Atrition Analysis using XG Boost and Support Vector Machine Algorithms

Bandung Prihanto
Graduate Student
Atma Jaya Catholic University of Indonesia Jakarta, Indonesia

Catherine Olivia Sereati, Maria A. Kartawidjaja, Marsul Siregar
Department of Electrical Engineering,
Atma Jaya Catholic University of Indonesia Jakarta, Indonesia

Abstract:- The presence of the internet network which is getting faster and the digital world which is growing rapidly in various fields have a very big influence on every aspect of human life, not limited to people who are related, especially jobs in the field of information technology but also outside the field of information technology. The massive development of the digital ecosystem and the entry of the industrial era 4.0 means that more and more data is available on the internet. A large amount of data available then raises various problems in terms of how to process and analyze data so that this data can be useful in human life both within the scope of individuals or companies and in the fields of education, health and so on. With the concept of Machine Learning technology, problems in processing and analyzing large amounts of data can be solved more quickly when compared to doing it manually by humans. The more data that is processed, the performance of Machine Learning in conducting analysis will increase. In this analysis process, the determined algorithm also affects the performance of Machine Learning. The Author will use Google's services in this research, namely Google Colaboratory. Then the Author will also compare the use of two algorithms, XG Boost, and Support Vector Machine, as well as carry out the feature selection process. The Author will use Pearson to find factors that have a high correlation value. Based on the results of prediction research on employee turnover using Machine Learning by comparing the two algorithms, XG Boost and Support Vector Machine it can be concluded that the accuracy obtained from each accuracy is that XG Boost obtained 86% and Support Vector Machine 84%.

***Keywords**:- Machine learning, pearson, xg boost, support vector machine, employee turnover.*

I. INTRODUCTION

The presence of the internet network which is getting faster and the digital world which is growing rapidly in various fields has a very big influence on every aspect of human life, not limited to people who are related, especially jobs in the information system and outside information system field. The massive development of the digital ecosystem and the entry of the industrial era 4.0 means that more and more data is available on the internet. A large amount of data available will be an advantage and can also be detrimental when viewed from a variety of different perspectives. A large amount of data available then raises various problems in terms of how to process and analyze data so that this data can be useful in human life both within the scope of individuals or companies and in the fields of education, health and so on. An example of a problem encountered within the scope of the company is the employee turnover rate in a company is 5% of the total employees, then the costs incurred by the company can be calculated with an estimate of around 1.5 times the annual income of an employee [1].

With the concept of Machine Learning technology, problems in processing and analyzing large amounts of data can be solved more quickly when compared to doing it manually by humans. Machine Learning utilizes a computer to run a processed program [2]. Computers in this program can do learning automatically based on the analysis of the data entered, the analysis process is carried out with an algorithm that produces certain mathematical patterns. Patterns that have been determined from the results of this study, can be used to determine the characteristics of new data by comparing patterns that already existed before [3]. Machine learning can be defined as computer applications and mathematical algorithms that are adopted by learning from data and generating predictions in the future [4]. There are 3 types of machine learning, Reinforcement Learning, Unsupervised and Supervised Learning [5], and Reinforcement Learning [6]. The more data that is processed, the performance of Machine Learning in carrying out the analysis will increase, in this analysis process the specified algorithm also affects the performance of Machine Learning.

Processing and analyzing data that uses a lot of Machine Learning also requires reliable and stable computing resources, so that the data analysis process is more reliable and accurate. To get these computing resources, we can rely on cloud computing technology. This technology is more commonly referred to as cloud computing technology (cloud computing) which is a paradigm for information technology infrastructure that managed by the service provider [7]. Some of cloud provider that serve the services in the global such Azure Cloud, Amazon Web Services and Google.

There are several studies related to predicting employee turnover. A summary of some of these studies is presented in Table 1. below.

Table 1: Study Literature

No.	Title	Discussion
1	HR Analytics: Employee Attrition Analysis Using Logistic Regression [8]	This study discusses the prediction of employee turnover, where the prediction results obtained an accuracy of 75% with feature selection using the Variance Inflation Factor (VIC) using the Logistic Regression algorithm. From this study, it is suggested to use another algorithm to find out the comparison of the correct prediction results between the Random Forest algorithm and other algorithms.
2	An Extensive Analytical Approach on Human Resources Using Random Forest Algorithm [9]	This study discusses employee predictions that need to be maintained based on several parameters. In this study, 14 parameters were used from a dataset sourced from Kaggle, totaling 19258 training data and 2129 test data. From the results of this study, 100% accuracy was obtained with the random forest algorithm [5]. This 100% accuracy result can be claimed as overfitting, this is due to noise or a less clear dataset. Suggestions in this study are to make predictions with other algorithmic models and perform data cleansing before learning.
3	Comparative Study of The Machine Learning Techniques for Predicting The Employee Attrition [10]	In this study, the prediction of employee reduction was carried out. Where it is done with a total of 35 attributes and using the Linear Discriminant Analysis (LDA) algorithm to get an accuracy of 86.39%.
4	Feature Selection pada Azure Machine Learning untuk Prediksi Calon Mahasiswa Berprestasi [11]	In this study, predictions were made of prospective student achievements using the SVM algorithm on the Azure Machine Learning platform. The results of this study obtained an accuracy of 82.7% with the use of 10 attributes.
5	Employee Attrition Rate Prediction Using Machine Learning Approach [12]	In this research, we develop a model to predict employee reduction, starting with some basic exploratory data analysis and continuing to feature engineering and applying a learning model in the form of a Random Forest, which has an accuracy of 85% in its predictions. Effort-reward imbalance is most likely the underlying common explanation for friction. For these situations, especially true of individuals who work longer hours than necessary and who often have generally low wages - it should be investigated whether our organization has an attractive extra time strategy. In the context of the future aspect, a certain heuristic-based approach can be followed in the coming years to predict the level of employee attrition and can solve real-world problems.

The author will use Google's services in this research, namely Google Colaboratory (Google Colab). Google Colab itself is a modified form of Jupyter Notebook provided to Google, where this platform is often used for Machine Learning and Deep Learning [13]. We can use google collabs with a lot of command and it provided with Memory dan Storage per user [14]. The author will compare the use of two algorithms, namely XG Boost and Support Vector Machine to determine the highest accuracy value, and to find out features that have a high correlation, the author will use the Pearson method.

II. RESEARCH METHOD

A. Dataset Collection

After the literature study process is carried out, the next process is to find data sources that will be used in research. These data are collected and then called a dataset. In this study, the author will use a secondary dataset from Kaggle.

B. Exploratory Data

Data Exploratory is intended to see how the spread of data from datasets that have been obtained into graphical forms that are easier to see and understand.

C. Cleansing Data

The next process after the dataset is obtained is to carry out the pre-processing process. As explained in the previous chapter, the pre-processing process consists of several stages. The first stage is data cleansing, it is necessary to check beforehand whether the dataset owned has duplicate data or not. Data contains more information that may not be needed to build a model or contains wrong information [11]. For example, if there is a dataset with 50 columns, many columns contain duplicate data from other columns. This affects the quality and efficiency of the resulting data model.

If the dataset you have does not contain duplicate/multiple data, then the next step is to check whether there are missing values or not. If a row/column with a missing value condition is found, it is necessary to find a way so that the data is no longer empty. Several ways that can be done to fill in the missing value include deleting rows or filling in the empty values.

D. Transformation Data

After there are no empty rows/columns of data, then the outliers are removed. Outliers are data that have values that are very far from the general value, or in other words have extreme values. The process is then continued by encoding features. Categorical data will be transformed into a form that can be understood by the system. Character data will be converted into numeric.

The encoding process uses One-Hot Encoding. One-hot encoding is used as a method for measuring categorical data. One-hot Encoding is the process of creating a unique feature value in a column. A nominal feature is a category feature type that cannot be sorted. After the encoding process is complete, it is followed by a data split process to break the data into 2 parts, namely test data and training data.

