Improving Employee Retention by Predicting Employee Attrition Using Machine Learning

Supported by:
Rakamin Academy
Career Acceleration School
www.rakamin.com

Rakamin Academy

Created by:
Kenneth Wahyudi

kennethwahyudi48@gmail.com
https://www.linkedin.com/in/kenneth-wahyudi-80b886209/

I am an aspiring data scientist with interest in machine learning and paddling through the data lake. I have experiences as a research intern in BATAN, and several data science and machine learning projects under my belt. Specifically intermediate to advanced knowledge in Exploratory Data Analysis, Data Pre-processing, Supervised & Unsupervised Learning, as well as Data Visualization and Storytelling. And I am always looking forward to exploring new and uncharted territories that might allow myself to grow and improve.

Overview

Overview

Human resources (HR) are the primary assets that must be managed effectively by a company to achieve its business goals. Today, we address a concern related to our workforce. Our focus is on understanding how to retain employees within our organization, thus reducing recruitment and onboarding costs. By identifying the key factors leading to employee dissatisfaction, we can create relevant programs to address their needs.

All results are from outputs of codes written in Python and its packages. JupyterLab was the GUI used in this project.

The dataset used in this project can be found in the following link.

Data Preprocesing

Data Preprocessing

In this section, our main focus will be in the aspect of data preprocessing. The outlines of the processes involved are as follows :

1. Checking and handling NULL values.
2. Replacing inappropriate values.
3. Discarding unnecessary columns.

Now, let’s delve deeper into the details of the processes outlined above.

The complete code and .ipynb file of the data preprocessing can be found in the following link.

Data Preprocessing

Rakamin Academy

```
Username                              0.000000
EnterpriseID                          0.000000
StatusPernikahan                      0.000000
JenisKelamin                          0.000000
StatusKepegawaian                     0.000000
Pekerjaan                             0.000000
JenjangKarir                          0.000000
PerformancePegawai                    0.000000
AsalDaerah                            0.000000
HiringPlatform                        0.000000
SkorSurveyEngagement                  0.000000
SkorKepuasanPegawai                   0.017422
JumlahKeikutsertaanProjek             0.010453
JumlahKeterlambatanSebulanTerakhir    0.003484
JumlahKetidakhadiran                  0.020906
NomorHP                               0.000000
Email                                 0.000000
TingkatPendidikan                     0.000000
PernahBekerja                         0.000000
IkutProgramLOP                        0.898955
AlasanResign                          0.229965
TanggalLahir                          0.000000
TanggalHiring                         0.000000
TanggalPenilaianKaryawan              0.000000
TanggalResign                         0.000000
dtype: float64
```

Let's first handle the null values.

To the right is the figure of the percentages of null values of each column to the total rows of the dataset. For the 'Skor Kepuasan Pegawai', 'Jumlah Keikutsertaan Projek', 'Jumlah Keterlambatan Sebulan Terakhir', 'Alasan Resign', and 'Jumlah Ketidakhadiran' columns, we imputed the null values with their mode values. As for the 'Ikut Program LOP' column, we will just drop the column since there are too many null values in the columns.

Data Preprocessing

Rakamin Academy

```
df['PernahBekerja'].value_counts()

1      286
yes      1
Name: PernahBekerja, dtype: int64
```

If we look at the value counts of the 'PernahBekerja' column (top left figure), we can see that it actually only has one unique value. However, due to errors, there are two types of values. They are 'yes' (with only one row) and 1 (the rest of the rows). We can change the value so that there are only one unique value, but a column with only one unique value is actually useless for the modelling part later on. So, we're just going to drop it.

```
df['StatusPernikahan'].value_counts()

Belum_menikah    132
Menikah           57
Lainnya           48
Bercerai          47
-                  3
Name: StatusPernikahan, dtype: int64
```

```
Belum_menikah    135
Menikah           57
Lainnya           48
Bercerai          47
Name: StatusPernikahan, dtype: int64
```

If we look at the 'StatusPernikahan' column (middle left figure), we can see that one of the values is '-'. That value does not represent anything, and is basically a NULL value. Therefore, we are going to change it into the mode, which is 'Belum_menikah’. Now the 'StatusPernikahan' column has the correct unique values (bottom left figure).

Data Preprocessing

```
Username                               285
EnterpriseID                           287
StatusPernikahan                         4
JenisKelamin                             2
StatusKepegawaian                        3
Pekerjaan                               14
JenjangKarir                             3
PerformancePegawai                       5
AsalDaerah                               5
HiringPlatform                           9
SkorSurveyEngagement                     5
SkorKepuasanPegawai                      5
JumlahKeikutsertaanProjek                9
JumlahKeterlambatanSebulanTerakhir       7
JumlahKetidakhadiran                    22
NomorHP                                287
Email                                  287
TingkatPendidikan                        3
AlasanResign                            11
TanggalLahir                           284
TanggalHiring                           97
TanggalPenilaianKaryawan               127
TanggalResign                           53
dtype: int64
```

Now, if we look at the number of unique values from each column (left figure), there are no longer any columns with only one unique value.

Annual Report on Employee Number Changes

Annual Report on Employee Number Changes

In this section, we are focusing on the aspect of the analysis of annual report on employee number changes. The outlines of the processes involved are as follows :

1. Creating a table to display resign counts each year.
2. Creating a table to display hiring counts each year.
3. Joining the two tables above and displaying the total employees every year.
4. Before displaying the changes in amounts of employees each year.
5. Finally, creating a plot to visualize the changes, and interpret the plot.

Now, let’s delve deeper into the details of the outlined processes above.

The complete code and .ipynb file of the analysis can be found in the following link.

Annual Report on Employee Number Changes

	tahun	hiring_count
0	2006	1
1	2007	2
2	2008	2
3	2009	7
4	2010	8
5	2011	76
6	2012	41
7	2013	43
8	2014	56
9	2015	31
10	2016	14
11	2017	5
12	2018	1

We first create a dataframe that houses the year and hiring counts of that specific year (figure on the left).

