FINAL PROJECT

HR ANALYTICS EMPLOYEE ATTRITION

Group 15

Faris Rizky Andika

Ikhsan Dwitama

Maria Reno

BACKGROUND

- ***Berkurangnya jumlah karyawan (attrition)*** adalah sesuatu yang tak terhindarkan di sebuah organisasi bisnis. Hal tersebut bisa terjadi baik karena berbagai alasan
- Ketika ***tingkat attrition*** mencapai ambang batas tertentu, hal tersebut dapat merugikan perusahaan.
- Oleh karena itu, penting sekali bagi sebuah perusahaan untuk menganalisa faktor apa saja yang menyebabkan terjadinya ***pengurangan karyawan*** dan membuat model untuk memprediksi hal tersebut.

Source: https://www.techfunnel.com/hr-tech/employee-attrition/

DATASET

- ***Dataset Name :*** **HR Analytics Case Study-Employee Attrition Analysis**
- ***Dataset URL:*** **https://www.kaggle.com/vjchoudhary7/hr-analytics-case-study**
- ***Contents:*** **Five .csv files with 1 data dictionary file (.xlsx)**

File name	File description
general_data.csv	Informasi umum mengenai karyawan
employee_survey_data.csv	Hasil survey terhadap karyawan dalam kepuasan kerja, kepuasan lingkungan kerja dan *work-life balance*
manager_survey_data.csv	Hasil survey terhadap manager mengenai kinerja dan tingkat keterlibatan kerja karyawan
in_time.csv	Waktu absensi masuk tiap karyawan
out_time.csv	Waktu absensi pulang tiap karyawan
data_dictionary.xlsx	Penjelasan mengenai variable-variable dalam dataset

DATASET

- ***Dataset info:*** (setelah melakukan *merging* terhadap **general_data.csv** dengan **employee_survey_data.csv** dan **manager_survey_data.csv**)

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4410 entries, 0 to 4409
Data columns (total 29 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      4410 non-null   int64
 1   Attrition                4410 non-null   object
 2   BusinessTravel           4410 non-null   object
 3   Department               4410 non-null   object
 4   DistanceFromHome         4410 non-null   int64
 5   Education                4410 non-null   int64
 6   EducationField           4410 non-null   object
 7   EmployeeCount            4410 non-null   int64
 8   EmployeeID               4410 non-null   int64
 9   Gender                   4410 non-null   object
 10  JobLevel                 4410 non-null   int64
 11  JobRole                  4410 non-null   object
 12  MaritalStatus            4410 non-null   object
 13  MonthlyIncome            4410 non-null   int64
 14  NumCompaniesWorked       4391 non-null   float64
 15  Over18                   4410 non-null   object
 16  PercentSalaryHike        4410 non-null   int64
 17  StandardHours            4410 non-null   int64
 18  StockOptionLevel         4410 non-null   int64
 19  TotalWorkingYears        4401 non-null   float64
 20  TrainingTimesLastYear    4410 non-null   int64
 21  YearsAtCompany           4410 non-null   int64
 22  YearsSinceLastPromotion  4410 non-null   int64
 23  YearsWithCurrManager     4410 non-null   int64
 24  EnvironmentSatisfaction  4385 non-null   float64
 25  JobSatisfaction          4390 non-null   float64
 26  WorkLifeBalance          4372 non-null   float64
 27  JobInvolvement           4410 non-null   int64
 28  PerformanceRating        4410 non-null   int64
dtypes: float64(5), int64(16), object(8)
memory usage: 1.0+ MB
```

DATASET

- ***Dataset heads:***

```
In [2]:
gen = pd.read_csv('general_data.csv')
gen.head()
```

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeID
0	51	No	Travel_Rarely	Sales	6	2	Life Sciences	1	1
1	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	1	2
2	32	No	Travel_Frequently	Research & Development	17	4	Other	1	3
3	38	No	Non-Travel	Research & Development	2	5	Life Sciences	1	4
4	32	No	Travel_Rarely	Research & Development	10	1	Medical	1	5

5 rows × 24 columns

```
In [3]:
mgr = pd.read_csv('manager_survey_data.csv')
mgr.head()
```

	EmployeeID	JobInvolvement	PerformanceRating
0	1	3	3
1	2	2	4
2	3	3	3
3	4	2	3
4	5	3	3

```
In [4]:
emp = pd.read_csv('employee_survey_data.csv')
emp.head()
```

	EmployeeID	EnvironmentSatisfaction	JobSatisfaction	WorkLifeBalance
0	1	3.0	4.0	2.0
1	2	3.0	2.0	4.0
2	3	2.0	2.0	1.0
3	4	4.0	4.0	3.0
4	5	4.0	1.0	3.0

DATASET

- ***Dataset heads:***

Keterangan
- *EmployeeID*
- Tanggal

In [5]:
```
checkin = pd.read_csv('in_time.csv')
checkin.head()
```

	Unnamed: 0	2015-01-01	2015-01-02	2015-01-05	2015-01-06	2015-01-07	2015-01-08	2015-01-09	2015-01-12	2015-01-13	...	2015-12-18	2015-12-21	2015-12-22	201 12-2
0	1	NaN	2015-01-02 09:43:45	2015-01-05 10:08:48	2015-01-06 09:54:26	2015-01-07 09:34:31	2015-01-08 09:51:09	2015-01-09 10:09:25	2015-01-12 09:42:53	2015-01-13 10:13:06	...	NaN	2015-12-21 09:55:29	2015-12-22 10:04:06	2015-12-23 10:14:
1	2	NaN	2015-01-02 10:15:44	2015-01-05 10:21:05	NaN	2015-01-07 09:45:17	2015-01-08 10:09:04	2015-01-09 09:43:26	2015-01-12 10:00:07	2015-01-13 10:43:29	...	2015-12-18 10:37:17	2015-12-21 09:49:02	2015-12-22 10:33:51	2015-12-23 10:12:
2	3	NaN	2015-01-02 10:17:41	2015-01-05 09:50:50	2015-01-06 10:14:13	2015-01-07 09:47:27	2015-01-08 10:03:40	2015-01-09 10:05:49	2015-01-12 10:03:47	2015-01-13 10:21:26	...	2015-12-18 10:15:14	2015-12-21 10:10:28	2015-12-22 09:44:44	2015-12-23 10:15:
3	4	NaN	2015-01-02 10:05:06	2015-01-05 09:56:32	2015-01-06 10:11:07	2015-01-07 09:37:30	2015-01-08 10:02:08	2015-01-09 10:08:12	2015-01-12 10:13:42	2015-01-13 09:53:22	...	2015-12-18 10:17:38	2015-12-21 09:58:21	2015-12-22 10:04:25	2015-12-23 10:11:
4	5	NaN	2015-01-02 10:28:17	2015-01-05 09:49:58	2015-01-06 09:45:28	2015-01-07 09:49:37	2015-01-08 10:19:44	2015-01-09 10:00:50	2015-01-12 10:29:27	2015-01-13 09:59:32	...	2015-12-18 09:58:35	2015-12-21 10:03:41	2015-12-22 10:10:30	2015-12-23 10:13:

5 rows × 262 columns

In [6]:
```
checkout = pd.read_csv('out_time.csv')
checkout.head()
```

	Unnamed: 0	2015-01-01	2015-01-02	2015-01-05	2015-01-06	2015-01-07	2015-01-08	2015-01-09	2015-01-12	2015-01-13	...	2015-12-18	2015-12-21	2015-12-22	201 12-2
0	1	NaN	2015-01-02 16:56:15	2015-01-05 17:20:11	2015-01-06 17:19:05	2015-01-07 16:34:55	2015-01-08 17:08:32	2015-01-09 17:38:29	2015-01-12 16:58:39	2015-01-13 18:02:58	...	NaN	2015-12-21 17:15:50	2015-12-22 17:27:51	2015-12-23 16:44:
1	2	NaN	2015-01-02 18:22:17	2015-01-05 17:48:22	NaN	2015-01-07 17:09:06	2015-01-08 17:34:04	2015-01-09 16:52:29	2015-01-12 17:36:48	2015-01-13 18:00:13	...	2015-12-18 18:31:28	2015-12-21 17:34:16	2015-12-22 18:16:35	2015-12-23 17:38:
2	3	NaN	2015-01-02 16:59:14	2015-01-05 17:06:46	2015-01-06 16:38:32	2015-01-07 16:33:21	2015-01-08 17:24:22	2015-01-09 16:57:30	2015-01-12 17:28:54	2015-01-13 17:21:25	...	2015-12-18 17:02:23	2015-12-21 17:20:17	2015-12-22 16:32:50	2015-12-23 16:59:
3	4	NaN	2015-01-02 17:25:24	2015-01-05 17:14:03	2015-01-06 17:07:42	2015-01-07 16:32:40	2015-01-08 16:53:11	2015-01-09 17:19:47	2015-01-12 17:13:37	2015-01-13 17:11:45	...	2015-12-18 17:55:23	2015-12-21 16:49:09	2015-12-22 17:24:00	2015-12-23 17:36:
4	5	NaN	2015-01-02 18:31:37	2015-01-05 17:49:15	2015-01-06 17:26:25	2015-01-07 17:37:59	2015-01-08 17:59:28	2015-01-09 17:44:08	2015-01-12 18:51:21	2015-01-13 18:14:58	...	2015-12-18 17:52:48	2015-12-21 17:43:35	2015-12-22 18:07:57	2015-12-23 18:00:

5 rows × 262 columns

DATASET

- ***Dataset variables:***

Variable	Meaning	Levels
Age	Untuk mengetahu berapa lama karyawan bekerja	
Attrition	Untuk mengetahui apakah karyawan tersebut keluar pada tahun sebelumnya atau tidak.	
BusinessTravel	Seberapa sering karyawan melakukan perjalanan untuk tujuan bisnis dalam setahun terakhir.	
Department	Departmen/Divisi dalam Perusahaan	
DistanceFromHome	Jarak dari rumah (km)	
Education	Pendidikan	1 'Below College'
		2 'College'
		3 'Bachelor'
		4 'Master'
		5 'Doctor'
EducationField	Bidang Pendidikan	

DATASET

- ***Dataset variables:***

Variable	Meaning	Levels
EmployeeCount	Jumlah Karyawan	
EmployeeID	Nomor Identitas Karyawan	
EnvironmentSatisfaction	Tingkat kepuasaan Lingkungan Kerja	1 'Low'
		2 'Medium'
		3 'High'
		4 'Very High'
Gender	Jenis Kelamin Karyawan	
JobInvolvement	Tingkat keterlibatan pekerjaan	1 'Low'
		2 'Medium'
		3 'High'
		4 'Very High'

DATASET

- ***Dataset variables:***

Variable	Meaning	Levels
JobLevel	Tingkat level pekerjaan range 1 sampai 5	
JobRole	Tugas pekerjaan	
JobSatisfaction	Tingkat kepuasan kerja	1 'Low'
		2 'Medium'
		3 'High'
		4 'Very High'
MaritalStatus	Status perkawinan karyawan	
MonthlyIncome	Pendapatan per bulan (in Indian Rupees)	
NumCompaniesWorked	Jumlah total perusahaan tempat karyawan bekerja	
Over18	Karyawan yang bekerja berusia diatas 18 tahun	
PercentSalaryHike	Persen kenaikan gaji dengan tahun lalu	

DATASET

- ***Dataset variables:***

Variable	Meaning	Levels
PerformanceRating	Peringkat kinerja untuk tahun lalu	1 'Low'
		2 'Good'
		3 'Excellent'
		4 'Outstanding'
RelationshipSatisfaction	Tingkat kepuasan hubungan	1 'Low'
		2 'Medium'
		3 'High'
		4 'Very High'
StandardHours	Waktu standar untuk karyawan bekerja	
StockOptionLevel	Tingkat Opsi saham karyawan	
TotalWorkingYears	Total jumlah tahun karyawan tersebut bekerja	

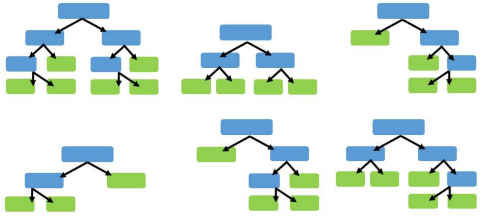
DATASET

- ***Dataset variables:***

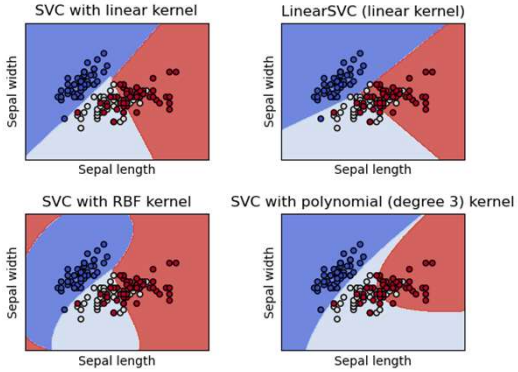
Variable	Meaning	Levels
TrainingTimesLastYear	Berapa kali pelatihan dilakukan untuk karyawan tahun lalu	
WorkLifeBalance	Tingkat keseimbangan kehidupan kerja	1 'Bad'
		2 'Good'
		3 'Better'
		4 'Best'
YearsAtCompany	Total jumlah tahun yang dihabiskan di perusahaan oleh karyawan	
YearsSinceLastPromotion	Jumlah tahun sejak promosi terakhir	
YearsWithCurrManager	Jumlah tahun di bawah manajer saat ini	

METHODOLOGY

- ***Supervised Learning***
  - Problem **Klasifikasi**
- ***Data Preprocessing***
  - *Exploratory Data Analysis* (EDA)
  - *Handling* missing values → berdasarkan metode imputansi (seperti Simple Imputer (Median), Random Forest atau KNN)
  - Kemungkinan data variable target *Attrition* tidak seimbang → Synthetic Minority Oversampling Technique (SMOTE)
  - Melakukan *encoding* dengan seperti cara *integer encoding* (Untuk variable category yang mempunyai 2 value) dan *Hot encoding* (variable yang memiliki 2 atau lebih value)
  - *Feature Engineering* (Berapa lama karyawan tersebut bekerja untuk manajer tertentu? Bagaimana hubungan pekerjaan dengan gelar karyawan?)
- ***Machine Learning Models***
  - *Random Forest Classifier*
  - *Support Vector Classifier*
  - XGBoost
  - Logistic Regression
  - Decision Tree Classifier
  - Hyperparameter Tuning menggunakan Grid Search Cross Validation



Modeling berdasarkan *Tree*



Support Vector Machine Classifier

DATA PREPROCESSING

- ***Handling Time data***
  - Mengisi null values dengan default timestamp (1970-01-01)

```
In [7]:
checkin = pd.read_csv('in_time.csv')
checkin=checkin.replace(np.nan,0)
checkin.iloc[:, 1:] = checkin.iloc[:, 1:].apply(pd.to_datetime, errors='coerce')
checkin.head()
```

	Unnamed: 0	2015-01-01	2015-01-02	2015-01-05	2015-01-06	2015-01-07	2015-01-08	2015-01-09	2015-01-12	2015-01-13	2015-01-14	2015-01-15	2015-01-16	2015-01-19	2 (
0	1	1970-01-01	2015-01-02 09:43:45	2015-01-05 10:08:48	2015-01-06 09:54:26	2015-01-07 09:34:31	2015-01-08 09:51:09	2015-01-09 10:09:25	2015-01-12 09:42:53	2015-01-13 10:13:06	1970-01-01	2015-01-15 10:01:24	2015-01-16 10:19:08	1970-01-01 00:00:00	20 01- 09:
1	2	1970-01-01	2015-01-02 10:15:44	2015-01-05 10:21:05	1970-01-01 00:00:00	2015-01-07 09:45:17	2015-01-08 10:09:04	2015-01-09 09:43:26	2015-01-12 10:00:07	2015-01-13 10:43:29	1970-01-01	2015-01-15 09:37:57	2015-01-16 09:57:18	2015-01-19 10:23:43	20 01- 09:
2	3	1970-01-01	2015-01-02 10:17:41	2015-01-05 09:50:50	2015-01-06 10:14:13	2015-01-07 09:47:27	2015-01-08 10:03:40	2015-01-09 10:05:49	2015-01-12 10:03:47	2015-01-13 10:21:26	1970-01-01	2015-01-15 09:55:11	2015-01-16 10:05:36	2015-01-19 09:47:53	20 01- 09:
3	4	1970-01-01	2015-01-02 10:05:06	2015-01-05 09:56:32	2015-01-06 10:11:07	2015-01-07 09:37:30	2015-01-08 10:02:08	2015-01-09 10:08:12	2015-01-12 10:13:42	2015-01-13 09:53:22	1970-01-01	2015-01-15 10:00:50	2015-01-16 09:58:06	2015-01-19 09:43:11	20 01- 10:
4	5	1970-01-01	2015-01-02 10:28:17	2015-01-05 09:49:58	2015-01-06 09:45:28	2015-01-07 09:49:37	2015-01-08 10:19:44	2015-01-09 10:00:50	2015-01-12 10:29:27	2015-01-13 09:59:32	1970-01-01	2015-01-15 10:06:12	2015-01-16 10:03:50	1970-01-01 00:00:00	20 01- 10:

- Menambahkan waktu check in dan check out karyawan menjadi satu frame.