ISSN No:-2456-2165

E. Split Data

In the next stage, the existing data will be divided into two parts, namely training data and test data. The dataset obtained from Kaggle will be divided by a simulation ratio of training data, namely 0.7 and test data, 0.3. After the encoding process is complete, it is continued with the process of overcoming data imbalance by oversampling.

F. Pre-processing Data

Then the process is continued by performing data processing which consists of checking the imbalance of the data using oversampling which aims to reduce/eliminate outliers in the data. Imbalance data occure if there is unbalance ratio from one data to another data [15]. Impact of accuracy in machine learning probably occurs on the misclassification due to of imbalanced data. The decrease in accuracy in imbalanced is caused by the fact that there are many noises or outliers found in the test dataset that come from the minority class. Imbalanced data can be solved using oversampling methos, adding the data synthetic data to class minority [16]. After that, data scaling is done using normalization. Using the right data scaling method can optimize the performance of the Machine Learning algorithm [17]. It aims to change the original data measurement scale into an accepted form without changing the value of the data. The data presented is already in numerical form.

G. Modeling

If the data is ready, then the next step is to create a modeling algorithm. In this study, the author will compare two algorithms, namely XG Boost and Support Vector Machine.

➢ *XG Boost*

Extreme Gradient Boosting (XGBoost) is a decision tree-based algorithm [19]. The model is an ensemble tree algorithm consisting of several classification and regression trees. The XG Boost algorithm performs optimization faster than other implementations of the Gradient Boosting Method both in classification and regression problems [20]. In a tree-based algorithm, the inner nodes represent values for the test attribute and the leaf nodes with scores represent decisions.

➢ *Support Vector Machine*

Support Vector Machine (SVM) is a Supervised Learning machine learning model [21]. Support Vector Machine uses a classification algorithm to solve the two-category classification problem [19]. SVM has the basic principle of a linear classifier, namely classification cases that can be separated linearly. Also SVM can work to the linear problem. SVM takes input the input data and predict the two different input and classified based on the hyperplane.

So when making modeling, the author makes the two algorithms and then does the training process with data that has previously been processed. If the modeling is complete, the next step is to carry out the trial process. The trial process expected that the system can provide optimal predictive results.

H. Evaluation

After the trial process is complete, an evaluation is then carried out to find out whether the results obtained from the trial results are optimal or not. In addition, from the evaluation stage, it can be known whether the performance of the Machine Learning model that has been made is good or not, the accuracy value of each algorithm used is also known and what features have the most influence on the prediction of employee turnover. already made.

Performance measurement/evaluation uses the Confusion Matrix. The performance evaluation process is carried out for each algorithm model used. A Confusion Matrix is a method that can be used to measure the performance of a classification method. The Confusion Matrix contains information that compares the results of the classification performed by the system with the results of the classification that should be. Based on the number of class outputs, the classification system can be divided into 4 (four) types, namely binary, multi-class, multi-label and hierarchical classification [22].

III. RESULT

This section contains coding designs and research results from predicting employee turnover using Machine Learning.

A. Dataset Collection

This study uses secondary data obtained from Kaggle. The data is used as a dataset. This dataset will then be processed to produce predictions regarding employee turnover using Machine Learning. The amount of data obtained is 1470 rows and 35 columns of data. Some examples of sample data can be seen in the appendix. From these data, no missing values were found. There are 34 features and 1 target (attrition). The form of feature data is numerical and categorical. There are 26 numerical features and 9 categorical features. When entering a dataset into Google Colaboratory, use the command as shown in the following image.

```
[ ] df_raw = pd.read_csv('datasetku.csv')
```

Fig. 1: Command input dataset

Then to check information related to column names, data types, and null data, use the command as shown in Fig.2 and the results are shown in Fig.3 to Fig.4.

```
[ ] df_raw.info()
```

Fig. 2: Command for checking column, data type, and null data

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

Fig. 3: Result checking column, data type, and null data

```
[ ] round(df_raw['Attrition'].value_counts(normalize=True)*100,2)
    #lihat persentase

    No     83.88
    Yes    16.12
    Name: Attrition, dtype: float64

[ ] cat = ['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','Over18','OverTime']
    num = df_raw.select_dtypes(exclude='object').columns.to_list()

[ ] cat

    ['BusinessTravel',
     'Department',
     'EducationField',
     'Gender',
     'JobRole',
     'MaritalStatus',
     'Over18',
     'OverTime']

[ ] num

    ['Age',
     'DailyRate',
     'DistanceFromHome',
     'Education',
     'EmployeeCount',
     'EmployeeNumber',
     'EnvironmentSatisfaction',
     'HourlyRate',
     'JobInvolvement',
     'JobLevel',
     'JobSatisfaction',
     'MonthlyIncome',
     'MonthlyRate',
     'NumCompaniesWorked',
     'PercentSalaryHike',
     'PerformanceRating',
     'RelationshipSatisfaction',
     'StandardHours',
     'StockOptionLevel',
     'TotalWorkingYears',
     'TrainingTimesLastYear',
     'WorkLifeBalance',
     'YearsAtCompany',
     'YearsInCurrentRole',
     'YearsSinceLastPromotion',
     'YearsWithCurrManager']
```

Fig. 4: Result checking numerical and categorical features

B. Exploratory Data

Data Exploratory is intended to view data distribution plots in a graphical form that is easier to see and understand.

The following is a command to view feature distribution plot graphs and feature distribution plot graphs against attrition in the form of stock graphs and line graphs.

```
[ ] #plot features distibution
    features = num
    plt.figure(figsize=(20,20))
    for i in range(0, len(features)):
        plt.subplot(5, 6, i+1)
        sns.boxplot(y=df_raw[features[i]], color='green', orient='v')
        plt.tight_layout()
```

