INSTITUTE FOR ADVANCED
COMPUTING AND
SOFTWARE
DEVELOPMENT
AKURDI, PUNE

Documentation On

“Employee Attrition Prediction Using Spark ML”

PG-DBDA SEPT 2022

Submitted By:

Group No: 02

Aparna Jore 229313

Prajakta Wadje 229332

Dr. Shantanu Pathak
Project Guide

Mr. Rohit Puranik
Centre Coordinator

CONTENTS

LIST OF FIGURES

1.Introduction

1.1 Problem Statement:

Employees are the most valuable assets of an organization. It is they who add value to the organization in terms of quantity and quality as well. Therefore, it is indispensable to maintain a permanent and promising workforce; which over the years has become a tough task for employers and thereby increased attrition in the organizations.

Organizations experience significant losses in productivity, increased recruitment and training costs, and reduced morale due to employee attrition. The inability to predict employee attrition hinders the development of effective strategies for employee retention. The problem is to develop an accurate machine learning model that can predict employee attrition based on Job Satisfaction, Monthly Income, Overtime, Salary Hike & Distance from Home, etc. and use this model to identify the factors that contribute to employee attrition. The solution aims to assist organizations in retaining top talent, reducing the cost of attrition, and ultimately improving organizational performance.

1.2 Abstract:

Employee attrition is a significant problem for organizations, leading to a loss of experienced personnel, reduced productivity, and increased recruitment and training costs. The use of machine learning algorithms to predict employee attrition has gained traction as a solution to this problem. This abstract highlights the importance of using machine learning for predicting employee attrition, the types of machine learning algorithms commonly used, and the benefits of this approach. The paper also discusses the factors that influence employee attrition including Job Satisfaction, Monthly Income, Overtime, Salary Hike & Distance from Home, etc.

The machine learning models used for predicting employee attrition consider these variables and can provide valuable insights into the likelihood of an employee leaving the organization. By predicting which employees are at risk of leaving, organizations can take proactive measures to retain top talent and reduce the cost of employee attrition. This project concludes that machine learning is a promising tool for predicting employee attrition and can provide organizations with a competitive edge in talent retention.

1.3 Scope of Project:

Machine learning models can be used to predict employee attrition, and they have a broad scope of application in this area. The scope of project is to develop a model that can analyze employee data including Job Satisfaction, Monthly Income, Overtime, Salary Hike & Distance From Home etc. to identify patterns and trends that may contribute to employee attrition.

The scope of project is –

- Exploratory Data Analysis: Exploratory Data Analysis (EDA) is the process of examining and analyzing data sets to summarize their main characteristics, often with visual methods. The goal of EDA is to gain an understanding of the underlying structure and patterns of the data in order to inform subsequent analysis or modeling.
- Model Selection: Choosing the appropriate machine learning model, such as logistic regression, Decision Tree, or neural networks, based on the characteristics of the data and the objectives of the analysis.
- Model Training and Evaluation: Training the model on historical data and evaluating its performance on a holdout dataset to ensure it provides accurate and reliable predictions.
- Interpretation of Results: Interpret the model's outputs to understand the underlying factors contributing to employee attrition and identify potential areas for improvement.

In conclusion, machine learning models can be a valuable tool in predicting employee attrition and identifying potential risk factors that contribute to employee turnover. By leveraging these models, organizations can take proactive measures to improve employee engagement, retention, and overall job satisfaction, leading to a more productive and successful workforce.

1.4 Aims & Objectives:

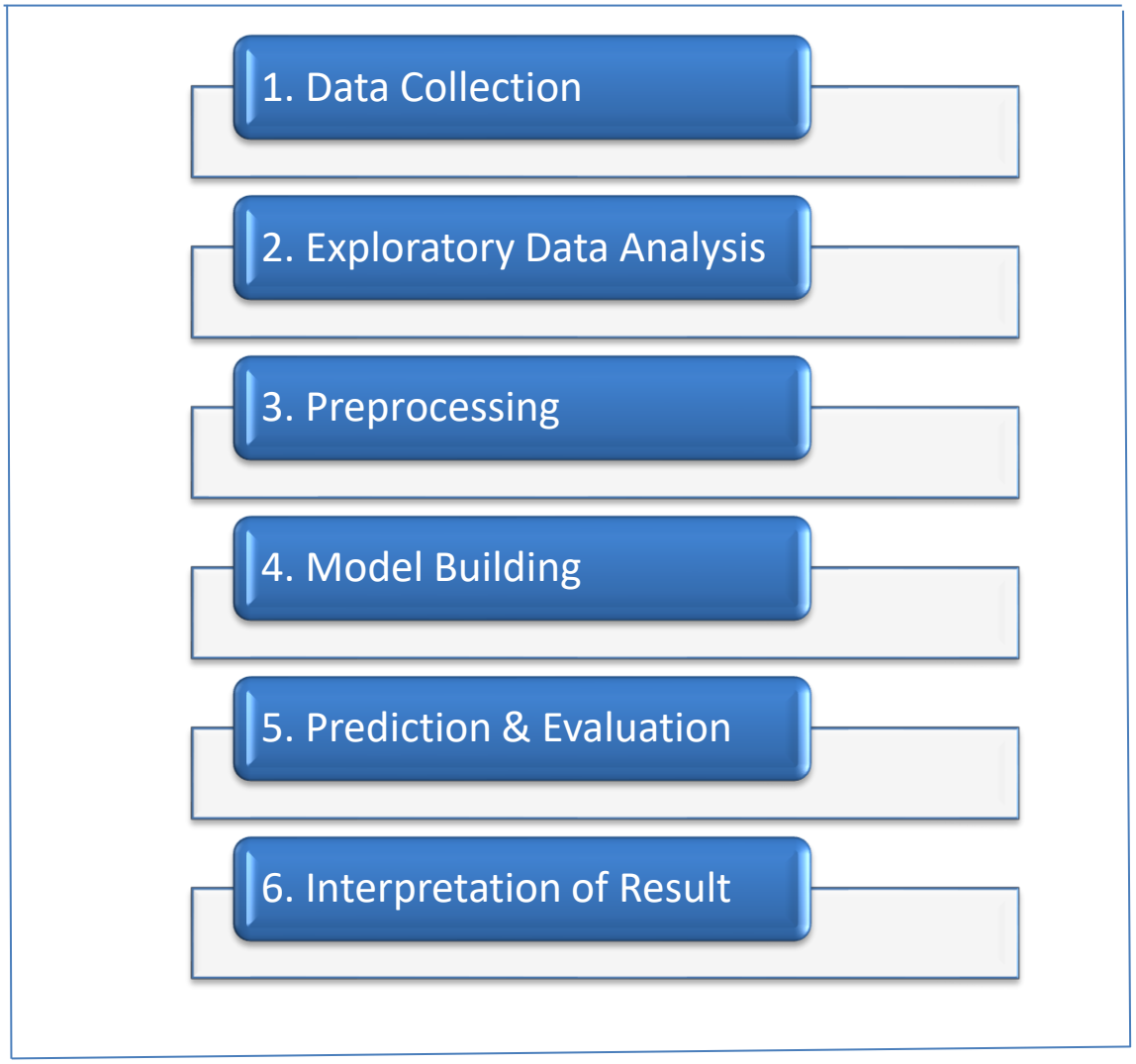
The aim of employee attrition prediction using machine learning is to develop a predictive model that can identify employees who are at a high risk of leaving their current job. This model can help organizations take proactive measures to retain their valuable employees and reduce the negative impact of attrition.

The objectives of this project may include:

- Retain valuable employees:
  By predicting which employees are likely to leave the company, organizations can take steps to retain their most valuable employees by offering them incentives and benefits that encourage them to stay.
- Reduce costs associated with employee turnover:
  Losing employees can be costly for organizations in terms of recruitment, training, and lost productivity. By predicting employee attrition, organizations can take steps to reduce these costs by addressing the underlying causes of turnover.
- Improve workforce planning:
  By predicting employee attrition, organizations can better plan their workforce needs and allocate resources accordingly. This can help ensure that the organization has the right people in the right positions at the right time.
- Enhance employee satisfaction and engagement:
  Predicting and addressing employee attrition can help improve employee satisfaction and engagement by addressing issues that may be causing employees to consider leaving the organization.
- Identify areas for improvement:
  Employee attrition prediction can also help organizations identify areas where they may need to improve, such as employee benefits, training and development programs, or the overall work environment. By addressing these areas, organizations can improve employee satisfaction and reduce turnover.

2.Overall Description

Workflow of project:



2.1 Data Collection:

We have taken the Employee Attrition dataset of IBM from Kaggle and contains information about the employees of a company, including their job roles, performance, and personal details. The dataset is designed to help organizations identify factors that contribute to employee turnover and attrition.

The dataset consists of 1,470 rows and 35 columns, including features such as age, gender, education level, job role, salary, work environment satisfaction, performance rating, and the reason for an employee's departure.

2.2 Exploratory Data Analysis Using Spark SQL:

EDA is an important first step in any data analysis project, providing a foundation for subsequent analysis and modeling. EDA is important because it allows analysts to understand the characteristics of the data and identify any potential problems or anomalies before conducting further analysis or modeling.
Following are some Spark SQL queries & visualization made on result of queries used to extract some useful information:

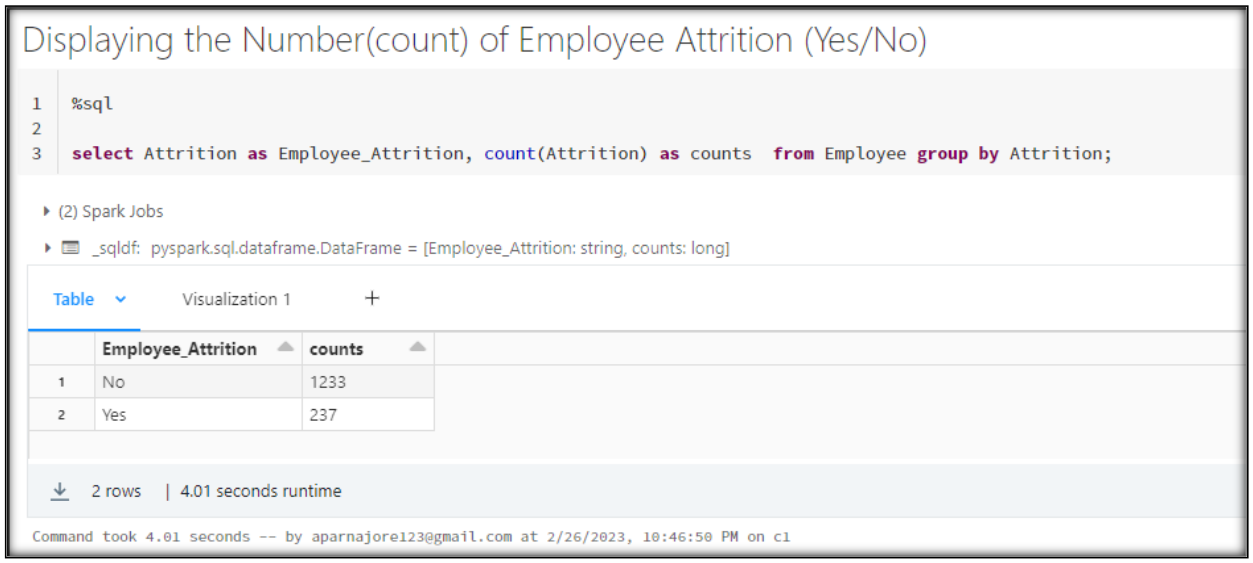


Figure 1. Displaying count of Employee Attrition (Yes/No)

