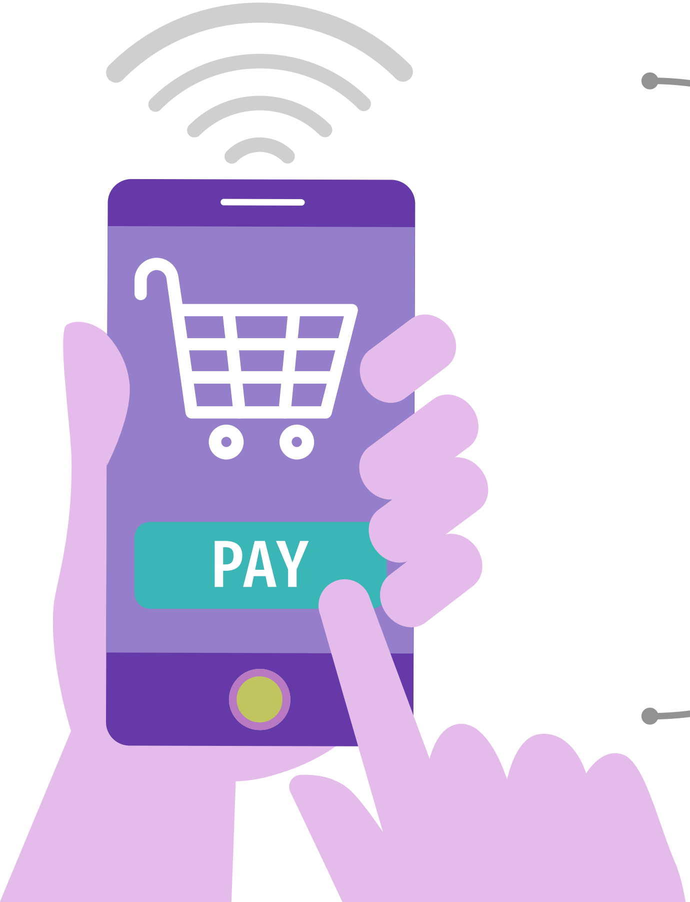
E-Commerce Churn Analysis

Team Project: Beta

1. Andhika Prakoso
2. Emil Supriatna
3. Monica Kristin Napitupulu

Purwadhika – JCDSOL-012

Content of Presentation:



01 Business Problem

Apa Masalah yang dihadapi?
Bagaimana menyelesaikannya?

02 Data Preprocessing

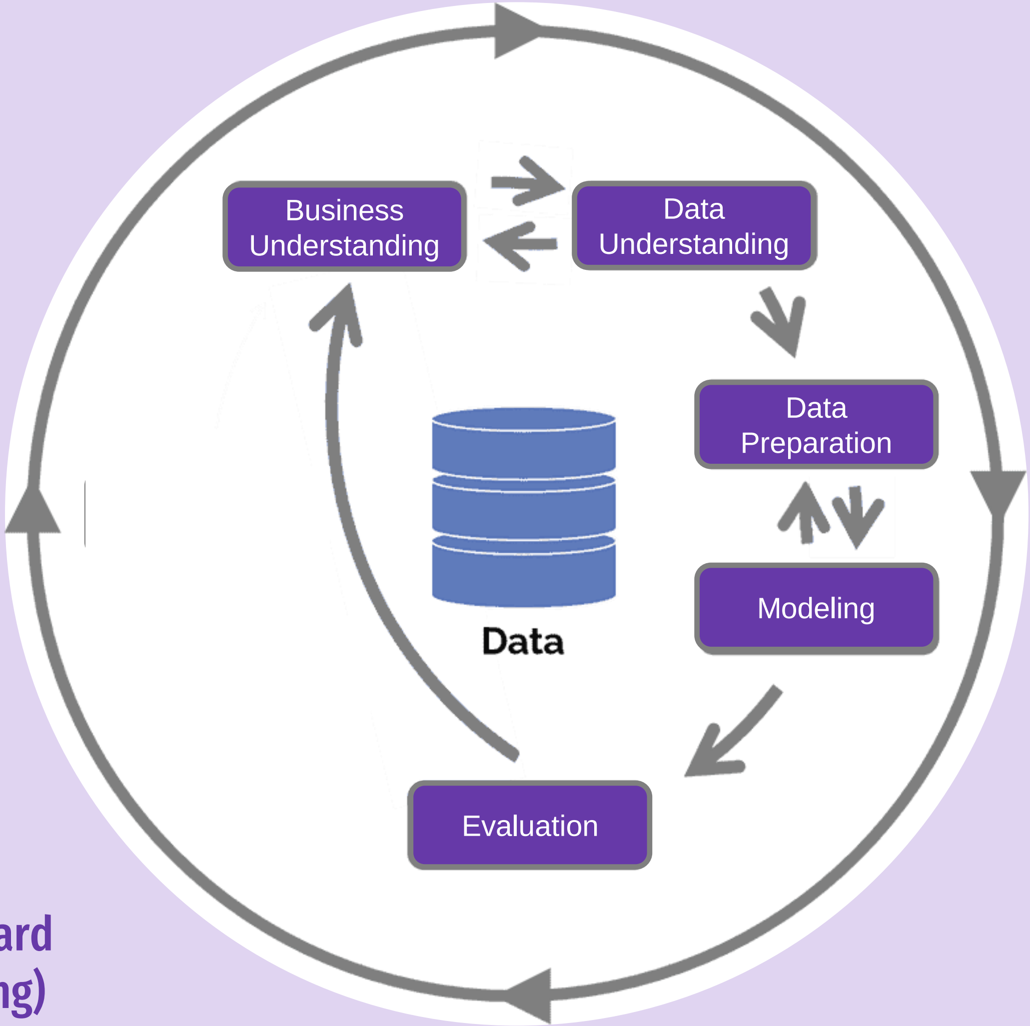
Bagaimana data dipahami?
Bagaimana data disiapkan untuk analisis tingkat lanjut?

03 Modeling & Evaluation

Apa model yang tepat digunakan?
Bagaimana kontribusinya terhadap permasalahan?