Fig. 5: Command features distribution

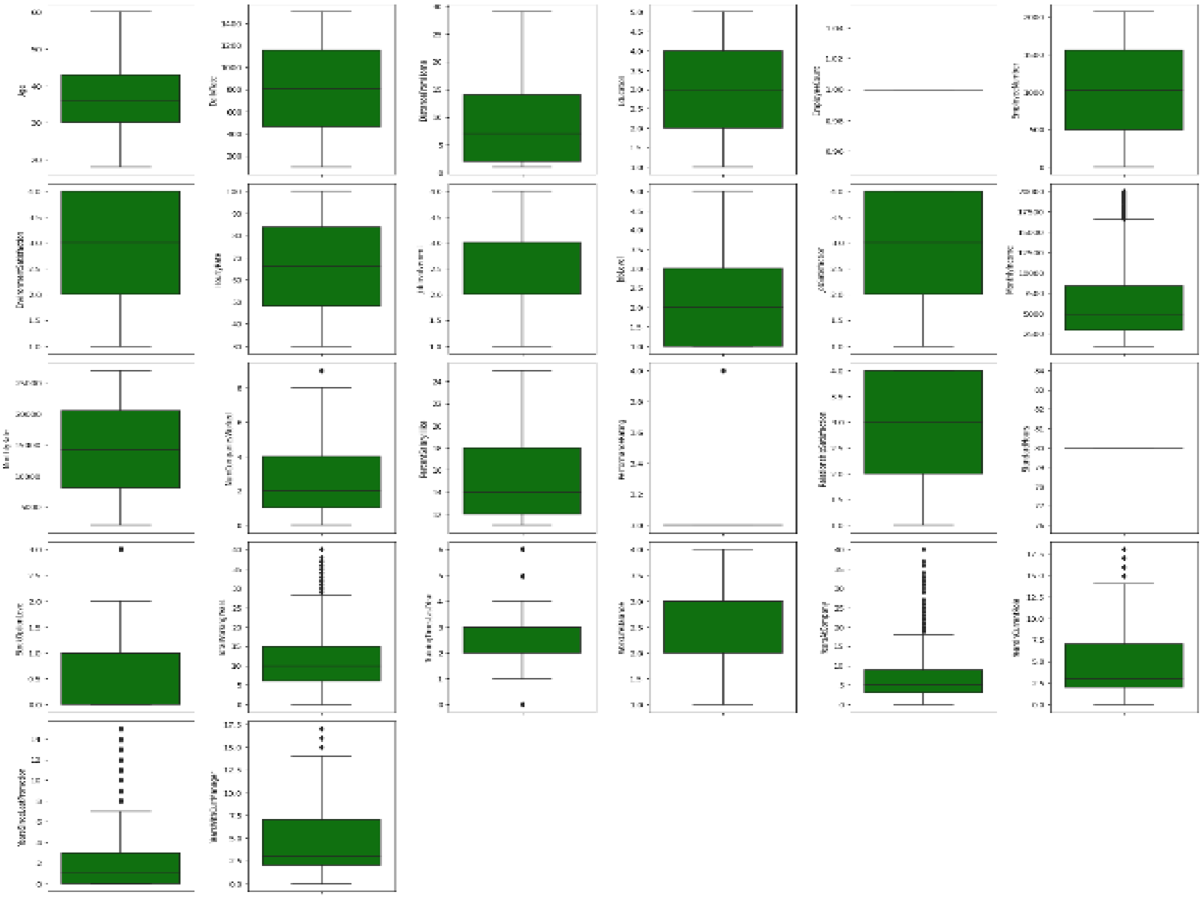

Fig. 6: Graph of feature distribution plots

```
[ ] #plot features distribution to atrition
    features = num
    plt.figure(figsize=(20,20))
    for i in range(0, len(features)):
        plt.subplot(5, 6, i+1)
        sns.boxplot(y=df_raw[features[i]], orient='v', x=df_raw['Attrition'])
        plt.tight_layout()
```

Fig. 7: Command features distribution to attrition

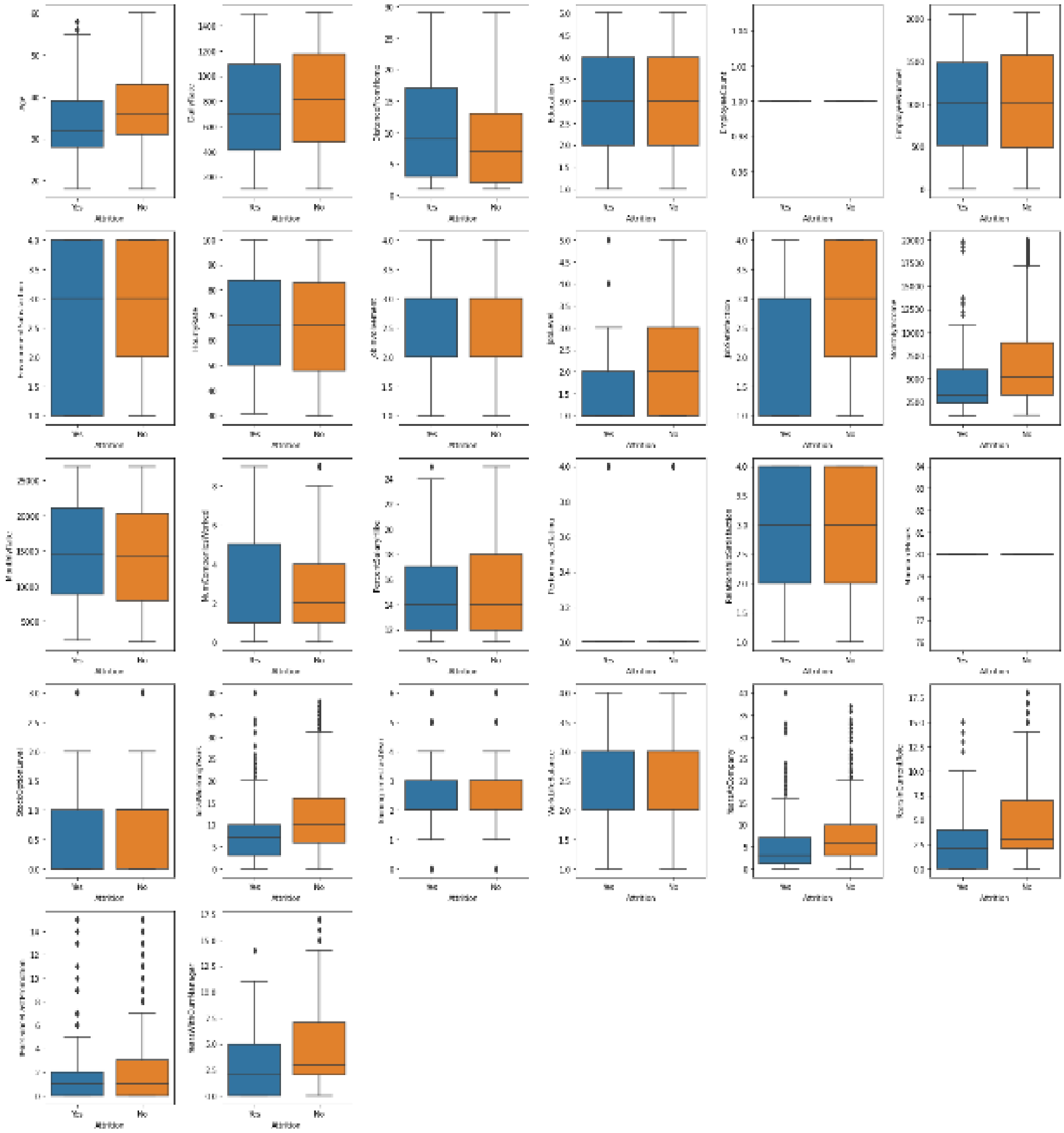

Fig. 8: Graphic plot of the distribution of features against attrition

```
[ ] #graph distibution
    features = num
    plt.figure(figsize=(20, 20))
    for i in range(0, len(num)):
        plt.subplot(7, 4, i+1)
        sns.kdeplot(x=df_raw[features[i]], color='green')
        plt.xlabel(features[i])
        plt.tight_layout()
```

Fig. 9: Command features distribution (graph)