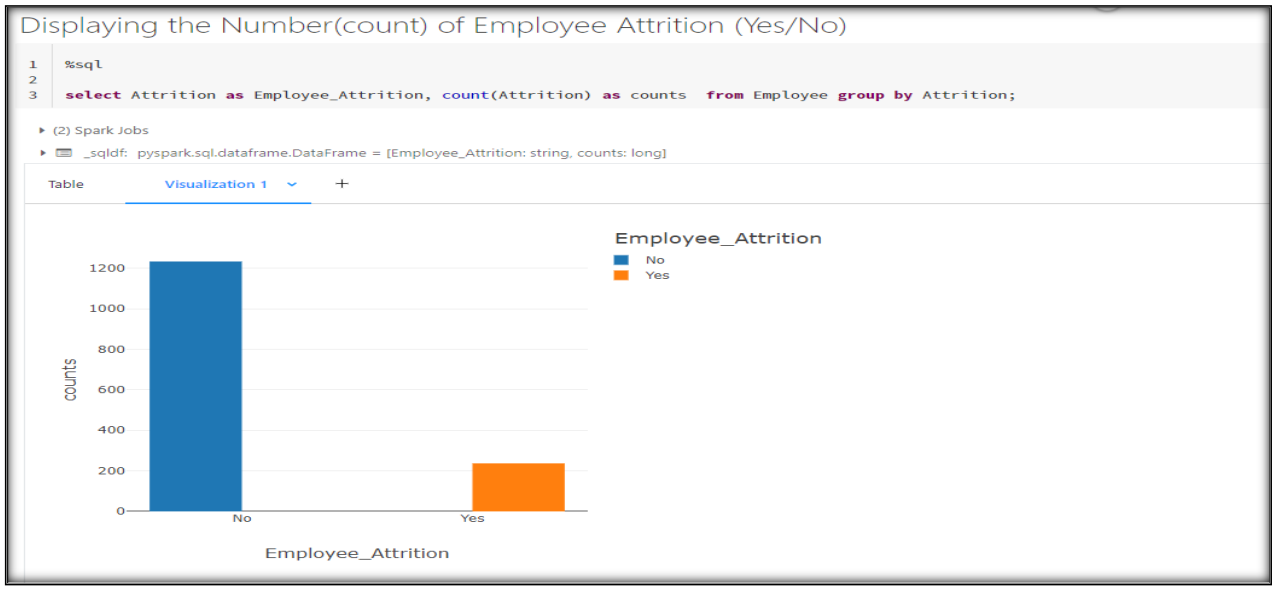


Figure 2. Displaying count of Employee Attrition (Yes/No)

Exp: Here we observed that there are 1233 employees who are not leaving organization & 237 employees leaving the organization.

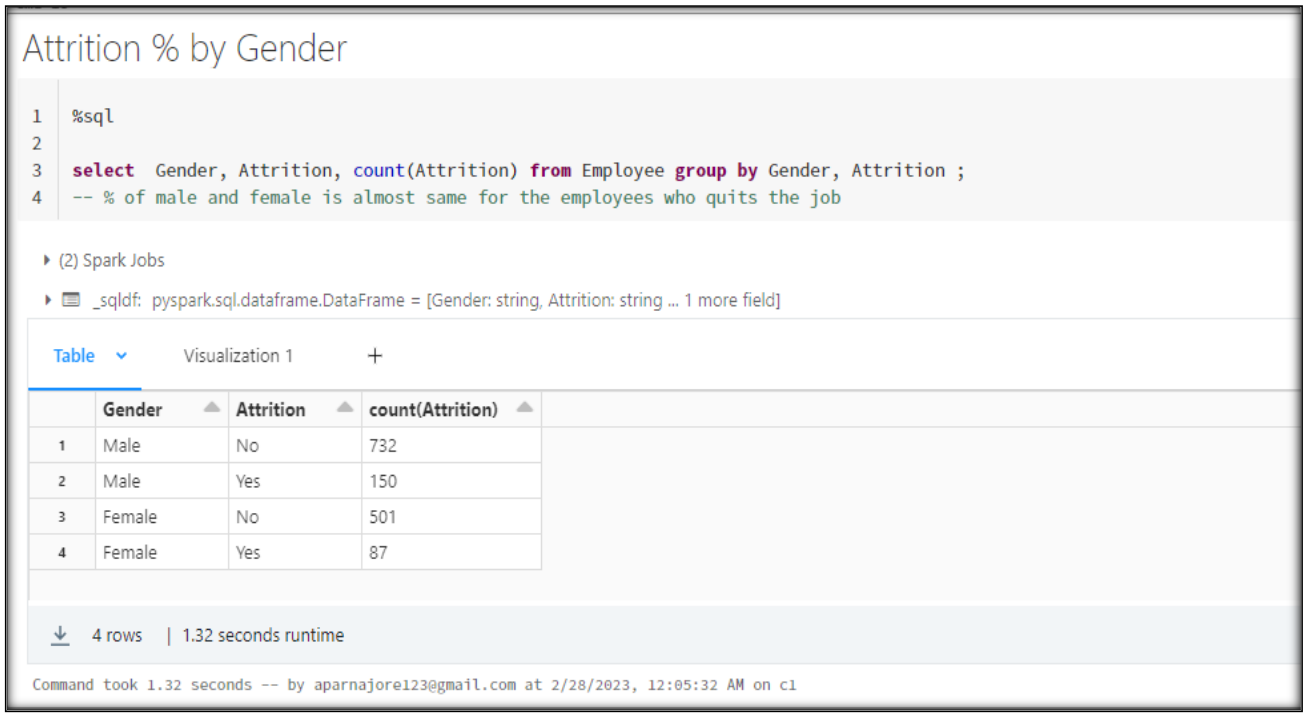


Figure 3. Displaying count of Employee Attrition (Yes/No)

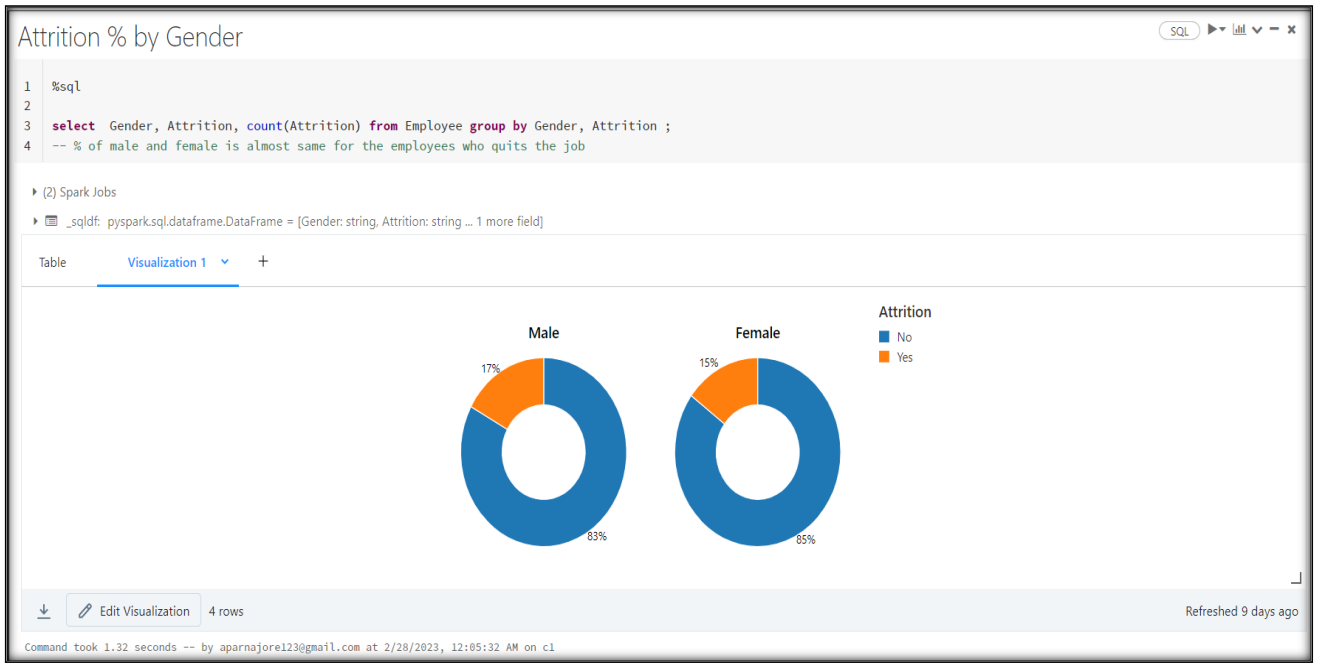


Figure 4. Donut chart showing the Attrition percentage by Gender

Exp: Here we observed that 12% Male employees leaving the organization & 15% Female employees leaving the organization.