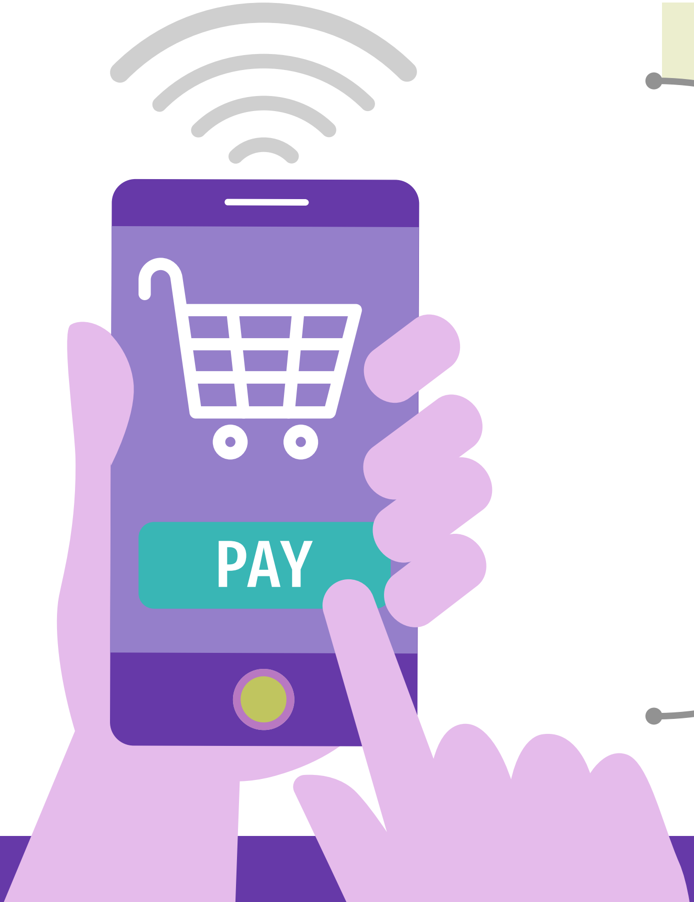
04 Recommendation

Apa rekomendasi bisnis yang dapat diberikan pada Stakeholder?



CRISP-DM

(Cross-Industry Standard Process for Data Mining)



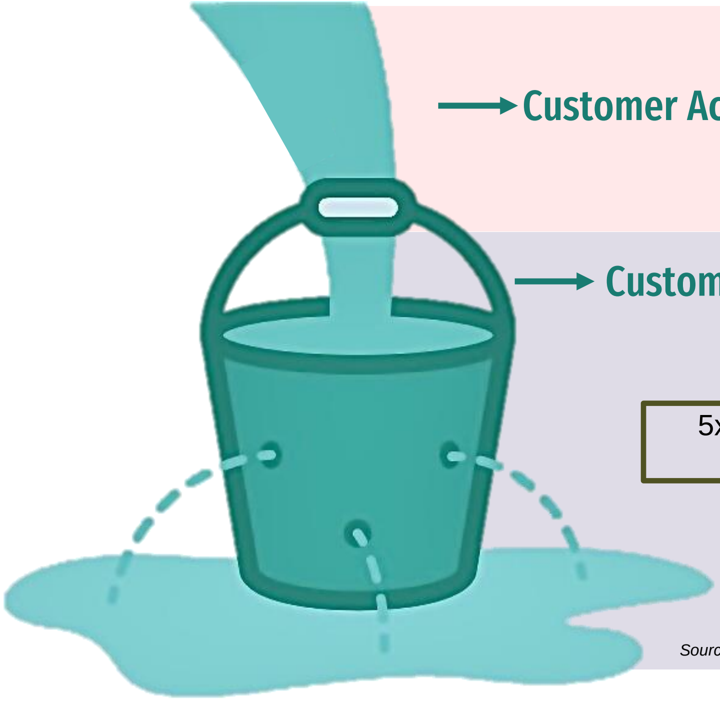
01 Business Problem

Apa Masalah yang dihadapi?
Bagaimana menyelesaikannya?

“Business understanding to understand the problem”

Predict Churn to Improve Business Revenue

Business revenue stream



→ Customer Acquisition

→ Customer Retention

Manfaat Customer Retention

5x biaya lebih rendah	Meningkatkan profit 25% - 95%	Kemudahan cross-selling produk

Source: https://hbr.org/2014/10/the-value-of-keeping-the-right-customers

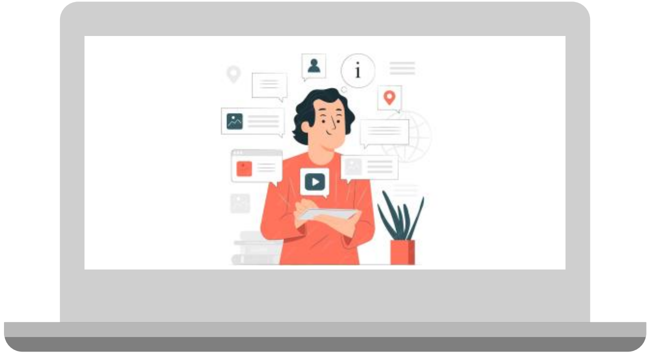
Business Problem:
- Fenomena Churn di dalam bisnis tidak dapat dihindari. Namun angka Churn bisa ditekan melalui berbagai macam strategi.
- Agar dapat memberikan strategi yang efektif, **bisnis membutuhkan *suatu sistem prediksi yang mampu membantu untuk mengurangi tingkat churn.***

Goals and Analytics Approach

Model Prediction

Membangun model prediksi terhadap potensi Churn di masa depan

Influential Factors



Mengetahui faktor-faktor paling berpengahuh terhadap Churn

Matrics Evaluation

Recall

Recall menyatakan seberapa besar persentase kejadian pada kelas positif yang berhasil dideteksi.

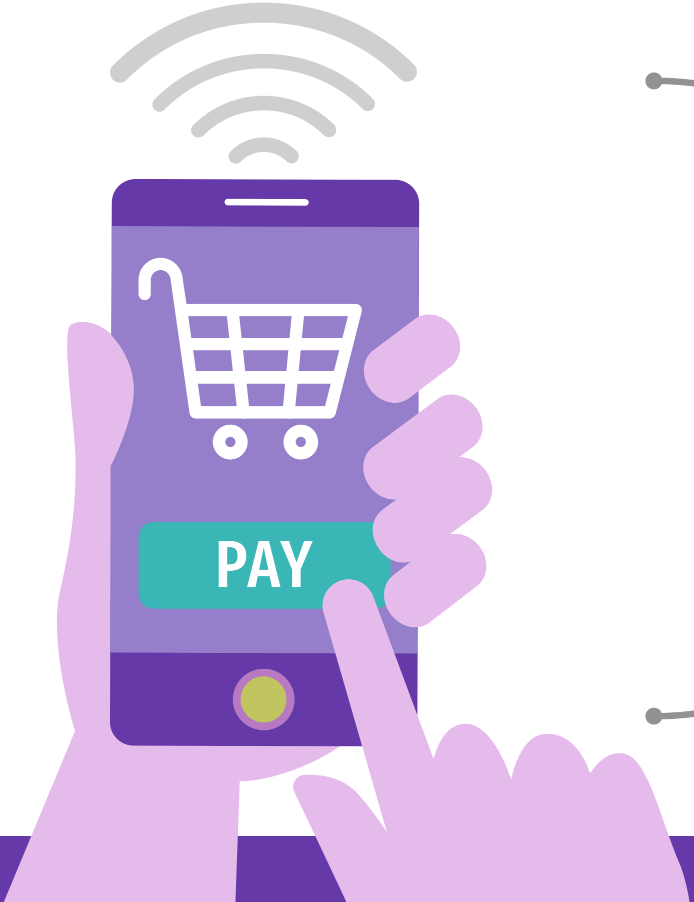
Cara kerjanya adalah semakin tinggi skor Recall, semakin kecil nilai False Negative, yang berarti model semakin baik dalam mendeteksi kasus Churn secara tepat.