ISSN No:-2456-2165

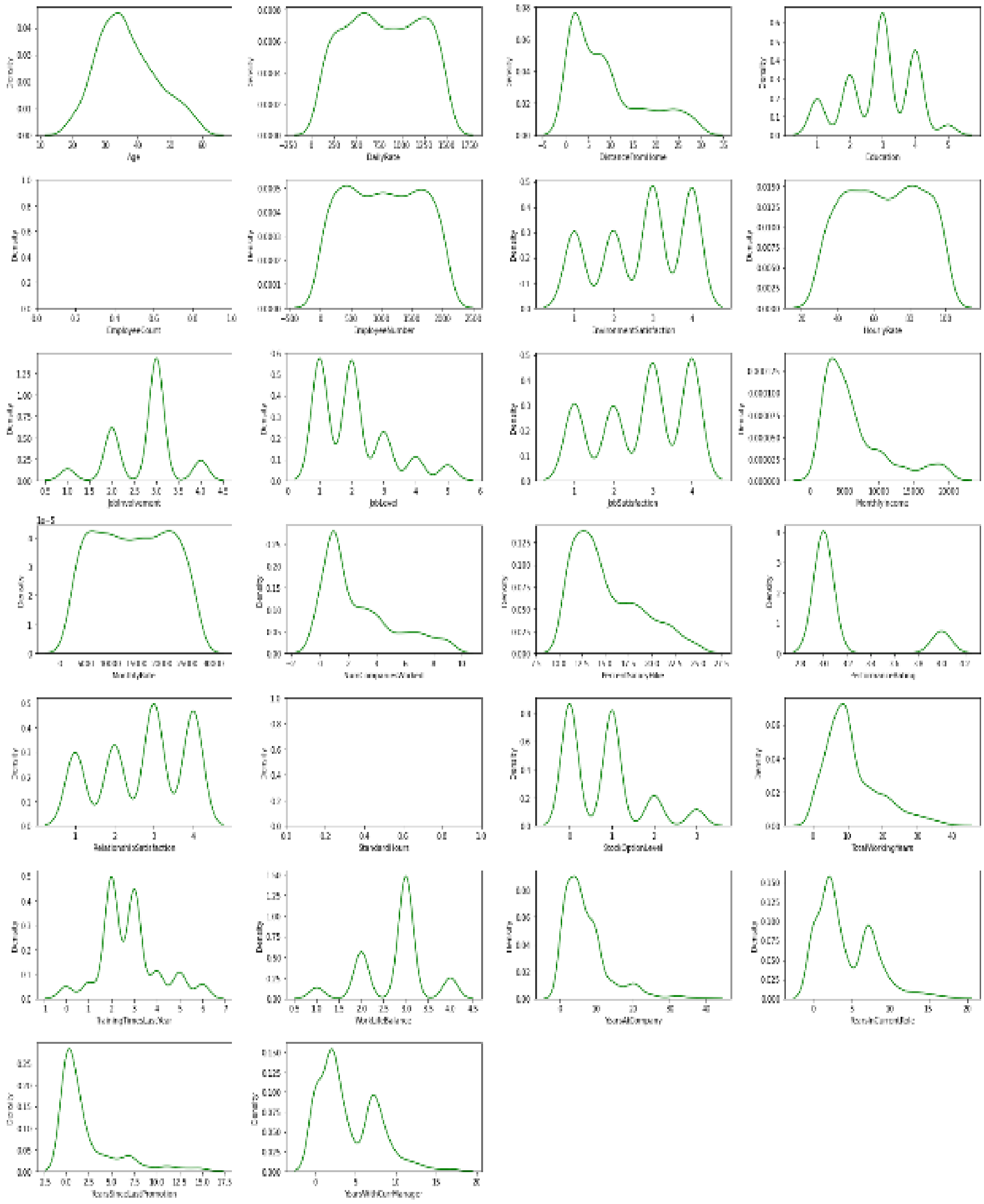

Fig. 10: Line graph of feature distribution plots

```
[ ] #graph distribution to atrition
    features = num
    plt.figure(figsize=(20, 20))
    for i in range(0, len(num)):
        plt.subplot(7, 4, i+1)
        sns.kdeplot(x=df_raw[features[i]], hue=df_raw['Attrition'])
        plt.xlabel(features[i])
        plt.tight_layout()
```

Fig. 11: Command features distribution to attrition (graph)

ISSN No:-2456-2165

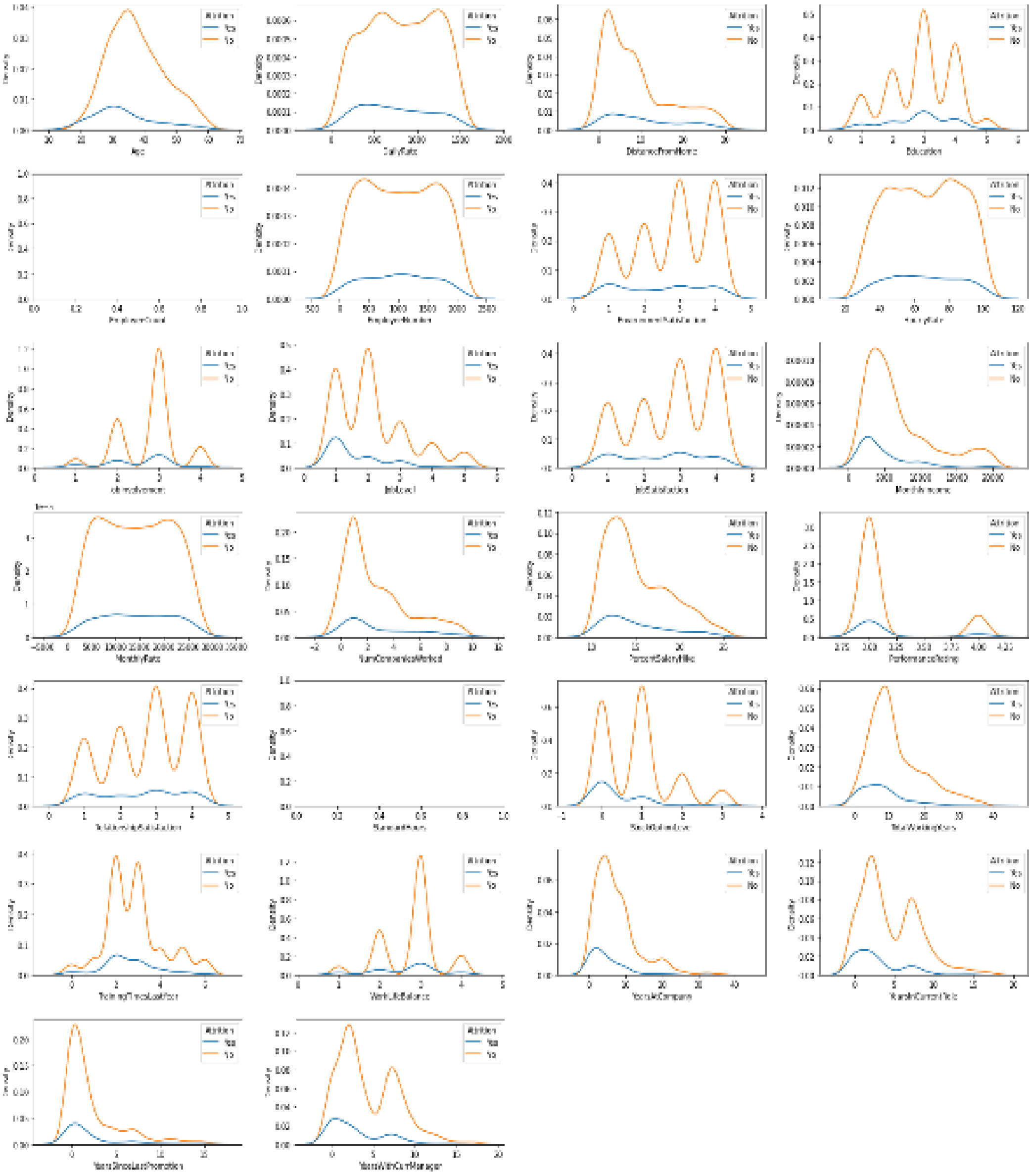

Fig. 12: Line graphic plot of the distribution of features against attrition

The graphic results show that attrition occurs in employees with the following features:

- Young age around 20-23 years
- Low daily rates
- Farther home
- Lower job satisfaction
- Lower monthly income
- Employees with little experience (4-10 years)
- New employees
- Employees with new roles (less than 4 years)
- New to current manager (less than 5 years)

When viewed from a large amount of each data, then if illustrated with a bar chart the graphical appearance is as follows.

```
[ ]  plt.figure(figsize=(20,20))
     for i in range(0, len(cat)):
         plt.subplot(3, 3, i+1)
         sns.countplot(x = df_raw[cat[i]], orient='v')
         plt.xticks(ha='left',rotation=-45)
         plt.tight_layout()
```