```
In [79]:
print('Shape before append:',checkin.shape)
attendance = checkin.append(checkout, ignore_index=True)
print('Shape after append:',attendance.shape)
attendance.head()
```

```
Shape before append: (4410, 262)
Shape after append: (8820, 262)
```

- ***Handling Time data***
  - Melakukan perhitungan check in dan check out dan perbedaan waktu dengan actual waktu kerja perhari
  - Menghapus data kolom hari libur, normalisasi nilai jam kerja, dan membuat kolom EmplyeeID.

In [9]:
```
attendance = attendance.diff(periods=4410)
attendance = attendance.iloc[4410:]
attendance.head()
```

	Unnamed: 0	2015-01-01	2015-01-02	2015-01-05	2015-01-06	2015-01-07	2015-01-08	2015-01-09	2015-01-12	2015-01-13	2015-01-14	2015-01-15	2015-01-16	2015-01-19
4410	0.0	0 days	0 days 07:12:30	0 days 07:11:23	0 days 07:24:39	0 days 07:00:24	0 days 07:17:23	0 days 07:29:04	0 days 07:15:46	0 days 07:49:52	0 days	0 days 07:20:49	0 days 07:16:03	0 days 00:00:00
4411	0.0	0 days	0 days 08:06:33	0 days 07:27:17	0 days 00:00:00	0 days 07:23:49	0 days 07:25:00	0 days 07:09:03	0 days 07:36:41	0 days 07:16:44	0 days	0 days 07:36:47	0 days 07:43:39	0 days 07:34:39
4412	0.0	0 days	0 days 06:41:33	0 days 07:15:56	0 days 06:24:19	0 days 06:45:54	0 days 07:20:42	0 days 06:51:41	0 days 07:25:07	0 days 06:59:59	0 days	0 days 07:26:18	0 days 07:12:37	0 days 07:04:20
4413	0.0	0 days	0 days 07:20:18	0 days 07:17:31	0 days 06:56:35	0 days 06:55:10	0 days 06:51:03	0 days 07:11:35	0 days 06:59:55	0 days 07:18:23	0 days	0 days 06:52:36	0 days 06:54:28	0 days 06:31:07
4414	0.0	0 days	0 days 08:03:20	0 days 07:59:17	0 days 07:40:57	0 days 07:48:22	0 days 07:39:44	0 days 07:43:18	0 days 08:21:54	0 days 08:15:26	0 days	0 days 08:15:36	0 days 08:24:13	0 days 00:00:00

```
att = attendance.reset_index().copy()
att.pop('index')
att.pop('Unnamed: 0')
# Normalize the actual daily work hour values
att = att/np.timedelta64(1, 'h')
# Create 'EmployeeID'
att['EmployeeID'] = range(1,4411)
# Remove public holiday columns
for i in att.columns:
    if ((att[i]==0).all()) == True:
        att.pop(i)
    else:
        continue
```

	2015-01-02	2015-01-05	2015-01-06	2015-01-07	2015-01-08	2015-01-09	2015-01-12	2015-01-13	2015-01-15	2015-01-16	2015-01-19	2015-01-20	2015-01-21	2 0
0	7.208333	7.189722	7.410833	7.006667	7.289722	7.484444	7.262778	7.831111	7.346944	7.267500	0.000000	6.775833	7.095000	7.05
1	8.109167	7.454722	0.000000	7.396944	7.416667	7.150833	7.611389	7.278889	7.613056	7.727500	7.577500	7.602778	7.905556	7.37
2	6.692500	7.265556	6.405278	6.765000	7.345000	6.861389	7.418611	6.999722	7.438333	7.210278	7.072222	6.920556	6.802778	7.47
3	7.338333	7.291944	6.943056	6.919444	6.850833	7.193056	6.998611	7.306389	6.876667	6.907778	6.518611	7.178889	6.705000	7.01
4	8.055556	7.988056	7.682500	7.806111	7.662222	7.721667	8.365000	8.257222	8.260000	8.403611	0.000000	7.815278	8.223056	8.29

- ***Handling Time data***
  - Variabel *Absent* : jumlah hari setiap karyawan tidak hadir di kantor
  - Variabel *AvgActualHours* : jam kerja aktual masing-masing karyawan selama setahun penuh. Variabel ini akan dinormalisasi menjadi *RelativeWorkHours* (proporsi jam kerja dibandingkan standar durasi kerja 8 jam)
  - Merge kedua kolom dengan *EmployeeID* menjadi data frame baru

New variable: *Absent*

```
In [13]:
att['Absent'] = ((att == 0).astype(int).sum(axis=1))-12
att['Absent'].head()
```

```
0    17
1    13
2     7
3    14
4     4
Name: Absent, dtype: int64
```

New variable: *AvgActualHours*

```
In [15]:
temp = att[['EmployeeID','Absent']].copy()
att.pop('EmployeeID')
att.pop('Absent')
att['ActualHours'] = att.mean(axis=1)
```

```
In [16]:
att['ActualHours']
```

```
0       6.870229
1       7.315971
2       6.816081
3       6.789215
4       7.877561
          ...
4405    8.316921
4406    5.897197
4407    7.149526
4408    9.187612
4409    6.511790
Name: ActualHours, Length: 4410, dtype: float64
```

```
In [17]:
worktime = temp.copy()
worktime['AvgActualHours'] = att['ActualHours'].copy()
worktime
```

	EmployeeID	Absent	AvgActualHours
0	1	17	6.870229
1	2	13	7.315971
2	3	7	6.816081
3	4	14	6.789215

- ***Handling Duplicated Values***
  - Tidak terdapat baris data yang merupakan duplicates dari baris data lainnya

```
df_whole[df_whole.duplicated()]
```

Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeID	Gender	JobLevel

- ***Handling Null Values***
  - Identifikasi variabel-variabel dengan null values terlebih dahulu
  - Sebelum eksekusi proses preprocessing null values dan outliers, perlu dilakukan **splitting** terhadap data untuk training dan testing, agar preprocessing data testing **tidak dipengaruhi** oleh data-data dari training set

```
#null di train data
X_trainp.isna().sum()
```

Age	0
BusinessTravel	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
Gender	0
JobLevel	0
JobRole	0
MaritalStatus	0
MonthlyIncome	0
NumCompaniesWorked	19
PercentSalaryHike	0
StockOptionLevel	0
TotalWorkingYears	6
TrainingTimesLastYear	0
YearsAtCompany	0
YearsSinceLastPromotion	0
YearsWithCurrManager	0
JobInvolvement	0
PerformanceRating	0
EnvironmentSatisfaction	18
JobSatisfaction	16
WorkLifeBalance	30
Absent	0
AvgActualHours	0
RelativeWorkHours	0
AgeGroup	0

• *Handling Null Values*

- Null terdapat di variabel *NumCompaniesWorked*, *TotalWorkingYears*, *EnvironmentSatisfaction*, *JobSatisfaction*, *WorkLifeBalance*
- Pada kolom ***NumCompaniesWorked***: Null values diisi dengan median jumlah perusahaan tempat karyawan bekerja berdasarkan kelompok usia karyawan (*AgeGroup*)
- Pada kolom ***TotalWorkingYears***: Null values diisi dengan median jumlah tahun karyawan tersebut berkarier sejauh ini berdasarkan kelompok usia karyawan
- ***AgeGroup*** dibuat dan digunakan karena merupakan faktor penentu karier seseorang.
- Median dipilih karena terjadi skewed distributions pada variable di beberapa grup usia.

```
#null di test data
X_testp.isna().sum()
```

Age	0
BusinessTravel	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
Gender	0
JobLevel	0
JobRole	0
MaritalStatus	0
MonthlyIncome	0
NumCompaniesWorked	0
PercentSalaryHike	0
StockOptionLevel	0
TotalWorkingYears	3
TrainingTimesLastYear	0
YearsAtCompany	0
YearsSinceLastPromotion	0
YearsWithCurrManager	0
JobInvolvement	0
PerformanceRating	0
EnvironmentSatisfaction	7
JobSatisfaction	4
WorkLifeBalance	8
Absent	0
AvgActualHours	0
RelativeWorkHours	0
AgeGroup	0

New variable: *AgeGroup*

```
idxb30 = df_whole[(df_whole['Age']<30)].index
idx30s = df_whole[(df_whole['Age']<40)&(df_whole['Age']>=30)].index
idx40s = df_whole[(df_whole['Age']<50)&(df_whole['Age']>=40)].index
idx50s = df_whole[(df_whole['Age']>=50)].index
df_whole.loc[idxb30,'AgeGroup'] = 'Under 30'
df_whole.loc[idx30s,'AgeGroup'] = '30s'
df_whole.loc[idx40s,'AgeGroup'] = '40s'
df_whole.loc[idx50s,'AgeGroup'] = '50 and above'
df_whole['AgeGroup'].value_counts()
```

```
30s             1866
40s             1047
Under 30         978
50 and above     519
Name: AgeGroup, dtype: int64
```

```
X_trainp.groupby('AgeGroup')['NumCompaniesWorked'].describe()
```

AgeGroup	count	mean	std	min	25%	50%	75%	max
30s	1501.0	2.584277	2.514039	0.0	1.0	1.0	4.0	9.0
40s	813.0	3.371464	2.507088	0.0	1.0	3.0	5.0	9.0
50 and above	408.0	3.796569	2.503126	0.0	2.0	3.0	6.0	9.0
Under 30	787.0	1.724269	2.106323	0.0	1.0	1.0	1.0	9.0

Median jumlah tempat kerja sebelumnya untuk:

- Karyawan muda di bawah 30 tahun = 1
- Karyawan usia 30-39 = 1
- Karyawan usia 40-49 = 3
- Karyawan usia 50 keatas = 3

n.b. Tidak terdapat null values di variabel *NumCompaniesWorked* pada test data

```
idxb30 = X_trainp[(X_trainp['Age']<30)&(X_trainp['NumCompaniesWorked'].isna()==True)].index
idx30s = X_trainp[(X_trainp['Age']<40)&(X_trainp['Age']>=30)
                  &(X_trainp['NumCompaniesWorked'].isna()==True)].index
idx40s = X_trainp[(X_trainp['Age']<50)&(X_trainp['Age']>=40)
                  &(X_trainp['NumCompaniesWorked'].isna()==True)].index
idx50s = X_trainp[(X_trainp['Age']>=50)&(X_trainp['NumCompaniesWorked'].isna()==True)].index
X_trainp.loc[idxb30,'NumCompaniesWorked'] = 1.0
X_trainp.loc[idx30s,'NumCompaniesWorked'] = 1.0
X_trainp.loc[idx40s,'NumCompaniesWorked'] = 3.0
X_trainp.loc[idx50s,'NumCompaniesWorked'] = 3.0
```

```
X_trainp.groupby('AgeGroup')['TotalWorkingYears'].describe()
```

AgeGroup	count	mean	std	min	25%	50%	75%	max
30s	1505.0	9.605316	4.579350	1.0	6.0	10.0	12.0	21.0
40s	814.0	14.977887	7.143547	1.0	9.0	14.0	21.0	31.0
50 and above	410.0	21.665854	10.109332	1.0	13.0	22.0	31.0	40.0
Under 30	793.0	5.036570	2.988602	0.0	2.0	5.0	7.0	11.0

Median jumlah tahun bekerja untuk:

- Karyawan muda di bawah 30 tahun = 5.0
- Karyawan usia 30-39 = 10.0
- Karyawan usia 40-49 = 14.0
- Karyawan usia 50 keatas = 22.0

```
idxb30 = X_trainp[(X_trainp['Age']<30)&(X_trainp['TotalWorkingYears'].isna()==True)].index
idx30s = X_trainp[(X_trainp['Age']<40)&(X_trainp['Age']>=30)
                  &(X_trainp['TotalWorkingYears'].isna()==True)].index
idx40s = X_trainp[(X_trainp['Age']<50)&(X_trainp['Age']>=40)
                  &(X_trainp['TotalWorkingYears'].isna()==True)].index
idx50s = X_trainp[(X_trainp['Age']>=50)&(X_trainp['TotalWorkingYears'].isna()==True)].index
X_trainp.loc[idxb30,'TotalWorkingYears'] = 5.0
X_trainp.loc[idx30s,'TotalWorkingYears'] = 10.0
X_trainp.loc[idx40s,'TotalWorkingYears'] = 14.0
X_trainp.loc[idx50s,'TotalWorkingYears'] = 22.0
```

```
X_testp.groupby('AgeGroup')['TotalWorkingYears'].describe()
```

AgeGroup	count	mean	std	min	25%	50%	75%	max
30s	357.0	9.492997	4.492576	1.0	6.0	10.0	12.0	21.0
40s	229.0	15.441048	6.993874	1.0	10.0	15.0	22.0	31.0
50 and above	109.0	21.165138	9.992139	1.0	14.0	21.0	30.0	40.0
Under 30	184.0	4.815217	2.883643	0.0	2.0	5.0	7.0	11.0

> Median jumlah tahun bekerja di test data untuk:
>
> - Karyawan muda di bawah 30 tahun = 5.0
> - Karyawan usia 30-39 = 10.0
> - Karyawan usia 40-49 = 15.0
> - Karyawan usia 50 keatas = 21.0

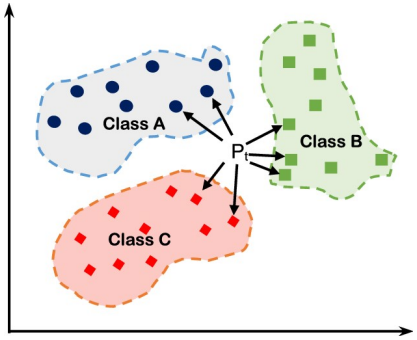
```
X_testp[(X_testp['TotalWorkingYears'].isna()==True)]
```

	Age	BusinessTravel	Department	DistanceFromHome	Education	EducationField	Gender	JobLevel	JobRole
308	47	Travel_Frequently	Research & Development	4	3	Life Sciences	1	1	Research Director
2367	39	Travel_Rarely	Sales	2	4	Life Sciences	1	1	Laboratory Technician
23	42	Travel_Rarely	Research & Development	4	4	Life Sciences	1	1	Manufacturing Director

```
X_testp.loc[308,'TotalWorkingYears'] = 15
X_testp.loc[2367,'TotalWorkingYears'] = 10
X_testp.loc[23,'TotalWorkingYears'] = 15
```

DATA PREPROCESSING

- ***Handling Null Values***
  - Pada variabel *EnvironmentSatisfaction*, *JobSatisfaction*, and *WorkLifeBalance*: Menggunakan method **K-Nearest Neighbors** (KNN)
    - Melakukan encoding data kategori menjadi data numeric dengan (pandas.get_dummies())
    - Mengisi null values menggunakan Teknik KNN algorithm (sklearn.impute.KNNImputer()) dengan 5 Neighbor terdekat.
    - Algoritma KNN melakukan klasifikasi terhadap objek berdasarkan data *training* yang jaraknya paling dekat dengan objek tersebut.
    - Algoritma KNN akan menentukan baris mana dalam data yang serupa (yaitu 'dekat') ke baris dengan nilai *null* dan memperkirakan nilai nol berdasarkan nilai dari baris serupa tersebut.