Rumus:

$$Recall = \frac{TP}{TP + FN}$$

		PREDICTED	
		Negative	Positive
ACTUAL	Negative	True Negative (TN)	False Positive (FP) Type I Error
	Positive	False Negative (FN) Type II Error	True Positive (TP)

Confusion Matrix



02

Data Preprocessing

Step 1: Data Understanding
Step 2: Data Preparation

Step 1:

Data Understanding

Data Understanding

	Variable	Description
0	CustomerID	Unique customer ID
1	Churn	Churn Flag
2	Tenure	Tenure of customer in organization
3	PreferredLoginDevice	Preferred login device of customer
4	CityTier	City tier
5	WarehouseToHome	Distance in between warehouse to home of customer
6	PreferredPaymentMode	Preferred payment method of customer
7	Gender	Gender of customer
8	HourSpendOnApp	Number of hours spend on mobile application or website
9	NumberOfDeviceRegistered	Total number of deceives is registered on particular customer
10	PreferedOrderCat	Preferred order category of customer in last month
11	SatisfactionScore	Satisfactory score of customer on service
12	MaritalStatus	Marital status of customer
13	NumberOfAddress	Total number of added added on particular customer
14	Complain	Any complaint has been raised in last month
15	OrderAmountHikeFromlastYear	Percentage increases in order from last year
16	CouponUsed	Total number of coupon has been used in last month
17	OrderCount	Total number of orders has been places in last month
18	DaySinceLastOrder	Day Since last order by customer
19	CashbackAmount	Average cashback in last month