Then, we create a dataframe that houses the year and resign counts of that specific year (figure on the right).

	tahun	resign_count
1	2013	5
2	2014	12
3	2015	8
4	2016	8
5	2017	19
6	2018	26
7	2019	5
8	2020	6

Annual Report on Employee Number Changes

	tahun	hiring_count	resign_count
0	2006	1.0	NaN
1	2007	2.0	NaN
2	2008	2.0	NaN
3	2009	7.0	NaN
4	2010	8.0	NaN
5	2011	76.0	NaN
6	2012	41.0	NaN
7	2013	43.0	5.0
8	2014	56.0	12.0
9	2015	31.0	8.0
10	2016	14.0	8.0
11	2017	5.0	19.0
12	2018	1.0	26.0
13	2019	NaN	5.0
14	2020	NaN	6.0

Let us join the two dataframe into one, with outer join (figure on the left).

And now, we replace the NULL values with zero, and change the datatypes into integer (figure on the right).

	tahun	hiring_count	resign_count
0	2006	1	0
1	2007	2	0
2	2008	2	0
3	2009	7	0
4	2010	8	0
5	2011	76	0
6	2012	41	0
7	2013	43	5
8	2014	56	12
9	2015	31	8
10	2016	14	8
11	2017	5	19
12	2018	1	26
13	2019	0	5
14	2020	0	6

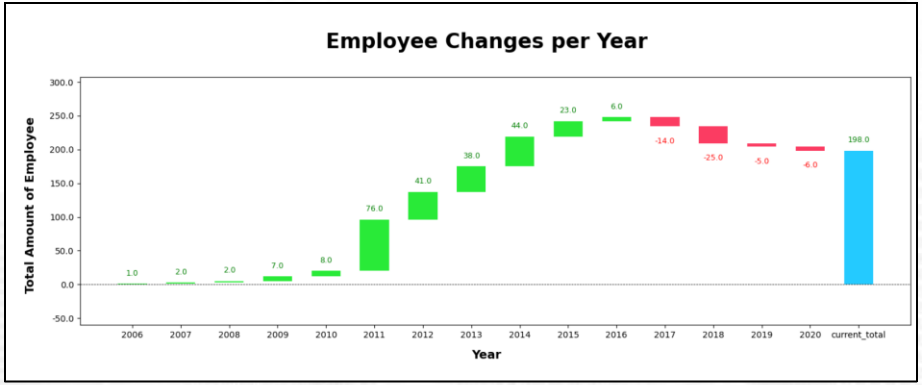
Annual Report on Employee Number Changes

	tahun	hiring_count	resign_count	total_employee	employee_changes
0	2006	1	0	1	1
1	2007	2	0	3	2
2	2008	2	0	5	2
3	2009	7	0	12	7
4	2010	8	0	20	8
5	2011	76	0	96	76
6	2012	41	0	137	41
7	2013	43	5	175	38
8	2014	56	12	219	44
9	2015	31	8	242	23
10	2016	14	8	248	6
11	2017	5	19	234	-14
12	2018	1	26	209	-25
13	2019	0	5	204	-5
14	2020	0	6	198	-6

We are now going to create the total employee present in every year column, and the employee changes every year (figure on the left).

Let us now plot the employee changes each year. We are going to use a waterfall chart for the visualization (next slide).

Annual Report on Employee Number Changes



From the chart to the left, we can see some interesting insights. The year 2011 is when we have a rapid expansion in our workforce, with a net of 76 employees joining the company.

The expansion slows down after that, before basically stagnating in 2016. Our workforce then starts to shrink in size in 2017, with 2018 having the highest amount of net employee loss at -28. After that, the shrinkage slows down, but we're still losing employees by the year. We hit a peak of 248 employees in 2016, and now we only have 198 employees contributing to the workforce. So as a conclusion, something(s) happened in 2016 and beyond, that caused the deterioration of the willingness of the employees to stay working in the company.

Resign Reason Analysis for Employee Attrition Management Strategy

Resign Reason Analysis for Employee Attrition Management Strategy

In this section, we are focusing on the aspect of analysis of resign reason for employee attrition management strategy. The outlines of the processes involved are as follows :

1. Creating a dataframe that displays the percentage of employee retention by division.
2. Creating a plot from the above dataframe and determine which division is the worst.
3. Creating a dataframe that displays the job level, employee performance, & reason of resignation of the employees from the worst division that have resigned.
4. Creating a sunburst plot from the above dataframe, interpreting the plot, and giving recommendations.

Now, let's dive into the details of the outlined processes above.

The complete code and .ipynb file of the analysis can be found in the following link.

Resign Reason Analysis for Employee Attrition Management Strategy

	Pekerjaan	stay_count
0	Data Analyst	8
1	Data Engineer	7
2	DevOps Engineer	3
3	Digital Product Manager	2
4	Machine Learning Engineer	2
5	Product Design (UI & UX)	15
6	Product Design (UX Researcher)	1
7	Product Manager	11
8	Scrum Master	3
9	Software Architect	1
10	Software Engineer (Android)	17
11	Software Engineer (Back End)	81
12	Software Engineer (Front End)	44
13	Software Engineer (iOS)	3

Creating the working employees dataframe (figure on the left).

Creating the resigned employees dataframe (figure on the bottom right).