```
X_trainp = pd.get_dummies(X_trainp)
X_testp = pd.get_dummies(X_testp)
from sklearn.impute import KNNImputer
# define imputer
imputer = KNNImputer()
# fit transform the dataset
X_trainp = pd.DataFrame(imputer.fit_transform(X_trainp),columns = X_trainp.columns)
# fit transform the dataset
X_testp = pd.DataFrame(imputer.fit_transform(X_testp),columns = X_testp.columns)
```

```
X_trainp.isna().sum()
```

```
Age                          0
DistanceFromHome             0
Education                    0
Gender                       0
MonthlyIncome                0
NumCompaniesWorked           0
PercentSalaryHike            0
StockOptionLevel             0
TotalWorkingYears            0
TrainingTimesLastYear        0
YearsAtCompany               0
YearsSinceLastPromotion      0
YearsWithCurrManager         0
JobInvolvement               0
PerformanceRating            0
EnvironmentSatisfaction      0
JobSatisfaction              0
WorkLifeBalance              0
Absent                       0
RelativeWorkHours            0
```

```
X_testp.isna().sum()
```

```
Age                          0
DistanceFromHome             0
Education                    0
Gender                       0
MonthlyIncome                0
NumCompaniesWorked           0
PercentSalaryHike            0
StockOptionLevel             0
TotalWorkingYears            0
TrainingTimesLastYear        0
YearsAtCompany               0
YearsSinceLastPromotion      0
YearsWithCurrManager         0
JobInvolvement               0
PerformanceRating            0
EnvironmentSatisfaction      0
JobSatisfaction              0
WorkLifeBalance              0
Absent                       0
RelativeWorkHours            0
```

DATA PREPROCESSING

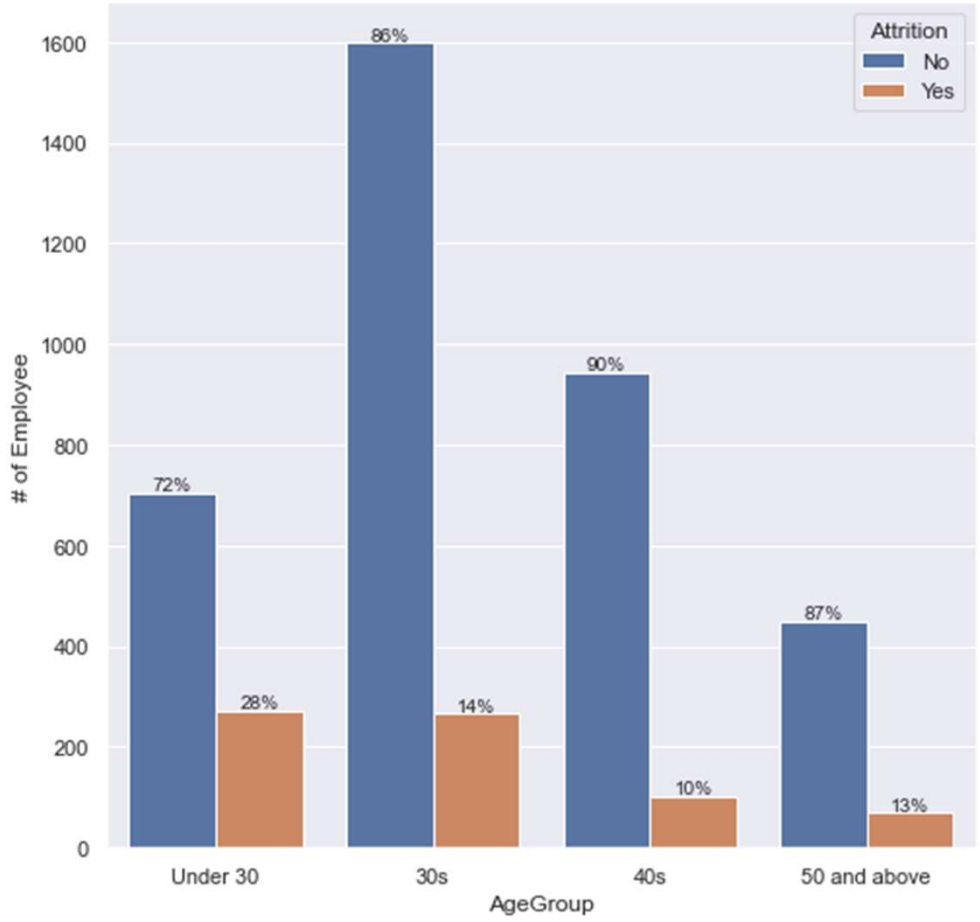
- ***Handling Irregular Values***
  - Melakukan pemeriksaan terhadap semua nilai unik yang ada pada dataset.

- ***Handling Outliers***
  - Mengimplementasikan metode scaling seperti *RobustScaler()* untuk mengatasi pengaruh outliers dalam variabel seperti *MonthlyIncome*, *DistanceFromHome*, *Absent,* etc.
  - Melakuan normalisasi variabel pada dataset yang memiliki skewed distribution
  - Scaling juga membantu menyamakan skala nilai di seluruh variabel
  - Scaling akan dilakukan sebelum modeling

```
for i in X_trainp.columns:
    print(i,':')
    print(X_trainp[i].sort_values().unique())
    print(X_testp[i].sort_values().unique())
```