Data Type Checking

```
df.info()
```

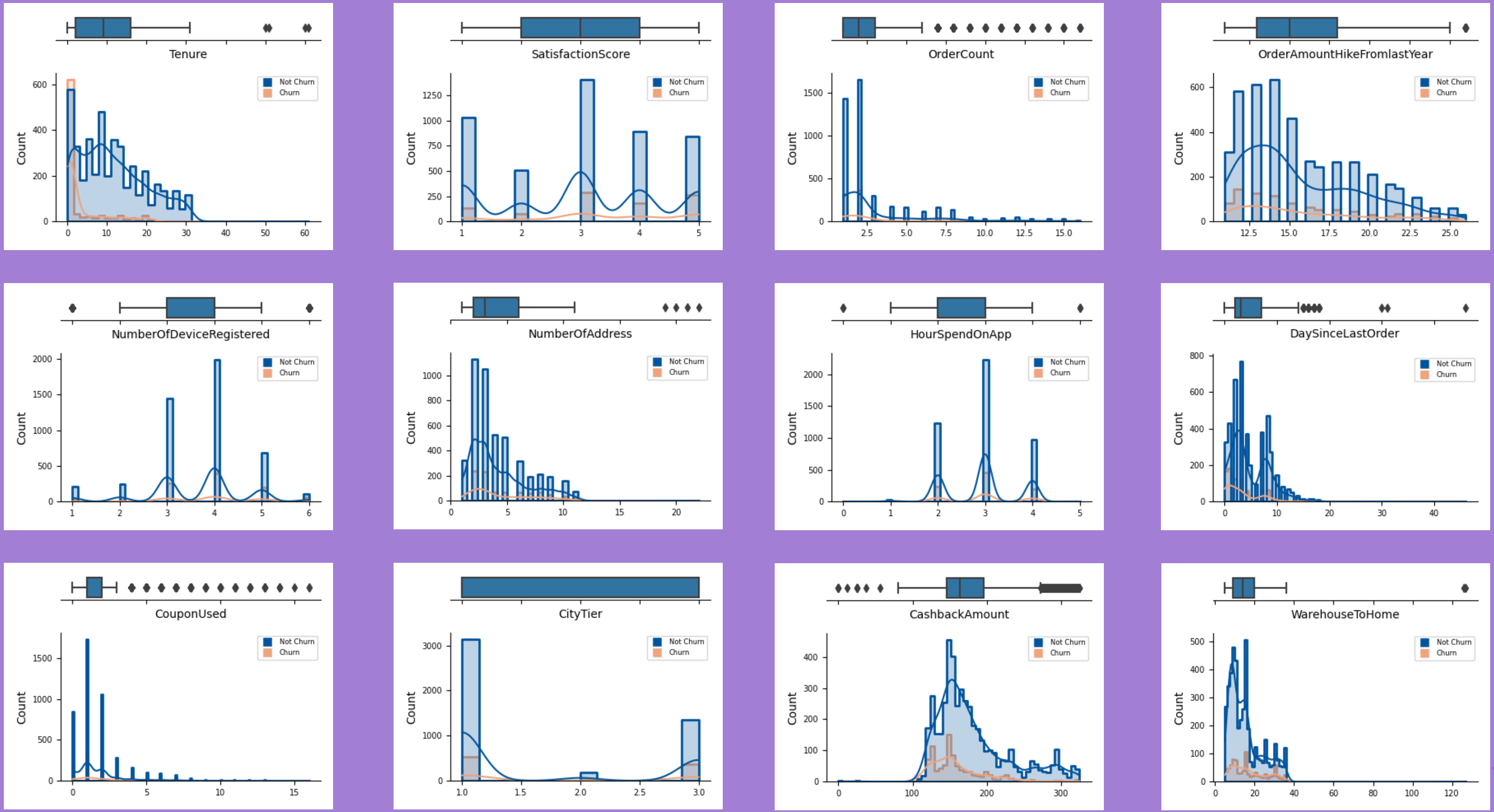
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5630 entries, 0 to 5629
Data columns (total 20 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   CustomerID                   5630 non-null   int64  
 1   Churn                        5630 non-null   int64  
 2   Tenure                       5366 non-null   float64
 3   PreferredLoginDevice         5630 non-null   object 
 4   CityTier                     5630 non-null   int64  
 5   WarehouseToHome              5379 non-null   float64
 6   PreferredPaymentMode         5630 non-null   object 
 7   Gender                       5630 non-null   object 
 8   HourSpendOnApp               5375 non-null   float64
 9   NumberOfDeviceRegistered     5630 non-null   int64  
 10  PreferedOrderCat             5630 non-null   object 
 11  SatisfactionScore            5630 non-null   int64  
 12  MaritalStatus                5630 non-null   object 
 13  NumberOfAddress              5630 non-null   int64  