Fig. 13: Command count features distribution

ISSN No:-2456-2165

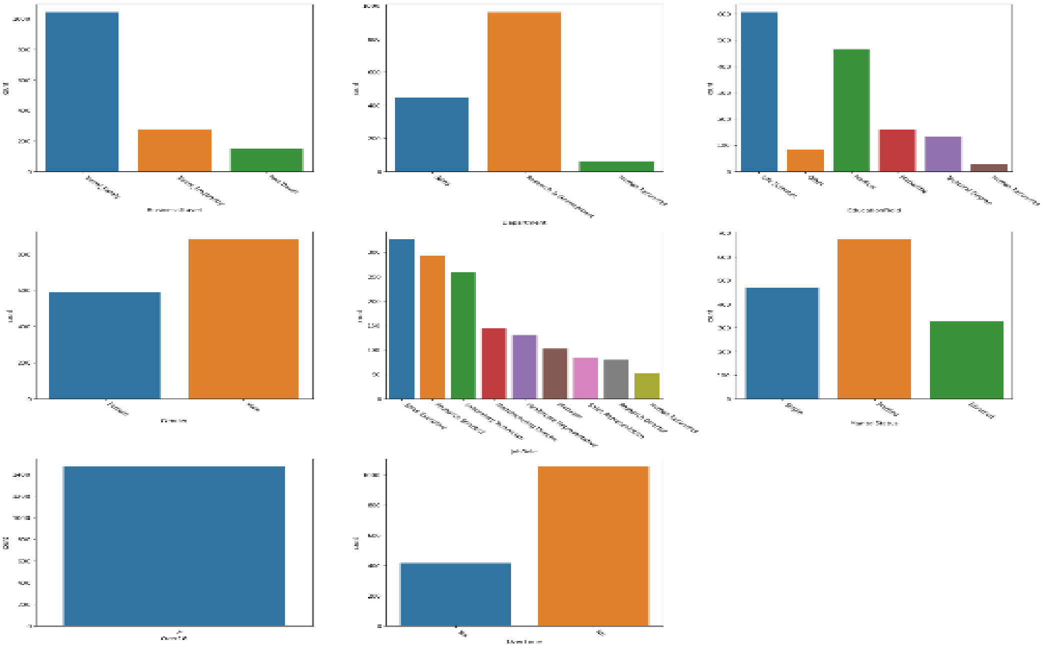

Fig. 14: Graph of the feature distribution count plot

```
[ ] plt.figure(figsize=(20,20))
    for i in range(0, len(cat)):
        plt.subplot(3, 3, i+1)
        sns.countplot(x = df_raw[cat[i]], orient='v', hue=df_raw['Attrition'])
        plt.xticks(ha='left',rotation=-45)
        plt.tight_layout()
```

Fig. 15: Command count features distribution

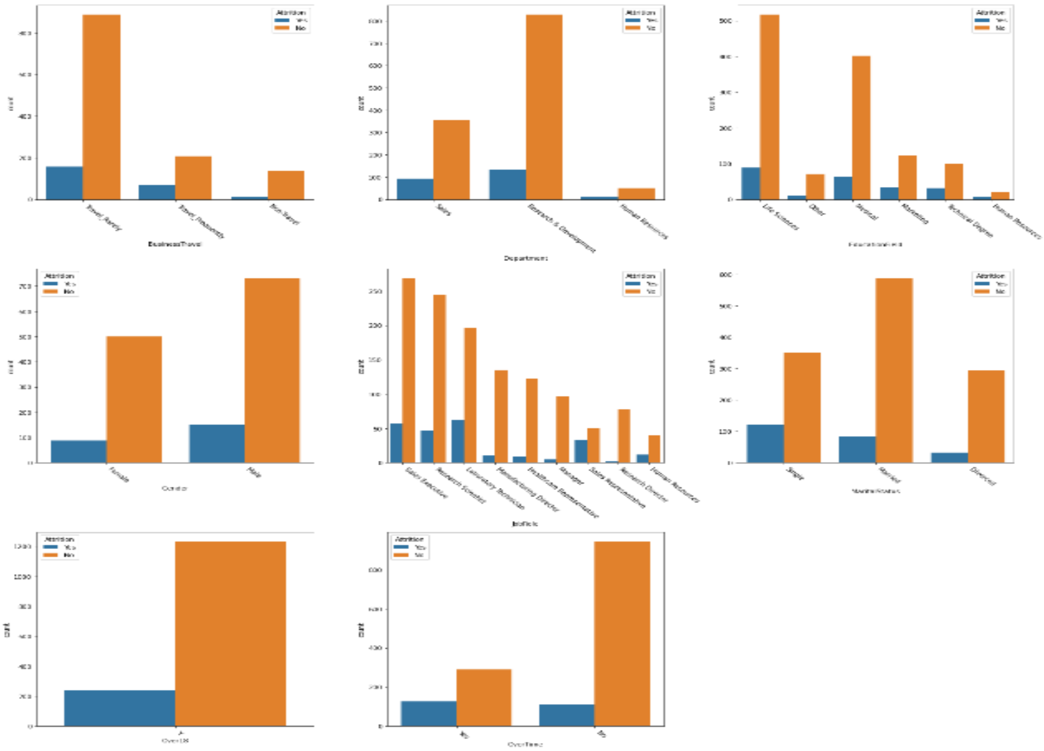

Fig. 16: Graph of count plot of feature distribution to attrition

ISSN No:-2456-2165

When this dataset is viewed in heatmap form, it will look like the image below.

```
[ ] df_att = df_raw.copy()
    df_att['is_attrition'] = df_att['Attrition'].apply(lambda x: 1 if x == 'Yes' else 0)

[ ] plt.figure(figsize=(20, 20))
    sns.heatmap(df_att.corr(method='pearson'), cmap='Blues', annot=True, fmt='.2f');
```

Fig. 17: The command displays the distribution features heatmap

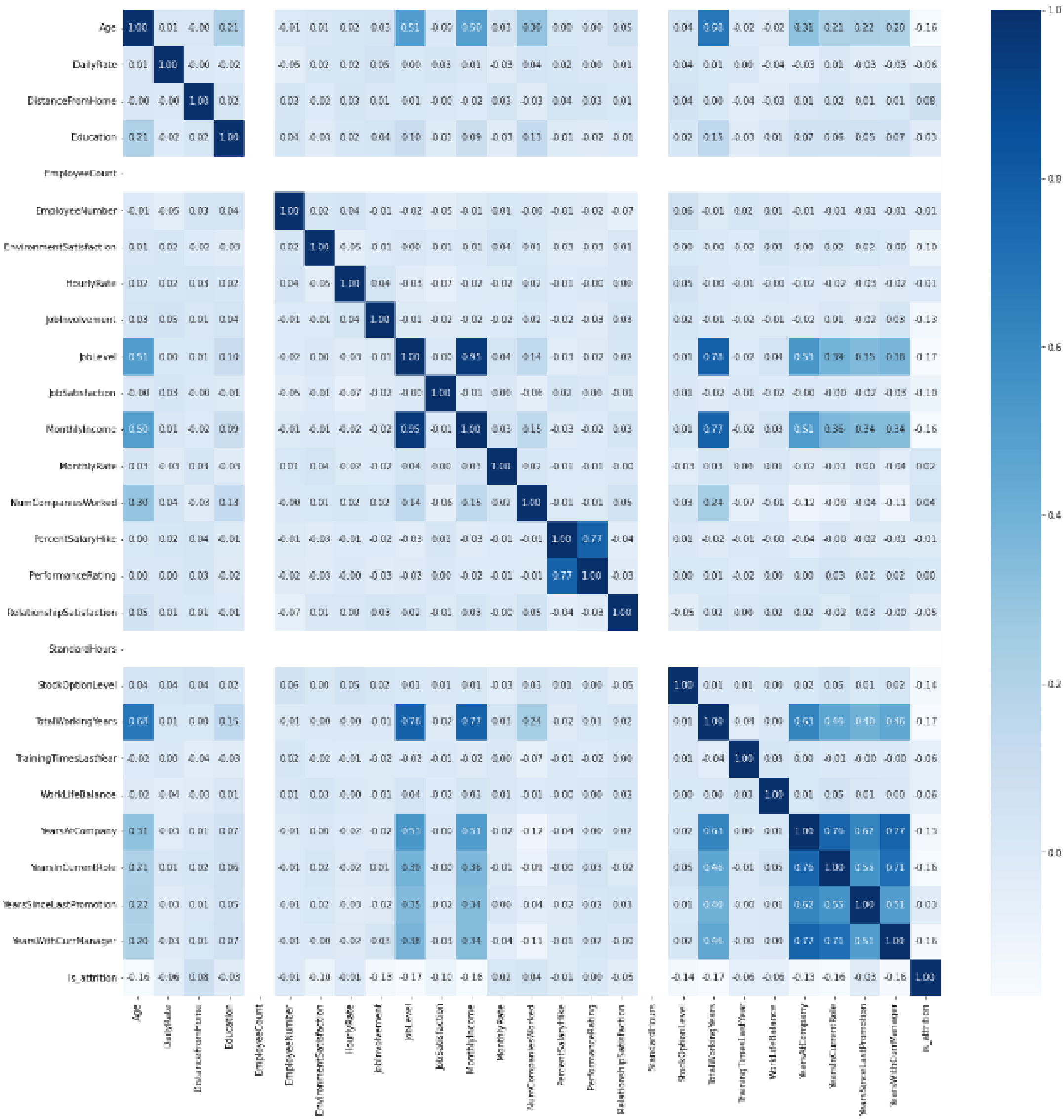

Fig. 18: Result from the distribution features heatmap

C. Cleansing Data

The next stage is data cleansing. The dataset that is owned needs to be checked first whether the dataset that is owned has duplicate data or not. If the dataset you have does not contain duplicate/multiple data, then the next step is to check whether there are missing values or not. If a row/column with a missing value condition is found, it is necessary to find a way so that the data is no longer empty. Then proceed with removing features that don't play an important role if deemed necessary.