```
Age :
[18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53.
 54. 55. 56. 57. 58. 59. 60.]
[18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53.
 54. 55. 56. 57. 58. 59.]
DistanceFromHome :
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.
 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29.]
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.
 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29.]
Gender :
[0. 1.]
[0. 1.]
NumCompaniesWorked :
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
PercentSalaryHike :
[11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25.]
[11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25.]
StockOptionLevel :
[0. 1. 2. 3.]
[0. 1. 2. 3.]
TotalWorkingYears :
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17.
 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 40.]
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17.
 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 40.]
```

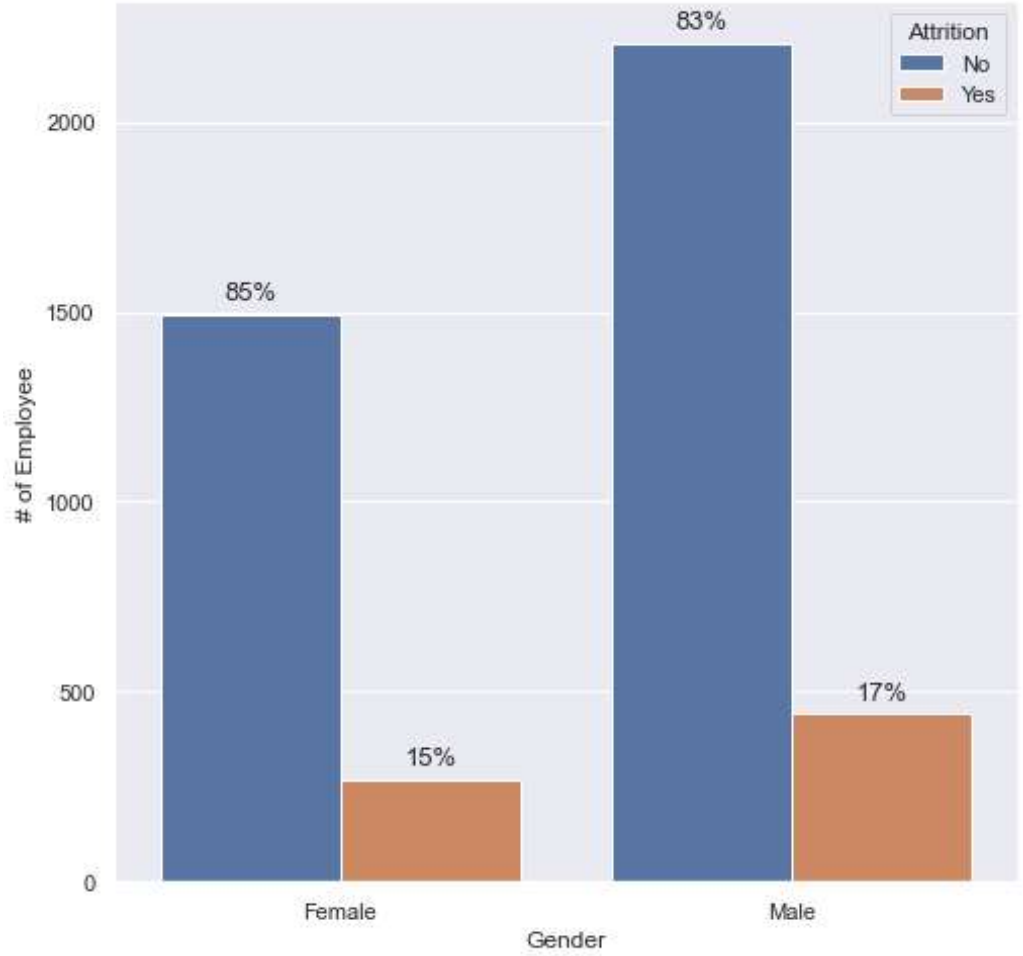
EDA : FINDINGS + GRAPH (1)



Mayoritas pegawai berada di kelompok usia **30s**.

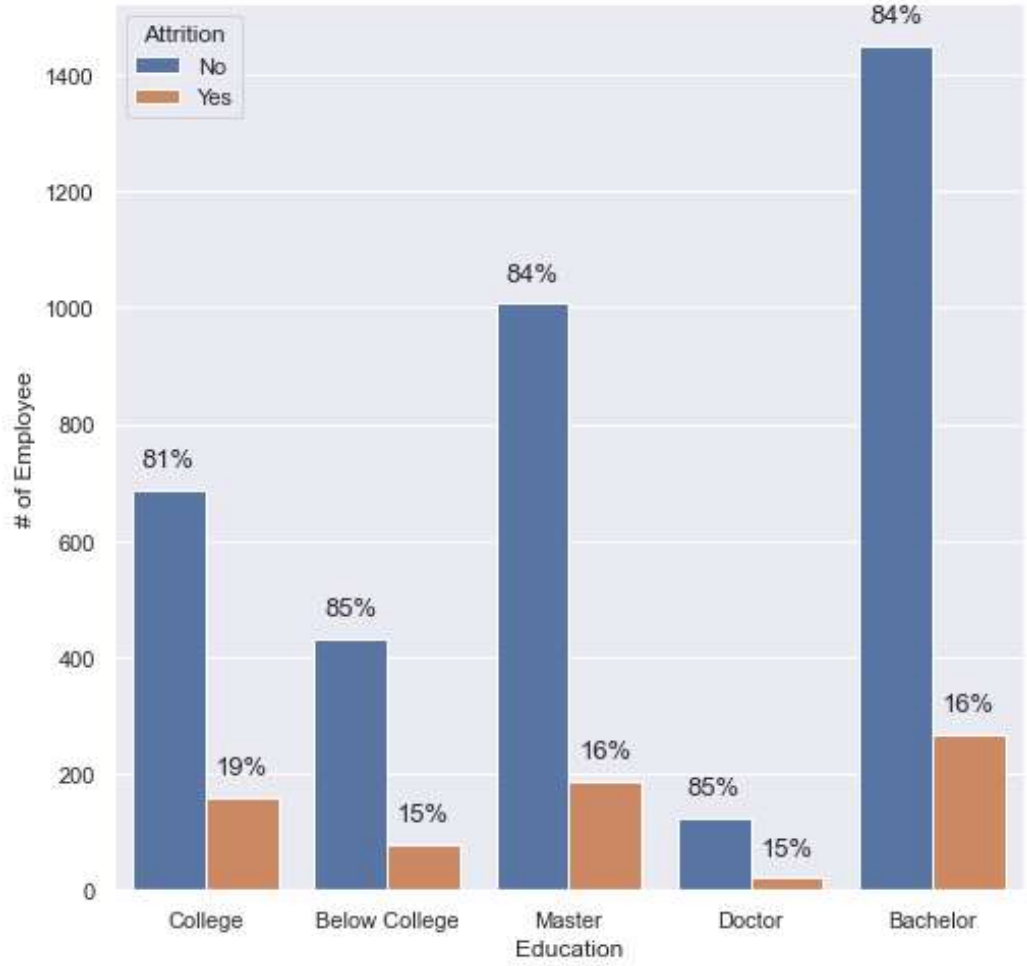
Tingkat attrition pegawai di kelompok usia **Under 30 lebih tinggi** dari kelompok usia lainnya.

EDA : FINDINGS + GRAPH (2)



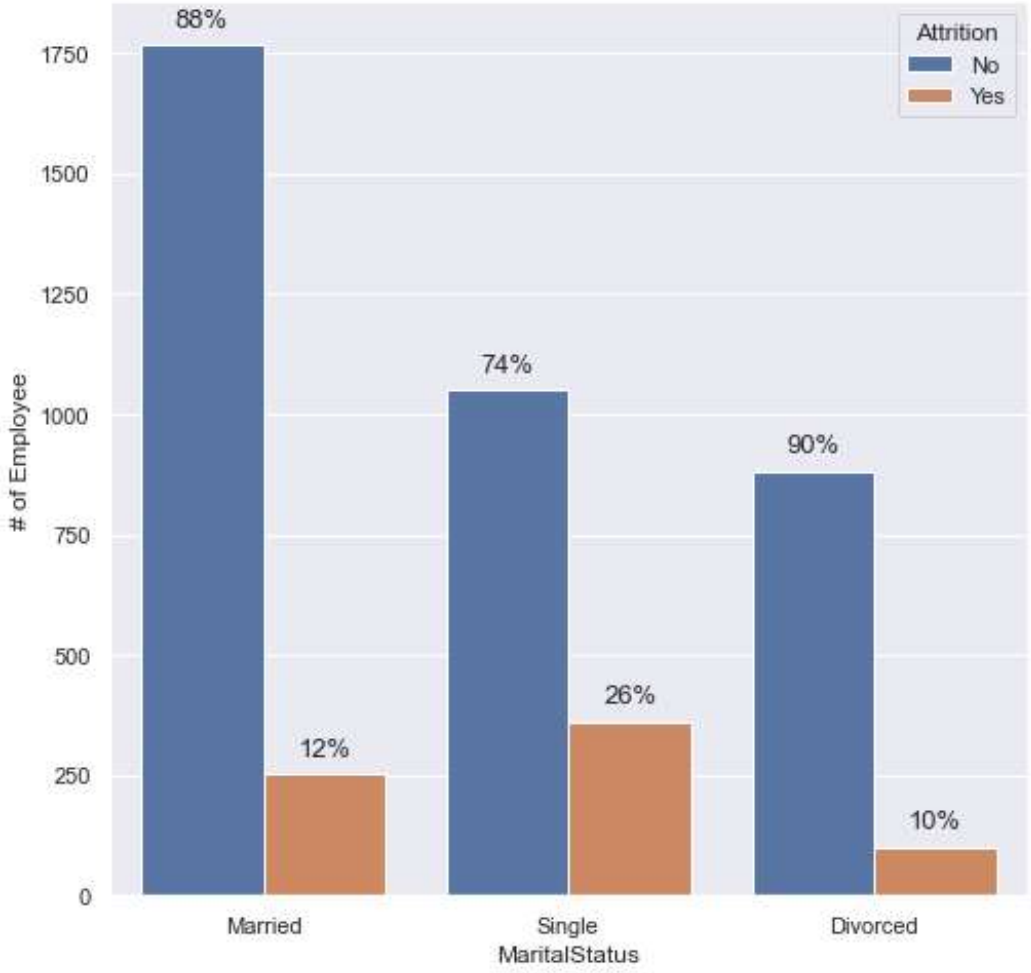
Mayoritas pegawai berjenis kelamin **laki-laki.** Tingkat attrition laki-laki cenderung **lebih tinggi** daripada perempuan

EDA : FINDINGS + GRAPH (3)



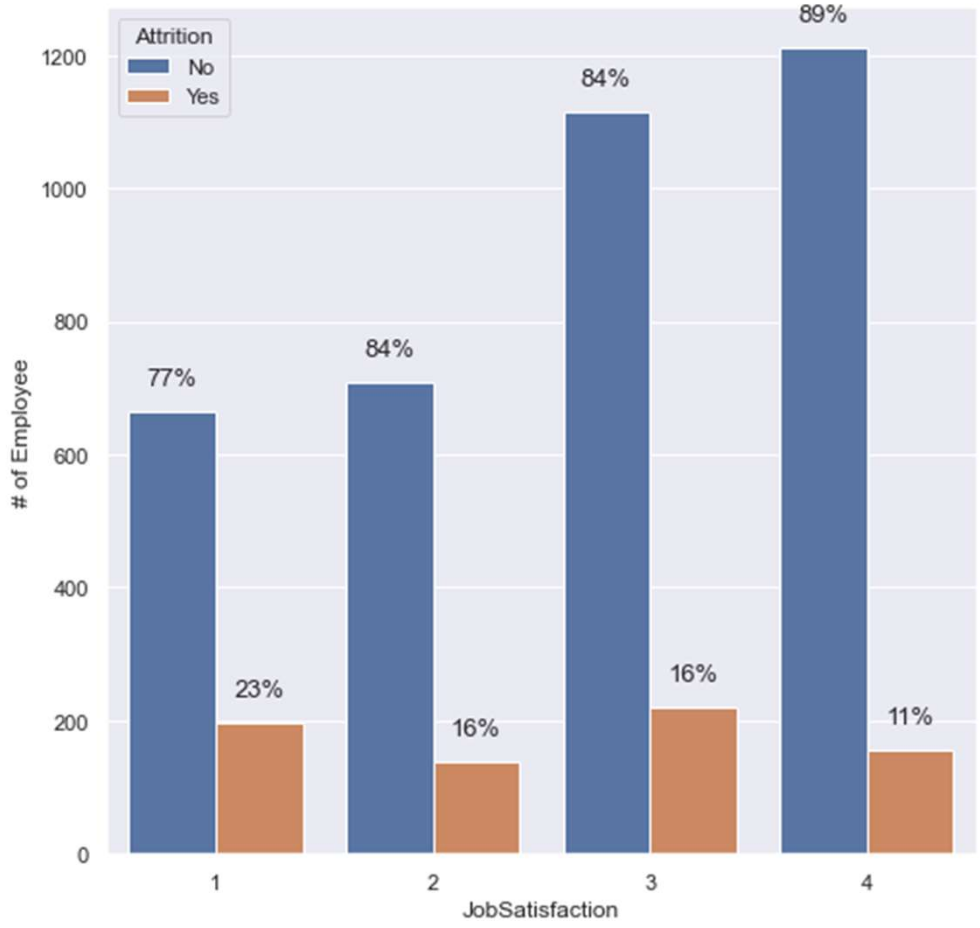
Mayoritas pegawai berpendidikan **Sarjana/S1.** Tingkat attrition cenderung tinggi pada pegawai dengan jenjang pendidikan **College**

EDA : FINDINGS + GRAPH (4)



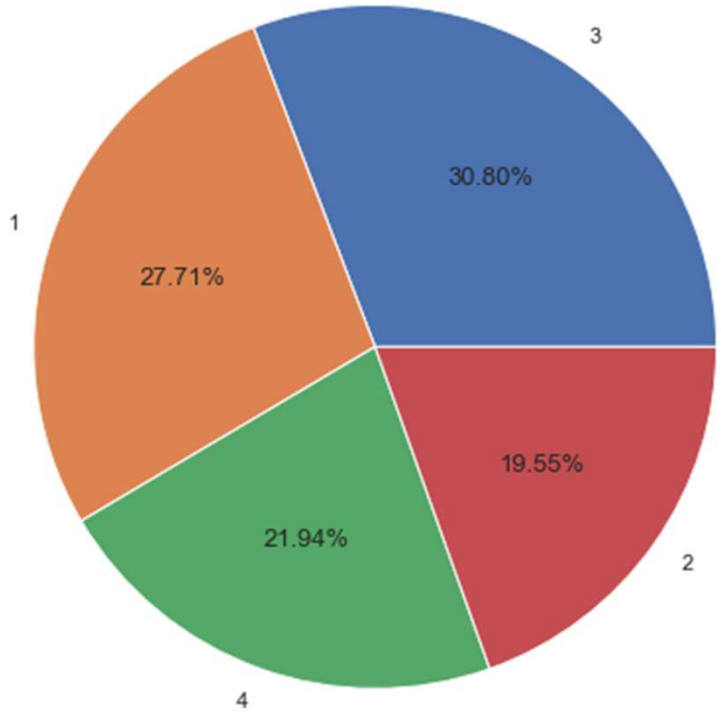
Mayoritas pegawai berstatus **menikah.** Tingkat attrition cenderung tinggi pada pegawai berstatus **single**

EDA : FINDINGS + GRAPH (5)



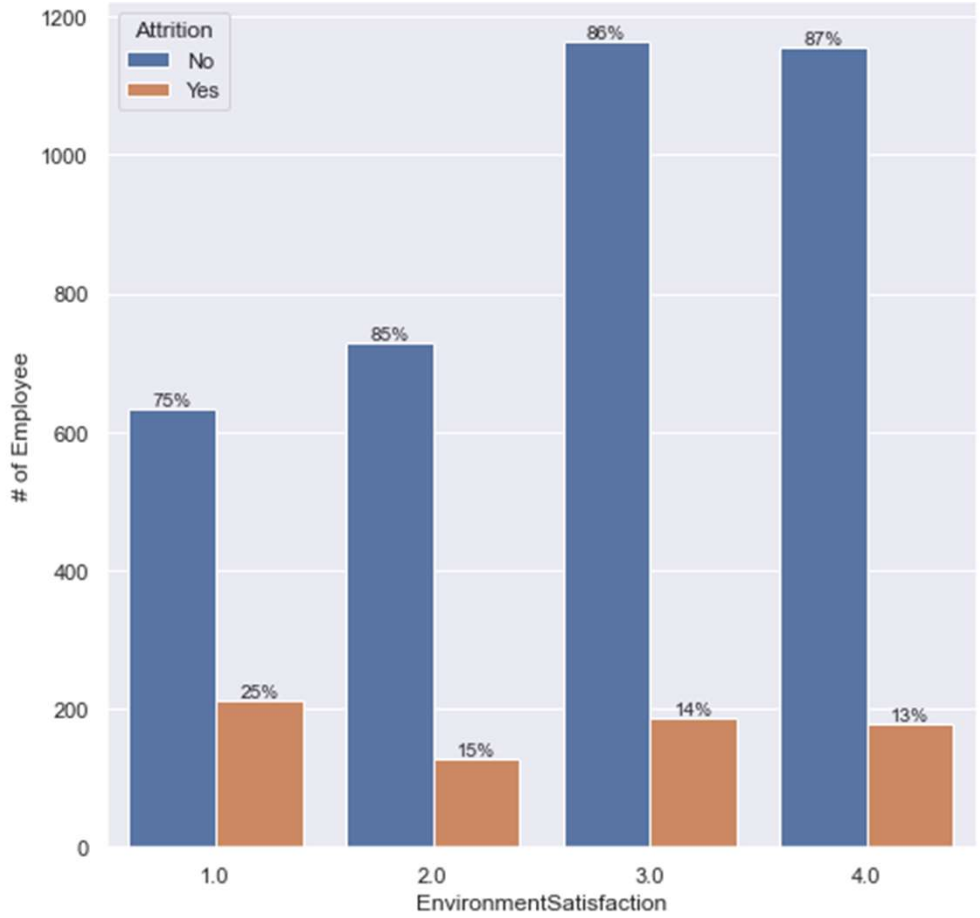
Semakin puas pegawai terhadap pekerjaannya, tingkat attrition cenderung **menurun**

EDA : FINDINGS + GRAPH (5 cont...)



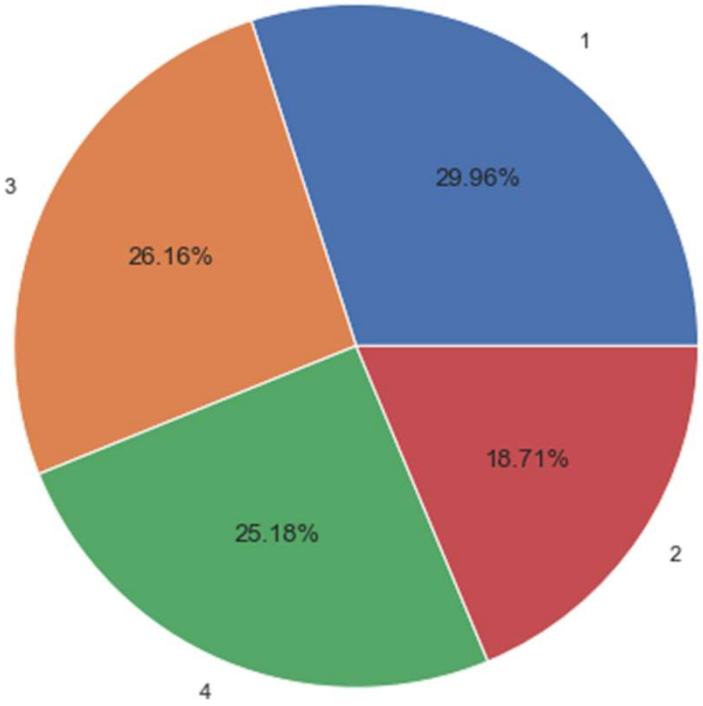
Sebagian besar pegawai yang berhenti bekerja memberikan penilaian *job satisfaction* antara **1** atau **3 dari 4**

EDA : FINDINGS + GRAPH (6)



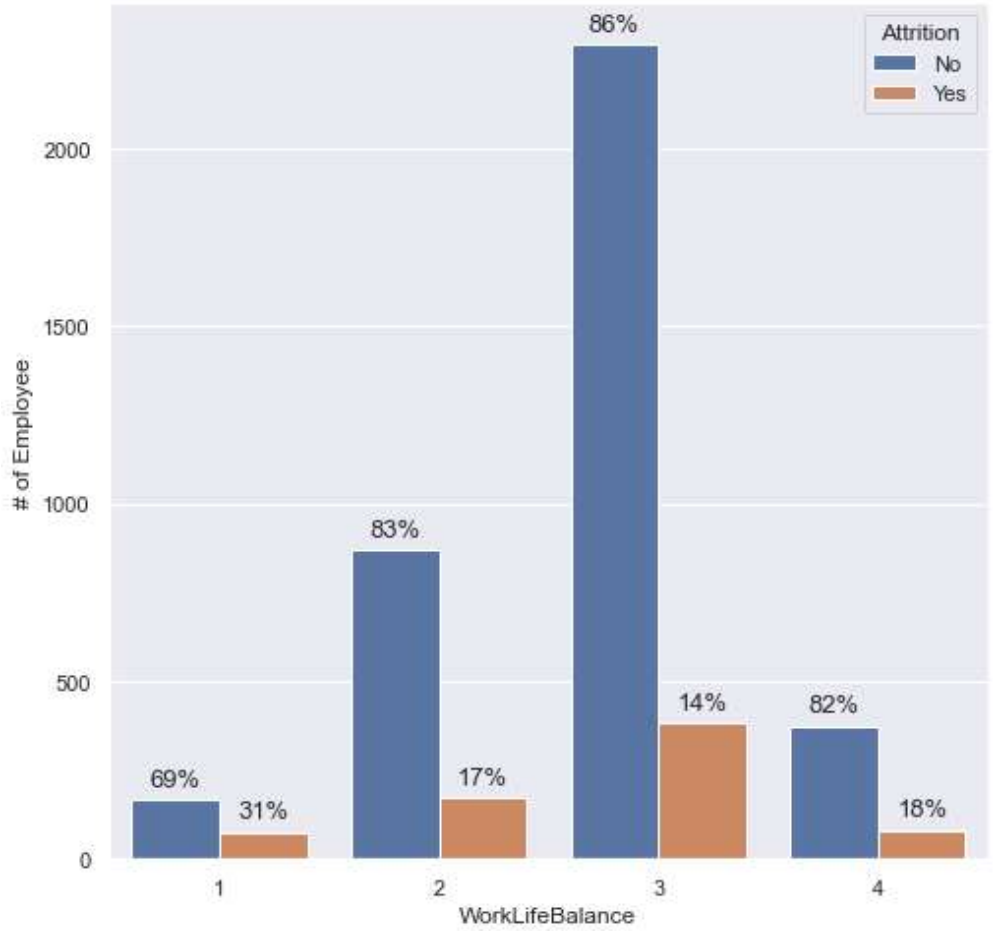
Semakin puas seorang pegawai dengan lingkungan pekerjaannya, tingkat attrition cenderung **menurun**

EDA : FINDINGS + GRAPH (6 cont...)



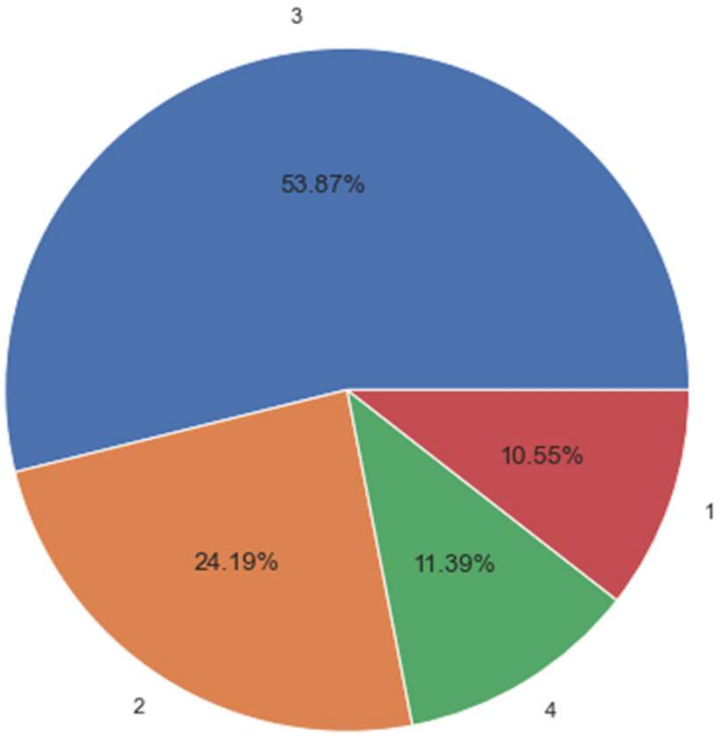
Sebagian besar pegawai yang berhenti bekerja memberikan penilaian kepuasan terhadap lingkungan kerja antara **1** atau **3** **dari 4**

EDA : FINDINGS + GRAPH (7)



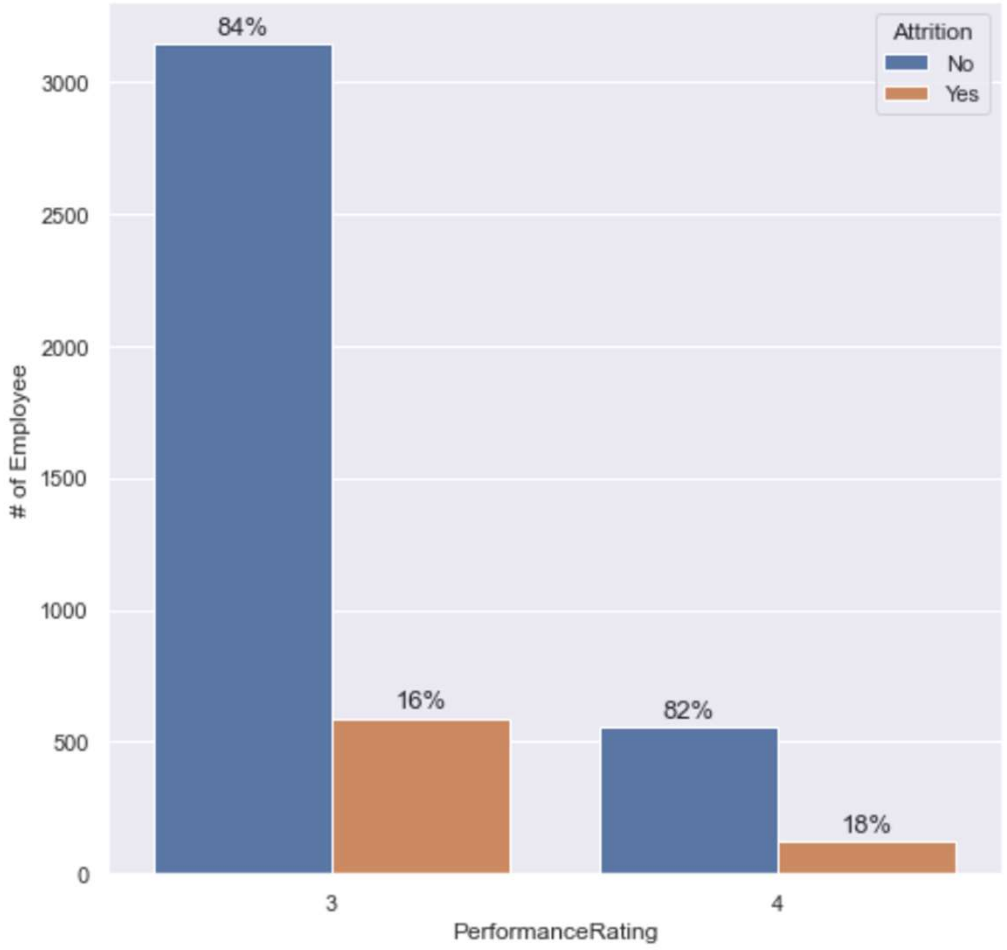
Semakin tinggi tingkat *work life balance* pegawai, tingkat attrition cenderung **menurun**

EDA : FINDINGS + GRAPH (7 cont...)



Sebagian besar pegawai yang berhenti bekerja memberikan penilaian *work life balance* mereka sebesar **3 dari 4**

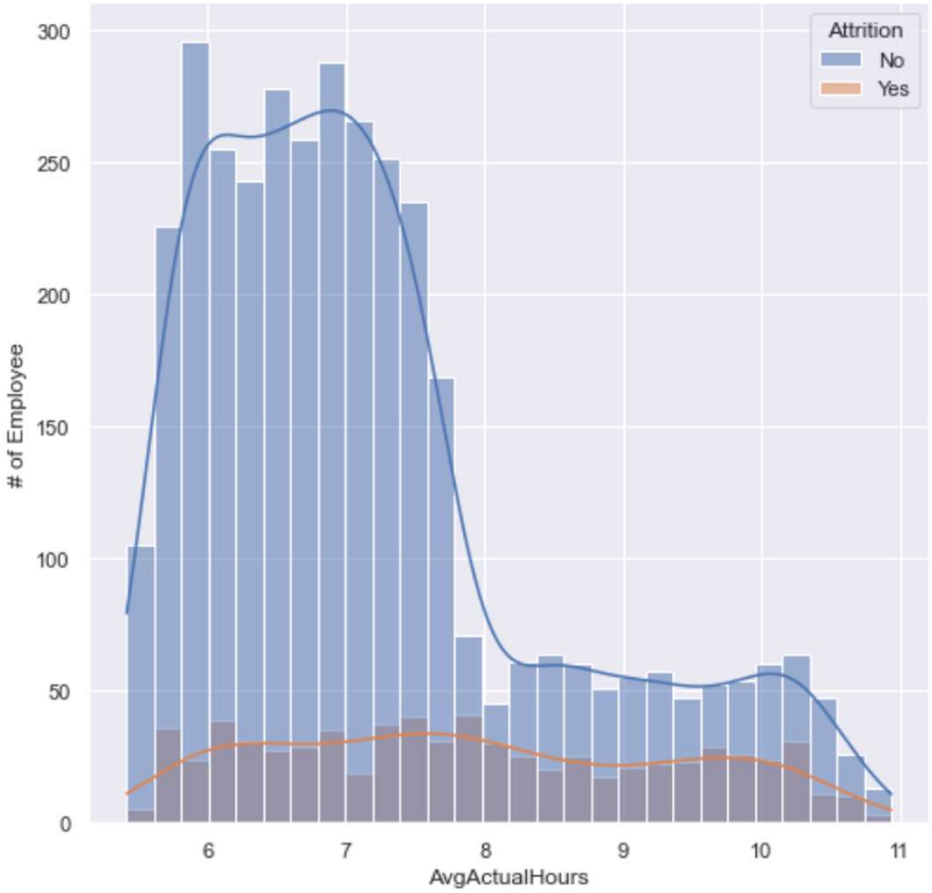
EDA : FINDINGS + GRAPH (8)



Mayoritas pegawai mendapatkan performance rating **3 dari 4.**

Tingkat attrition pada pegawai dengan performance rating **4 dari 4** cenderung **lebih tinggi** daripada pegawai dengan performance rating 3 dari 4

EDA : FINDINGS + GRAPH (9)

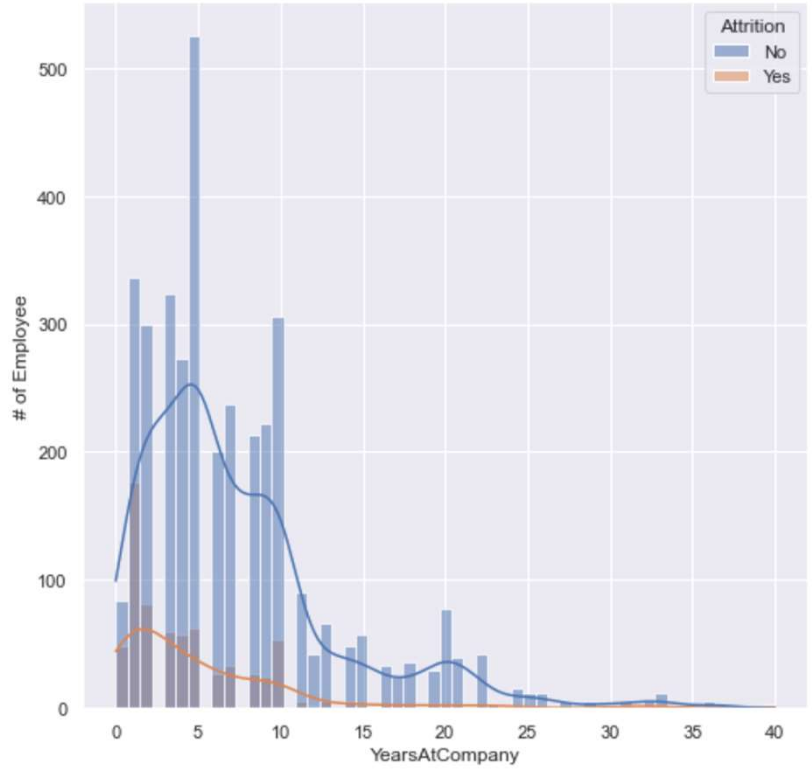


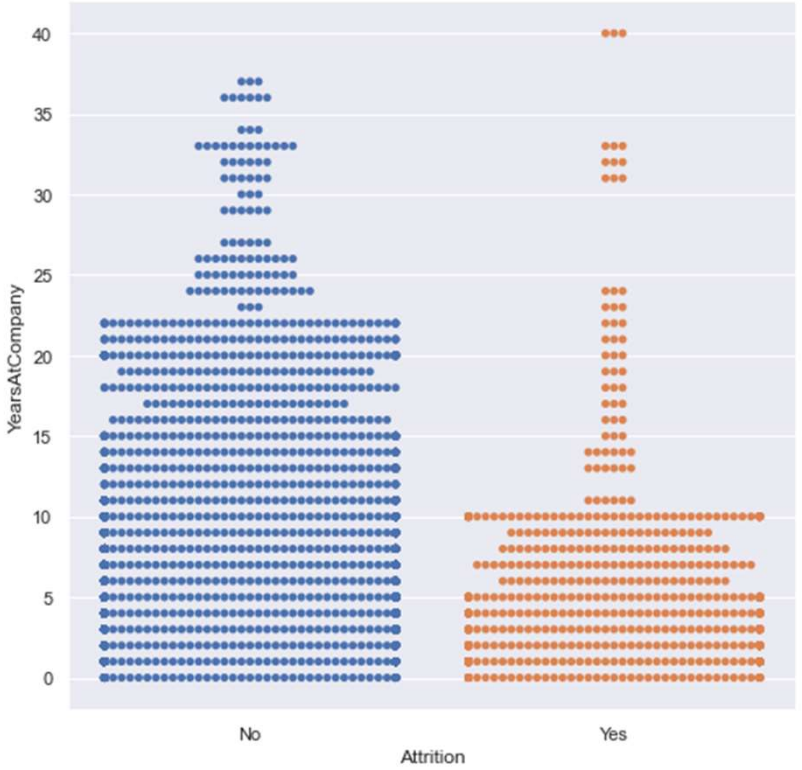
Rata-rata jam kerja pegawai yang berhenti bekerja **lebih tinggi** dibandingkan pegawai yang tidak berhenti

Distribusi jam kerja pegawai yang berhenti bekerja beragam dan cenderung mengikuti kurva **normal**

Distribusi jam kerja pegawai yang tidak berhenti bekerja tahun lalu cenderung **positively skewed** dengan median jam kerja pegawai lebih rendah dari nilai rata-rata

EDA : FINDINGS + GRAPH (10)

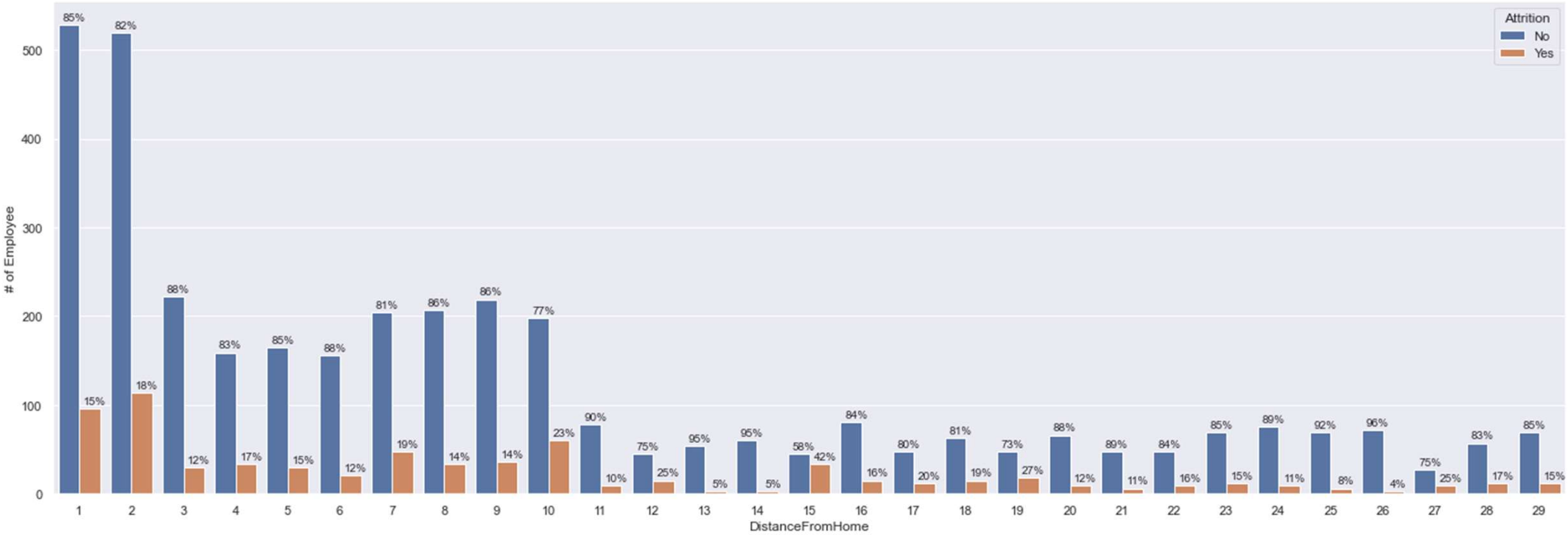




Semakin lama pegawai bekerja di perusahaan, tingkat attrition cenderung **menurun**

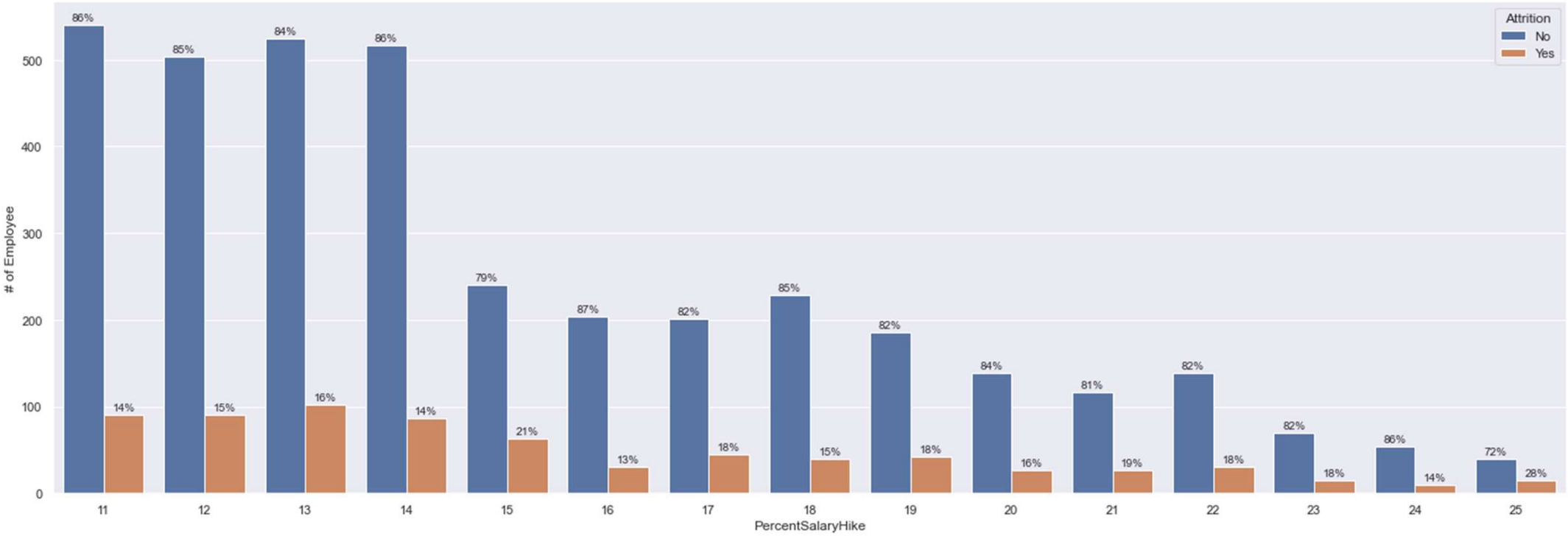
EDA : FINDINGS + GRAPH (11)

Semakin dekat jarak kantor dari rumah pegawai, tingkat attrition cenderung semakin **menurun**



EDA : FINDINGS + GRAPH (12)

Semakin besar presentase kenaikan gaji, tingkat attritionnya cenderung **meningkat**



EDA : FINDINGS + GRAPH (13)

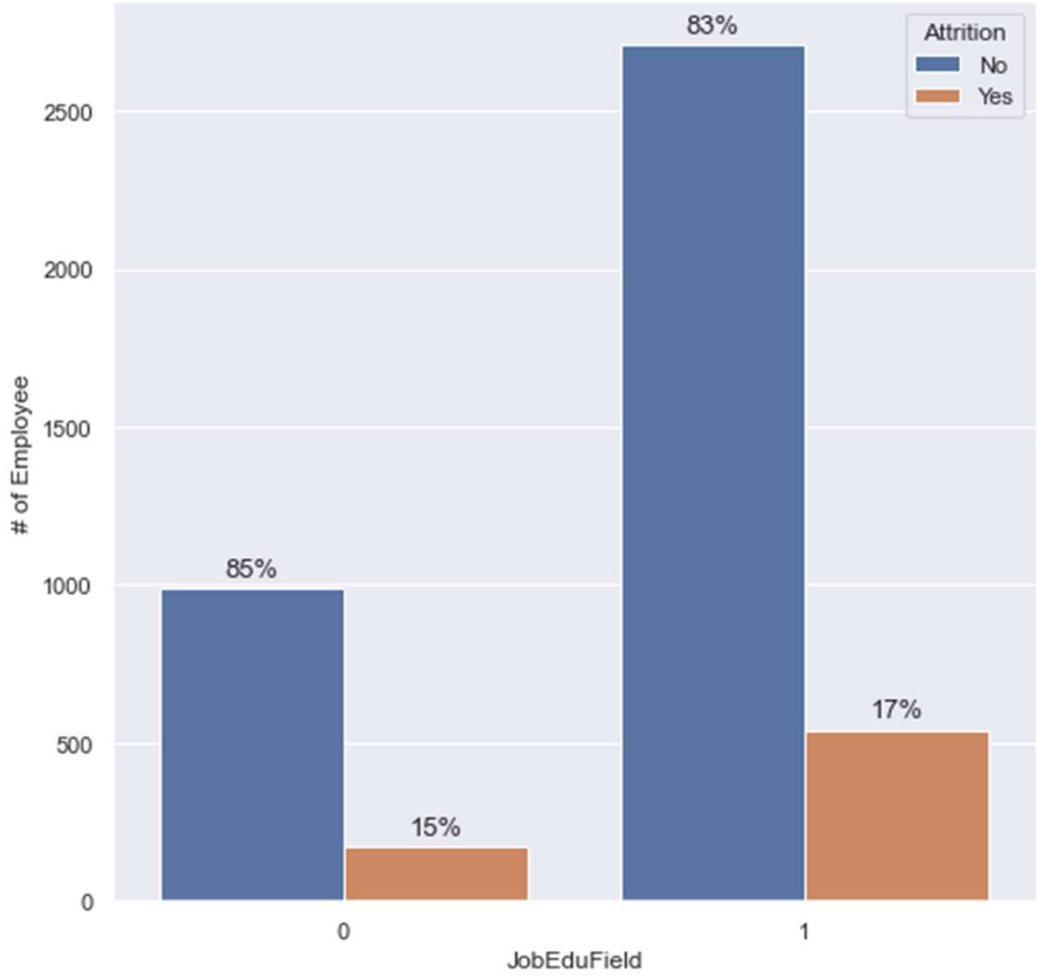
In [68]:
```
df_whole['JobEduField']=0
idxHR = df_whole[(df_whole['EducationField']=='Human Resources')
                 &(df_whole['Department']=='Human Resources')].index
idxRD = df_whole[((df_whole['EducationField']=='Life Sciences')|
                  (df_whole['EducationField']=='Medical')|
                  (df_whole['EducationField']=='Technical Degree'))
                 &(df_whole['Department']=='Research & Development')].index
idxS = df_whole[(df_whole['EducationField']=='Marketing')
                 &(df_whole['Department']=='Sales')].index
```

In [69]:
```
df_whole.loc[idxHR,'JobEduField'] = 1
df_whole.loc[idxRD,'JobEduField'] = 1
df_whole.loc[idxS,'JobEduField'] = 1
df_whole['JobEduField'].value_counts()
```

