLbalanceradio,YrsComp,YrsCurrent,YrsPromotion,
YrsCurrManager,

travel_r,travel_freq,Non_travel,sales,RnD,hRes,ls_field,med_field,fm
_field,
```

```
td_field,hr_field,o_field,rs_jr,lt_jr,se_jr,md_jr,health_jr,sr_jr,rd_jr,
        man_jr,hr_jr,sing,mar,div]
    body_string = ",".join([str(body) for body in body_list])
    print(body_string)
    data_body = json.dumps({"body": body_string})
    print(type(body_string),type(data_body))
    print(data_body)
    r = requests.post(url = URL, data = data_body)
    print(r)
    print(r.status_code, r.reason, r.text)
    r_text  = r.text.strip('\"')
    if(r_text == "Bad"):
        final_text = 'Rating : {}.Needs to improve a lot.'.format(r_text)
    elif(r_text == "Not Satisfactory"):
        final_text = 'Rating : {}. A lot of room for
improvement.'.format(r_text)
    elif(r_text == "Satisfactory"):
        final_text = 'Rating : {}.There is still room for
improvement.'.format(r_text)
    elif(r_text == "Good!"):
        final_text = 'Rating : {} Performance is above
average!'.format(r_text)
    elif(r_text == "Great!"):
        final_text = 'Rating : {} Outstanding Performance!'.format(r_text)
```

```
    return render_template('index.html',final_text = final_text)


if __name__ == '__main__':
    app.run(debug =True)
```