Monthly Income by Gender

```
%sql

select Gender, MonthlyIncome from Employee;
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Gender: string, MonthlyIncome: string]

Table | Visualization 1 | +

	Gender	MonthlyIncome
1	Female	5993
2	Male	5130
3	Male	2090
4	Female	2909
5	Male	3468
6	Male	3068
7	Female	2670
8	Male	2693
9	Male	9526
10	Male	5237

1,000 rows | Truncated data | 0.48 seconds runtime

Command took 0.48 seconds -- by aparnajore123@gmail.com at 2/26/2023, 10:46:50 PM on c1

Figure 5. Monthly Income by Gender

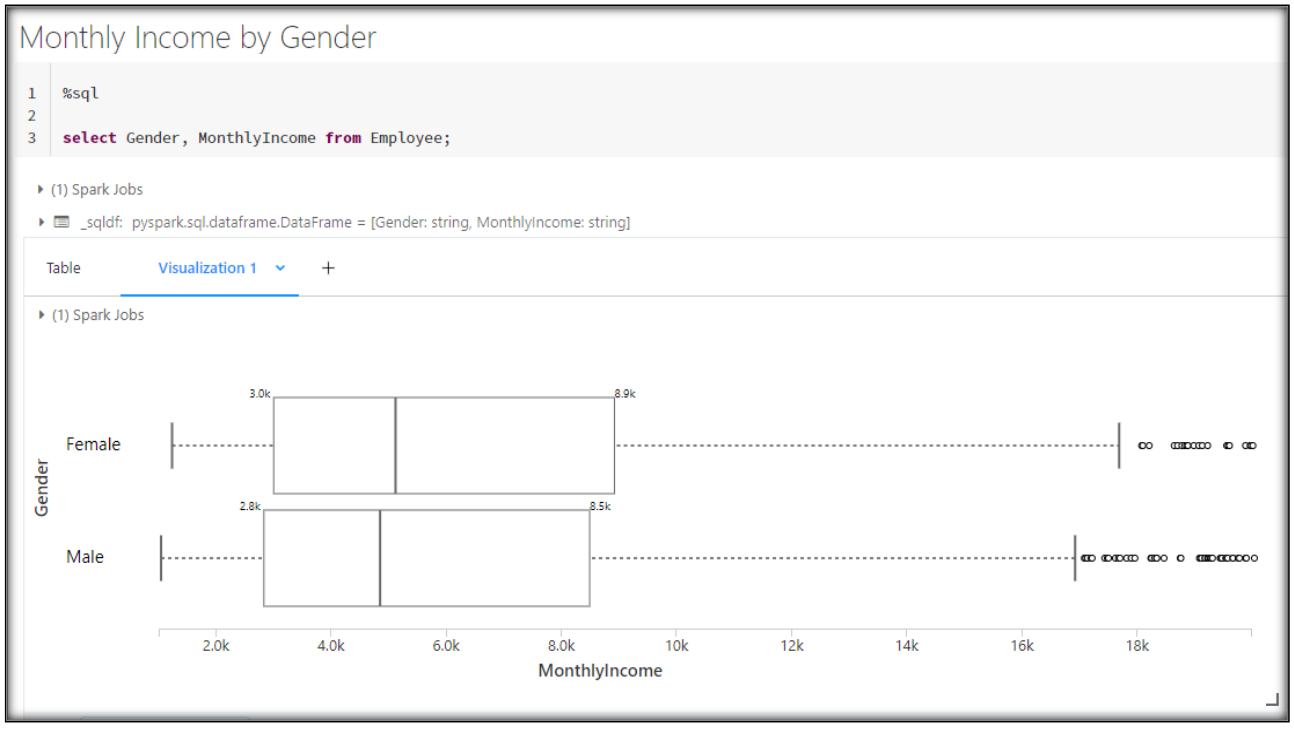


Figure 6. Box plot showing Monthly Income by Gender

Exp: Here we observed that for Female employees the average range of salary is between 3k to 8.9k. And for Male the range is from 2.8k to 8.5k.

`

Average_MonthlyIncome & Effect on Attrition

```
%sql

select  Gender,Attrition, avg(MonthlyIncome) from Employee group by Gender,Attrition;

-- The avg(MonthlyIncome) has same effect on both male & female, the employess with avg salary 4700 has left the organization
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Gender: string, Attrition: string ... 1 more field]

Table | Visualization 1 | +

	Gender	Attrition	avg(MonthlyIncome)
1	Male	No	6704.964480874317
2	Male	Yes	4797.16
3	Female	No	7019.429141716567
4	Female	Yes	4769.735632183908

4 rows | 0.95 seconds runtime

Figure 7. Average _Monthly Income and Effect on Attrition

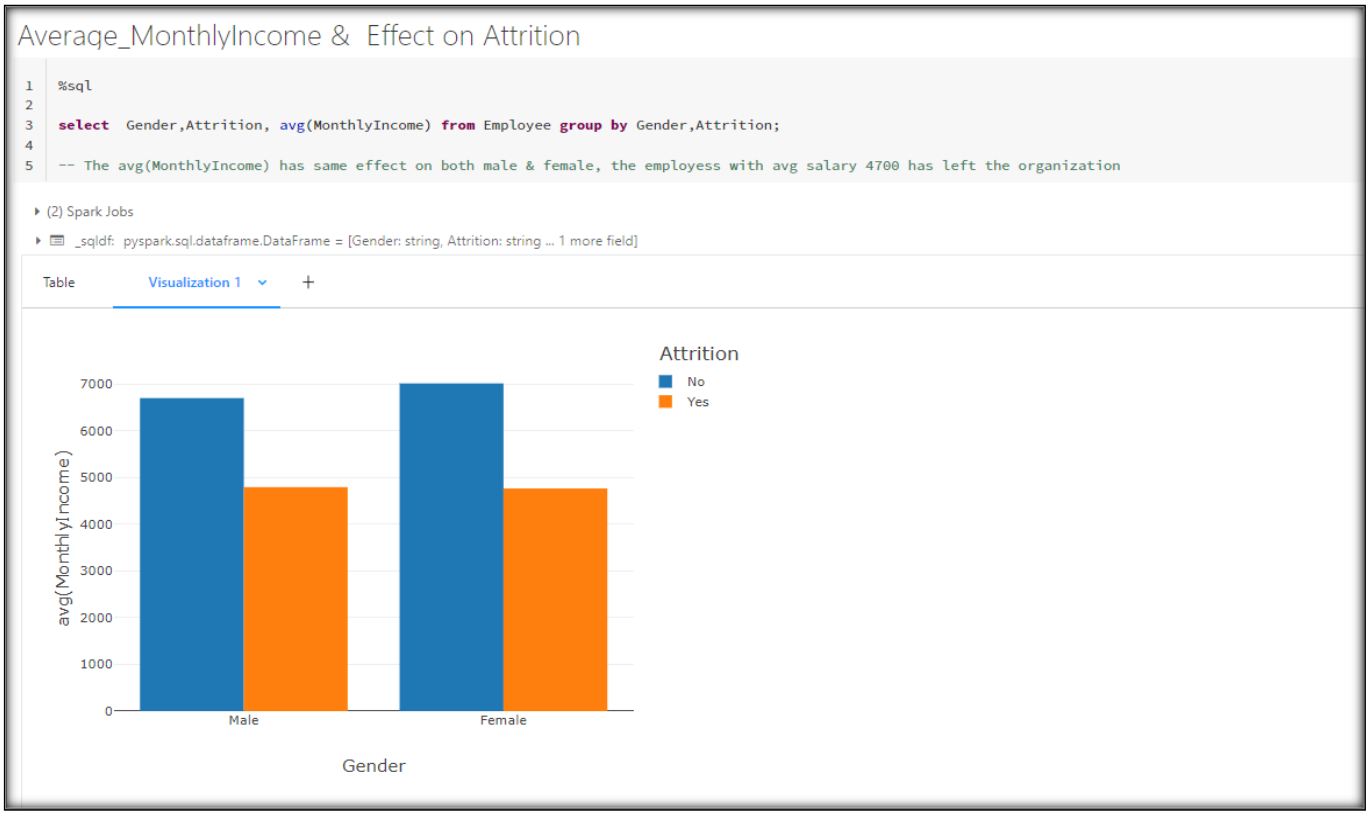


Figure 8. Side by Side Bar Chart showing Average Monthly Income & its effect on Attrition

Exp: Here we observed that for both Female and Male the employees having Average Monthly Income around 4700 has leaved the organization.

Distribution of Job Satisfaction:

```
%sql

select JobSatisfaction, Attrition from Employee;

--  (1 = Low satisfied, 2 = Medium satisfied, 3 = Highly satisfied,  4 = Very High satisfied)
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [JobSatisfaction: string, Attrition: string]

Table | Visualization 1 | +

	JobSatisfaction	Attrition
1	4	Yes
2	2	No
3	3	Yes
4	3	No
5	2	No
6	4	No
7	1	No

1,000 rows | Truncated data | 0.59 seconds runtime

Figure 9. Distribution of Job Satisfaction

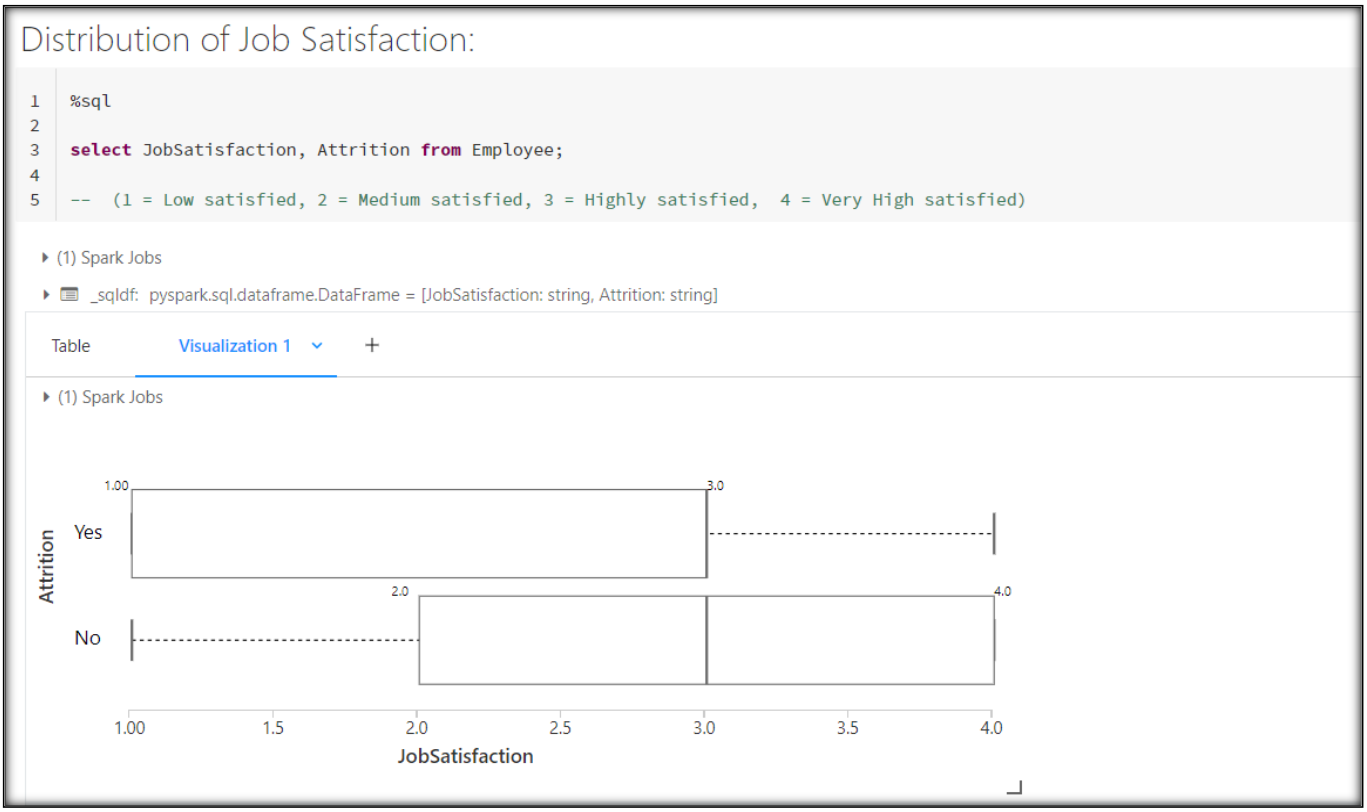


Figure 10. Box Plot showing distribution of Job Satisfaction

Exp: Here we observed that the employees who leaved the organization has job satisfaction level between 1 to 3 and the employees who are not leaving the organization has job satisfaction level between 2 to 4. Means the employees who are leaving the organization are less satisfied than those who are not leaving.

Satisfaction level of employees in organization