```
1    3249
0    1161
Name: JobEduField, dtype: int64
```

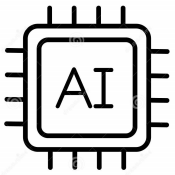
JobEduField adalah variable yang dibuat yang bernilai 1 jika **bidang kerja** pegawai **sama dengan bidang studi** pegawai semasa kuliah/sekolah.

EDA : FINDINGS + GRAPH (13 cont.)



Mayoritas pegawai bekerja di bidang yang **sama** dengan bidang studi pada saat kuliah/sekolah

Tingkat attrition pegawai yang memiliki bidang kerja sama dengan bidang studi cenderung **lebih tinggi** daripada pegawai yang bidang kerjanya berbeda dari *background* pendidikan dirinya



MODELING

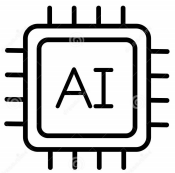
```
df_prep['Attrition'] = df_prep['Attrition'].replace({ 'Yes' : 1, 'No': 0 })
df_prep['Gender'] = df_prep['Gender'].replace({ 'Male' : 1, 'Female': 0 })
df_prep.pop('EmployeeID')
from sklearn.model_selection import train_test_split
X = df_prep.drop(['Attrition'], axis = 1)
y = df_prep['Attrition']
X_trainp, X_testp, y_trainp, y_testp = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

