

1. Create a notebook instance("EmployeeAttrition")

Amazon SageMaker > Notebook instances

Notebook instances info

Search notebook instances

Status: InService X Clear Filters

	Name	Instance	Creation time	Status	Actions
<input type="radio"/>	TsehsDiabetesInstance	ml.t2.medium	8/19/2024, 8:50:59 AM	InService	Open Jupyter Open JupyterLab
<input type="radio"/>	Diabetes	ml.t3.medium	8/19/2024, 8:36:54 AM	InService	Open Jupyter Open JupyterLab
<input type="radio"/>	EmployeeAttrition	ml.t3.medium	8/19/2024, 8:34:42 AM	InService	Open Jupyter Open JupyterLab
<input type="radio"/>	Diabeticsdepl	ml.t3.medium	8/19/2024, 8:34:38 AM	InService	Open Jupyter Open JupyterLab
<input type="radio"/>	TeeFirstInstance	ml.t3.medium	8/19/2024, 8:34:25 AM	InService	Open Jupyter Open JupyterLab
<input type="radio"/>	nontobekoinstance	ml.t3.medium	8/19/2024, 8:34:20 AM	InService	Open Jupyter Open JupyterLab
<input type="radio"/>	diabetinstance	ml.t3.medium	8/19/2024, 8:34:11 AM	InService	Open Jupyter Open JupyterLab

2. Create a s3 bucket("evy1234")

General purpose buckets (10) info All AWS Regions

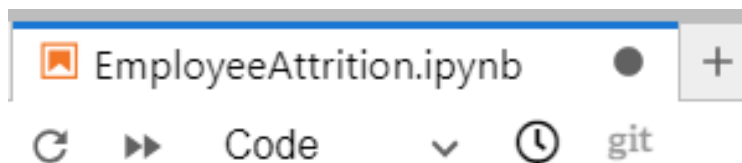
Buckets are containers for data stored in S3.

Find buckets by name

< 1 >

Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/> demobucketsfrancis	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 19, 2024, 11:09:56 (UTC+02:00)
<input type="radio"/> diabetesbuckets	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 19, 2024, 09:03:12 (UTC+02:00)
<input type="radio"/> diabeticsinstancebucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 19, 2024, 09:04:21 (UTC+02:00)
<input type="radio"/> evy1234	Europe (Ireland) eu-west-1	View analyzer for eu-west-1	August 19, 2024, 09:04:55 (UTC+02:00)
<input type="radio"/> motsobucket	Europe (Ireland) eu-west-1	View analyzer for eu-west-1	August 19, 2024, 09:03:50 (UTC+02:00)
<input type="radio"/> nontobekobucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 19, 2024, 09:04:28 (UTC+02:00)
<input type="radio"/> romeodiabetbucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 19, 2024, 09:03:37 (UTC+02:00)
<input type="radio"/> sagemakeremployeeattrition	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 18, 2024, 02:26:30 (UTC+02:00)
<input type="radio"/> teediabucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 19, 2024, 09:03:33 (UTC+02:00)

3. Create a notebook



4. Import necessary libraries

```
[84]: # Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
pd.set_option('display.max_columns', None)
warnings.filterwarnings('ignore')
```

5. Import the train data

```
[85]: #import the train data
df=pd.read_csv('train.csv')
df.head()
```

```
[85]:
```

	Employee ID	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Number of Dependents
0	8410	31	Male	19	Education	5390	Excellent	Medium	Average	2	No	22	Associate Degree	Married	0
1	64756	59	Female	4	Media	5534	Poor	High	Low	3	No	21	Master's Degree	Divorced	3
2	30257	24	Female	10	Healthcare	8159	Good	High	Low	0	No	11	Bachelor's Degree	Married	3
3	65791	36	Female	7	Education	3989	Good	High	High	1	No	27	High School	Single	2
4	65026	56	Male	41	Education	4821	Fair	Very High	Average	0	Yes	71	High School	Divorced	0

6. Check for missing values

```
[83]: # Checking for missing
df.isnull().sum()
```

```
[83]: Employee ID      0
Age      0
Gender    0
Years at Company    0
Job Role    0
Monthly Income    0
Work-Life Balance    0
Job Satisfaction    0
Performance Rating    0
Number of Promotions    0
Overtime    0
Distance from Home    0
Education Level    0
Marital Status    0
Number of Dependents    0
Job Level    0
Company Size    0
Company Tenure    0
Remote Work    0
Leadership Opportunities    0
Innovation Opportunities    0
Company Reputation    0
Employee Recognition    0
Attrition    0
dtype: int64
```