 14  Complain                     5630 non-null   int64  
 15  OrderAmountHikeFromlastYear  5365 non-null   float64
 16  CouponUsed                   5374 non-null   float64
 17  OrderCount                   5372 non-null   float64
 18  DaySinceLastOrder            5323 non-null   float64
 19  CashbackAmount               5630 non-null   float64
dtypes: float64(8), int64(7), object(5)
memory usage: 879.8+ KB
```

Simple Statistic

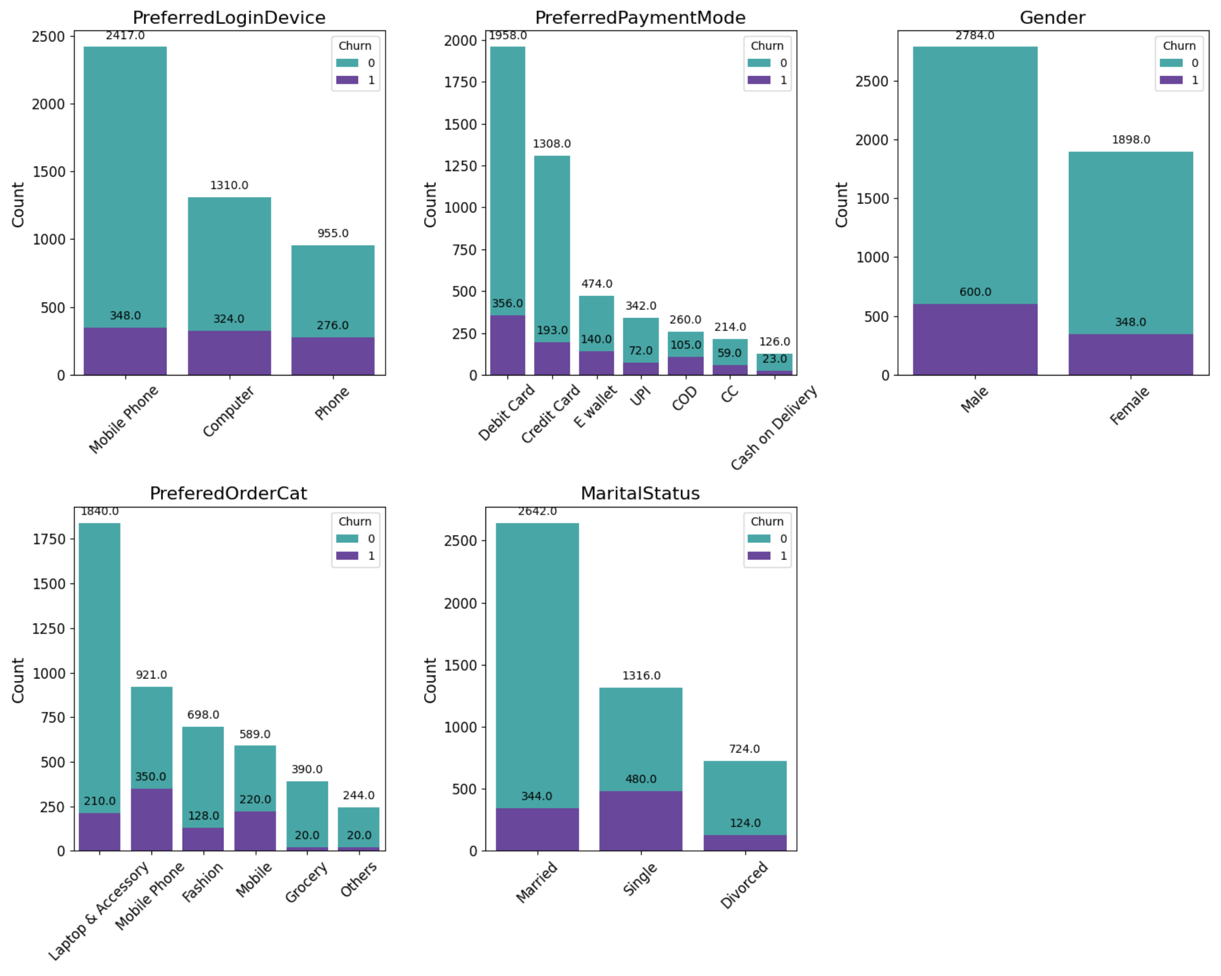
	count	mean	std	min	25%	50%	75%	max
Tenure	5366.0	10.189899	8.557241	0.0	2.00	9.00	16.0000	61.00
CityTier	5630.0	1.654707	0.915389	1.0	1.00	1.00	3.0000	3.00
WarehouseToHome	5379.0	15.639896	8.531475	5.0	9.00	14.00	20.0000	127.00
HourSpendOnApp	5375.0	2.931535	0.721926	0.0	2.00	3.00	3.0000	5.00
NumberOfDeviceRegistered	5630.0	3.688988	1.023999	1.0	3.00	4.00	4.0000	6.00
SatisfactionScore	5630.0	3.066785	1.380194	1.0	2.00	3.00	4.0000	5.00
NumberOfAddress	5630.0	4.214032	2.583586	1.0	2.00	3.00	6.0000	22.00
