

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

2.LITERATURE SURVEY

2.1 Existing problem

2.2 Proposed solution

3.THEORETICAL ANALYSIS

3.1 Block diagram

3.2 Hardware/Software designing

4.EXPERIMENTAL INVESTIGATIONS

5.RESULT

6.ADVANTAGES & DISADVANTAGES

7.APPLICATIONS

8. FUTURE SCOPE AND CONCLUSION

9.BIBLIOGRAPHY

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:

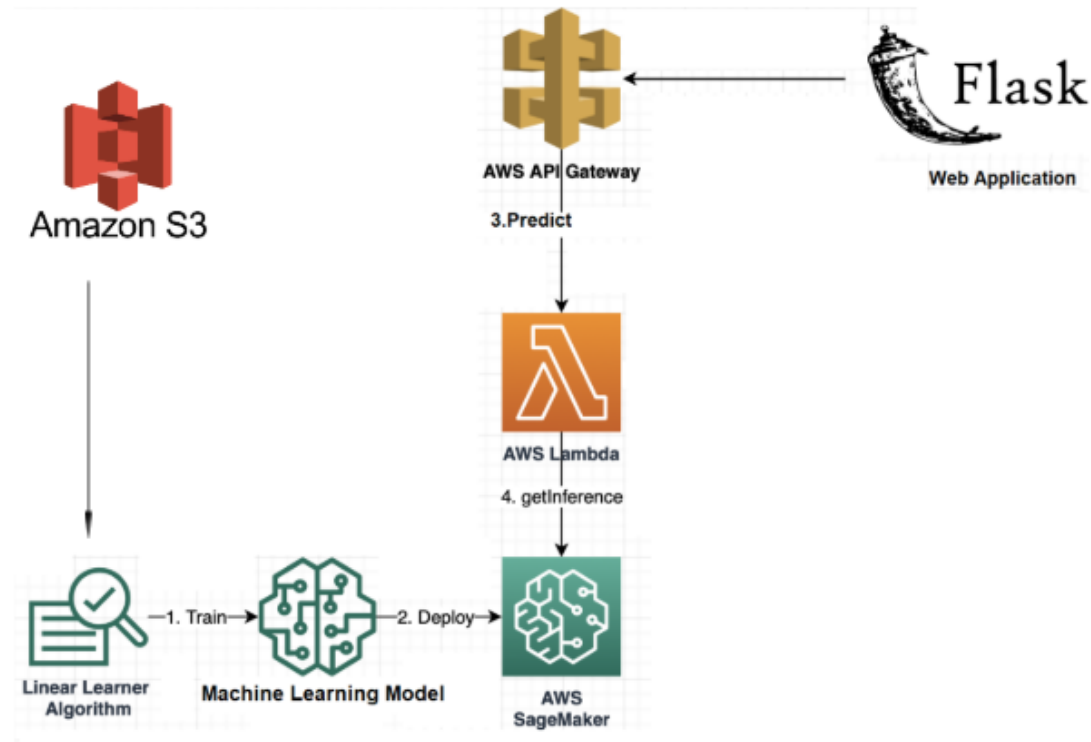
Employees are the key resources of the organization. The success or failure of an organization depends on the employee. Most of the organizations or companies have a formal performance evaluation system in which employee job performance is graded on a regular basis, usually once or twice a year. A good 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. SageMaker removes the heavy lifting from each step of the machine learning process to make it easier to develop 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, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud. API Gateway creates RESTful APIs that are HTTP-based.

5.RESULT:

Browser tabs: Fwd: - prabha.mathi519@gmail.com, Student Dashboard, Sample Form

Address bar: 127.0.0.1:5000

Navigation bar: Apps, Neural machine tra..., MT Tab-delimited Bilin..., Time series forecast..., Language Translat..., Intuitive Understan..., Word Level English..., Image captioning w..., Text generation wit...

Form fields:

- Enter Dailyrate
- Enter Distance From Home
- LifeScience ☐ Medical ☐ Marketing ☐
- Technical Degree ☐ Human Resources ☐ Others ☐
- Enter Envirinment Satisfaction
- Enter Hourly Rate
- Enter Monthly rate
- Enter Training Times Last Year
- Enter Years With Current Manager
- Submit

Windows watermark: Activate Windows. Go to Settings to activate Windows. Show all X

Taskbar: Type here to search, File Explorer, Google Chrome, Microsoft Edge, VS Code, PC icon

System tray: 15:10, 03-10-2020

Browser tabs: Fwd: - prabha.mathi519@gmail.com, Student Dashboard, Sample Form

Address bar: 127.0.0.1:5000

Navigation bar: Apps, Neural machine tra..., MT Tab-delimited Bilin..., Time series forecast..., Language Translat..., Intuitive Understan..., Word Level English..., Image captioning w..., Text generation wit...