	Pekerjaan	resign_count
0	Data Analyst	8
1	Data Engineer	3
2	Product Design (UI & UX)	9
3	Product Manager	6
4	Software Engineer (Android)	7
5	Software Engineer (Back End)	28
6	Software Engineer (Front End)	28

Resign Reason Analysis for Employee Attrition Management Strategy

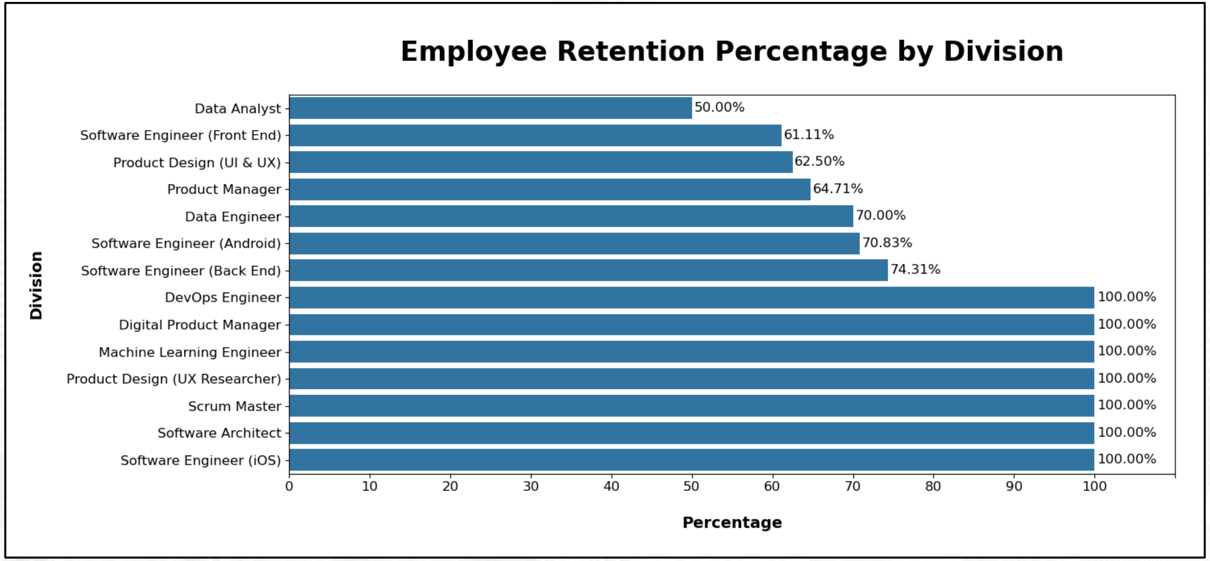
Rakamin Academy

	Pekerjaan	stay_count	resign_count
0	Data Analyst	8	8
1	Data Engineer	7	3
2	DevOps Engineer	3	0
3	Digital Product Manager	2	0
4	Machine Learning Engineer	2	0
5	Product Design (UI & UX)	15	9
6	Product Design (UX Researcher)	1	0
7	Product Manager	11	6
8	Scrum Master	3	0
9	Software Architect	1	0
10	Software Engineer (Android)	17	7
11	Software Engineer (Back End)	81	28
12	Software Engineer (Front End)	44	28
13	Software Engineer (iOS)	3	0

	Pekerjaan	stay_count	resign_count	total_employee	percentage
0	Data Analyst	8	8	16	50.000000
12	Software Engineer (Front End)	44	28	72	61.111111
5	Product Design (UI & UX)	15	9	24	62.500000
7	Product Manager	11	6	17	64.705882
1	Data Engineer	7	3	10	70.000000
10	Software Engineer (Android)	17	7	24	70.833333
11	Software Engineer (Back End)	81	28	109	74.311927
2	DevOps Engineer	3	0	3	100.000000
3	Digital Product Manager	2	0	2	100.000000
4	Machine Learning Engineer	2	0	2	100.000000
6	Product Design (UX Researcher)	1	0	1	100.000000
8	Scrum Master	3	0	3	100.000000
9	Software Architect	1	0	1	100.000000
13	Software Engineer (iOS)	3	0	3	100.000000

Joining the two, and filling the NULL values with zero. Also changing the data type to integer (figure on the left). And creating two new columns that house the number of total employees (stay_count + resign_count), and the percentage of retention (stay_count / total_employees) (figure on the right). Now that we have the percentages, we can visualize the table above to see which division is the worst when it comes to employee retention.

Resign Reason Analysis for Employee Attrition Management Strategy



From the barplot above, we can see that the 'Data Analyst' division is the worst when it comes to employee retention. With only 50% of hired data analysts still working at the company. Next, we need to know why that is the case. And to do so, we need to create a dataframe that shows the rank/seniority level of the jobs, the employee performance, and the reason of resignation.

Resign Reason Analysis for Employee Attrition Management Strategy

Rakamin Academy

	Username	JenjangKarir	PerformancePegawai	AlasanResign
0	jealousGelding2	Freshgraduate_program	Sangat_kurang	toxic_culture
1	hushedSeahorse7	Freshgraduate_program	Sangat_bagus	internal_conflict
2	sincereGatorade8	Freshgraduate_program	Sangat_bagus	toxic_culture
3	brainyFish3	Freshgraduate_program	Sangat_bagus	toxic_culture
4	troubledThrushe9	Freshgraduate_program	Bagus	toxic_culture
5	jealousIguana3	Freshgraduate_program	Biasa	toxic_culture
6	jumpyBuck8	Freshgraduate_program	Sangat_bagus	toxic_culture
7	finickySwift5	Freshgraduate_program	Biasa	internal_conflict