Complain	5630.0	0.284902	0.451408	0.0	0.00	0.00	1.0000	1.00
OrderAmountHikeFromlastYear	5365.0	15.707922	3.675485	11.0	13.00	15.00	18.0000	26.00
CouponUsed	5374.0	1.751023	1.894621	0.0	1.00	1.00	2.0000	16.00
OrderCount	5372.0	3.008004	2.939680	1.0	1.00	2.00	3.0000	16.00
DaySinceLastOrder	5323.0	4.543491	3.654433	0.0	2.00	3.00	7.0000	46.00
CashbackAmount	5630.0	177.223030	49.207036	0.0	145.77	163.28	196.3925	324.99

	count	unique	top	freq
PreferredLoginDevice	5630	3	Mobile Phone	2765
PreferredPaymentMode	5630	7	Debit Card	2314
Gender	5630	2	Male	3384
PreferedOrderCat	5630	6	Laptop & Accessory	2050
MaritalStatus	5630	3	Married	2986

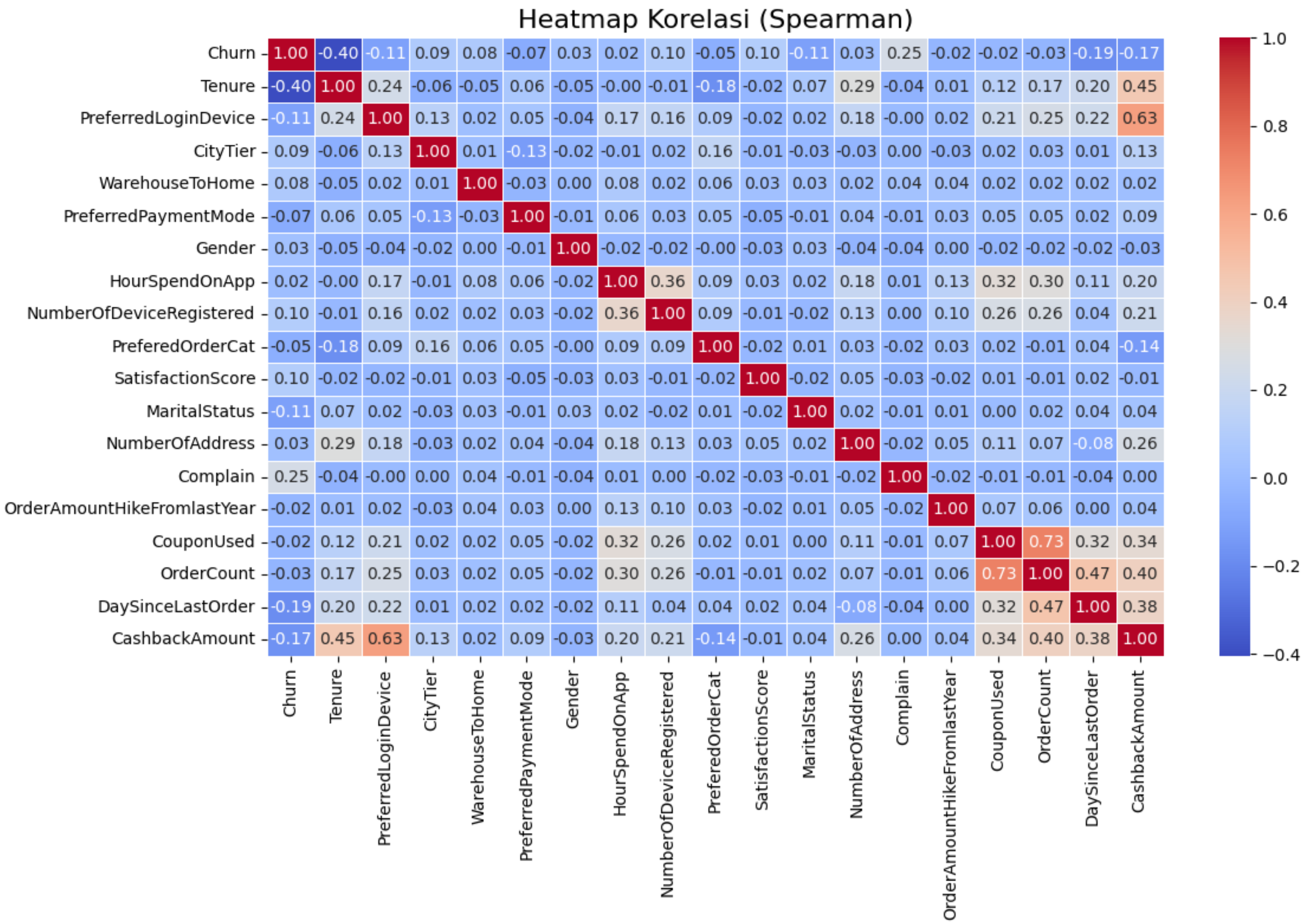
Data Distribution (numeric)



Data Distribution (categoric)



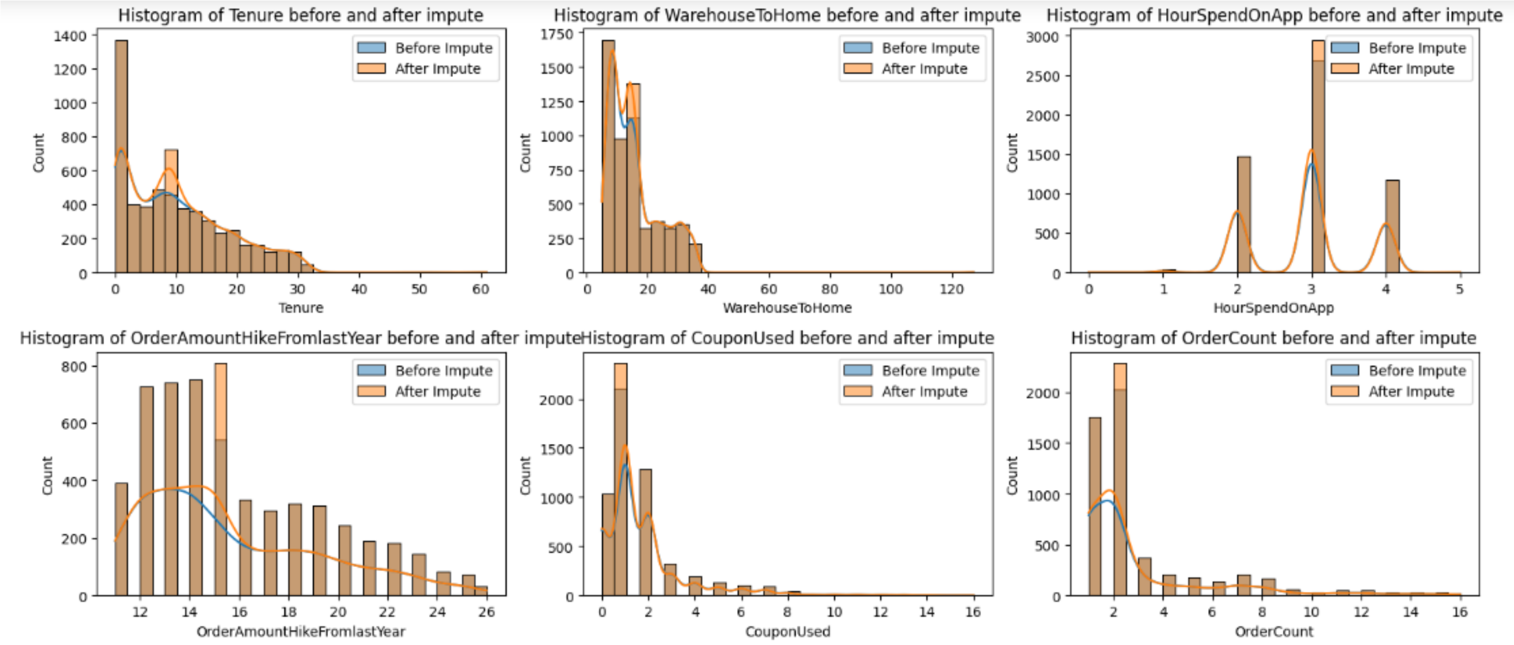
Data Correlation

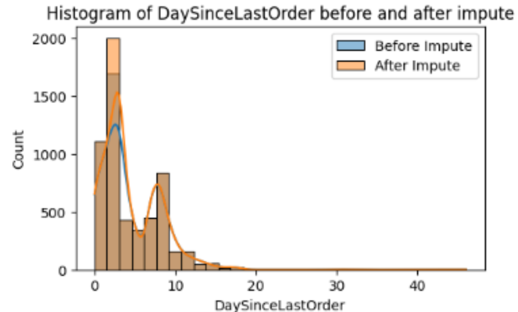


Step 2:

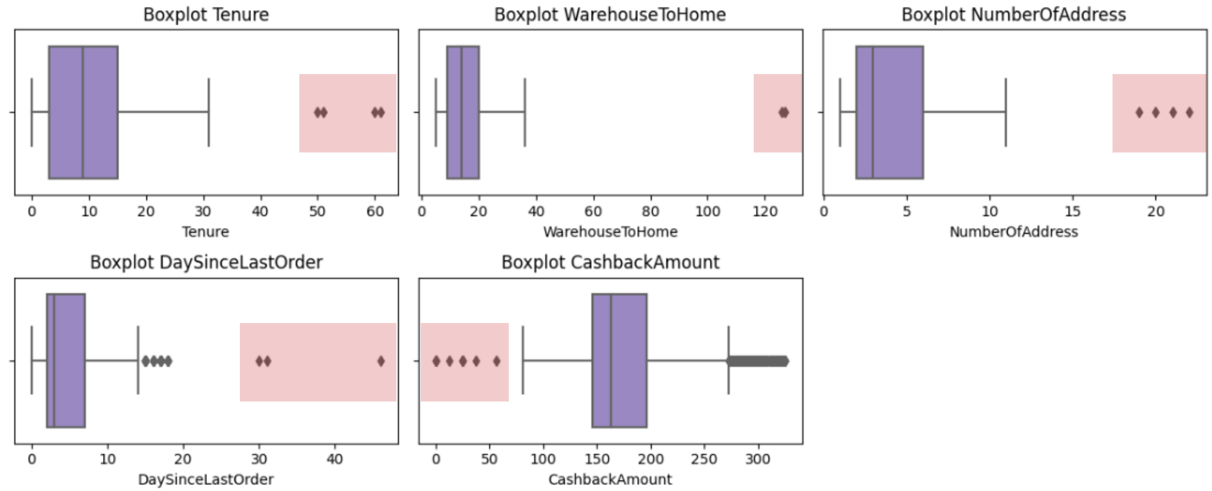
Data Preparation

Missing Values Handling

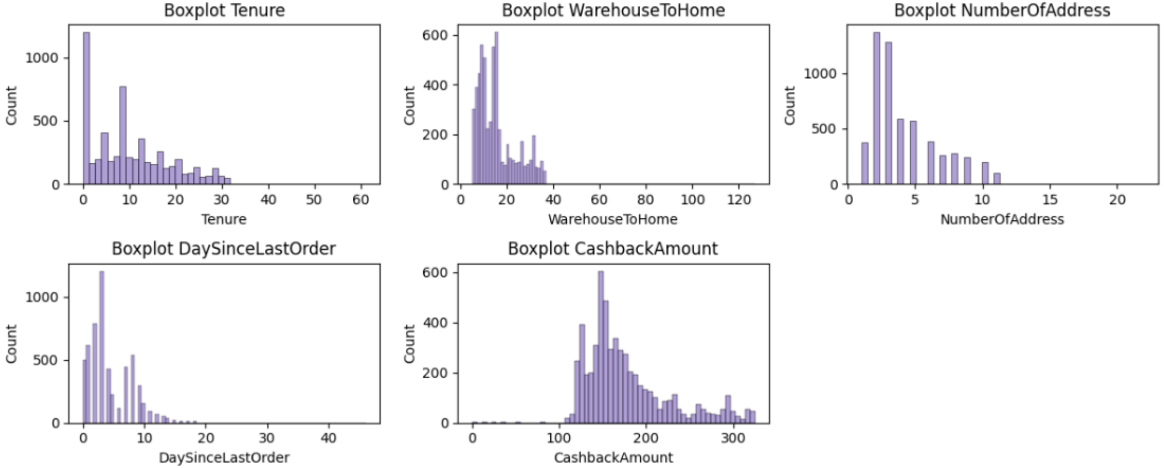


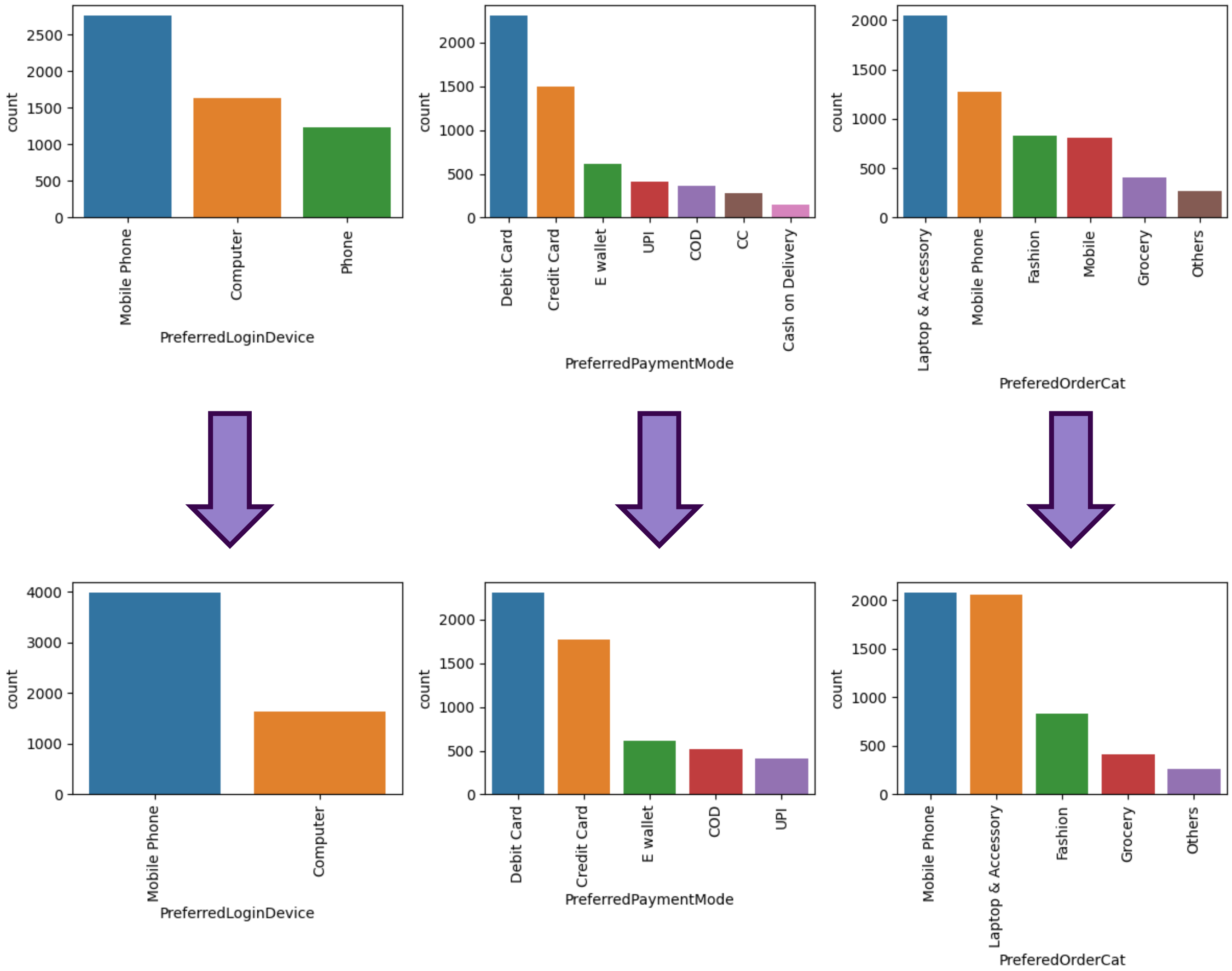