- ***Handling Variables***
  - Encode variabel *Attrition* dan *Gender*
  - Menghapus variabel yang tidak diperlukan dalam modeling :
    - *AvgActualHours*
    - *AgeGroup*
    - *EmployeeID*
  - *Encoding* dengan membuat variabel *dummy* untuk *categorical variables* yang memiliki lebih dari 2 *levels*
  - Semua proses telah dilakukan sebelum imputasi data pada saat preprocessing

```
X_trainp.pop('AgeGroup')
X_trainp.pop('AvgActualHours')
X_testp.pop('AgeGroup')
X_testp.pop('AvgActualHours')
```

```
X_trainp = pd.get_dummies(X_trainp)
X_testp = pd.get_dummies(X_testp)
```

```
20  BusinessTravel_Non-Travel          3528 non-null   uint8
21  BusinessTravel_Travel_Frequently   3528 non-null   uint8
22  BusinessTravel_Travel_Rarely       3528 non-null   uint8
23  Department_Human Resources         3528 non-null   uint8
24  Department_Research & Development  3528 non-null   uint8
25  Department_Sales                   3528 non-null   uint8
26  EducationField_Human Resources     3528 non-null   uint8
27  EducationField_Life Sciences       3528 non-null   uint8
28  EducationField_Marketing           3528 non-null   uint8
29  EducationField_Medical             3528 non-null   uint8
30  EducationField_Other               3528 non-null   uint8
31  EducationField_Technical Degree    3528 non-null   uint8
32  JobLevel_1                         3528 non-null   uint8
33  JobLevel_2                         3528 non-null   uint8
34  JobLevel_3                         3528 non-null   uint8
35  JobLevel_4                         3528 non-null   uint8
36  JobLevel_5                         3528 non-null   uint8
```



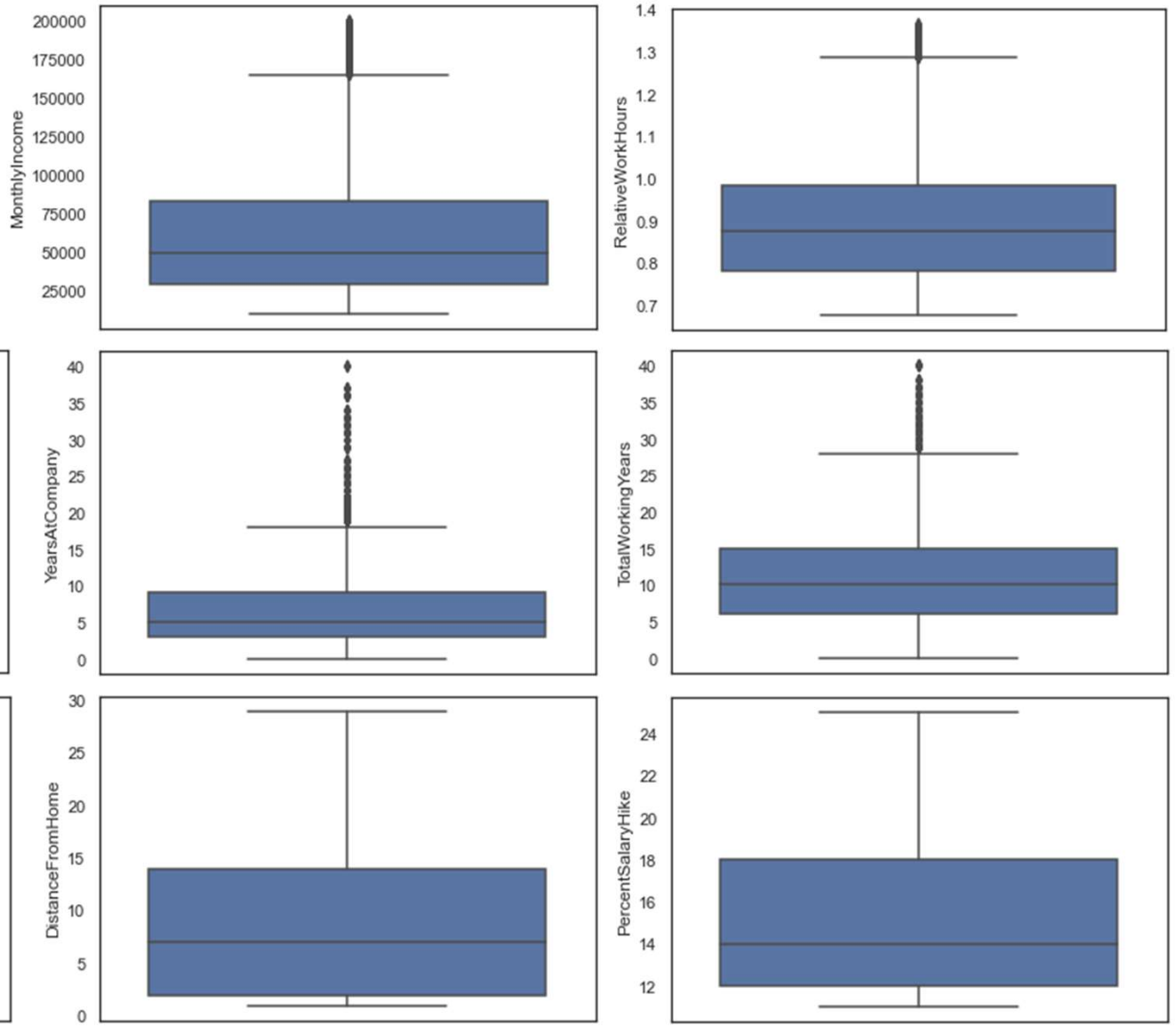
MODELING

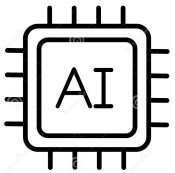
- ***Train Test Split***
  - Memisahkan data menjadi *training* dan *test* data
  - Menggunakan *train_test_split()* dari *sklearn.model_selection*
  - 20% dataset untuk *test* data
  - Telah dilakukan sejak data preprocessing

```
from sklearn.model_selection import train_test_split
X = df_prep.drop(['Attrition'], axis = 1)
y = df_prep['Attrition']
X_trainp, X_testp, y_trainp, y_testp = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

MODELING

- ***Handling Outliers***
  - Cek variabel yang memiliki outliers dengan boxplot





MODELING

- ***Handling Outliers***
  - Ada 6 variabel yang memiliki cukup banyak *outliers*
    - MonthlyIncome
    - RelativeWorkHours
    - YearsAtCompany
    - YearsSinceLastPromotion
    - YearsWithCurrManager
    - TotalWorkingYears
  - ***Scaling*** untuk seluruh variabel akan coba dilakukan menggunakan *RobustScaler()* dan *MinMaxScaler()*

```
from sklearn.preprocessing import MinMaxScaler, RobustScaler
robs = RobustScaler()
mms = MinMaxScaler()
cols = X_train.columns
X_train[cols] = robs.fit_transform(X_train[cols])
X_test[cols] = robs.fit_transform(X_test[cols])
X_train
```

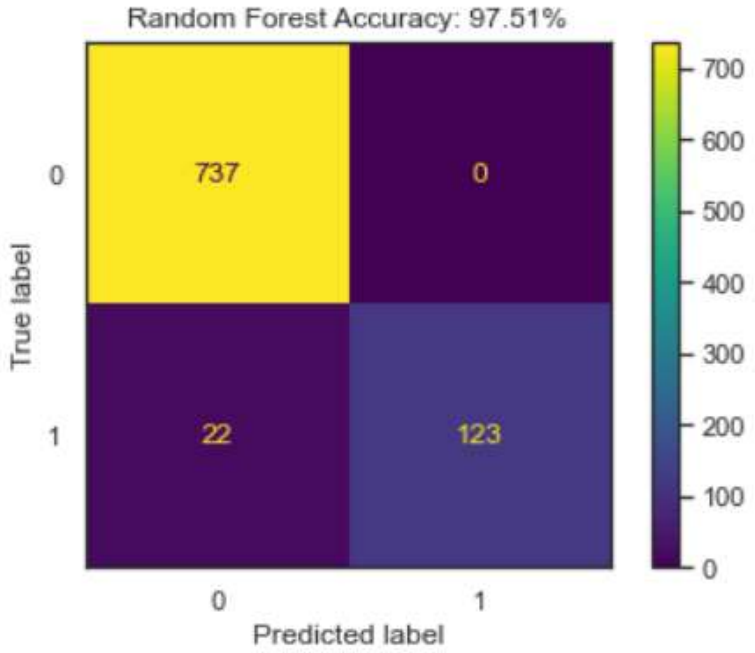
	Age	DistanceFromHome	Gender	MonthlyIncome	NumCompaniesWorked
0	-0.538462	-0.083333	0.0	-0.489972	-0.333333
1	-0.153846	0.000000	-1.0	0.683165	-0.666667
2	1.307692	1.000000	0.0	-0.432015	-0.666667
3	0.384615	0.250000	-1.0	0.558970	0.666667
4	-1.076923	0.083333	0.0	0.000000	-0.333333

```
X_train[cols] = mms.fit_transform(X_train[cols])
X_test[cols] = mms.fit_transform(X_test[cols])
X_train
```

	Age	DistanceFromHome	Gender	MonthlyIncome	NumCompaniesWorked
0	0.261905	0.178571	1.0	0.065034	0.111111
1	0.380952	0.214286	0.0	0.400790	0.000000
2	0.833333	0.642857	1.0	0.081622	0.000000
3	0.547619	0.321429	0.0	0.365245	0.444444
4	0.095238	0.250000	1.0	0.205266	0.111111

MODELING WITH ROBUSTSCALER()

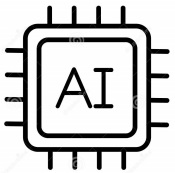
- ***Random Forest Classifier***
  - *random_state* = 0
  - Akurasi = 97,51%
  - AUC ≈ 0,92
  - Recall = 0.85
  - F1 Score = 0.92




```
from sklearn import model_selection as ms
print('Five-fold Cross Validation Mean Accuracy =',
      ms.cross_val_score(rfor, X_train.append(X_test), y_train.append(y_test),
                         cv=ms.StratifiedKFold(n_splits=5), n_jobs=-1).mean())
print('Ten-fold Cross Validation Mean Accuracy =',
      ms.cross_val_score(rfor, X_train.append(X_test), y_train.append(y_test),
                         cv=ms.StratifiedKFold(n_splits=10), n_jobs=-1).mean())
```

```
Five-fold Cross Validation Mean Accuracy = 0.9859410430839002
Ten-fold Cross Validation Mean Accuracy = 0.9936507936507937
```

```
Accuracy: 0.9751
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       737
           1       1.00      0.85      0.92       145

    accuracy                           0.98       882
   macro avg       0.99      0.92      0.95       882
weighted avg       0.98      0.98      0.97       882

AUC : 0.9241379310344828
```



MODELING WITH ROBUSTSCALER()

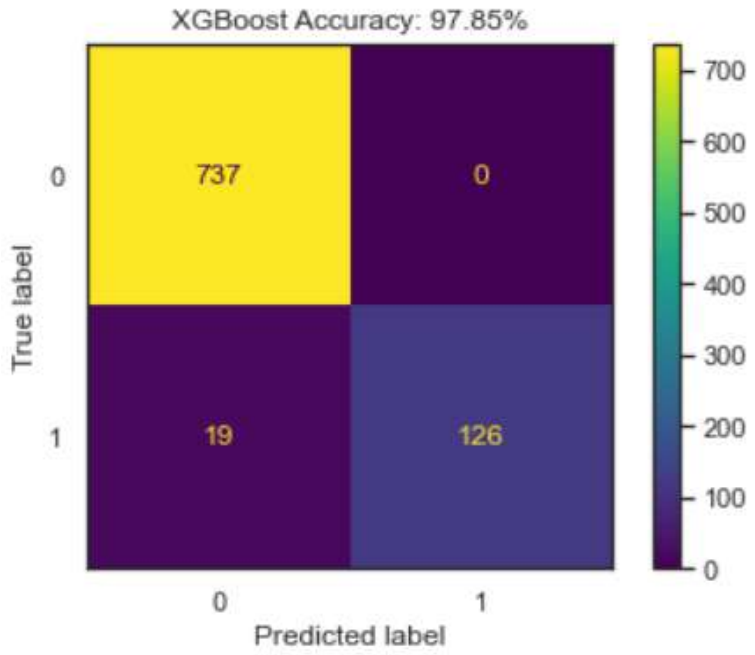
- ***Random Forest Classifier***
  - 5 variabel paling berpengaruh dalam proses modeling menggunakan Random Forest
    - *RelativeWorkHours*
    - *Age*
    - *TotalWorkingYears*
    - *MonthlyIncome*
    - *YearsAtCompany*

```
importances = rfor.feature_importances_
# printing all feature importance
indices = np.argsort(importances)[::-1]
feat_labels = X_train.columns[:]

for i in range(X_train.shape[1]):
    print("%2d) %-*s %f" % (i + 1, 30,
                            feat_labels[indices[i]],
                            importances[indices[i]]))
```