```
%sql

select count(JobSatisfaction), JobSatisfaction from Employee group by JobSatisfaction order by JobSatisfaction desc;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [count(JobSatisfaction): long, JobSatisfaction: string]

Table | Visualization 1 | +

	count(JobSatisfaction)	JobSatisfaction
1	459	4
2	442	3
3	280	2
4	289	1

4 rows | 1.19 seconds runtime

Command took 1.19 seconds -- by aparnajore123@gmail.com at 2/26/2023, 10:46:50 PM on c1

Figure 11. Satisfaction level of employees

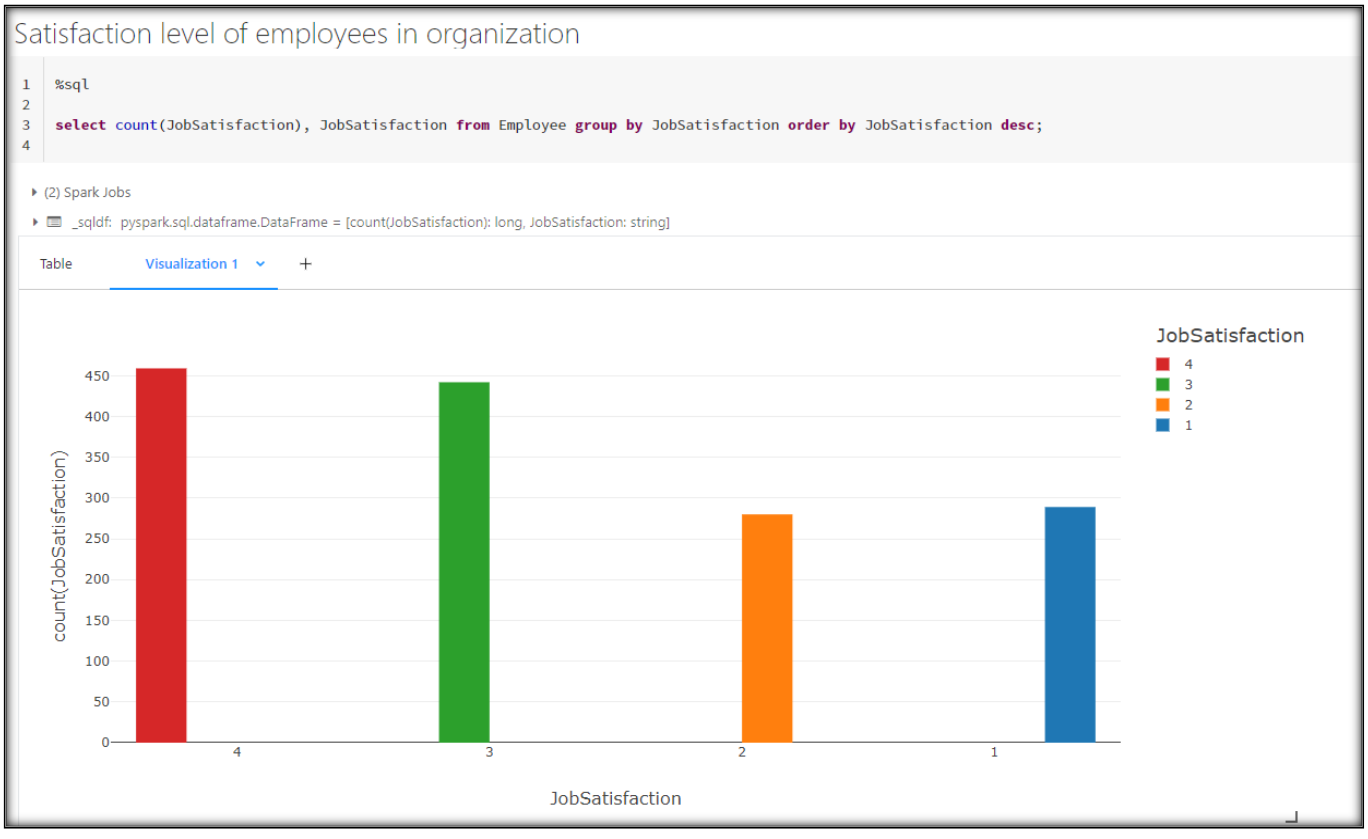


Figure 12. Bar plot showing satisfaction level of employees

Exp: Here we observed that most of the employees in organization are highly satisfied. There are 459 employees having job satisfaction level 4, 442 employees having job satisfaction level 3, 280 employees having job satisfaction level 2 and 289 having job satisfaction level 1.

Department-wise Attrition Percentage

```
1  %sql
2
3  select  Department, Attrition,count(Department) as Number_of_Employee from Employee group by Department, Attrition;
4
5  -- sales has more attrition rate as compared to R&D and HR department
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Department: string, Attrition: string ... 1 more field]

Table | Visualization 1 | +

	Department	Attrition	Number_of_Employee
1	Research & Development	No	828
2	Human Resources	No	51
3	Sales	Yes	92
4	Sales	No	354
5	Research & Development	Yes	133
6	Human Resources	Yes	12

6 rows | 0.61 seconds runtime

Figure 13. Department-wise attrition percentage

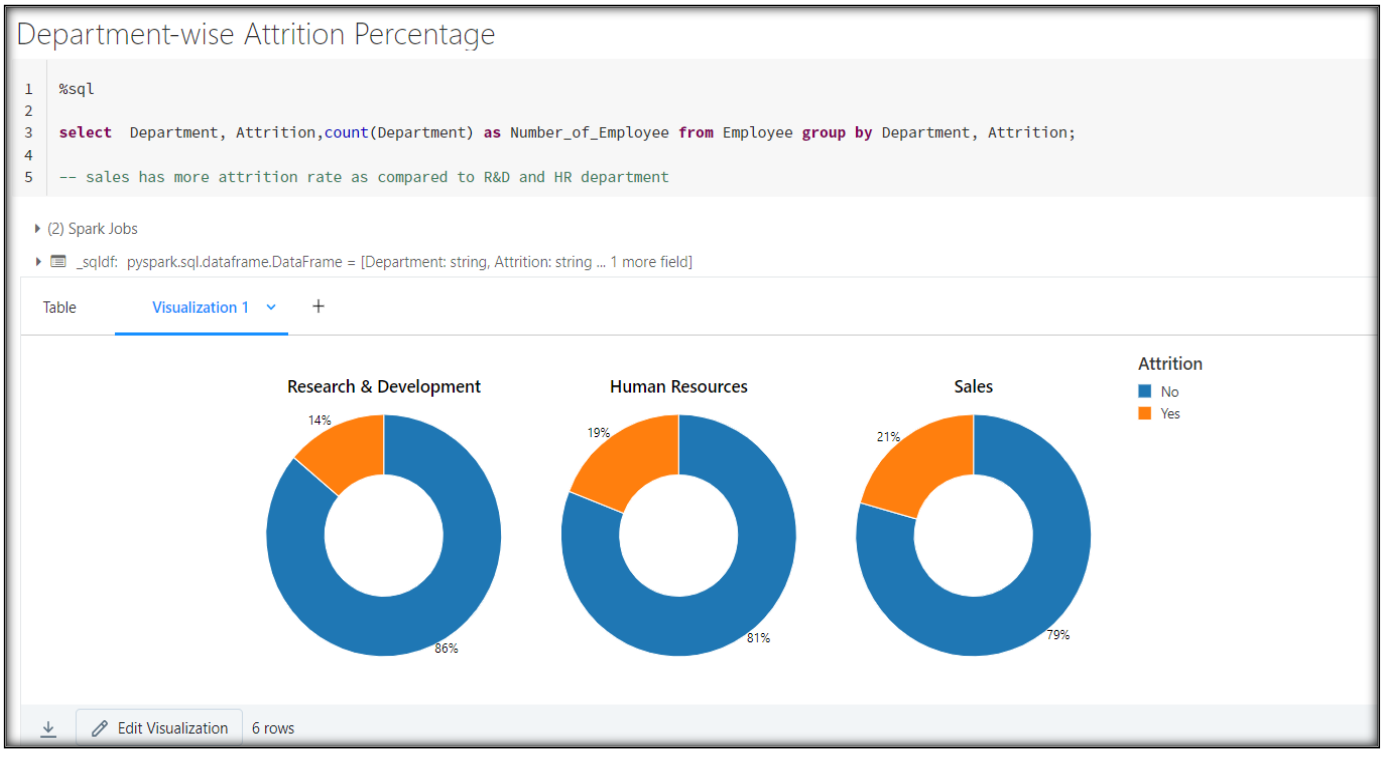


Figure 14. Donut chart showing department-wise attrition percentage

Exp: Here we can observe that the Sales department has more attrition rate as compared to HR and R&D department.

Level of Attrition by Overtime Status:

```
%sql

select  OverTime,count(OverTime), Attrition from Employee group by OverTime, Attrition;

-- Exhaustion at Work: Over 31% of workers who left the organization worked overtime! Will this be a reason why employees are leaving?
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [OverTime: string, count(OverTime): long ... 1 more field]

Table | Visualization 1 | +

	OverTime	count(OverTime)	Attrition
1	Yes	127	Yes
2	No	944	No
3	Yes	289	No
4	No	110	Yes