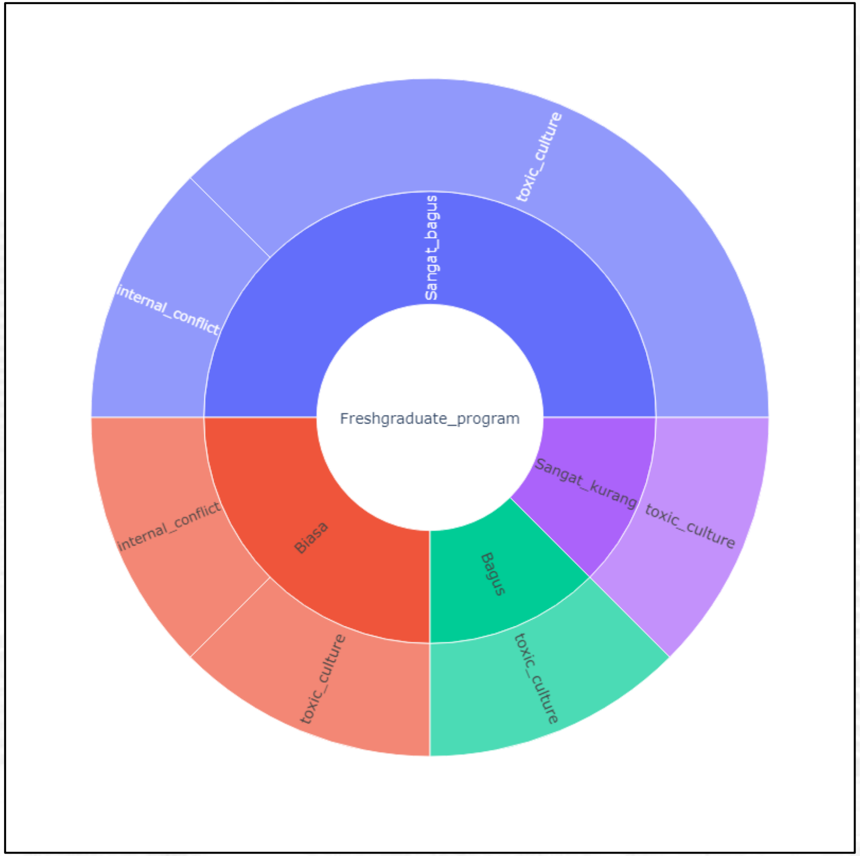
	JenjangKarir	PerformancePegawai	AlasanResign	count
0	Freshgraduate_program	Bagus	toxic_culture	1
1	Freshgraduate_program	Biasa	internal_conflict	1
2	Freshgraduate_program	Biasa	toxic_culture	1
3	Freshgraduate_program	Sangat_bagus	internal_conflict	1
4	Freshgraduate_program	Sangat_bagus	toxic_culture	3
5	Freshgraduate_program	Sangat_kurang	toxic_culture	1

We have created a dataframe that shows the rank/seniority level of the jobs, the employee performance, and the reason of resignation (top left figure).

We have grouped the table above by the rank/seniority level of the jobs, the employee performance, and the reason of resignation (bottom left figure).

Now, let's visualize the table above with a sunburst chart. To see what factors in the resignation of employees working in the data analyst division (next slide).

Resign Reason Analysis for Employee Attrition Management Strategy



From the left chart, we can see that all of the employees hired were in a freshgraduate level position. And out of all that resigned (a total of 8 people), only 1 has a performance rating of 'very poor'. And 50% (4 people) had an employee performance of 'very good'. This clearly means that we are losing talented and valuable employees, and therefore need to make some changes to stop this from happening again. And if we look at the reasons of resignation, there are only two reasons. They are toxic culture and internal conflict.

In order to solve the internal conflict reason, we recommend that the HR (or whoever that has the authority) to conduct a regular survey to see who are in conflict with whom. That way, internal conflicts can be mitigated by looking at the reason for the conflict, and trying to find a middle ground for both/all parties involved.

That being said, internal conflict only accounts for 25% (2/8) of the reasons. The rest are toxic culture. For this matter, we recommend that surveys also be done in order to see what kind of toxic culture are present within the division. If it is within the division itself, then we should consider talking to or even replacing the head/lead of the division (since all of the employees that resigned are in a fresh graduate level). If the toxic culture is in the company as a whole however, we should consider bringing it up to the board members in order for systematic changes to be done universally within the company. To maximize the detoxification of the working culture within the company.

Build an Automated Resignation Behavior Prediction using Machine Learning

In this section, we are focusing in the aspect of building an automated resignation behavior prediction using machine learning. The outlines of the processes involved are as follows :

1. Feature Engineering
2. Feature Encoding
3. Feature Scaling
4. Developing a predictive model
5. Hyperparameter tuning
6. Model Evaluation

Now, let's delve deeper into the details of the outlined processes above.

The complete code and .ipynb file of the analysis can be found in the following link.

	umur
0	48.251882
1	34.138261
2	40.711841
3	40.854209
4	43.830253
...	...
282	46.814511
283	42.956879
284	48.605065
285	36.498289
286	39.143053

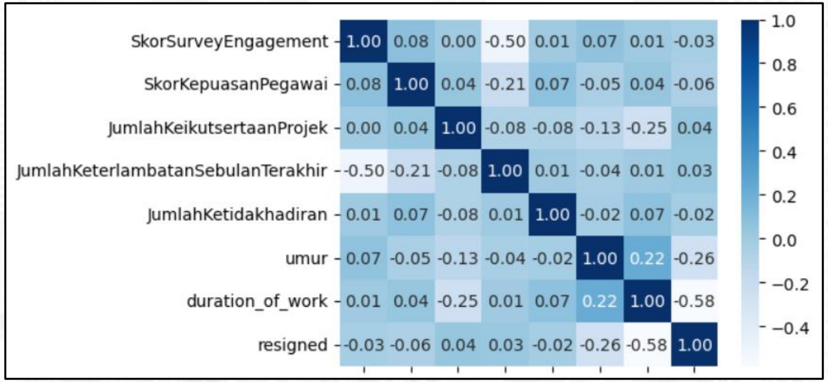
	duration_of_work
0	9.724846
1	4.440794
2	3.704312
3	6.620123
4	4.818617
...	...
282	9.015743
283	4.509240
284	4.572211
285	6.877481
286	8.125941

We have created two new features. They are age (from the 'TanggalLahir' feature), and duration of work (from the 'TanggalHiring' feature). Because the most recent date is from the 'TanggalResign' (2020-9-27), we are going to use that date as reference for the two features mentioned earlier. We are going to assume that the current date is (2020-10-1). For the age & duration of work however, if the employee has resigned, then the age & duration of work will only be counted up to the resignation date.

Build an Automated Resignation Behavior Prediction using Machine Learning

Rakamin Academy

Because our dataset only consists of 282 rows, we need to be very careful in not summoning the curse of dimensionality. Therefore, one way to help mitigate that is through the means of feature selection.