```
 1) RelativeWorkHours              0.091101
 2) Age                            0.073040
 3) TotalWorkingYears              0.071903
 4) MonthlyIncome                  0.062049
 5) YearsAtCompany                 0.051413
 6) DistanceFromHome               0.042842
 7) PercentSalaryHike              0.041943
 8) YearsWithCurrManager           0.041719
 9) NumCompaniesWorked             0.036753
10) EnvironmentSatisfaction        0.032652
```

MODELING WITH ROBUSTSCALER()

- ***Extreme Gradient Boosting (XGBoost)***
  - *Default hyperparameters*
  - Akurasi = 97,85%
  - AUC ≈ 0.93
  - Recall = 0.87
  - F1 score = 0.93

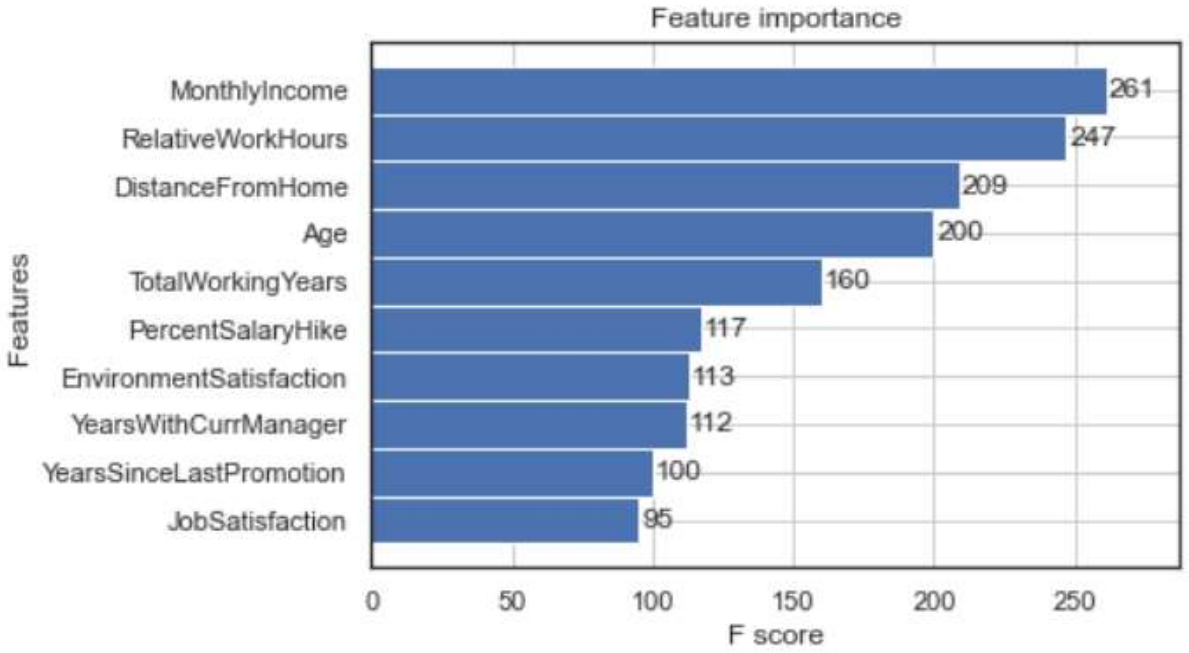



```
Five-fold Cross Validation Mean Accuracy = 0.9884353741496599
Ten-fold Cross Validation Mean Accuracy = 0.9947845804988662
```

```
Accuracy: 0.9785
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       737
           1       1.00      0.87      0.93       145

    accuracy                           0.98       882
   macro avg       0.99      0.93      0.96       882
weighted avg       0.98      0.98      0.98       882

AUC : 0.9344827586206896
```

MODELING WITH ROBUSTSCALER()

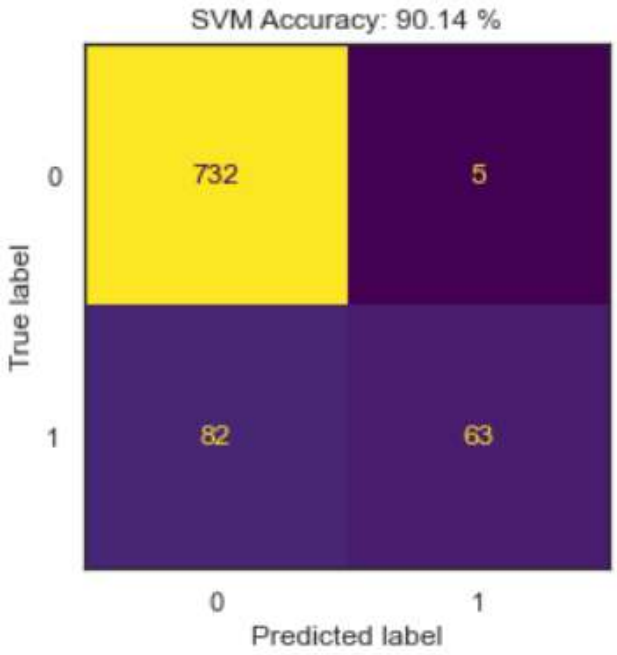
- ***XGBoost***
  - 5 variabel paling berpengaruh dalam proses modeling menggunakan XGBoost
    - *MonthlyIncome*
    - *RelativeWorkHours*
    - *DistanceFromHome*
    - *Age*
    - *TotalWorkingYears*



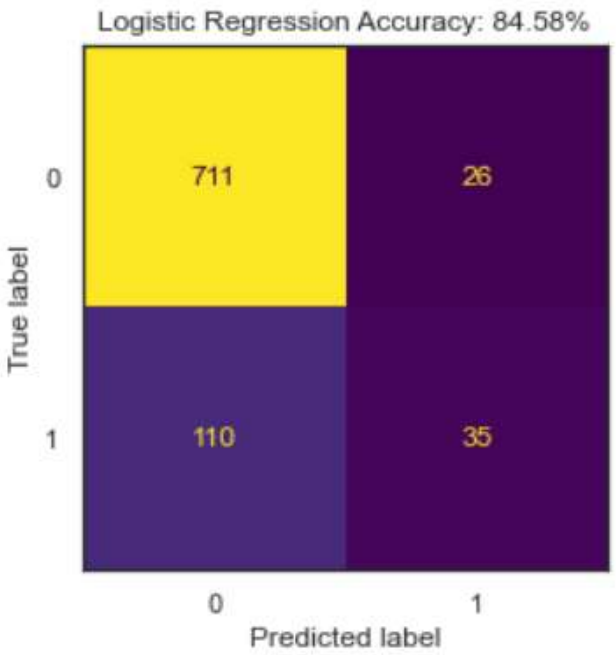
MODELING WITH ROBUSTSCALER()

Keterangan
1 *Attrition*
0 No *Attrition*

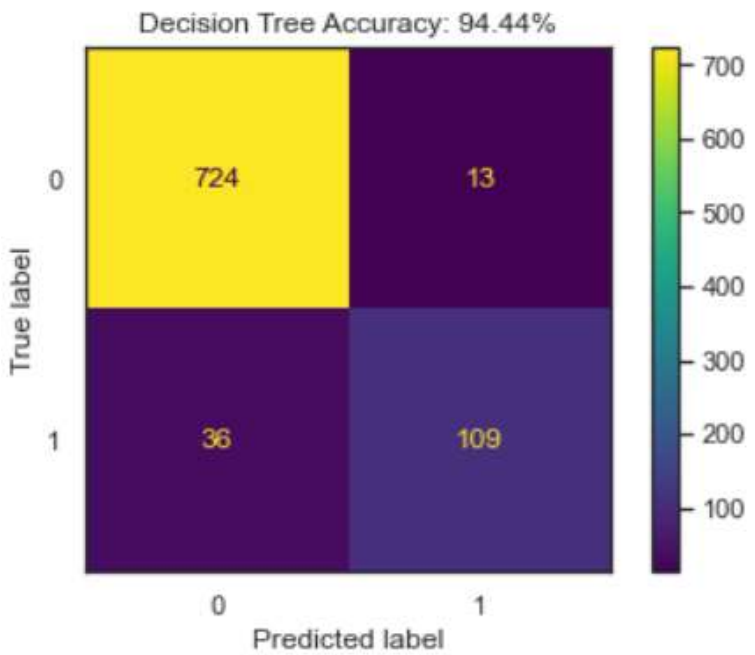
- ***Other Models***



Support Vector Machine Classifier
(AUC = 0.71; Recall = 0.43; F1 = 0.59)



Logistic Regression Classifier
(AUC = 0.6; Recall = 0.24; F1 = 0.34)

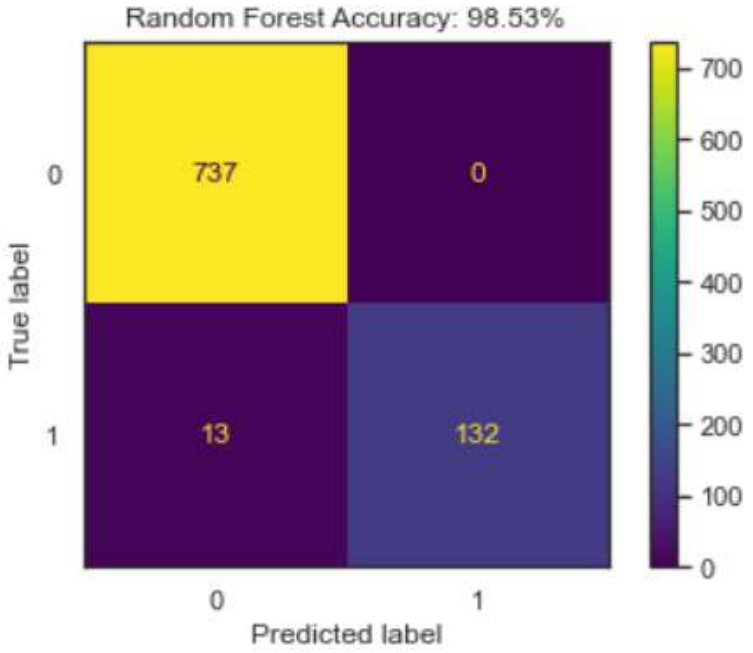


Decision Tree Classifier
(AUC = 0.87; Recall = 0.75; F1 = 0.82)

MODELING WITH MINMAXSCALER()

- ***Random Forest Classifier***
  - *random_state* = 0
  - Akurasi = 98,53%
  - AUC ≈ 0,96
  - Recall = 0.91
  - F1 Score = 0.95
  - Hasil identifikasi feature importance sama dengan saat menggunakan RobustScaler()

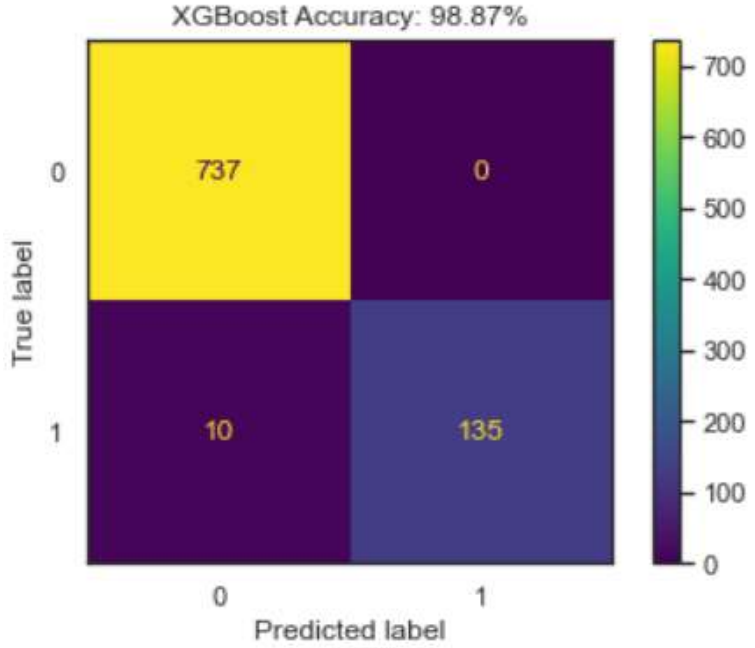
```
Five-fold Cross Validation Mean Accuracy = 0.9882086167800453
Ten-fold Cross Validation Mean Accuracy = 0.9956916099773243
```




```
Accuracy: 0.9853
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       737
           1       1.00      0.91      0.95       145

    accuracy                           0.99       882
   macro avg       0.99      0.96      0.97       882
weighted avg       0.99      0.99      0.98       882

AUC : 0.9551724137931035
```

MODELING WITH MINMAXSCALER()

- ***Extreme Gradient Boosting (XGBoost)***
  - *Default hyperparameters*
  - Akurasi = 98,87%
  - AUC ≈ 0.97
  - Recall = 0.93
  - F1 score = 0.96
  - Hasil identifikasi feature importance sama dengan saat menggunakan RobustScaler()



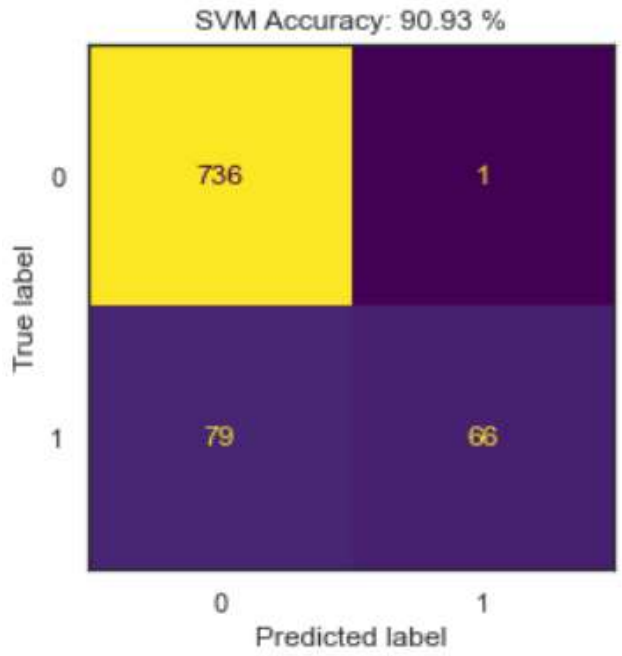

```
Accuracy: 0.9887
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       737
           1       1.00      0.93      0.96       145

    accuracy                           0.99       882
   macro avg       0.99      0.97      0.98       882
weighted avg       0.99      0.99      0.99       882

AUC : 0.9655172413793103
```

```
Five-fold Cross Validation Mean Accuracy = 0.9931972789115646
Ten-fold Cross Validation Mean Accuracy = 0.9959183673469388
```

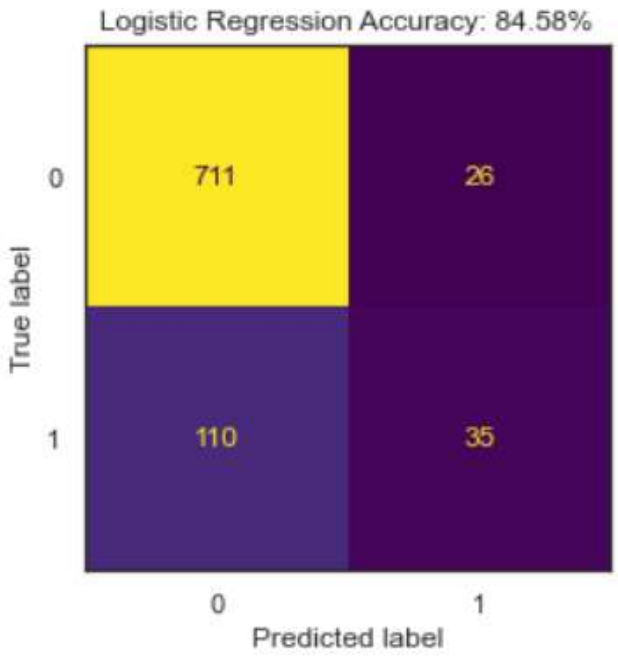
MODELING WITH MINMAXSCALER()

Keterangan
1 *Attrition*
0 No *Attrition*

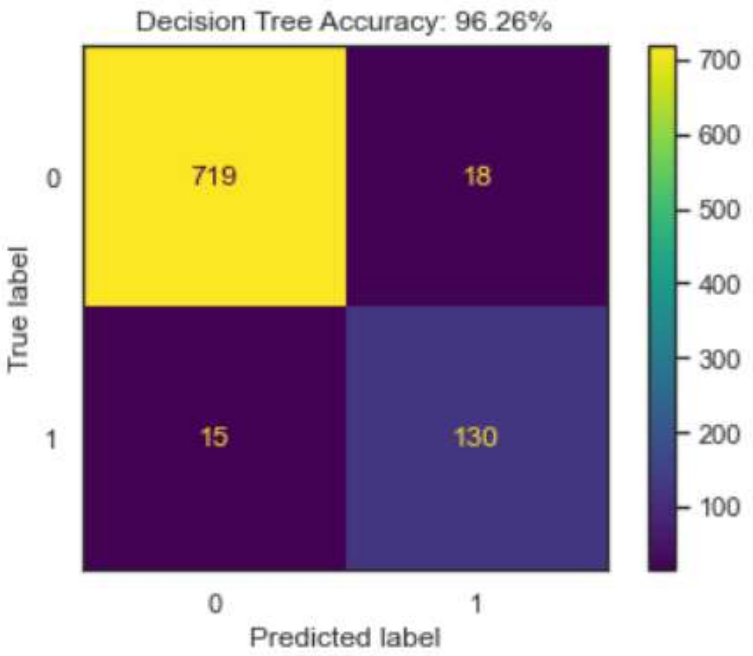
- ***Other Models***



Support Vector Machine Classifier
(AUC = 0.73; Recall = 0.46; F1 = 0.62)



Logistic Regression Classifier
(AUC = 0.6; Recall = 0.24; F1 = 0.34)



Decision Tree Classifier
(AUC = 0.94; Recall = 0.9; F1 = 0.89)

MODELING

- ***Handling Target Imbalance***
  - Implementasi Synthetic Minority Oversampling Technique (SMOTE)