4 rows | 1.17 seconds runtime

Figure 15. Level of Attrition by Overtime Status

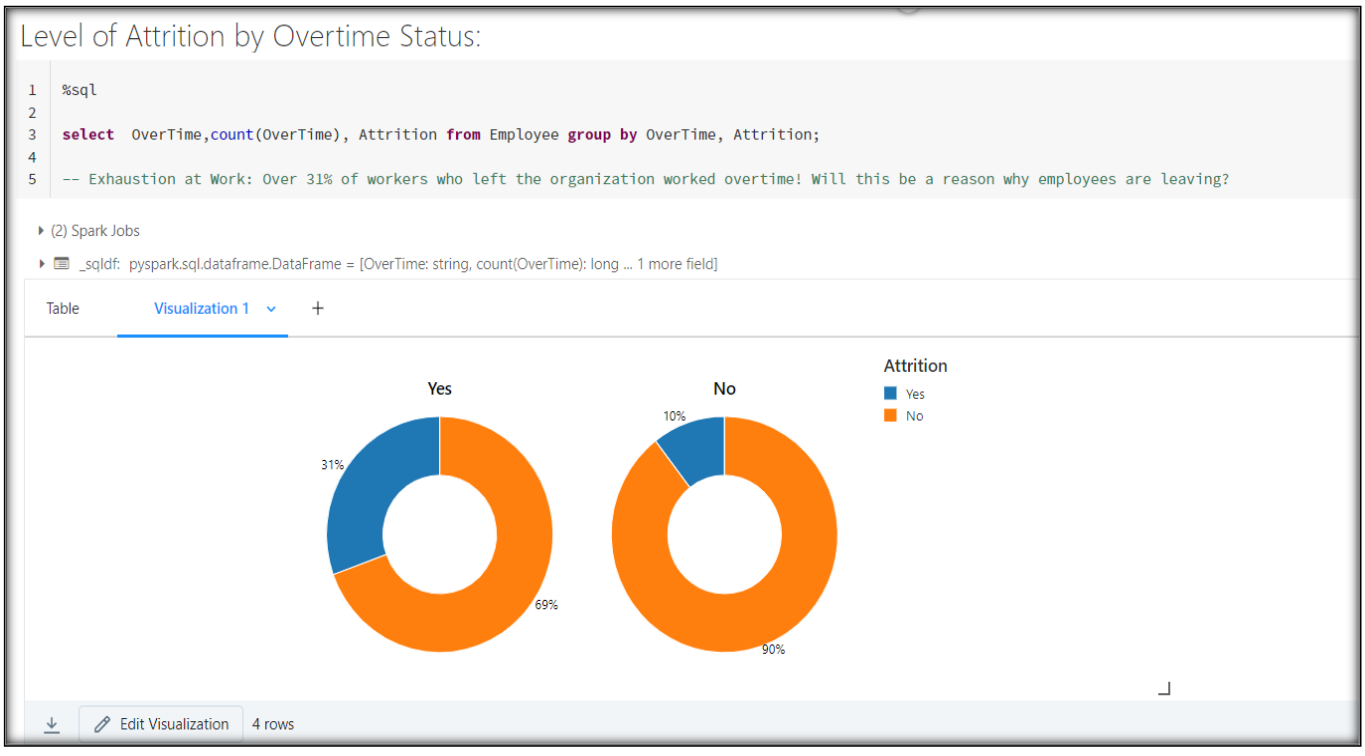


Figure 16. Donut chart for Comparing Level of Attrition by Overtime Status

Exp: Here we can observe that there are 31% employees with overtime are leaving the organization and the employees are who are not doing overtime have less percentage i.e., only 10% of them are leaving the organization.

Attrition by Job Role

```
1  %sql
2
3  select JobRole, Attrition, count(Attrition) from  Employee group by JobRole, Attrition;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [JobRole: string, Attrition: string ... 1 more field]

Table | Visualization 1 | +

	JobRole	Attrition	count(Attrition)
1	Research Scientist	No	245
2	Manufacturing Director	Yes	10
3	Manager	No	97
4	Manufacturing Director	No	135
5	Laboratory Technician	No	197
6	Healthcare Representative	Yes	9
7	Research Director	Yes	2

18 rows | 1.33 seconds runtime

Command took 1.33 seconds -- by aparnajore123@gmail.com at 2/26/2023, 11:36:57 PM on c1

Figure 17. Attrition by Job Role

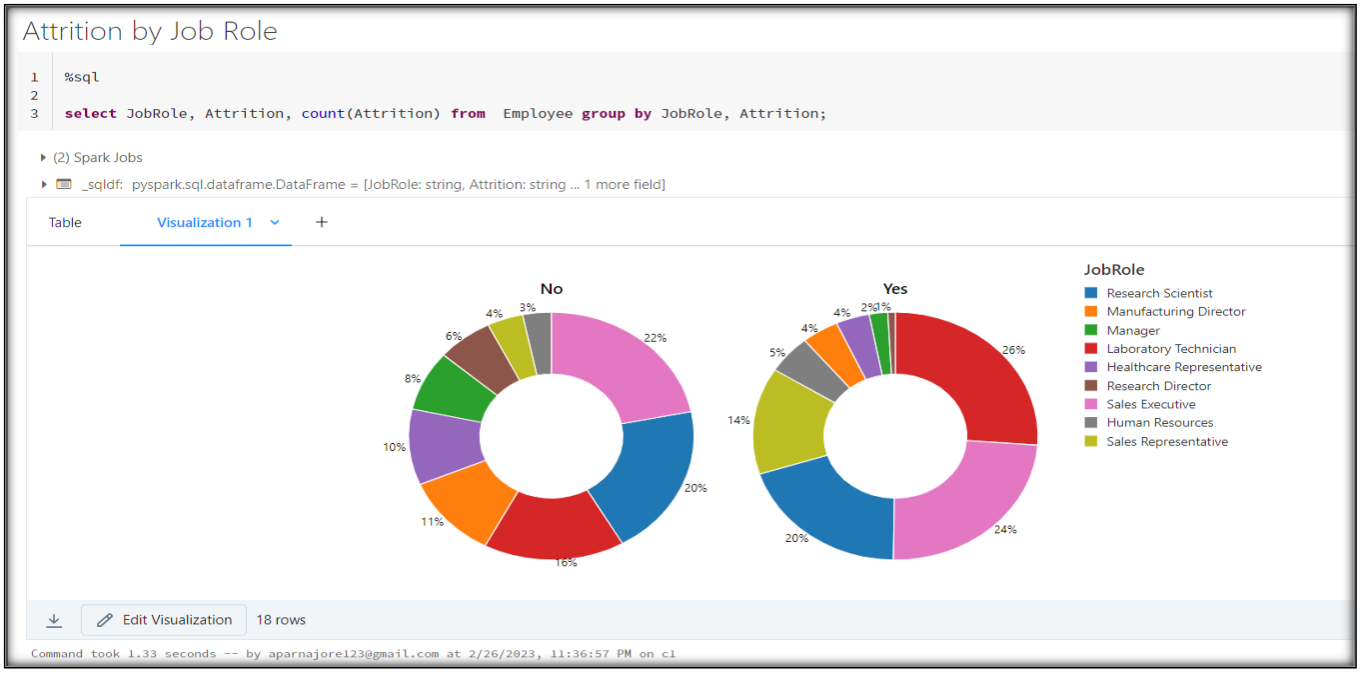


Figure 18. Donut chart for comparison of Attrition by Job Role

Exp: Here we can observe that there out of total employees who are leaving there are 26% employees having job role Laboratory Technician then 24% employees having job role Sales Executive and so on.

```
%sql

select Attrition,
CASE
    WHEN DistanceFromHome < 7 THEN "Below Average"
    ELSE "Above Average"
END AS DistanceFromHome
from Employee group by DistanceFromHome, Attrition;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Attrition: string, DistanceFromHome: string]

Table | Visualization 1 | +

	Attrition	DistanceFromHome
1	Yes	Above Average
2	Yes	Below Average
3	Yes	Above Average
4	Yes	Below Average
5	Yes	Above Average
6	Yes	Above Average
7	Yes	Above Average
8	Yes	Above Average

58 rows | 0.68 seconds runtime

Figure 19. Distance from home & Attrition

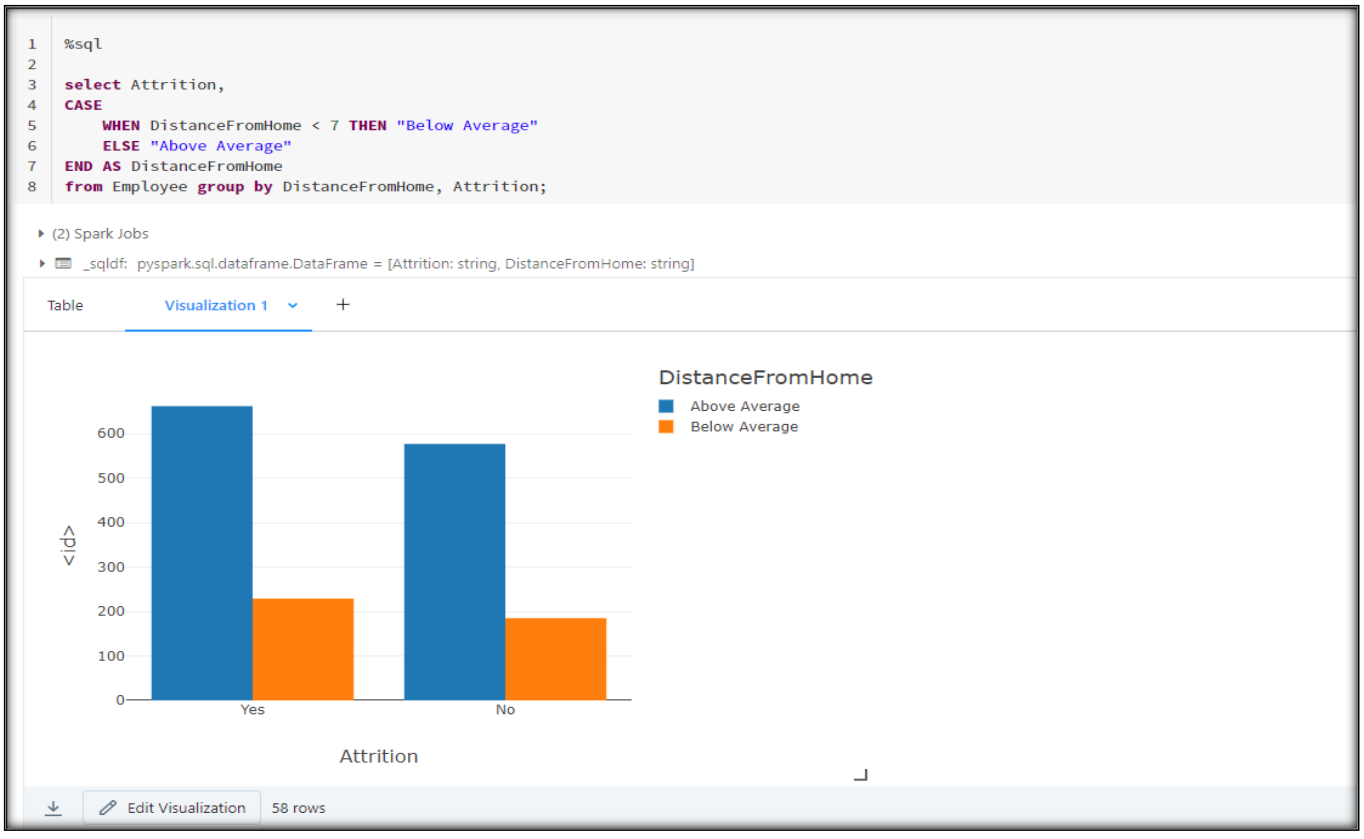


Figure 20. Side by Side Bar Chart showing Distance from home & Attrition

Exp: Here we can observe that there is no major effect of distance from home on Attrition. There are 662 employees having above average distance and they are leaving organization and there are 577 employees having above average distance but they are not leaving.