The command for checking duplicate data and the results are shown in the following Fig.19.

```
[ ] df_raw.duplicated().any()

    False

[ ] df_raw['EmployeeNumber'].duplicated().any()
    #employeenumber penting sbg indetify, karena tidak ada nama

    False

[ ] #memastikan tidak ada duplicate
    print(f'before: {df_raw.shape[0]}')
    df_raw = df_raw.drop_duplicates()
    print(f'after: {df_raw.shape[0]}')

    before: 1470
    after: 1470
```

Fig. 19: Command checks for data duplication and results

Furthermore, to carry out the missing value-checking process, the command used is as shown in the image below. The results of checking shows that the dataset has no missing values.

```
[ ] df_raw.isna().sum()
```

Fig. 20: Command for checking missing values

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

Fig. 21: Missing value checking results

Then the data is checked again whether there is feature data that needs to be dropped or not. In this dataset, it turns out that there is feature data that needs to be dropped or removed because some of these data features have the same value in all rows. The data that needs to be dropped include EmployeeCount, StandardHours and Over18 data.

```
[ ] df_prep = df_raw.copy()

[ ] drop_list = ['EmployeeCount', 'StandardHours', 'Over18'] #These features are useless because they have the same value for all rows
    df_prep = df_prep.drop(drop_list, axis=1)
```

Fig. 22: Command drop feature data

After that, the process continues by converting the EmployeeNumber data into a string with the following command.

```
[ ] df_prep['EmployeeNumber'] = df_prep['EmployeeNumber'].astype(str)
    #EmployeeNumber is not actually numerical feature
    #diubah menjadi string supaya tidak merubah pengolahan ML karena ML mengolah number. Tidak di drop karena identifier
```

Fig. 23: Command converts EmployeeNumber to a string

D. Transformation Data

After the data cleansing process is complete, then the outliers are removed. Outliers are data that have values that are very far from the general value, or in other words have extreme values. The following is the command that is run to remove the outliers.

```
[ ] df_outliers = df_prep.copy()

    print(f'Rows before removing outliers: {len(df_outliers)}')

    filtered_entries = np.array([True] * len(df_outliers))

    for col in num:
      zscore = abs(stats.zscore(df_outliers[col]))
      filtered_entries = (zscore < 3) & filtered_entries

    df_outliers = df_outliers[filtered_entries]

    print(f'Rows after removing outliers: {len(df_outliers)}')

    #beberapa algoritma sangat sensitif dengan outlier, maka dari itu perlu dihilangkan

    Rows before removing outliers: 1470
    Rows after removing outliers: 1387
```

Fig. 24: The command and results remove outliers

The process is then continued by encoding features. Categorical data will be transformed into a form that can be understood by the system. Character data will be converted into numeric. The encoding process uses One-Hot Encoding. The following is the command that is executed during the encoding process.

```
[ ] df_outliers.describe(include='object')
    #biasa digunakan untuk fitur yang memiliki 2 nilai, misal laki-wanita, yes-no
```

	Attrition	BusinessTravel	Department	EducationField	EmployeeNumber	Gender	JobRole	MaritalStatus	OverTime
count	1387	1387	1387	1387	1387	1387	1387	1387	1387
unique	2	3	3	6	1387	2	9	3	2
top	No	Travel_Rarely	Research & Development	Life Sciences	1	Male	Sales Executive	Married	No
freq	1158	981	909	570	1	835	313	636	992

Fig. 25: The command and results labeling encoding

```
[ ] df_outliers['Attrition'].value_counts()

    No      1158
    Yes      229
    Name: Attrition, dtype: int64

[ ] df_outliers['Gender'].value_counts()

    Male      835
    Female    552
    Name: Gender, dtype: int64

[ ] remap_attr = {'Yes': 1, 'No':0}
    remap_ovt = {'Yes':1, 'No':0}
    remap_gender = {'Male':1, 'Female':0}

    df_outliers['Attrition'] = df_outliers['Attrition'].replace(remap_attr)
    df_outliers['OverTime'] = df_outliers['OverTime'].replace(remap_ovt)
    df_outliers['Gender'] = df_outliers['Gender'].replace(remap_gender)

[ ] df_outliers['Attrition'].value_counts()

    0    1158
    1     229
    Name: Attrition, dtype: int64

[ ] df_outliers['Gender'].value_counts()

    1    835
    0    552
    Name: Gender, dtype: int64
```

Fig. 26: Command labeling encoding for re-map outliers

After the labeling encoding process is complete, the process continues with data encoding using One-hot Encoding. The command for performing the One-hot Encoding process is shown in the following Fig. 27.

```
[ ] df_outliers.describe(include='object')
```

	BusinessTravel	Department	EducationField	EmployeeNumber	JobRole	MaritalStatus
count	1387	1387	1387	1387	1387	1387
unique	3	3	6	1387	9	3
top	Travel_Rarely	Research & Development	Life Sciences	1	Sales Executive	Married
freq	981	909	570	1	313	636

```
[ ] df_onehot = df_outliers.copy()
    for cat in ['BusinessTravel', 'Department','EducationField','JobRole','MaritalStatus']:
      onehots = pd.get_dummies(df_onehot[cat], prefix=cat)
      df_onehot = df_onehot.join(onehots)

[ ] df_onehot = df_onehot.drop(['BusinessTravel', 'Department','EducationField','JobRole','MaritalStatus'], axis=1)
    #drop the original categorical feature

[ ] df_onehot.sample()
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	Gender	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	MonthlyR
588	50	0	691	2	3	815	3	1	64	3	4	3	17639	6

```
[ ] df_onehot['Attrition'].value_counts()

    0    1158
    1     229
    Name: Attrition, dtype: int64
```

Fig. 27: Command One-hot Encoding process and the result

After the encoding process is complete, it is followed by a data split process to break the data into 2 parts, namely test data and training data

The dataset obtained from Kaggle will then be divided into 2 to obtain training data and test data with a simulated training data ratio of 0.7 and 0.3 for test data.

E. Split Data

```
[ ] df_ml = df_onehot.copy()

[ ] df_ml.info()