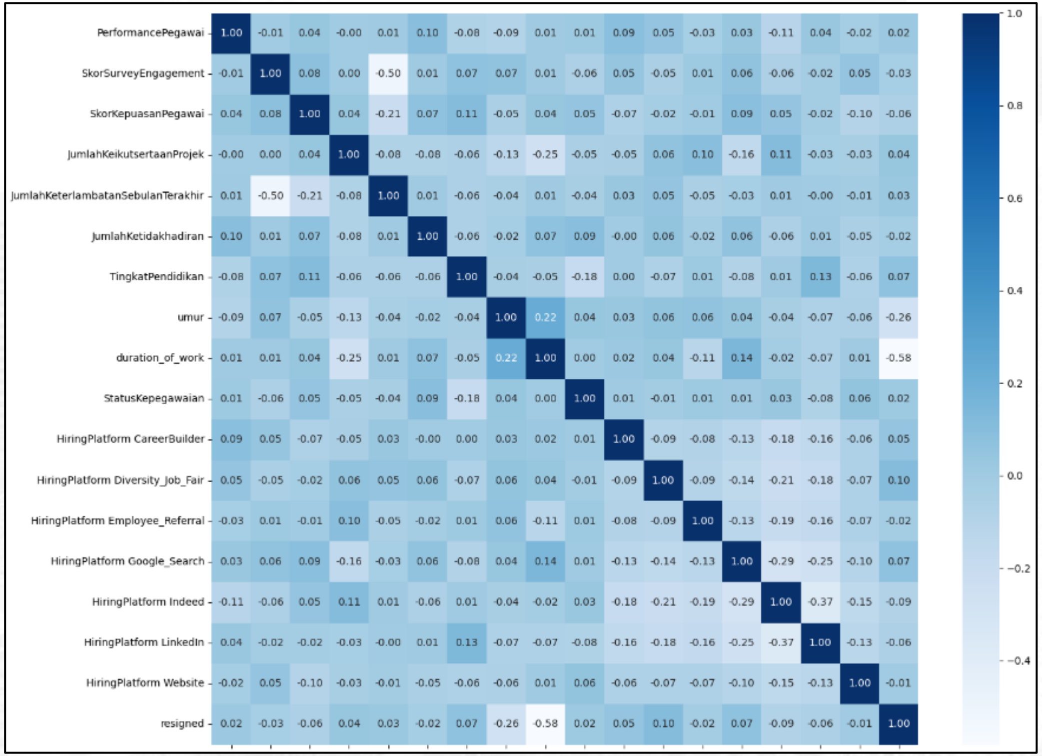
As we can see from the heatmap above, no categorical features (left figure) have a high correlation to the target compared to the numerical features (right figure),. However, there is no feature that is colinear with the other. From the categorical features (left figure), we are only going to use 4 features. They are 'StatusKepegawaian', 'PerformancePegawai', 'HiringPlatform', and 'TingkatPendidikan'. Let us now encode those features.

	PerformancePegawai	SkorSurveyEngagement	SkorKepuasanPegawai	JumlahKeikutsertaanProjek	JumlahKeterlambatanSebulanTerakhir	JumlahKetidakhadiran	TingkatPendidikan	umur	duration_of_work	StatusKepegawaian	HiringPlatform CareerBuilder	HiringPlatform Diversity_Job_Fair	HiringPlatform Employee_Referral	HiringPlatform Google_Search	HiringPlatform Indeed	HiringPlatform LinkedIn	HiringPlatform Website	resigned
0	4	4	4.0	0.0	0.0	9.0	1	48.251882	9.724846	0	0	0	1	0	0	0	0	0
1	0	4	4.0	4.0	0.0	3.0	0	34.138261	4.440794	1	0	0	0	0	0	0	1	1
2	3	4	3.0	0.0	0.0	11.0	1	40.711841	3.704312	1	0	0	0	0	1	0	0	1
3	3	3	3.0	0.0	4.0	6.0	0	40.854209	6.620123	0	0	0	0	0	0	1	0	0
4	3	3	3.0	0.0	0.0	11.0	0	43.830253	4.818617	1	0	0	0	0	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
282	3	2	5.0	0.0	0.0	16.0	0	46.814511	9.015743	1	0	0	0	1	0	0	0	0
283	2	4	5.0	0.0	0.0	11.0	0	42.956879	4.509240	1	0	0	0	0	0	1	0	1
284	2	2	5.0	0.0	0.0	17.0	0	48.605065	4.572211	1	0	1	0	0	0	0	0	1
285	3	4	5.0	0.0	0.0	20.0	0	36.498289	6.877481	1	0	0	0	1	0	0	0	1
286	3	4	3.0	0.0	0.0	2.0	0	39.143053	8.125941	0	0	0	0	0	1	0	0	0

Now that we have successfully encoded the features, let's see the correlation heatmap once again. To make sure whether or not some features are colinear to each other, and to see if we have more features to drop.

Build an Automated Resignation Behavior Prediction using Machine Learning

Rakamin Academy



It seems that we have no colinear features, and therefore we can proceed to the next stage of feature scaling.