Effect of Business Travel on Attrition

```
%sql

select BusinessTravel,Attrition  ,count(Attrition) as Attrition_count from Employee group by BusinessTravel, Attrition order by Attrition_count desc;
-- The employees who travel rarely having more attrition count as compared to employees who travel frequently.
-- there is no impact on attrition for the employees who not travel
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [BusinessTravel: string, Attrition: string ... 1 more field]

Table | Visualization 1 | +

	BusinessTravel	Attrition	Attrition_count
1	Travel_Rarely	No	887
2	Travel_Frequently	No	208
3	Travel_Rarely	Yes	156
4	Non-Travel	No	138
5	Travel_Frequently	Yes	69
6	Non-Travel	Yes	12

6 rows | 1.04 seconds runtime

Figure 21. Effect of Business Travel on Attrition

Figure 22. Side by Side Bar Chart showing Effect of Business Travel on Attrition

Exp: Here we can observe that the employees who are leaving organization are more from them who travel rarely and some of them are from who travel frequently and very few of them are from who do not travel.

2.3 Preprocessing

Preprocessing is required in employee attrition prediction to improve the quality of the data and to prepare it for machine learning algorithms.

2.3.1 Handling missing values:

The datasets may contain missing values, which can lead to incorrect predictions. We have checked if there are any null values in our dataset. In Spark, the Imputer is a transformer that is used to fill in missing values in a dataset. The Imputer works by computing the mean or median of the columns that have missing values and then filling in the missing values with that value. By default, the Imputer computes the mean of the columns that have missing values. However, you can also specify the strategy parameter to use the median or most frequent value instead of the mean.

```
1  Dict_Null = {col:employeeDF.filter(employeeDF[col].isNull()).count() for col in employeeDF.columns}
2  Dict_Null
3
4  #employeeDF = employeeDF.dropna() to remove rows with Null values.

▸ (67) Spark Jobs

Out[21]: {'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,
Command took 12.73 seconds -- by aparnajore123@gmail.com at 3/2/2023, 10:52:54 AM on cluster1
```

Figure 23. Check for null values

2.3.2 Convert categorical variables:

Decision trees work best with numerical data. If dataset contains categorical variables, we need to convert them into numerical data. We have used Pyspark's StringIndexer to convert the categorical variables into numerical values. StringIndexer is a feature transformation technique used in machine learning for converting categorical or string data into numerical data. It works by assigning a unique index value to each distinct category or string in the input data. A Pipeline consists of a sequence of stages, each stage is Transformer & When pipeline.fit() is called, the stages are executed in order.

```
#from pyspark.ml.attribute import Attribute
from pyspark.ml.feature import IndexToString, StringIndexer
from pyspark.ml import Pipeline, PipelineModel

indexers = [StringIndexer(inputCol=col, outputCol=col+"_indexed", handleInvalid="skip") for col in string_feature_col]

pipeline = Pipeline(stages=indexers)

empDF = pipeline.fit(employeeDF).transform(employeeDF)
```

▸ (18) Spark Jobs

▸ empDF: pyspark.sql.dataframe.DataFrame = [Age: string, Attrition: string ... 42 more fields]

Command took 16.72 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:40:09 PM on c1

Figure 24. StringIndexing

```
empDF.select("Attrition","Attrition_indexed").distinct().show()
empDF.select("BusinessTravel","BusinessTravel_indexed").distinct().show()
empDF.select("EducationField","EducationField_indexed").distinct().show()
```

▸ (6) Spark Jobs

```
+---------+-----------------+
|Attrition|Attrition_indexed|
+---------+-----------------+
|       No|              0.0|
|      Yes|              1.0|
+---------+-----------------+

+-----------------+----------------------+
|   BusinessTravel|BusinessTravel_indexed|
+-----------------+----------------------+
|       Non-Travel|                   2.0|
|Travel_Frequently|                   1.0|
|    Travel_Rarely|                   0.0|
+-----------------+----------------------+

+---------------+---------------------+
|  EducationField|EducationField_indexed|
+---------------+---------------------+
|          Other|                  4.0|
|      Marketing|                  2.0|
|  Life Sciences|                  0.0|
```

Command took 5.70 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:40:33 PM on c1

Figure 25. Output of StringIndexer

2.3.3 Feature transformation using VectorAssembler:

VectorAssembler is a feature transformer in Apache Spark that is used to combine multiple feature columns into a single feature vector column. It takes a set of input columns (of numeric, boolean, or vector type) and combines them into a single vector column. The resulting vector column can then be used as input for machine learning algorithms.

The VectorAssembler is often used in data preprocessing pipelines, where multiple columns need to be combined into a single feature vector before training a machine learning model. This can simplify the code needed to prepare the data for the model and make it easier to manage the feature engineering process.

`

```
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(inputCols =[ "Age",    "BusinessTravel_indexed",    "DailyRate",    "Department_indexed",    "DistanceFromHome",    "Education",    "EducationField_indexed",    "EnvironmentSatisfaction",    "Gender_indexed",
   "HourlyRate",    "JobInvolvement",    "JobLevel",    "JobRole_indexed",    "JobSatisfaction",    "MaritalStatus_indexed",    "MonthlyIncome",    "MonthlyRate",    "NumCompaniesWorked",    "Over18_indexed",    "OverTime_indexed",
   "PercentSalaryHike",    "PerformanceRating",    "RelationshipSatisfaction",    "StandardHours",    "StockOptionLevel",    "TotalWorkingYears",    "TrainingTimesLastYear",    "WorkLifeBalance",    "YearsAtCompany",    "YearsInCurrentRole",
   "YearsSinceLastPromotion",    "YearsWithCurrManager"], outputCol = "features" )

training = assembler.transform(trainingData).select("features", "Attrition_indexed")
```

```
▸ training:  pyspark.sql.dataframe.DataFrame = [features: udt, Attrition_indexed: double]
Command took 0.52 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:46:29 PM on c1
```

```
End 41
```

```
training.show(truncate=0)
```

```
▸ (1) Spark Jobs
```

```
+----------------------------------------------------------------------------------------------------------------------------------------------------------------+-----------------+
|features                                                                                                                                                        |Attrition_indexed|
+----------------------------------------------------------------------------------------------------------------------------------------------------------------+-----------------+
|[41.0,0.0,1102.0,1.0,1.0,2.0,0.0,2.0,1.0,94.0,3.0,2.0,0.0,4.0,1.0,5993.0,19479.0,8.0,0.0,1.0,11.0,3.0,1.0,80.0,0.0,8.0,0.0,1.0,6.0,4.0,0.0,5.0]                 |1.0              |
|[49.0,1.0,279.0,0.0,8.0,1.0,0.0,3.0,0.0,61.0,2.0,2.0,1.0,2.0,0.0,5130.0,24907.0,1.0,0.0,0.0,23.0,4.0,4.0,80.0,1.0,10.0,3.0,3.0,10.0,7.0,1.0,7.0]               |0.0              |
|[33.0,1.0,1392.0,0.0,3.0,4.0,0.0,4.0,1.0,56.0,3.0,1.0,1.0,3.0,0.0,2909.0,23159.0,1.0,0.0,1.0,11.0,3.0,3.0,80.0,0.0,8.0,3.0,3.0,8.0,7.0,3.0,0.0]               |0.0              |
|[27.0,0.0,591.0,0.0,2.0,1.0,1.0,1.0,0.0,40.0,3.0,1.0,2.0,2.0,0.0,3468.0,16632.0,9.0,0.0,0.0,12.0,3.0,4.0,80.0,1.0,6.0,3.0,3.0,2.0,2.0,2.0,2.0]               |0.0              |
|[32.0,1.0,1005.0,0.0,2.0,2.0,0.0,4.0,0.0,79.0,3.0,1.0,2.0,4.0,1.0,3068.0,11864.0,0.0,0.0,0.0,13.0,3.0,3.0,80.0,0.0,8.0,2.0,2.0,7.0,7.0,3.0,6.0]               |0.0              |
|[30.0,0.0,1358.0,0.0,24.0,1.0,0.0,4.0,0.0,67.0,3.0,1.0,2.0,3.0,2.0,2693.0,13335.0,1.0,0.0,0.0,22.0,4.0,2.0,80.0,1.0,1.0,2.0,3.0,1.0,0.0,0.0,0.0]               |0.0              |
|[35.0,0.0,809.0,0.0,16.0,3.0,1.0,1.0,0.0,84.0,4.0,1.0,2.0,2.0,0.0,2426.0,16479.0,0.0,0.0,0.0,13.0,3.0,3.0,80.0,1.0,6.0,5.0,3.0,5.0,4.0,0.0,3.0]               |0.0              |
|[29.0,0.0,153.0,0.0,15.0,2.0,0.0,4.0,1.0,49.0,2.0,2.0,2.0,3.0,1.0,4193.0,12682.0,0.0,0.0,0.0,12.0,3.0,4.0,80.0,0.0,10.0,3.0,3.0,9.0,5.0,0.0,8.0]             |0.0              |
|[31.0,0.0,670.0,0.0,26.0,1.0,0.0,1.0,0.0,31.0,3.0,1.0,1.0,3.0,2.0,2911.0,15170.0,1.0,0.0,0.0,17.0,3.0,4.0,80.0,1.0,5.0,1.0,2.0,5.0,2.0,4.0,3.0]               |0.0              |
|[32.0,0.0,334.0,0.0,5.0,2.0,0.0,1.0,0.0,80.0,4.0,1.0,1.0,2.0,2.0,3298.0,15053.0,0.0,0.0,1.0,12.0,3.0,4.0,80.0,2.0,7.0,5.0,2.0,6.0,2.0,0.0,5.0]               |0.0              |
|[22.0,2.0,1123.0,0.0,16.0,2.0,1.0,4.0,0.0,96.0,4.0,1.0,2.0,4.0,2.0,2935.0,7324.0,1.0,0.0,1.0,13.0,3.0,2.0,80.0,2.0,1.0,2.0,2.0,1.0,0.0,0.0,0.0]             |0.0              |
|[53.0,0.0,1219.0,1.0,2.0,4.0,0.0,1.0,1.0,78.0,2.0,4.0,5.0,4.0,0.0,15427.0,22021.0,2.0,0.0,0.0,16.0,3.0,3.0,80.0,0.0,31.0,3.0,3.0,25.0,8.0,3.0,7.0]           |0.0              |
|[24.0,2.0,673.0,0.0,11.0,2.0,4.0,1.0,1.0,96.0,4.0,2.0,3.0,3.0,2.0,4011.0,8232.0,0.0,0.0,0.0,18.0,3.0,4.0,80.0,1.0,5.0,5.0,2.0,4.0,2.0,1.0,3.0]               |0.0              |
|[34.0,0.0,419.0,0.0,7.0,4.0,0.0,1.0,1.0,53.0,3.0,3.0,7.0,2.0,1.0,11994.0,21293.0,0.0,0.0,0.0,11.0,3.0,3.0,80.0,0.0,13.0,4.0,3.0,12.0,6.0,2.0,11.0]           |0.0              |
|[53.0,0.0,1282.0,0.0,5.0,3.0,4.0,3.0,1.0,58.0,3.0,5.0,5.0,3.0,2.0,19094.0,10735.0,4.0,0.0,0.0,11.0,3.0,4.0,80.0,1.0,26.0,3.0,2.0,14.0,13.0,4.0,8.0]         |0.0              |
|[32.0,1.0,1125.0,0.0,16.0,1.0,0.0,2.0,1.0,72.0,1.0,1.0,1.0,1.0,1.0,3919.0,4681.0,1.0,0.0,1.0,22.0,4.0,2.0,80.0,0.0,10.0,5.0,3.0,10.0,2.0,6.0,7.0]           |1.0              |
|[42.0,0.0,691.0,1.0,8.0,4.0,2.0,3.0,0.0,48.0,3.0,2.0,0.0,2.0,0.0,6825.0,21173.0,0.0,0.0,0.0,11.0,3.0,4.0,80.0,1.0,10.0,2.0,3.0,9.0,7.0,4.0,2.0]               |0.0              |
|[44.0,0.0,1459.0,0.0,10.0,4.0,4.0,4.0,0.0,41.0,3.0,2.0,4.0,4.0,0.0,6465.0,19121.0,2.0,0.0,0.1,13.0,3.0,4.0,80.0,0.0,9.0,5.0,4.0,4.0,2.0,1.0,3.0]           |0.0              |
```