```python
# Inisialisasi SimpleImputer
imputer = SimpleImputer(strategy='median')
```



Outliers Handling





Inconsistent Variable

Other Data Preparation

DATA DUPLIKAT

```
duplicated_data = df.duplicated(subset='CustomerID')
print(df[duplicated_data])
```

```
Empty DataFrame
```

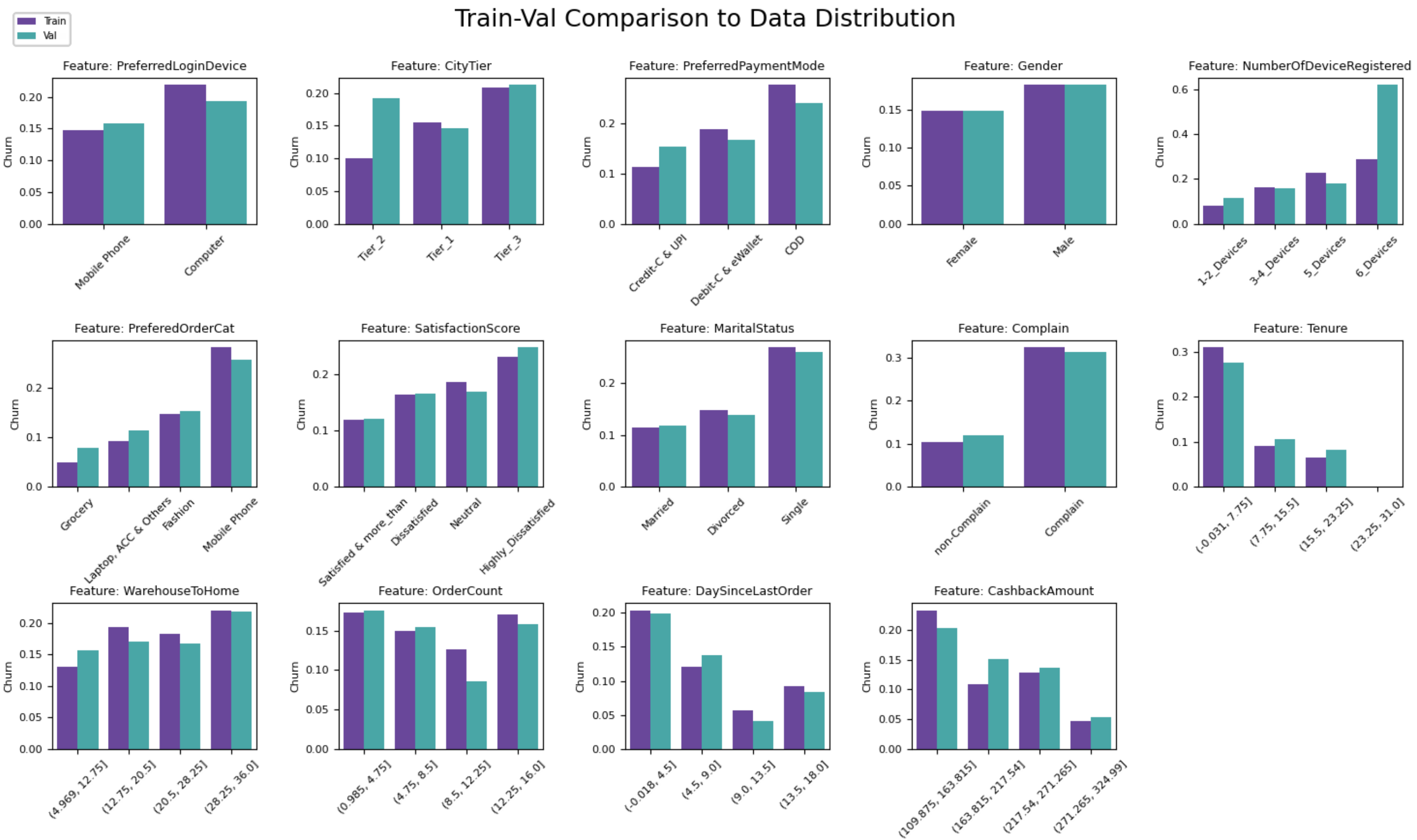
DROP CUSTOMER ID

```
# Drop Customer ID
df = df.drop(['CustomerID'], axis=1)
```

MAPPING FOR VISUALIZATION

```
df['CityTier'] = df['CityTier'].replace({1: 'Tier_1', 2 : 'Tier_2', 3 : 'Tier_3'})
df['SatisfactionScore'] = df['SatisfactionScore'].replace({1 : 'Very_Satisfied',
                                                            2 : 'Satisfied',
                                                            3 : 'Neutral',
                                                            4 : 'Dissatisfied',
                                                            5 : 'Highly_Dissatisfied'})
df['Complain'] = df['Complain'].replace({1: 'Complain', 0 : 'non-Complain'})

df['PreferedOrderCat'] = df['PreferedOrderCat'].replace('Laptop & Accessory', 'Laptop & Acc')
df['NumberOfDeviceRegistered'] = df['NumberOfDeviceRegistered'].replace({1 : '1_Device',
                                                            2 : '2_Devices',
                                                            3 : '3_Devices',
                                                            4 : '4_Devices',
                                                            5 : '5_Devices',
                                                            6 : '6_Devices'})
```

EDA for Modelling

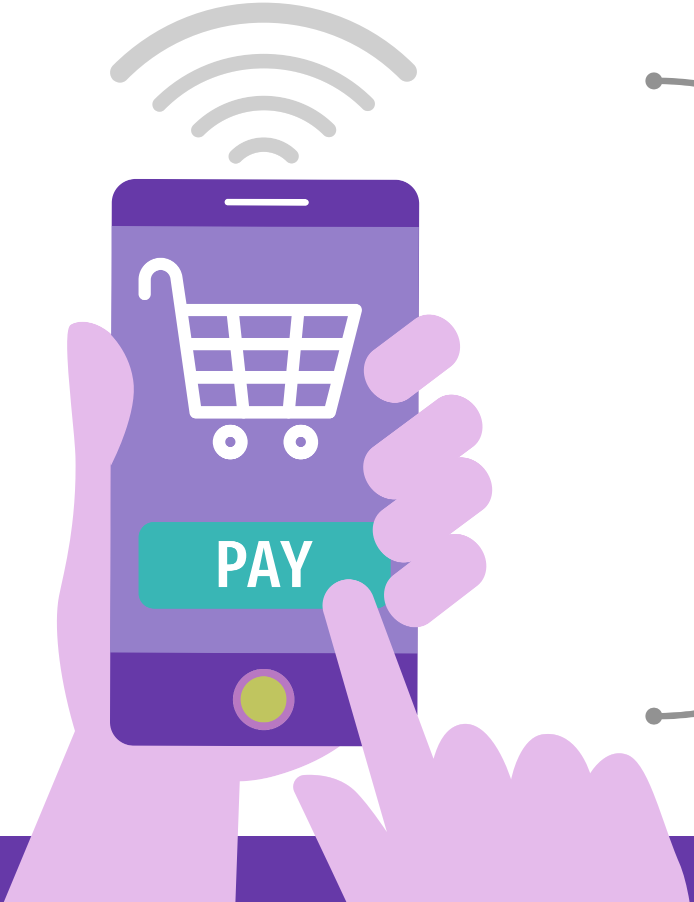


ENCODING DAN SCALING