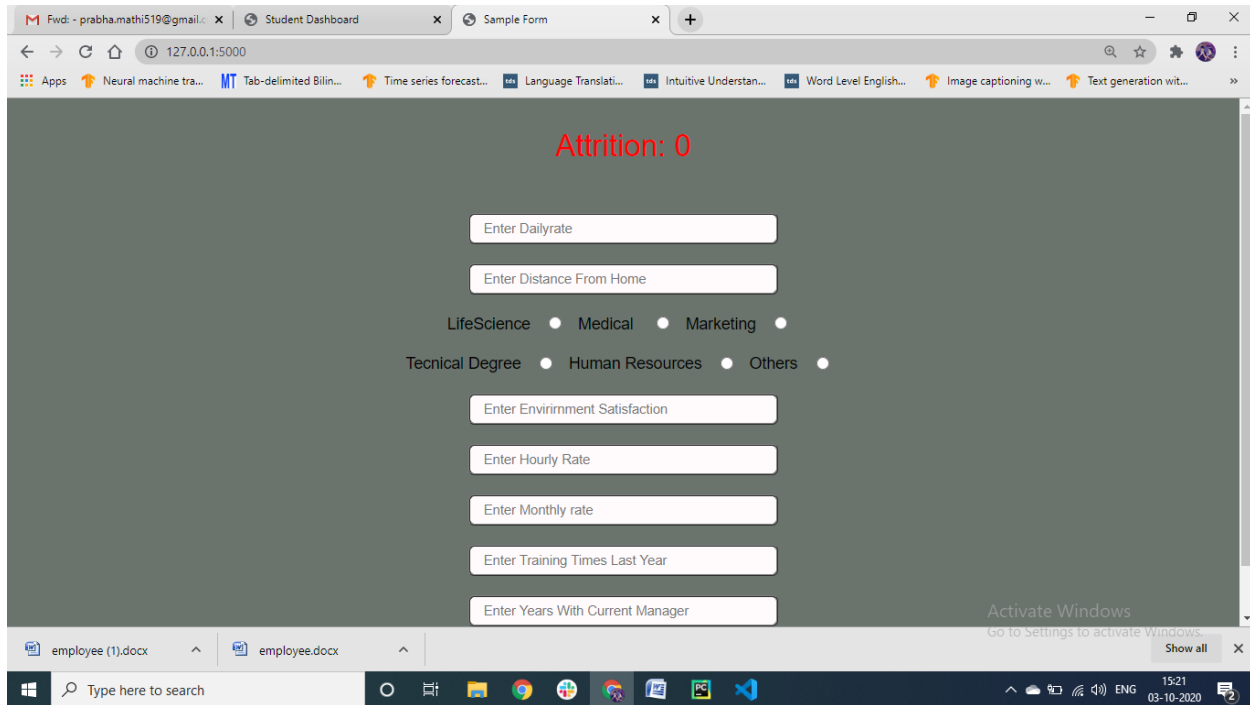
Form fields:

- 800
- 4
- LifeScience ☐ Medical ☐ Marketing ☒
- Technical Degree ☐ Human Resources ☐ Others ☐
- 4
- 90
- 20000
- 3
- 2
- Submit

Windows watermark: Activate Windows. Go to Settings to activate Windows. Show all X

Taskbar: Type here to search, File Explorer, Google Chrome, Microsoft Edge, VS Code, PC icon

System tray: 15:20, 03-10-2020



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 best thing is the capture of data around interventions done by the organization for at-risk 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 it is recommended to use XGBoost for accurately predicting employee turnover, thus enabling organizations to take actions for retention or succession of employees.