```
y_train.value_counts()
```

```
Attrition
0           2962
1            566
dtype: int64
```

```
y_test.value_counts()
```

```
Attrition
0           737
1           145
dtype: int64
```

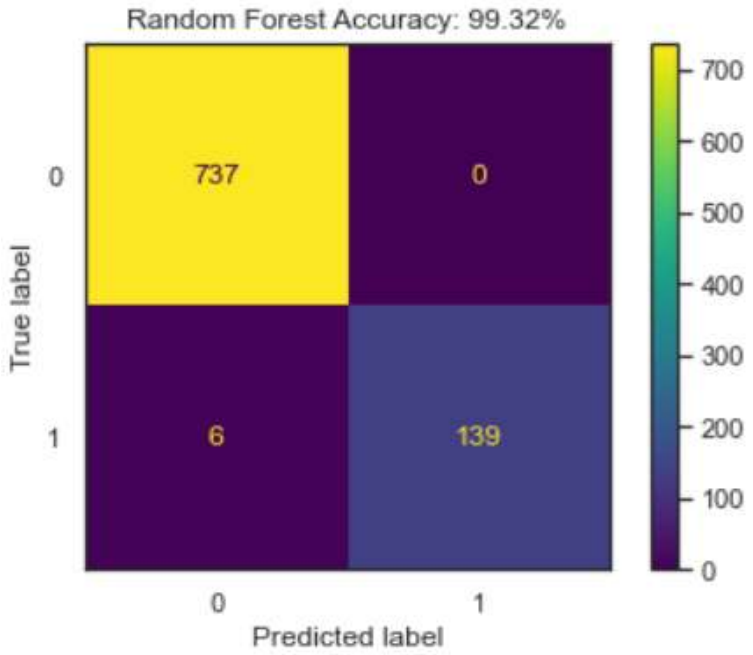
```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=0)
X_train, y_train = sm.fit_sample(X_train, y_train)
```

```
y_train.value_counts()
```

```
Attrition
1           2962
0           2962
dtype: int64
```

MODELING WITH MINMAXSCALER() & SMOTE

- ***Random Forest Classifier***
  - *random_state* = 0
  - Akurasi = 99,32%
  - AUC ≈ 0,98
  - Recall = 0.96
  - F1 Score = 0.98

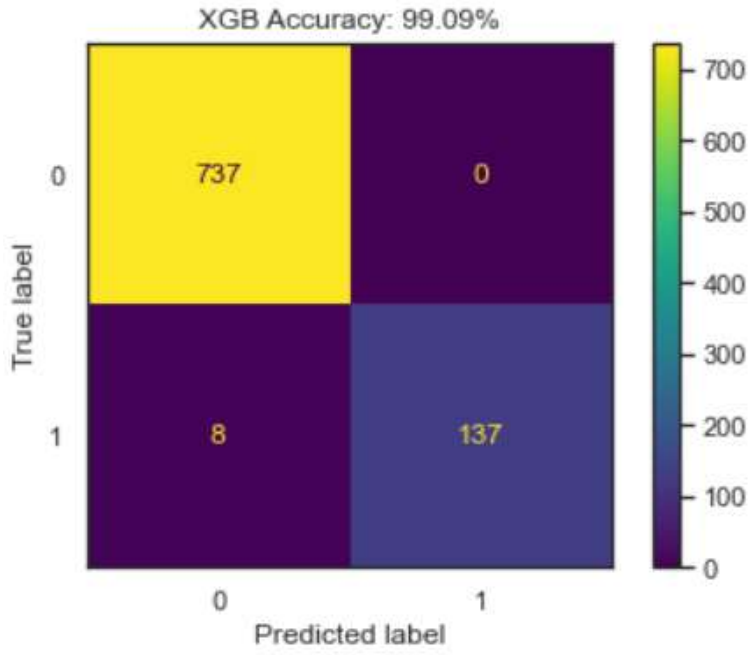



```
Accuracy: 0.9932
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       737
           1       1.00      0.96      0.98       145

    accuracy                           0.99       882
   macro avg       1.00      0.98      0.99       882
weighted avg       0.99      0.99      0.99       882

AUC : 0.9793103448275862
```

```
Five-fold Cross Validation Mean Accuracy = 0.9970612003569113
Ten-fold Cross Validation Mean Accuracy = 0.9980884512395267
```

MODELING WITH MINMAXSCALER() & SMOTE

- ***Extreme Gradient Boosting (XGBoost)***
  - *Default hyperparameters*
  - Akurasi = 99,09%
  - AUC ≈ 0.97
  - Recall = 0.94
  - F1 score = 0.97



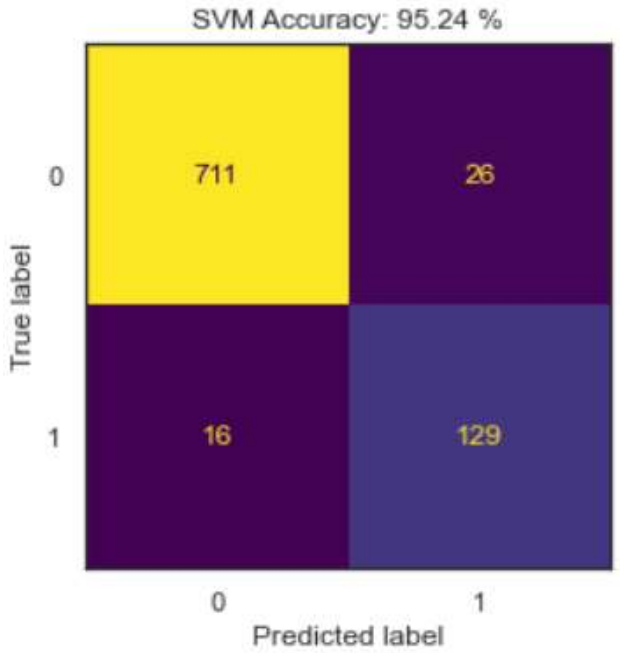

```
Accuracy: 0.9909
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       737
           1       1.00      0.94      0.97       145

    accuracy                           0.99       882
   macro avg       0.99      0.97      0.98       882
weighted avg       0.99      0.99      0.99       882

AUC : 0.9724137931034482
```

```
Five-fold Cross Validation Mean Accuracy = 0.9966209954026632
Ten-fold Cross Validation Mean Accuracy = 0.9982357260084651
```

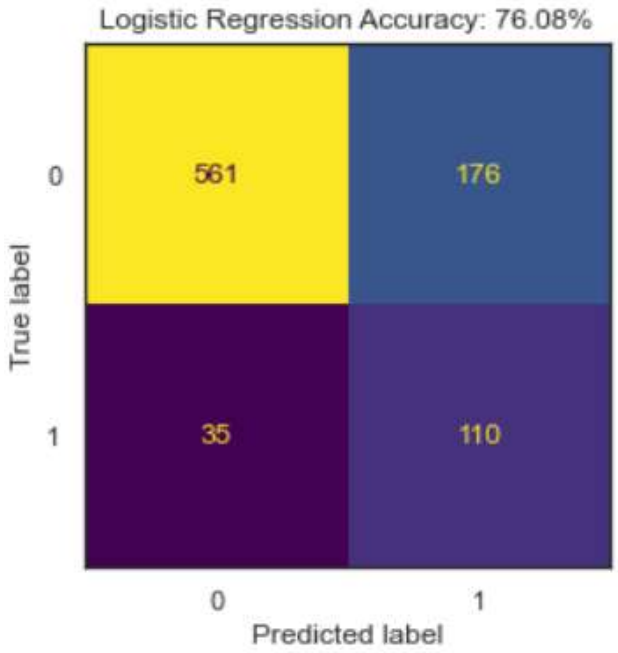
MODELING WITH MINMAXSCALER() & SMOTE

Keterangan
1 *Attrition*
0 No *Attrition*

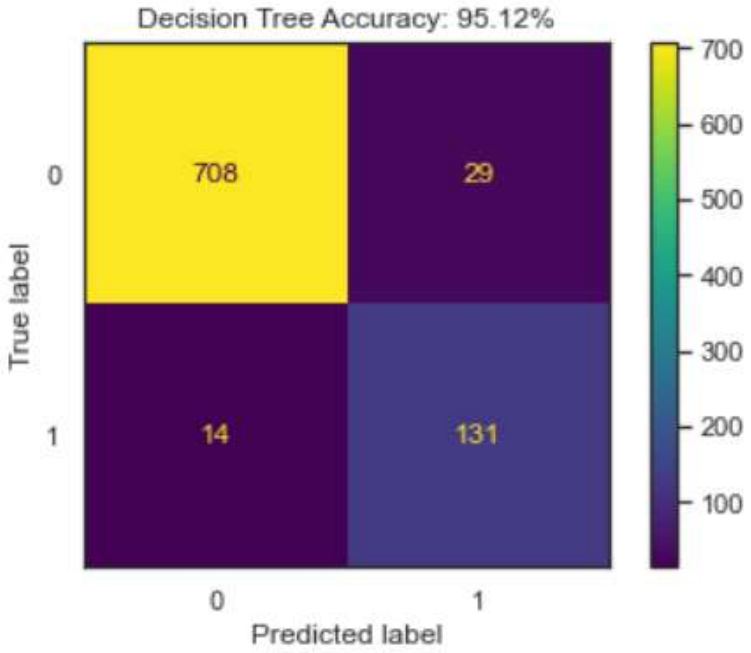
- ***Other Models***



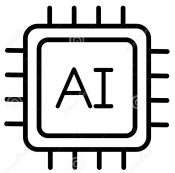
Support Vector Machine Classifier
(AUC = 0.93; Recall = 0.89; F1 = 0.86)



Logistic Regression Classifier
(AUC = 0.76; Recall = 0.76; F1 = 0.51)



Decision Tree Classifier
(AUC = 0.93; Recall = 0.9; F1 = 0.86)



MODELING

- ***Hyperparameter Tuning***
  - Parameter *Random Forest* yang akan di*tuning* :

```
# create a dictionary of parameters values we want to try
optimization_dict = {
                    'n_estimators': [100, 500, 800],
                    'max_features': ['auto','sqrt','log2'],
                    'max_depth': [None, 10, 15, 20],
                    'min_samples_split': [2,5,10],
                    'min_samples_leaf': [1,2,5],
                    'random_state' : [0]
                    }
```

```
# set GridSearchCV parameters
model = ms.GridSearchCV(rfor, optimization_dict,
                        scoring='accuracy', verbose = 1, n_jobs = -1,
                        cv = ms.StratifiedKFold(n_splits=5, shuffle=True))

# use training data
model.fit(X_train, y_train)
print(model.best_score_)
print(model.best_params_)
```

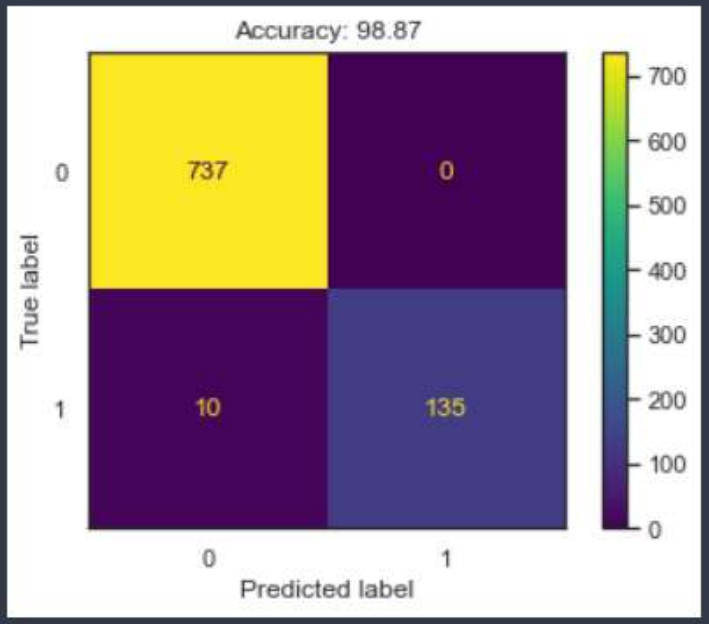
```
Fitting 5 folds for each of 324 candidates, totalling 1620 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   54.6s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  7.6min
[Parallel(n_jobs=-1)]: Done 792 tasks      | elapsed: 13.9min
[Parallel(n_jobs=-1)]: Done 1242 tasks      | elapsed: 20.9min
[Parallel(n_jobs=-1)]: Done 1620 out of 1620 | elapsed: 27.4min finished


0.9810069716512968
{'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 800, 'random_state': 0}
```

MODELING

- ***Hyperparameter Tuning***
  - Penggunaan nilai parameter *Random Forest* hasil dari *tuning* menggunakan *GridSearchCV()*
  - Dalam kasus ini, *tuning* sedikit menambah akurasi model klasifikasi *Random Forest* terhadap dataset
  - Akurasi lebih rendah dari hasil model Random Forest setelah SMOTE

```
rfort = RandomForestClassifier(max_depth = None, max_features='log2',min_samples_leaf=1,
                               min_samples_split=2,n_estimators=500,random_state=0)
rfort.fit(X_train, y_train)
y_rfort_pred = rfort.predict(X_test)
```

```
Accuracy: 0.9887
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       737
           1       1.00      0.93      0.96       145

    accuracy                           0.99       882
   macro avg       0.99      0.97      0.98       882
weighted avg       0.99      0.99      0.99       882

AUC : 0.9655172413793103
```



CONCLUSION

- ***Pengurangan Karyawan*** dapat diklasifikasi atau diprediksi menggunakan model klasifikasi ***Random Forest*** maupun ***XGBoost*** dengan hasil akurasi prediksi mencapai **99%**
- Menurut kedua model, faktor-faktor terpenting yang menentukan apakah seorang karyawan akan berhenti kerja di antaranya ***gaji pokok***, ***rata-rata jam kerja per hari***, ***usia***, ***sudah berapa lama berkarier***, ***sudah berapa lama bekerja di perusahaan***, dan ***jarak antara rumah dan kantor***
- ***Supervised learning*** mampu menciptakan sebuah model klasifikasi yang bermanfaat bagi sebuah perusahaan untuk ***memprediksi potensi pengurangan karyawan*** di masa depan

REFERENCE

- Boyle, T. (2019). Methods for Dealing with Imbalanced Data. Retrieved 5 February 2021, from https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18
- Hale, J. (2019). Scale, Standardize, or Normalize with Scikit-Learn. Retrieved 3 February 2021, from https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02
- Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal Of Artificial Intelligence Research, 16, 321-357. doi: 10.1613/jair.953
- Toshniwal, R. (2020). How to select Performance Metrics for Classification Models. Retrieved 4 February 2021, from https://medium.com/analytics-vidhya/how-to-select-performance-metrics-for-classification-models-c847fe6b1ea3
- Brownlee, J. (2016). A Gentle Introduction to XGBoost for Applied Machine Learning. Retrieved 4 February 2021, from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
- XGBoost Documentation — xgboost 1.4.0-SNAPSHOT documentation. (2021). Retrieved 6 February 2021, from https://xgboost.readthedocs.io/en/latest/