```
Command took 1.23 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:46:37 PM on c1
```

Figure 26. Feature Transformation using Vector Assembler

2.4 Model Building:

2.4.1 Train Test Split:

Train-test split is a common technique used in machine learning to evaluate the performance of a model. It involves splitting a dataset into two subsets: one for training the model and another for testing the model. The training set is used to train the model, while the test set is used to evaluate the model's performance on unseen data. Our dataset is imbalanced that's why here we have used Stratified Sampling.

Stratified Sampling: This technique is used when the dataset is imbalanced, meaning that some classes have fewer samples than others. In this technique, the dataset is divided into subsets based on the class labels, and then a train-test split is performed on each subset separately. The objective of stratified sampling is to ensure that the sample is representative of the entire population by reducing sampling bias and increasing precision. By dividing the population into strata, the variability within each stratum is reduced, and the differences between strata are maximized. This approach ensures that each stratum is represented in the sample, which provides a more accurate estimate of the population parameters.

Stratified Sampling

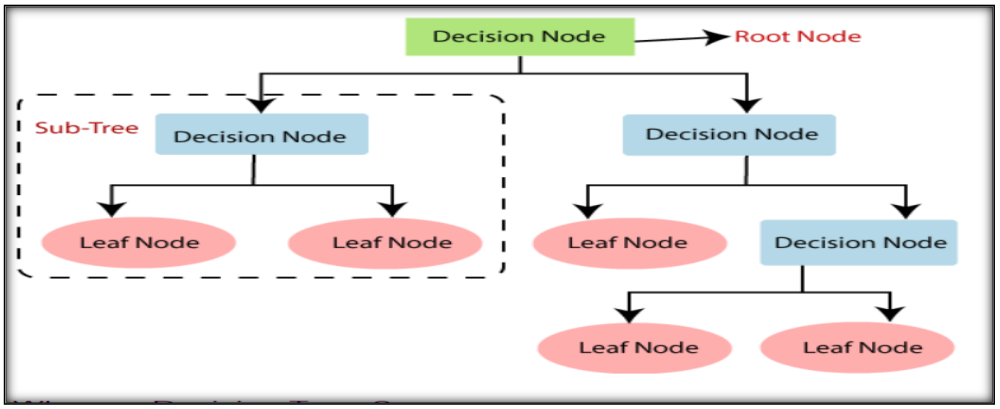
```
# specify the column containing the labels
label_col = "Attrition_indexed"

# specify the fraction of data to use for testing
test_fraction = 0.3 #1-test_fraction

# perform the stratified sampling
trainingData = empDF.sampleBy(
    label_col,
    fractions={label: 1-test_fraction for label in empDF.select(label_col).distinct().rdd.flatMap(lambda x: x).collect()},
    seed=42
)
testData = empDF.subtract(trainingData)

#print(empDF.select(label_col).distinct().rdd.flatMap(lambda x: x).collect())
#[0.0, 1.0]
#print({label: 1-test_fraction for label in empDF.select(label_col).distinct().rdd.flatMap(lambda x: x).collect()})
#{0.0: 0.7, 1.0: 0.7}
```

```
▸ (2) Spark Jobs
▸ trainingData: pyspark.sql.dataframe.DataFrame = [Age: string, Attrition: string ... 42 more fields]
▸ testData: pyspark.sql.dataframe.DataFrame = [Age: string, Attrition: string ... 42 more fields]
Command took 1.71 seconds -- by aparnajorel23@gmail.com at 3/8/2023, 8:14:05 PM on c1
```

Figure 27. Stratified Sampling

2.4.2 Decision Tree Classification Model:

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:



2.4.2.1 Decision Tree Terminologies:

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

2.4.2.2 Attribute Selection Measures:

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain:** A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.
- **Gini Index:** Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
  An attribute with the low Gini index should be preferred as compared to the high Gini index.

2.4.2.3 Model Training:

Model training is the process of training a machine learning model on a set of training data in order to learn a function that can predict outputs for new, unseen inputs.

```
1  from pyspark.ml.classification import DecisionTreeClassificationModel, DecisionTreeClassifier
2  from pyspark.ml.evaluation import MulticlassClassificationEvaluator
3
4  lr = DecisionTreeClassifier(labelCol="Attrition_indexed", featuresCol="features")
5
6  model = lr.fit(training)
7
8  print("Model Trained!")
```

```
▸ (8) Spark Jobs
Model Trained!
Command took 6.30 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:46:44 PM on c1
```

Figure 28. Model Training

2.4.2.4 Preparing test data:

Used StringIndexer for converting categorical columns into numerical and VectorAssembler for combining multiple columns into a single feature vector

Preparing Test Data

```
for col_name in string_cols:
    testData = testData.withColumn(col_name, col(col_name).cast("double"))

test = assembler.transform(testData).select("features", "Attrition_indexed")
test.show()
```

▸ (3) Spark Jobs

▸ testData: pyspark.sql.dataframe.DataFrame = [Age: double, Attrition: string ... 42 more fields]

▸ test: pyspark.sql.dataframe.DataFrame = [features: udt, Attrition_indexed: double]

```
+--------------------+-----------------+
|            features|Attrition_indexed|
+--------------------+-----------------+
|[40.0,0.0,1124.0,...|              0.0|
|[36.0,1.0,541.0,1...|              0.0|
|[50.0,0.0,691.0,0...|              0.0|
|[32.0,1.0,116.0,0...|              0.0|
|[29.0,0.0,1329.0,...|              0.0|
|[35.0,0.0,1229.0,...|              0.0|
|[35.0,0.0,1204.0,...|              1.0|
|[50.0,1.0,959.0,1...|              1.0|
|[35.0,0.0,982.0,0...|              0.0|
|[30.0,0.0,634.0,0...|              0.0|
|[35.0,0.0,384.0,1...|              0.0|
|[30.0,0.0,945.0,1...|              1.0|
|[31.0,0.0,525.0,1...|              0.0|
|[30.0,0.0,125.0,0...|              0.0|
|[35.0,0.0,802.0,0...|              0.0|
|[30.0,1.0,160.0,0...|              0.0|
|[40.0,0.0,1492.0,...|              0.0|
|[33.0,0.0,211.0,1...|              1.0|
```

Command took 4.98 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:46:55 PM on c1

Figure 29. Preparing test data

2.4.2.4 Predictions on test data:

The transform method is an important part of the machine learning pipeline, as it allows us to test the performance of the trained model on new data and evaluate its ability to generalize to unseen examples.

Predictions on test data

```
prediction = model.transform(test)
prediction.select("prediction", "Attrition_indexed", "features").show(10)
```

▸ (3) Spark Jobs

▸ prediction: pyspark.sql.dataframe.DataFrame = [features: udt, Attrition_indexed: double ... 3 more fields]

```
+----------+-----------------+--------------------+
|prediction|Attrition_indexed|            features|
+----------+-----------------+--------------------+
|       1.0|              0.0|[40.0,0.0,1124.0,...|
|       0.0|              0.0|[36.0,1.0,541.0,1...|
|       0.0|              0.0|[50.0,0.0,691.0,0...|
|       0.0|              0.0|[32.0,1.0,116.0,0...|
|       0.0|              0.0|[29.0,0.0,1329.0,...|
|       0.0|              0.0|[35.0,0.0,1229.0,...|
|       0.0|              1.0|[35.0,0.0,1204.0,...|
|       0.0|              1.0|[50.0,1.0,959.0,1...|
|       0.0|              0.0|[35.0,0.0,982.0,0...|
|       0.0|              0.0|[30.0,0.0,634.0,0...|
+----------+-----------------+--------------------+
only showing top 10 rows
```

Command took 3.20 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:47:06 PM on c1

Figure 30. Predictions on test data

2.4.2.4 Model Evaluation:

The accuracy of decision tree model is 79.93%.

Model Evaluation