9. BIBLIOGRAPHY:

- 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

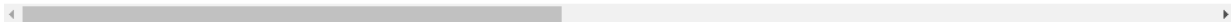
Code:

```
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   PercentsSalaryHike                 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
1	49	0	1	279	1	8	1	1	1	2	...	2
2	37	1	2	1373	1	2	2	4	1	4	...	4
3	33	0	1	1392	1	3	4	1	1	5	...	5
4	27	0	2	591	1	2	1	3	1	7	...	7

5 rows × 35 columns

```

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

```

```

4     if i==1:
5         continue
6     else:
7
        person_cof,p_value=stats.pearsonr(dataset.iloc[:,i],dataset
        .iloc[:,25])
8         print('{0}    person cofficent {1}        p value {2}
        '.format(keys[i],person_cof,p_value))

```

```

Age    person cofficent 0.05353471967122897    p value 0.04014266700714043
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
RelationshipSatisfaction    person cofficent 0.9999999999999996    p value 0.0
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

```

```

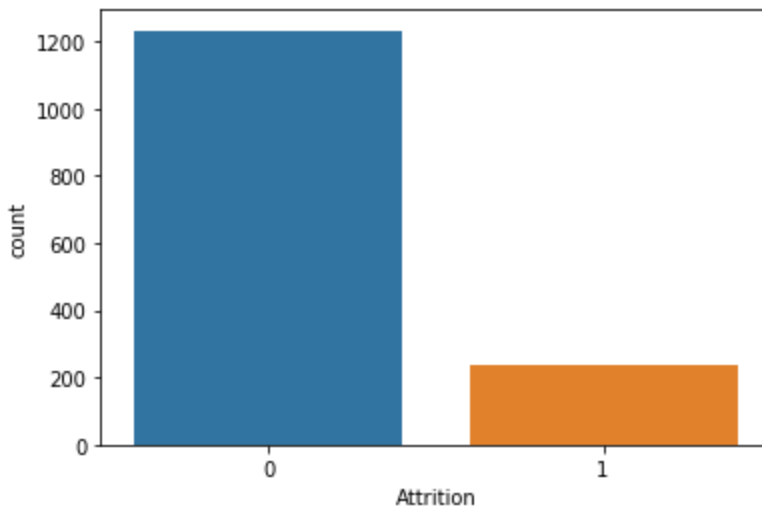
1  data=dataset[['DailyRate','DistanceFromHome','EducationFiel
    d','EnvironmentSatisfaction','HourlyRate','MonthlyRate','Tr
    ainingTimesLastYear','YearsWithCurrManager','Attrition']]
2  data.head()

```

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

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

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

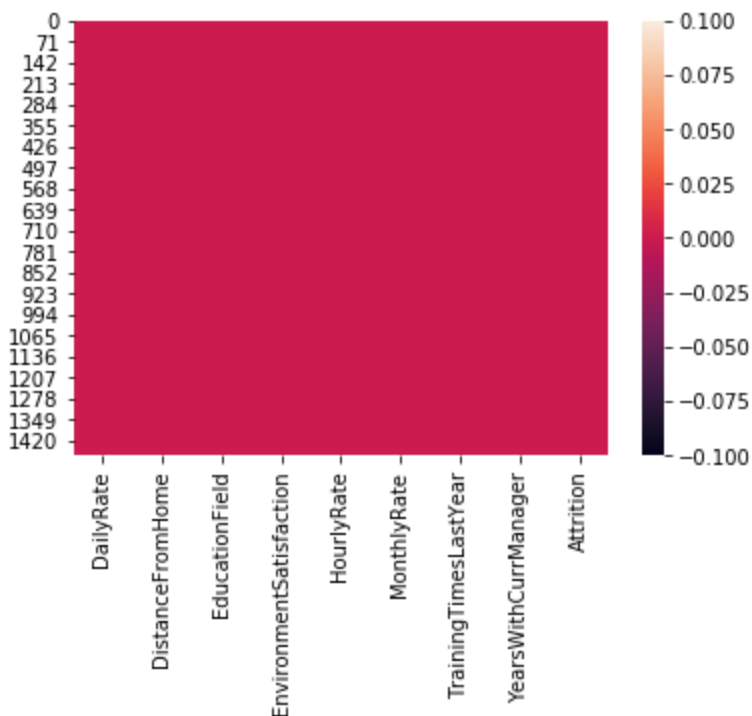


```
1 data.isnull().sum()
```

```
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]})
9 dataset=pd.DataFrame(dici)
10 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