[ ] <class 'pandas.core.frame.DataFrame'>
    Int64Index: 1387 entries, 0 to 1469
    Data columns (total 51 columns):
     #   Column                             Non-Null Count  Dtype
    ---  ------                             --------------  -----
     0   Age                                1387 non-null   int64
     1   Attrition                          1387 non-null   int64
     2   DailyRate                          1387 non-null   int64
     3   DistanceFromHome                   1387 non-null   int64
     4   Education                          1387 non-null   int64
     5   EmployeeNumber                     1387 non-null   object
     6   EnvironmentSatisfaction            1387 non-null   int64
     7   Gender                             1387 non-null   int64
     8   HourlyRate                         1387 non-null   int64
     9   JobInvolvement                     1387 non-null   int64
     10  JobLevel                           1387 non-null   int64
     11  JobSatisfaction                    1387 non-null   int64
     12  MonthlyIncome                      1387 non-null   int64
     13  MonthlyRate                        1387 non-null   int64
     14  NumCompaniesWorked                 1387 non-null   int64
     15  OverTime                           1387 non-null   int64
     16  PercentSalaryHike                  1387 non-null   int64
     17  PerformanceRating                  1387 non-null   int64
     18  RelationshipSatisfaction           1387 non-null   int64
     19  StockOptionLevel                   1387 non-null   int64
     20  TotalWorkingYears                  1387 non-null   int64
     21  TrainingTimesLastYear              1387 non-null   int64
     22  WorkLifeBalance                    1387 non-null   int64
     23  YearsAtCompany                     1387 non-null   int64
     24  YearsInCurrentRole                 1387 non-null   int64
     25  YearsSinceLastPromotion            1387 non-null   int64
     26  YearsWithCurrManager               1387 non-null   int64
     27  BusinessTravel_Non-Travel          1387 non-null   uint8
     28  BusinessTravel_Travel_Frequently   1387 non-null   uint8
     29  BusinessTravel_Travel_Rarely       1387 non-null   uint8
     30  Department_Human Resources         1387 non-null   uint8
     31  Department_Research & Development  1387 non-null   uint8
     32  Department_Sales                   1387 non-null   uint8
     33  EducationField_Human Resources     1387 non-null   uint8
     34  EducationField_Life Sciences       1387 non-null   uint8
     35  EducationField_Marketing           1387 non-null   uint8
     36  EducationField_Medical             1387 non-null   uint8
     37  EducationField_Other               1387 non-null   uint8
     38  EducationField_Technical Degree    1387 non-null   uint8
     39  JobRole_Healthcare Representative  1387 non-null   uint8
     40  JobRole_Human Resources            1387 non-null   uint8
     41  JobRole_Laboratory Technician      1387 non-null   uint8
     42  JobRole_Manager                    1387 non-null   uint8
     43  JobRole_Manufacturing Director     1387 non-null   uint8
     44  JobRole_Research Director          1387 non-null   uint8
     45  JobRole_Research Scientist         1387 non-null   uint8
     46  JobRole_Sales Executive            1387 non-null   uint8
     47  JobRole_Sales Representative       1387 non-null   uint8
     48  MaritalStatus_Divorced             1387 non-null   uint8
     49  MaritalStatus_Married              1387 non-null   uint8
     50  MaritalStatus_Single               1387 non-null   uint8
    dtypes: int64(26), object(1), uint8(24)
    memory usage: 368.2+ KB

[ ] X = df_ml.drop(['Attrition', 'EmployeeNumber'], axis=1)
    y = df_ml[['Attrition']]

[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
```

Fig. 28: Command split data process

F. Pre-processing Data

Entering the data processing, the stages consist of checking imbalanced data and scaling data. To check the imbalance of data using oversampling which aims to reduce/eliminate outliers in the data.

```
[ ] y_train.value_counts(normalize=True)

    Attrition
    0           0.835052
    1           0.164948
    dtype: float64

[ ] y_train.value_counts()

    Attrition
    0           810
    1           160
    dtype: int64

[ ] # Create an instance of the RandomOverSampler class
    # menggunakan oversampling
    ros = RandomOverSampler(random_state=42)

    # Fit the resampler to the training data and resample
    X_train_resampled, y_train_resampled = ros.fit_resample(X_train, y_train)

[ ] y_train_resampled.value_counts()
    #y_train_resampled.value_counts(normalize=True)

    Attrition
    0           810
    1           810
    dtype: int64
```

Fig. 29: Command handling imbalance data process

ISSN No:-2456-2165

After that, data scaling is done using normalization. It aims to change the original data measurement scale into an accepted form without changing the value of the data. The data presented is already in numerical form. The following is a command to perform data scaling commands using normalization.

```
[ ] X_train_resampled.sample()
```

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	MonthlyRate	NumCompaniesWorked	Ov
843	29	992	1	3	3	1	85	3	1	3	2058	19757	0	

```
[ ] num = X_train_resampled.columns.to_list()

[ ] X_train_scaled = X_train_resampled.copy()
    X_test_scaled = X_test.copy()

[ ] scaler = StandardScaler()
    X_train_scaled[num] = scaler.fit_transform(X_train_resampled[num])
    X_test_scaled[num] = scaler.transform(X_test[num])

[ ] X_train_scaled.sample()
```

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	MonthlyRate	NumCompaniesWo
1147	-0.349597	-0.865312	-0.941729	1.102062	0.381031	0.776127	0.821294	-0.809201	0.235698	-0.556786	1.172255	1.646481	1.54

Fig. 30: Command for data scaling

G. Modeling

After the data processing is complete, now the data is ready to be used in the modeling phase. Because it will compare two algorithms, namely XG Boost and Support Vector Machine, it is necessary to model the two algorithms and then carry out the experimental process with previously processed data.

If the modeling is complete, the next step is to carry out the trial process. The trial process expected that the system can provide optimal predictive results. The following is the command used to carry out the testing process and its results.

```
svm = SVC()
xgb = XGBClassifier()

model = [logreg, svm, xgb]
me_summary = pd.DataFrame({'model':[],'accuracy_train':[],'accuracy':[],'precision':[],'recall':[],'f1':[],'auc_train':[],'auc':[]})

for i in model:
    name = f'{i}'.split('(')[0]
    print(name)
    i.fit(X_train_scaled, y_train_resampled)
    accuracy_train = round(accuracy_score(y_train_resampled, i.predict(X_train_scaled)),2) * 100
    accuracy = round(accuracy_score(y_test, i.predict(X_test_scaled)),2) * 100
    precision = round(precision_score(y_test, i.predict(X_test_scaled)),2) * 100
    recall = round(recall_score(y_test, i.predict(X_test_scaled)),2) * 100
    f1 = round(f1_score(y_test, i.predict(X_test_scaled)),2) * 100
    auc_train = round(roc_auc_score(y_train_resampled, i.predict(X_train_scaled)),2) * 100
    auc = round(roc_auc_score(y_test, i.predict(X_test_scaled)),2) * 100
    me_summary = me_summary.append
    ({'model':name, 'accuracy_train':accuracy_train, 'accuracy':accuracy, 'precision':precision, 'recall':recall, 'f1':f1, 'auc_train':auc_train,'auc':auc}, ignore_index=True)
    # print(f'{name}, acc_tr:{accuracy_train}, acc:{accuracy}, prc:{precision}, rcl:{recall}, f1:{f1}, auc:{auc}')
me_summary