Build an Automated Resignation Behavior Prediction using Machine Learning

Rakamin Academy

```
X_train[numericals.columns.tolist()].describe()
```

	SkorSurveyEngagement	SkorKepuasanPegawai	JumlahKeikutsertaanProjek	JumlahKeterlambatanSebulanTerakhir	JumlahKetidakhadiran	umur	duration_of_work
count	197.000000	197.000000	197.000000	197.000000	197.000000	197.000000	197.000000
mean	0.519036	0.632826	0.177665	0.072758	0.175315	0.344055	0.448490
std	0.212686	0.295439	0.332582	0.220947	0.124585	0.208813	0.187554
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.500000	0.333333	0.000000	0.000000	0.055556	0.177109	0.373628
50%	0.500000	0.666667	0.000000	0.000000	0.185185	0.335409	0.457132
75%	0.750000	1.000000	0.000000	0.000000	0.259259	0.492230	0.592524
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
X_test[numericals.columns.tolist()].describe()
```

	SkorSurveyEngagement	SkorKepuasanPegawai	JumlahKeikutsertaanProjek	JumlahKeterlambatanSebulanTerakhir	JumlahKetidakhadiran	umur	duration_of_work
count	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000
mean	0.538235	0.615686	0.121008	0.062745	0.158824	0.333121	0.467236
std	0.202315	0.327351	0.288680	0.197498	0.137497	0.185350	0.144567
min	0.000000	-0.333333	0.000000	0.000000	0.000000	-0.040577	0.105635
25%	0.500000	0.333333	0.000000	0.000000	0.055556	0.214110	0.377720
50%	0.500000	0.666667	0.000000	0.000000	0.111111	0.309386	0.466803
75%	0.750000	1.000000	0.000000	0.000000	0.259259	0.416256	0.592524
max	0.750000	1.000000	1.000000	1.000000	0.907407	0.931857	0.901060

We have successfully scaled the features, using MinMaxScaler (X_train on top figure & X_test on bottom figure). Now it's time to handle the imbalance of the target class with SMOTE.

```python
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_train_over, y_train_over = smote.fit_resample(X_train, y_train)
```

```python
y_train_over.value_counts()
```

```
resigned
0          138
1          138
dtype: int64
```

The oversampling has been implemented successfully. As the target classes are now balanced, as seen in the value counts above. Now, it's time to develop the predictive model.

Since the target is imbalanced (about 70% of 0 and 30% of 1), we cannot rely on the accuracy metric. And because suppressing the false negatives is important here (since we want to minimize the amount of employees that we predict won't resign, but actually will resign), we need to use the recall metric. Lastly, we need to use the ROC-AUC metric, since we need a metric that quantifies the model's ability in separating the two target classes (since our target is imbalanced). And as complimentary, we also will use the accuracy and precision metrics.

For this project, we are going to try 5 different model types. They are :

1. Linear Model (Logistic Regression)
2. Neighbor Model (KNN)
3. Tree-based Model (Decision Tree)
4. Bagging Algorithm (Random Forest)
5. Boosting Algorithm (XGBoost)

And see which is the best in terms of metrics scores, but also in terms of not overfitting.

Model	Accuracy	Recall	Precision	ROC-AUC
Logistic Regression	**0.7946** (test set) **0.8428** (train set)	**0.7262** (test set) **0.7923** (train set)	**0.6476** (test set) **0.7124** (train set)	**0.8465** (test set) **0.9122** (train set)
KNN	**0.6019** (test set) **0.7935** (train set)	**0.5516** (test set) **0.8479** (train set)	**0.3857** (test set) **0.6119** (train set)	**0.6230** (test set) **0.8995** (train set)
Decision Tree	**0.7494** (test set) **1.0** (train set)	**0.6429** (test set) **1.0** (train set)	**0.5618** (test set) **1.0** (train set)	**0.7154** (test set) **1.0** (train set)
Random Forest	**0.8299** (test set) **1.0** (train set)	**0.7183** (test set) **1.0** (train set)	**0.7589** (test set) **1.0** (train set)	**0.8744** (test set) **1.0** (train set)
XGBoost	**0.8274** (test set) **1.0** (train set)	**0.7103** (test set) **1.0** (train set)	**0.7223** (test set) **1.0** (train set)	**0.8764** (test set) **1.0** (train set)

From the results above, we can see that the tree based models (decision tree, random forest, and xgboost) are overwhelmingly overfit. The KNN model is subpar in performance metrics, and therefore for the hyperparameter tuning stage, we are only going to include the logistic regression model.

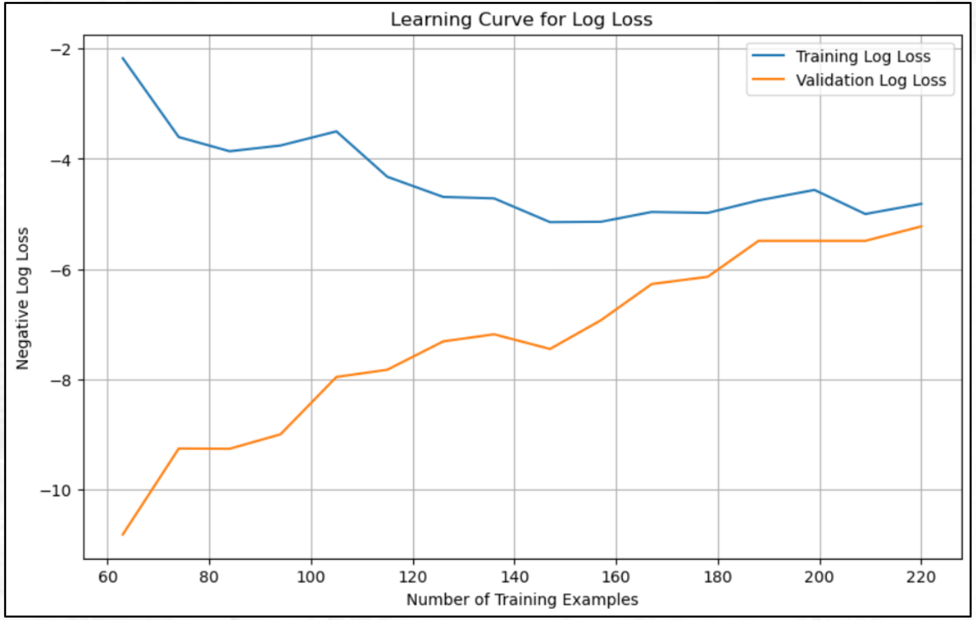
As we said previously, we want to maximize the recall score. Therefore we are going to prioritize the recall score in the hyperparameter tuning, and be fine with a slight accuracy decrease, as long as it is still significantly above the base classifier. A base classifier is defined as a classifier that only predicts the majority class, and in this case would have an accuracy score of 70%.