```
1  evaluator = MulticlassClassificationEvaluator(
2      labelCol="Attrition_indexed", predictionCol="prediction")
3  accuracy = evaluator.evaluate(prediction)
4  print(accuracy)
```

▸ (3) Spark Jobs

```
0.7993855587946903
Command took 2.43 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:47:20 PM on c1
```

2.4.3 Support Vectors Machine (SVM):

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

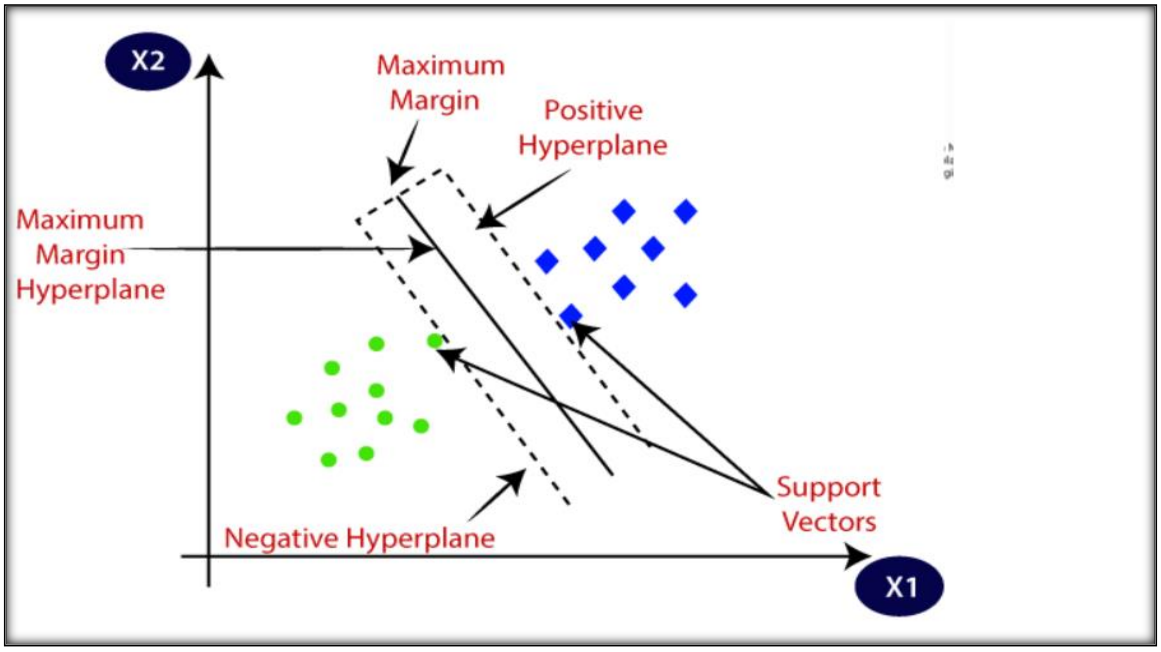


Figure 31. Support Vector Machine

2.4.3.1 Model Training:

Model training is the process of training a machine learning model on a set of training data in order to learn a function that can predict outputs for new, unseen inputs.

```
1  from pyspark.ml.classification import LinearSVC

Command took 0.12 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:47:46 PM on c1

Cmd 47

1  svm = LinearSVC(labelCol="Attrition_indexed", featuresCol="features")
2  svm_model = svm.fit(training)

▸ (100) Spark Jobs

Command took 23.14 seconds -- by aparnajore123@gmail.com at 3/8/2023, 3:59:33 PM on c1
```

Figure 32. Model Training

2.4.3.2 Predictions on test data:

The transform method is an important part of the machine learning pipeline, as it allows us to test the performance of the trained model on new data and evaluate its ability to generalize to unseen examples.

```
1  predictions = model.transform(test)
2  predictions.select("prediction", "Attrition_indexed", "features").show(10)

▸ (3) Spark Jobs
▸ predictions: pyspark.sql.dataframe.DataFrame = [features: udt, Attrition_indexed: double ... 2 more fields]
+----------+-----------------+--------------------+
|prediction|Attrition_indexed|            features|
+----------+-----------------+--------------------+
|       0.0|              0.0|[40.0,0.0,1124.0,...|
|       0.0|              0.0|[36.0,1.0,541.0,1...|
|       0.0|              0.0|[50.0,0.0,691.0,0...|
|       0.0|              0.0|[32.0,1.0,116.0,0...|
|       0.0|              0.0|[29.0,0.0,1329.0,...|
|       0.0|              0.0|[35.0,0.0,1229.0,...|
|       0.0|              1.0|[35.0,0.0,1204.0,...|
|       0.0|              1.0|[50.0,1.0,959.0,1...|
|       0.0|              0.0|[35.0,0.0,982.0,0...|
|       0.0|              0.0|[30.0,0.0,634.0,0...|
+----------+-----------------+--------------------+
only showing top 10 rows

Command took 3.47 seconds -- by aparnajore123@gmail.com at 3/8/2023, 4:00:03 PM on c1
```

Figure 33. Predictions on test data

2.4.3.3 Model Evaluation:

The accuracy of Linear SVM Classification model is 80.16%.

```
1  evaluator = MulticlassClassificationEvaluator(labelCol="Attrition_indexed", predictionCol="prediction")
2  accuracy = evaluator.evaluate(predictions)
3  print("Accuracy:", accuracy)
```

▸ (3) Spark Jobs

```
Accuracy: 0.8016431818043899
```

Command took 2.59 seconds -- by aparnajore123@gmail.com at 3/8/2023, 4:07:09 PM on c1

Figure 34. Model Evaluation

3. Automation using AWS services

Automating Spark tasks on AWS using S3, Glue, and Lambda can provide a powerful and flexible solution for processing large-scale data sets. The services used for automation are **S3(Simple Storage Service), IAM, Lambda, Glue, Cloudwatch, Athena**. Here is a high-level overview of how these services can be used together to create an automated data processing pipeline:

- **Data ingestion**: First, data is uploaded to an S3 bucket, which can serve as the central storage location for input and output data.
- **Job definition**: Next, a Glue job is defined to process the data using Spark. Glue is a fully managed ETL (extract, transform, load) service that can handle both batch and streaming data processing. Glue allows users to define and execute Spark jobs in a serverless environment.
- **Job configuration**: The Glue job configuration includes details such as the location of the input data in S3, the code for the Spark application, and any necessary settings for the Spark job (such as the number of nodes to use, memory allocation, etc.).
- **Triggering the job**: Once the Glue job is defined and configured, it can be triggered by various events such as the arrival of new data in S3 or a schedule.
- **Job execution**: When the job is triggered, Glue launches a Spark cluster to execute the job code, retrieves the input data from S3, and writes the output data back to S3. The job progress and results can be monitored in the Glue console or through CloudWatch logs.
- **Post-processing**: If additional processing or analysis is required on the output data, this can be performed using other AWS services such as Athena
- **Automation**: Finally, the entire data processing pipeline can be automated using AWS Lambda, a serverless compute service that can execute code in response to various events. For example, a Lambda function can be triggered when new data is added to the input S3 bucket, which can in turn trigger the Glue job to process the data automatically.

Overall, using S3, Glue, and Lambda together provides a powerful and scalable platform for processing large-scale data sets on AWS.

4. Conclusion

After analyzing employee attrition data using decision tree and SVM models, we can conclude that both decision tree and SVM models are effective in predicting employee attrition. The decision tree model shows good performance with a higher accuracy score and faster computation time, making it a suitable choice for companies with large datasets. The SVM model has a higher accuracy, indicating that it is better at correctly identifying employees who will leave the company. This is crucial for companies tha want to prevent attrition and retain valuable employees. The most significant factors that contribute to employee attrition are job satisfaction, monthly income and overtime. Companies can use the insights gained from the models to develop strategies to improve job satisfaction, increase employee compensation, and create a work environment that promotes work-life balance to reduce employee turnover. Overall, decision tree and SVM models are useful tools for predicting employee attrition, and the insights gained from these models can help companies take proactive measures to reduce attrition and retain their top-performing employees.

5. Future Scope

The scope for employee attrition prediction is quite significant and is expected to grow in the future due to the increasing need for companies to retain their employees. Here are some potential areas where employee attrition prediction projects can be applied:

- Human resources: Employee attrition prediction can be a valuable tool for human resources departments in companies of all sizes. Predictive models can be used to identify factors that may cause employees to leave, such as low pay or lack of career growth opportunities. This information can then be used to make changes that help retain employees.
- Recruitment: Attrition prediction models can help companies make better hiring decisions by identifying candidates who are more likely to stay with the company in the long term. This can help reduce the cost of hiring and training new employees.
- Performance management: Attrition prediction models can also be used to monitor employee performance and identify those who may be at risk of leaving. This information can then be used to provide additional support or training to help these employees succeed and stay with the company.
- Succession planning: Attrition prediction models can also be used to identify employees who are likely to leave in the near future and develop succession plans to fill their positions. This can help ensure that the company is prepared for any changes that may occur.

Overall, the future scope of employee attrition prediction projects is quite promising, and as technology advances, there will be more data available to develop even more accurate predictive models.

6. References

1. C. Cortes and V. Vapnik, Support-vector networks. Machine learning, 20(3), 273-297, 1995

2. V. Nagadevara, V. Srinivasan, and R. Valk, "Establishing a link between employee turnover and withdrawal behaviors: Application of data mining techniques", Research and Practice in Human Resource Management, 16(2), 81-97, 2008

3. Dilip Singh Sisodia, Somdutta Vishwakarma, Abinash Pujahari "Evaluation of Machine Learning Models for Employee Churn Prediction", Proceedings of the International Conference on Inventive Computing and Informatics (ICICI 2017) IEEE [13] Xplore Compliant - Part Number: CFP17L34-ART, ISBN: 978-1-5386-4031-9.

4. W. C. Hong, S. Y. Wei, and Y. F. Chen, "A comparative test of two employee turnover prediction models", International Journal of Management, 24(4), 808, 2007.

5. L. K. Marjorie, "Predictive Models of Employee Voluntary Turnover in a North American Professional Sales Force using Data-Mining Analysis", Texas, A&M University College of Education, 2007.

6. D. Alao and A. B. Adeyemo, "Analyzing employee attrition using decision tree algorithms", Computing, Information Systems, Development Informatics and Allied Research Journal, 4, 2013.

7. V. V. Saradhi and G. K. Palshikar, "Employee churn prediction", Expert Systems with Applications, 38(3), 1999-2006, 2011.

8. Usha, P.; Balaji, N. Analysing Employee attrition using machine learning. Karpagam J. Comput. Sci. 2019, 13, 277–282.