#me summary untuk mengeluarkan matriks
```

Fig. 31: Command for the testing process

ISSN No:-2456-2165

H. Evaluation

After the trial process was completed, it was found that among the two algorithm models, namely XG Boost and Support Vector Machine, the results showed that best accuracy was found in the XG Boost model with an accuracy of 86%, followed by a Support Vector Machine of 84 %.

Performance measurement/evaluation uses the Confusion Matrix. The performance evaluation process is carried out for each algorithm model used. The command used along with the results of the Confusion Matrix is shown in the image below.

Fig. 37: Command of XG Boost performance evaluation process

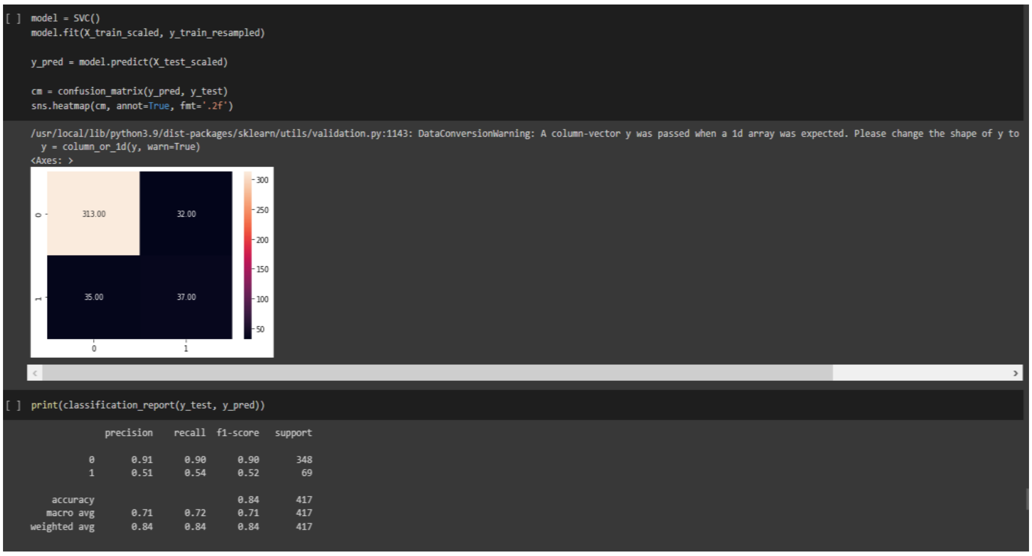

Fig. 38: Command of SVM performance evaluation process

IV. DISCUSSION

The initial hypothesis of this study is that it is suspected that the two algorithms used (XG Boost and Support Vector Machine) will obtain accurate results that reach >80%.

V. CONCLUSSION

Based on the results of prediction research using Machine Learning by comparing the two XG Boost and Support Vector Machine it can be concluded that the accuracy obtained from each accuracy is that XG Boost obtained 86% and Support Vector Machine 84%. In terms of accuracy, much of the predicted data is correct. Precision shows low false positives. Recall shows low false negatives.

REFERENCES

[1.] S. Yadav, A. Jain, and D. Singh, “Early Prediction of Employee Attrition using Data Mining Techniques,” *Proc. 8th Int. Adv. Comput. Conf. IACC 2018*, no. April, pp. 349–354, 2018, doi: 10.1109/IADCC.2018.8692137.

[2.] D. Prasetyawan and R. Gatra, “Model Convolutional Neural Network untuk Mengukur Kepuasan Pelanggan Berdasarkan Ekspresi Wajah,” *J. Tek. Inform. dan Sist. Inf.*, vol. 8, no. 3, pp. 661–673, 2022, doi: 10.28932/jutisi.v8i3.5493.

[3.] Ö. Çelik, “A Research on Machine Learning Methods and Its Applications,” *J. Educ. Technol. Online Learn.*, vol. 1, no. 3, pp. 25–40, 2018, doi: 10.31681/jetol.457046.

[4.] A. Roihan, P. A. Sunarya, and A. S. Rafika, “Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper,” *IJCIT (Indonesian J. Comput. Inf. Technol.*, vol. 5, no. 1, pp. 75–82, 2020, doi: 10.31294/ijcit.v5i1.7951.

[5.] P. Santoso, H. Abijono, and N. L. Anggreini, “Algoritma Supervised Learning Dan Unsupervised Learning Dalam Pengolahan Data,” *J. Teknol. Terap. G-Tech*, vol. 4, no. 2, pp. 315–318, 2021, doi: 10.33379/gtech.v4i2.635.

[6.] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli, “Applications of Deep Learning and Reinforcement Learning to Biological Data,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 6, pp. 1–33, 2018, doi: 10.1109/TNNLS.2018.2790388.

[7.] W. Punlumjeak, N. Rachburee, and J. Arunrerk, “Big Data Analytics: Student Performance Prediction Using Feature Selection and Machine Learning on Microsoft Azure Platform,” *J. Telecommun. Electron. Comput. Eng.*, vol. 9, no. 1–4, pp. 113–117, 2017.

[8.] Setiawan, S. Suprihanto, A. C. Nugraha, and J. Hutahaean, “HR Analytics: Employee Attrition Analysis Using Logistic Regression,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 830, no. 3, pp. 1–7, 2020, doi: 10.1088/1757-899X/830/3/032001.

[9.] S. L. V. Papineni, A. Mallikarjuna Reddy, S. Yarlagadda, S. Yarlagadda, and H. Akkineni, “An Extensive Analytical Approach on Human Resources Using Random Forest Algorithm,” *Int. J. Eng. Trends Technol.*, vol. 69, no. 5, pp. 119–127, 2021, doi: 10.14445/22315381/IJETT-V69I5P217.

[10.] K. Bhuva and K. Srivastava, “Comparative Study of The Machine Learning Techniques for Predicting The Employee Attrition,” *IJRAR*, vol. 5, no. 3, pp. 568–577, 2018, [Online]. Available: www.ijrar.org.

[11.] H. Ariesta and M. A. Kartawidjaja, “Feature Selection pada Azure Machine Learning untuk Prediksi Calon Mahasiswa Berprestasi,” *TESLA J. Tek. Elektro*, vol. 20, no. 2, pp. 166–174, 2018, doi: 10.24912/tesla.v20i2.2993.

[12.] A. Sethy and D. A. K. Raut, “Employee Attrition Rate Prediction Using Machine Learning Approach,” *Turkish J. Physiother. Rehabil.*, vol. 32, no. 3, pp. 14024–14031, 2020, [Online]. Available: www.turkjphysiotherrehabil.org.

[13.] D. F. Sengkey, F. D. Kambey, S. P. Lengkong, S. R. Joshua, and H. V. F. Kainde, “Pemanfaatan Platform Pemrograman Daring dalam Pembelajaran Probabilitas dan Statistika di Masa Pandemi CoVID-19,” *J. Inform.*, vol. 15, no. 4, pp. 257–264, 2020, [Online]. Available: https://ejournal.unsrat.ac.id/index.php/informatika/article/view/31685.

[14.] R. R. Zaveri and P. P. M. Chawan, “Google Colab : A White Paper,” *IJSRD - Int. J. Sci. Res. Dev.*, vol. 9, no. 6, pp. 124–125, 2021.

[15.] R. D. Fitriani, H. Yasin, and Tarno, “Penanganan Klasifikasi Kelas Data Tidak Seimbang dengan Random Oversampling Pada Naive Bayes,” *J. Gaussian*, vol. 10, no. 1, pp. 11–20, 2021.

[16.] M. C. Untoro and J. L. Buliali, “Penanganan Imbalance Class Data Laboratorium Kesehatan dengan Majority Weighted Minority Oversampling Technique,” *Regist. J. Ilm. Teknol. Sist. Inf.*, vol. 4, no. 1, pp. 23–29, 2018, doi: 10.26594/register.v4i1.1184.

[17.] A. Ambarwari, Q. J. Adrian, and Y. Herdiyeni, “Analisis Pengaruh Data Scaling Terhadap Performa Algoritme Machine Learning untuk Identifikasi Tanaman,” *Jurmal Resti (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 1, pp. 117–122, 2020.

[18.] R. Maharjan, “Employee Churn Prediction Using Logistic Regression and Support Vector Machine,” San Jose State University, 2021.

[19.] Suwarno and R. Kusnadi, “Analisis Perbandingan SVM, XGBoost dan Neutral Network pada Klasifikasi Ujaran Kebencian,” *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 5, pp. 896–903, 2021, doi: 10.29207/resti.v5i5.3506.

[20.] M. Guo, Z. Yuan, B. Janson, Y. Peng, Y. Yang, and W. Wang, “Older Pedestrian Traffic Crashes Severity Analysis Based on An Emerging Machine Learning XGBoost,” *Sustainability*, vol. 13, no. 926, pp. 1–26, 2021, doi: 10.3390/su13020926.

[21.] A. Raza, K. Munir, M. Almutairi, F. Younas, and M. M. S. Fareed, “Predicting Employee Attrition Using Machine Learning Approaches,” *Appl. Sci.*, vol. 12, no. 13, pp. 1–17, 2022, doi: 10.3390/app12136424.

[22.] M. Sokolova and G. Lapalme, “A Systematic Analysis of Performance Measures for Classification Tasks,” *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009, doi: 10.1016/j.ipm.2009.03.002.