```
Accuracy (Cross-Val Test):  0.7507712944332664
Accuracy (Cross-Val Train):  0.8002660287937048

Recall (Cross-Val Test):  0.8095238095238094
Recall (Cross-Val Train):  0.9074074074074074

Precision (Cross-Val Test):  0.5648723258287309
Precision (Cross-Val Train):  0.6120680331046101

ROC-AUC (Cross-Val Test):  0.8609637188208615
ROC-AUC (Cross-Val Train):  0.918941112733059
```

As we can see to the left, the recall has improved significantly from around 0.72 to about 0.81. Whilst the accuracy score only decreased from 0.79 to 0.75. The ROC-AUC score is also very good, increasing from 0.84 to 0.86. This is a good result, and there are no signs of overfitting. Since the train set scores are not significantly more than the test set scores. However, to be completely sure, let's check the graph of the log loss of the model.

Build an Automated Resignation Behavior Prediction using Machine Learning

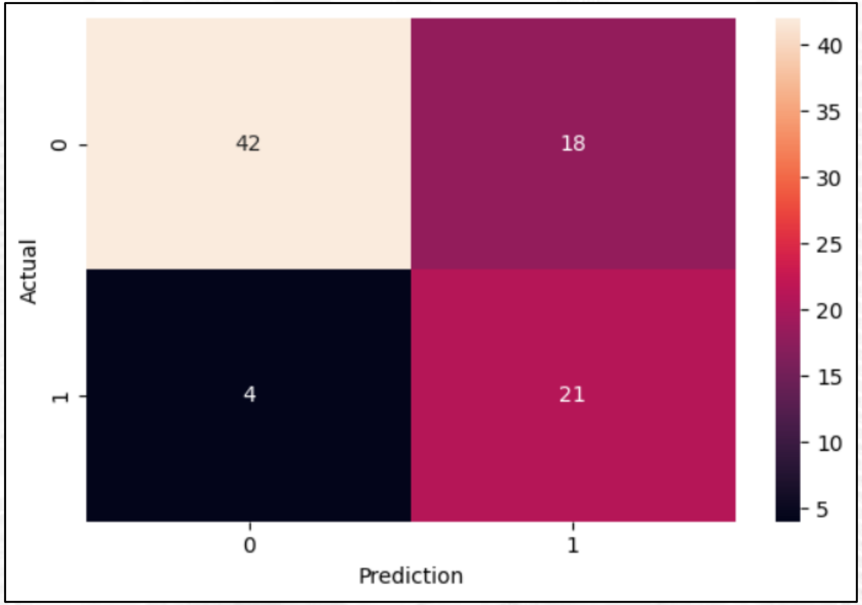
Rakamin Academy



From the log loss graph on the left, we can see that with more training samples, the training and validation sets eventually converge. This means that the model is a good fit. Now, it's time to move on to the confusion matrix.

Build an Automated Resignation Behavior Prediction using Machine Learning

Rakamin Academy



From the confusion matrix on the left, we can calculate the accuracy, recall, and precision. And they are as follows :

1. **Accuracy** : (True Positive + True Negative) / Total. (21 + 42) / 85 = **0.741**
2. **Recall** : True Positive / (True Positive + False Negative). 21 / (21 + 4) = **0.84**
3. **Precision** : True Positive / (True Positive + False Positive). 21 / (21 + 18) = **0.538**

As we can see above, we have successfully developed the model in a way that would maximize the recall score, whilst also not losing out too much on the accuracy score.

Presenting Machine Learning Products to the Business Users

In this section, we are focusing in the aspect of presenting machine learning products to the business users. The outlines of the processes involved are as follows :

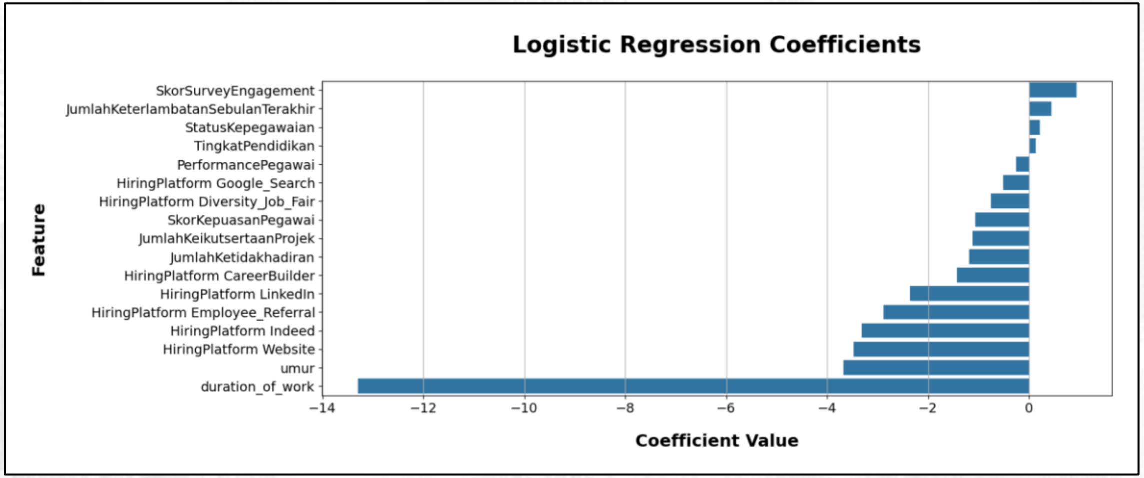
1. Displaying and plotting the model's coefficients.
2. Displaying and plotting the model's odds ratio.
3. Displaying and plotting the SHAP values for 2 instances of the prediction (positive & negative).
4. Business recommendation.

Now, let's dive into the details of the outlined processes above.

The complete code and .ipynb file of the analysis can be found in the following link.

Presenting Machine Learning Products to the Business Users

The model that we chose (logistic regression) is already a very interpretable model. And to interpret it, we can start from its coefficients.



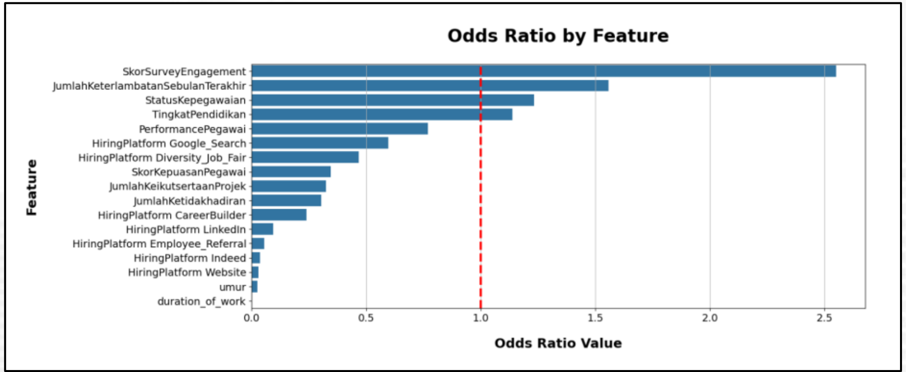
From the plot above, we can see that the duration_of_work feature is by far the most impactful feature in the model. The relationship is negative however, so as the duration_of_work value increases, the likelihood of the target being a positive or a 1 (resigning) decreases.

For an even more interpretable metric, we can use the odds ratio metric. An odds ratio quantifies the change in the odds of the positive class for a one-unit change in the predictor variable. The odds ratios provide information about the effect of each predictor on the odds of the binary outcome. Here are some general guidelines for interpreting odds ratios:

1. An odds ratio of 1 means that the predictor has no effect on the odds of the outcome.
2. An odds ratio greater than 1 indicates that the predictor increases the odds of the outcome.
3. An odds ratio less than 1 indicates that the predictor decreases the odds of the outcome.

For example, if the odds ratio for a predictor is 1.5, it means that a one-unit increase in that predictor is associated with a 50% increase in the odds of the outcome, holding all other predictors constant.

Presenting Machine Learning Products to the Business Users

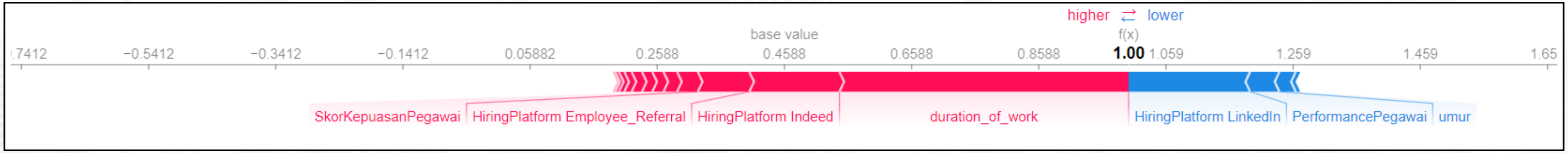