7. Describe the data

```
[49]: # Describe the data
df.describe()
```

```
[49]:
```

	Employee ID	Age	Years at Company	Monthly Income	Number of Promotions	Distance from Home	Number of Dependents	Company Tenure
count	59598.000000	59598.000000	59598.000000	59598.000000	59598.000000	59598.000000	59598.000000	59598.000000
mean	37227.118729	38.565875	15.753901	7302.397983	0.832578	50.007651	1.648075	55.758415
std	21519.150028	12.079673	11.245981	2151.457423	0.994991	28.466459	1.555689	25.411090
min	1.000000	18.000000	1.000000	1316.000000	0.000000	1.000000	0.000000	2.000000
25%	18580.250000	28.000000	7.000000	5658.000000	0.000000	25.000000	0.000000	36.000000
50%	37209.500000	39.000000	13.000000	7354.000000	1.000000	50.000000	1.000000	56.000000
75%	55876.750000	49.000000	23.000000	8880.000000	2.000000	75.000000	3.000000	76.000000
max	74498.000000	59.000000	51.000000	16149.000000	4.000000	99.000000	6.000000	128.000000

8. Checking for any repeating column names or same

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

```
Gender : ['Male' 'Female']
Male      32739
Female    26859
Name: Gender, dtype: int64
Job Role : ['Education' 'Media' 'Healthcare' 'Technology' 'Finance']
Technology    15507
Healthcare    13642
Education     12490
Media          9574
Finance        8385
Name: Job Role, dtype: int64
Work-Life Balance : ['Excellent' 'Poor' 'Good' 'Fair']
Good          22528
Fair          18046
Excellent     10719
Poor           8305
Name: Work-Life Balance, dtype: int64
Job Satisfaction : ['Medium' 'High' 'Very High' 'Low']
High          25779
Very High     12111
Medium        11817
Low           5891
Name: Job Satisfaction, dtype: int64
Performance Rating : ['Average' 'Low' 'High' 'Below Average']
Average       35810
High          11888
Below Average  8050
Low           2950
```

9. Visualization of the data



10. Label encoding

```
[74]: # Label encoding
columns = ['Gender', 'Monthly Income', 'Job Role', 'Work-Life Balance', 'Job Satisfaction', 'Performance Rating', 'Overtime', 'Education Level', 'Marital Status', 'Remote Work', 'Leadership Opportunities', 'Innovation Opportunities', 'Company Reputation', 'Employee Recognition', 'Attrition']
le = LabelEncoder()
df[columns] = df[columns].apply(le.fit_transform)
```

```
[75]: df.head()
```

	Employee ID	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Number of Dependents	Job Level
0	8410	31	1	19	0	2611	0	2	0	2	0	22	0	1	0	1
1	64756	59	0	4	3	2755	3	0	3	3	0	21	3	0	3	1
2	30257	24	0	10	2	5380	2	0	3	0	0	11	1	1	3	1
3	65791	36	0	7	0	1212	2	0	2	1	0	27	2	2	2	1
4	65026	56	1	41	0	2042	1	3	0	0	1	71	2	0	0	2

11. Predict target

```
[53]: # Predict target
y = df.Attrition
y.tail()
```

```
[53]: 59593    0
      59594    0
      59595    1
      59596    0
      59597    1
      Name: Attrition, dtype: int64
```

```
[54]: y.head()
```

```
[54]: 0    1
      1    1
      2    1
      3    1
      4    1
      Name: Attrition, dtype: int64
```

12. Features selection

```
[76]: # Feature selection
columns = ['Age', 'Gender', 'Years at Company', 'Job Role', 'Monthly Income', 'Work-Life Balance', 'Job Satisfaction', 'Performance Rating', 'Number of Promotions', 'Overtime', 'Distance from Home', 'Education Level', 'Marital Status', 'Job Level', 'Company Size', 'Company Tenure', 'Revenue']
x = x[columns]
x.head()
```

[76]:

	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Job Level	Company Size	Company Tenure	Revenue
0	31	1	19	0	2611	0	2	0	2	0	22	0	1	1	1	89	
1	59	0	4	3	2755	3	0	3	3	0	21	3	0	1	1	21	
2	24	0	10	2	5380	2	0	3	0	0	11	1	1	1	1	74	
3	36	0	7	0	1212	2	0	2	1	0	27	2	2	1	2	50	
4	56	1	41	0	2042	1	3	0	0	1	71	2	0	2	1	68	

13. Split the data and save the x_train and x_test

```
[77]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

[79]: print(x.shape, x_train.shape, x_test.shape)
(59598, 21) (47678, 21) (11920, 21)

[80]: # saving the data
x_train.to_csv('xtrain.csv', index=False, index_label='Row', header=False)

[81]: x_test.to_csv('xtest.csv', index=False, index_label='Row', header=False)
```