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

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

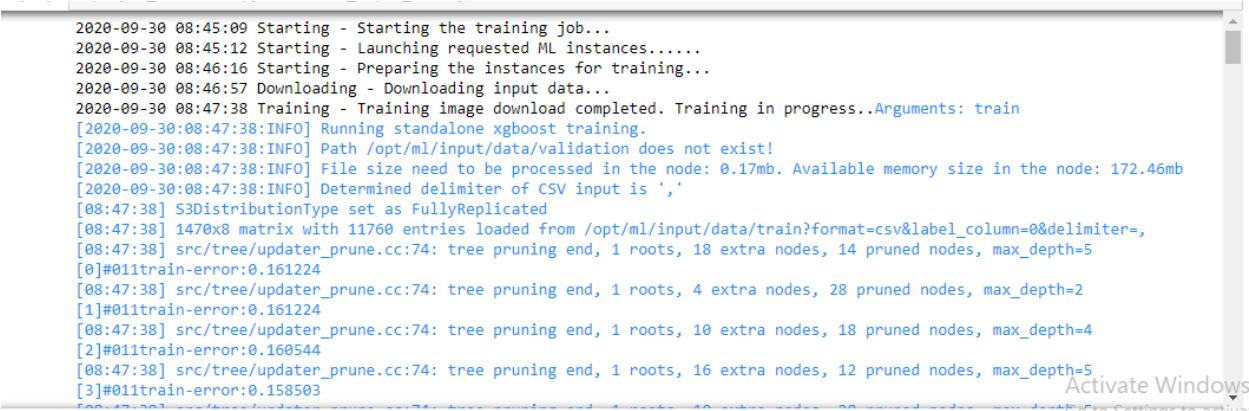
```
1 train,test=train_test_split(final_data,test_size=0.2)
```

```
1 import boto3,re,os,json,sagemaker
2 from sagemaker import get_execution_role
3 role=get_execution_role()
4 my_region=boto3.session.Session().region_name
5 containers = {'us-west-2':
6               '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:lates
7               t',
8               'us-east-1':
9               '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:lates
10              t',
11              'us-east-2':
12              '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:lates
13              t',
14              'eu-west-1':
15              '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:lates
16              t'}
17 prefix='sagemaker/Employe'
18 bucket_name='buildathonproject1'
19 final_data.to_csv('train.csv',index=False,header=False)
20 boto3.Session().resource('s3').Bucket(bucket_name).Object(o
21 s.path.join(prefix,'train/train.csv')).upload_file('train.c
22 sv')
23 s3_input_train=sagemaker.s3_input(s3_data='s3://{}/{}/train
24 '.format(bucket_name, prefix),content_type='csv')
25 sess=sagemaker.Session()
26 employee_model=sagemaker.estimator.Estimator(containers[my_
27 region],role,train_instance_count=1,train_instance_type='ml
28 .m5.large',output_path='s3://{}/{}/output'.format(bucket_na
29 me,prefix),sagemaker_session=sess)
```

```

16 employee_model.set_hyperparameters(max_depth=5,eta=0.2,gamm
   a=4,min_child_weight=6,subsample=0.8,silent=0,objective='bi
   nary:logistic',num_round=100)
17 employee_model.fit({'train':s3_input_train})

```



```

2020-09-30 08:45:09 Starting - Starting the training job...
2020-09-30 08:45:12 Starting - Launching requested ML instances.....
2020-09-30 08:46:16 Starting - Preparing the instances for training...
2020-09-30 08:46:57 Downloading - Downloading input data...
2020-09-30 08:47:38 Training - Training image download completed. Training in progress..Arguments: train
[2020-09-30:08:47:38:INFO] Running standalone xgboost training.
[2020-09-30:08:47:38:INFO] Path /opt/ml/input/data/validation does not exist!
[2020-09-30:08:47:38:INFO] File size need to be processed in the node: 0.17mb. Available memory size in the node: 172.46mb
[2020-09-30:08:47:38:INFO] Determined delimiter of CSV input is ','
[08:47:38] S3DistributionType set as FullyReplicated
[08:47:38] 1470x8 matrix with 11760 entries loaded from /opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
[08:47:38] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18 extra nodes, 14 pruned nodes, max_depth=5
[0]#011train-error:0.161224
[08:47:38] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 28 pruned nodes, max_depth=2
[1]#011train-error:0.161224
[08:47:38] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10 extra nodes, 18 pruned nodes, max_depth=4
[2]#011train-error:0.160544
[08:47:38] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16 extra nodes, 12 pruned nodes, max_depth=5
[3]#011train-error:0.158503