In an odds ratio metric, the least impactful feature is a feature that has an odds ratio that is equal to 1 or very close to it. And from the plot to the left, we can see that that feature is 'TingkatPendidikan'. The closer the value is to zero, the more the presence of the feature describes a negative outcome in the target. And the further away the value is from 1, the more the presence of the feature describes a positive outcome in the target.

The feature with the highest positive relationship with the target is the 'SkorSurveyEngagement' feature. It has a 2.54 odds ratio. Which means that a one-unit increase in the 'SkorSurveyEngagement' feature is associated with a 154% increase in the odds of the outcome (or 2.54 times more likely), holding all other features constant. The feature with the highest positive relationship with the target is the 'duration_of_work' feature. It has a 0.000002 odds ratio. Which means that the odds of the target outcome being a positive (or a 1) are 0.000002 times (or 0.0002%) as likely when the feature is present or increases by one unit, compared to when it's not present or unchanged. A very low odds ratio suggests a strong negative association between the feature and the outcome. In other words, the presence of this feature is indicative of a very low probability of the target outcome being a positive (or a 1). As a summary, the model thinks that with a high positive value of the 'SkorSurveyEngagement' feature, the employee is more likely to resign, citing a positive relationship between the two. In contrast, the model thinks that with a high positive value of the 'duration_of_work' feature, the employee is less likely to resign, citing a highly negative relationship between the two.

Now, let's look at the local explanation. Which basically is taking a look at a specific instance, and see why the model predicts a certain way. We are going to show 2 instances, one when the model predicts the employee to not resign, and the other when the model predicts the employee to resign. For this next section, we are going to use the SHAP library (SHapley Additive exPlanations).

Starting with the instance where the model predicts that the employee won't resign. In the above plot, we can see that the 'duration_of_work' feature is a really powerful predictor in this model compared to the other features. As the feature plays a pivotal role in predicting the target outcome as not resigning.



Moving on with the instance where the model predicts that the employee will resign. In the above plot, we can see that the 'duration_of_work' feature again is a really powerful predictor in this model compared to the other features. As the feature plays a pivotal role in predicting the target outcome as resigning.

Presenting Machine Learning Products to the Business Users

From the modelling & analysis above, we can see that some of the features stand out. Especially the duration of work feature. Using the predictive modelling, we can concur that these are the aspects that determine whether or not an employee will resign in the near future. The recommendation is to suppress the reason to resign in these employees. We should maximize the effectiveness of the working hours, clarity of their career prospects, efficiency in leadership & management, detoxification of the working culture, and resolve internal conflicts when they are present. Again, these things could be done in an anonymous survey, where we ask the employees what their biggest challenges are working in the company. And for specific employees, we also might consider giving them a raise if they deserve it, relocate them to a different team under a different supervisor, etc. The most susceptible are the ones in a young age group and a young duration of work. We recommend that for these employee demographics that special and extra care to be taken as the model suggest that with time, as they get older and as they have worked for longer, the likelihood of them resigning is slimmer than it used to be.