```

```

1 detector=employee_model.deploy(initial_instance_count=1,ins
   tance_type='ml.m5.large')

```

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

-----!

```

1 detector.endpoint

```

```
'xgboost-2020-09-30-08-45-09-662'
```

```

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

```


0.23525457583, 0.0621659755707, 0.19459120321, 0.154024705291, 0.0698856264353, 0.424703359604, 0.136716663837, 0.0600500926375, 0.0924533009529, 0.20742220382, 0.127422377467, 0.240950152278, 0.0462073560119, 0.128579318523, 0.077395199387, 0.0982446074486, 0.055473282298, 0.129804804921, 0.22340019584, 0.474873393774, 0.14097569026, 0.0789168924093, 0.048984404603, 0.139528542757, 0.252831488848, 0.512037336826, 0.595387458801, 0.0783490091562, 0.128582835197, 0.0694169476628, 0.126682803035, 0.248284742236, 0.103055216372, 0.34826235877, 0.0495620518923, 0.109172788989, 0.0324510075152, 0.11202154033, 0.085127800703, 0.062438054886, 0.160732358694, 0.159619902349, 0.247369021014, 0.188158448068, 0.291853129864, 0.3474984169, 0.087158113718, 0.404733866453, 0.12131700885, 0.168491870165, 0.0381562262774, 0.0317437201738, 0.103937707841, 0.100567348301, 0.107712209225, 0.0325755253434, 0.0343840196729, 0.210582122207, 0.0646713227034, 0.619333148003, 0.0303777884692, 0.084062756284, 0.272539705038, 0.168336028653, 0.149052456021, 0.102004952729, 0.18825968914, 0.276600748301, 0.521488070488, 0.11724550277, 0.0642483370653, 0.141158148602, 0.221530467272, 0.0541391409034, 0.328638139201, 0.186246216297, 0.0669659078121, 0.181038767099, 0.344874113798, 0.288093686104, 0.117713622749, 0.175055190921, 0.154918834567, 0.023147308639, 0.106085784733, 0.131366148591, 0.213091850281, 0.319343566895, 0.15413069725, 0.0265471227467, 0.024955118075, 0.09494987726, 0.196030810475, 0.081547526895, 0.143944576383, 0.066569674854, 0.19584585796, 0.090035259723, 0.217776224017, 0.0624685436487, 0.0657070875168, 0.0723061934114, 0.143281638622, 0.273711735191, 0.474400192499, 0.063778012991, 0.0620041005313, 0.116630025298, 0.066384114325, 0.037969255023, 0.105074733496, 0.15095712245, 0.193218648434, 0.0675120577216, 0.0768635720015, 0.196035072207, 0.038662515807, 0.134858774532, 0.252915978432, 0.11115366955, 0.0409122332931, 0.126744747162, 0.107104249299, 0.785858915615, 0.393085926771, 0.162514582276, 0.297258943318, 0.447130350256, 0.0730208158493, 0.116772949606, 0.196521967649, 0.0278613567352, 0.544190168381, 0.11268620193, 0.168384194374, 0.0234868694097, 0.224179700017, 0.14389680326, 0.3006259799, 0.093551017344, 0.305000334978, 0.11037582569, 0.302375048399, 0.131376355886, 0.0637858584523, 0.076270468533, 0.0965035632253, 0.128745958209, 0.100917465985, 0.16458337009, 0.0656297802925, 0.0687073841691, 0.322709530592, 0.0795113816857, 0.0519857369363, 0.192684188485, 0.151869475842, 0.24930883944, 0.163331031799, 0.040809719305, 0.600890162922, 0.171190023724, 0.143961474299, 0.0365794781983, 0.158125415444, 0.155979901552, 0.059639429097, 0.0469980239868, 0.337509691715, 0.50577044487, 0.154477447271, 0.048874668777, 0.058345789123, 0.077598616447, 0.0371638685465, 0.0573840253055, 0.577352702618, 0.109033301473, 0.0608575753868, 0.148531973362, 0.0728976801038, 0.157082349062, 0.0439004890954, 0.1026501971407, 0.0646234840456, 0.181817904115, 0.277896255255, 0.146952703595, 0.131492272019, 0.122336044008, 0.10883502293, 0.0397947058082, 0.0823908671737, 0.0640965476632, 0.0690795853734, 0.364885628223, 0.083915157723, 0.146860390902, 0.0746128931642, 0.499233566225, 0.0785308020115, 0.0977258532329, 0.052973091118, 0.0488270968199, 0.0316100360869, 0.376551886927, 0.111923471093, 0.131381481886, 0.365643769503, 0.0674703121185, 0.0497312247753, 0.0722003951669, 0.144517555833, 0.682386696339, 0.0788387656212, 0.0508754013676, 0.0756733119488, 0.071564061565, 0.151822179556, 0.414350509644, 0.0586323998868, 0.156209903082, 0.354884922504, 0.05113354

LambdaFunction:

```

1 import os
2 import io
3 import boto3
4 import json
5 import csv
6 def lambda_handler(event, context):
7     ENDPOINT_NAME = os.environ['environment_variable']
8     runtime= boto3.client('runtime.sagemaker')
9     print(ENDPOINT_NAME)
10    print("Received event: " , json.dumps(event, indent=2))
11    data = json.loads(json.dumps(event))
12    print("Data:",data)
13    payload = data['data']
14    print("Payload:",payload)
15    response =
runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
16
Content-Type='text/csv',
Body=payload)
17
18    print(response)

```



```
19     result = json.loads(response['Body'].read().decode())
20     print(result)
21
22     if result>0.5:
23         return "P"
24     else:
25         return "N"
```

UI:

hello.py

```
1  from flask import Flask,render_template,request,url_for
2  import requests
3  app=Flask(__name__)
4  @app.route('/',methods=['POST','GET'])
5  def hello():
6      if request.method=='POST':
7          dailyrate=request.form['a']
8          dfm=request.form['b']
9          ef=request.form['r1']
10         es=request.form['d']
11         hr=request.form['e']
12         mr=request.form['f']
13         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':
    'beaead3198f7dale70d03ab969765e0821b24fc913697e929e726aeaeb
    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[3]
42     print(result)
43     if result=='N':
44         return
    render_template('data.html',result=str(0))

```

```

45         else:
46             return
47         render_template('data.html',result=str(1))
48     else:
49         return render_template('data.html')
50 if __name__ == '__main__':
51     app.run(debug=True)
52

```

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:#ffffbf;
11            border-style: ridge;
12            border-color:gray;
13            border-radius:6px;
14        }
15        body
16        {
17            font-family:sans-serif;
18        }
19        #sub
20        {
21            width:200px;
22            height:25px;
23            background-color:#9f6cff;

```

```

24         border-style: ridge;
25         border-color:gray;
26         border-radius:6px;
27     }
28 </style>
29 </head>
30 <body style="background-color:#6a736c;">
31     <center>
32         {% if result %}
33         <p style="color:red;font-size:30px;"> Attrition:
34         {{result}}</p>
35         {% endif %}
36         {% if err_msg %}
37         <p
38         style="color:red;font-size:15px;">{{err_msg}}</p>
39         {% endif %}
40         <br/>
41         <form method="post" action="/">
42             <input type="text" name="a" class="emp"
43             placeholder="    Enter Dailyrates" required><br/><br/>
44             <input type="text" name="b" class="emp"
45             placeholder="    Enter Distance From Home"
46             required><br/><br/>
47             LifeScience&nbsp;&nbsp;&nbsp;<input type="radio"
48             value="1" name="r1" required>&nbsp;&nbsp;&nbsp;
49             Medical &nbsp;&nbsp;&nbsp;<input type="radio"
50             value="3" name="r1" required>&nbsp;&nbsp;&nbsp;
51             Marketing &nbsp;&nbsp;&nbsp;<input type="radio"
52             value="2" name="r1" required>&nbsp;&nbsp;&nbsp;
53             <br/><br/>
54             Tecnical Degree&nbsp;&nbsp;&nbsp;<input type="radio"
55             value="5" name="r1" required>&nbsp;&nbsp;&nbsp;
56             Human Resources &nbsp;&nbsp;&nbsp;<input type="radio"
57             value="0" name="r1" required>&nbsp;&nbsp;&nbsp;
58             Others &nbsp;&nbsp;&nbsp;<input type="radio"
59             value="4" name="r1" required>&nbsp;&nbsp;&nbsp;

```

```
49         <br/><br/>
50         <input type="text" name="d" class="emp"
placeholder="    Enter Envirinment 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>
```