```
ordinal_mapping = [
    {"col": "CityTier", "mapping": {"Tier_1": 1, "Tier_2": 2, "Tier_3": 3}},
    {"col": "NumberOfDeviceRegistered", "mapping": {"1-2_Devices": 1, "3-4_Devices": 2, "5_Devices": 3, "6_Devices": 4}},
    {"col": "SatisfactionScore", "mapping": {"Satisfied & more_than": 1, "Neutral": 2, "Dissatisfied": 3,
                                             "Highly_Dissatisfied": 4}},
    {"col": "PreferedOrderCat", "mapping": {"Grocery": 1, "Fashion": 2, "Mobile Phone": 3, "Laptop, ACC & Others": 4}}]

transformer = ColumnTransformer([
    ('One Hot Encoding', OneHotEncoder(drop='first'), ['PreferredLoginDevice', 'PreferredPaymentMode',
                                                       'MaritalStatus', 'Complain']),
    ('Ordinal Encoding', ce.OrdinalEncoder(cols=['CityTier', 'NumberOfDeviceRegistered',
                                                 'SatisfactionScore', 'PreferedOrderCat'],
                                           mapping=ordinal_mapping), ['CityTier', 'NumberOfDeviceRegistered',
                                                                      'SatisfactionScore', 'PreferedOrderCat']),
    ('Robust', RobustScaler(), ['Tenure', 'DaySinceLastOrder', 'CashbackAmount']),
])
```

ENCODED and SCALED FEATURES

```
 #   Column                                                Non-Null Count  Dtype  
---  ------                                                --------------  -----  
 0   One Hot Encoding__PreferredLoginDevice_Mobile Phone   3923 non-null   float64
 1   One Hot Encoding__PreferredPaymentMode_Credit-C & UPI 3923 non-null   float64
 2   One Hot Encoding__PreferredPaymentMode_Debit-C & eWallet 3923 non-null float64
 3   One Hot Encoding__MaritalStatus_Married               3923 non-null   float64
 4   One Hot Encoding__MaritalStatus_Single                3923 non-null   float64
 5   One Hot Encoding__Complain_non-Complain               3923 non-null   float64
 6   Ordinal Encoding__CityTier                            3923 non-null   float64
 7   Ordinal Encoding__NumberOfDeviceRegistered            3923 non-null   float64
 8   Ordinal Encoding__SatisfactionScore                   3923 non-null   float64
 9   Ordinal Encoding__PreferedOrderCat                    3923 non-null   float64
 10  Robust__Tenure                                        3923 non-null   float64
 11  Robust__DaySinceLastOrder                             3923 non-null   float64
 12  Robust__CashbackAmount                                3923 non-null   float64
dtypes: float64(13)
```



03

Modeling & Evaluation

Apa model yang tepat digunakan?
Bagaimana evaluasi dan kontribusinya terhadap permasalahan?

Model Benchmarking: K-Fold

model	mean recall	StdDev
XGBoost	0.794	0.059
AdaBoost	0.779	0.031
Decision Tree	0.778	0.045
Random Forest	0.754	0.064
LightGBM	0.749	0.059
GBoost	0.572	0.070
KNN	0.466	0.025

Model Benchmarking: Data Validation

	Model	recall score
1	Decision Tree	0.838
5	XGBoost	0.824
3	AdaBoost	0.817
2	Random Forest	0.782
6	LightGBM	0.754
4	GBoost	0.542
0	KNN	0.521

Oversampling

```
# Import oversampling method
Smote = SMOTE(random_state = 2020)
Ros = RandomOverSampler(random_state=2020)
```

model	mean recall Score	StdDev
XGB_ros	0.838	0.063
XGB	0.794	0.059
XGB_smote	0.784	0.036

model	mean recall Score	StdDev
TREE_smote	0.781	0.028
TREE	0.778	0.045
TREE_ros	0.758	0.050

model	mean recall Score	StdDev
ADA_smote	0.781	0.032
ADABoost	0.779	0.031
ADA_ros	0.763	0.055

feature selection

```
feature_selection_XGB = RFE(estimator=XGBClassifier(random_state=2020), n_features_to_select=11)
```

model	mean recall Score	StdDev
XGB_ROS_FS	0.843	0.060
XGB_ROS	0.838	0.063

model	mean recall Score	StdDev
TREE_SMOTE_FS	0.775	0.057
TREE_SMOTE	0.758	0.050

model	mean recall Score	StdDev
ADA_FS	0.785	0.043
model	0.779	0.031

Machine Learning we use

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. *(geeksforgeeks.org/xgboost)*

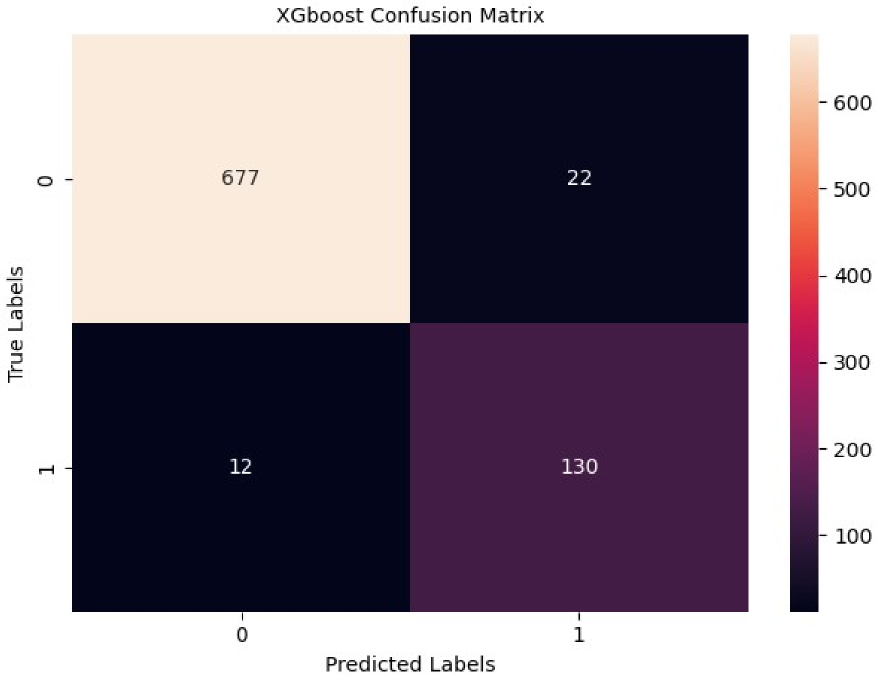
Hyperparameter Tuning

```
# Hyperparameter Tuning pada XGB dengan Random Over Sampling dan Feature Selection
xgb = XGBClassifier(random_state=2020)

estimator = Pipeline([
    ('preprocess', transformer),
    ('resampler',Ros),
    ('feature_selector', feature_selection_XGB),
    ('model', xgb)])

hyperparam_space = {
    'model__n_estimators': [100, 200, 300],
    'model__max_depth': [3, 5, 7],
    'model__learning_rate': [0.01, 0.1, 0.3],
    'model__subsample': [0.7, 0.8, 0.9],
}

grid_search_xgb = GridSearchCV(
    estimator,
    param_grid = hyperparam_space,
    cv = skfold,
    scoring = 'recall',
    n_jobs = -1
)
```

```
Recall Score XGB Default dengan  Feature Selection :  0.8873239436619719
Recall Score XGB Tuned dengan  Feature Selection:  0.8591549295774648
```

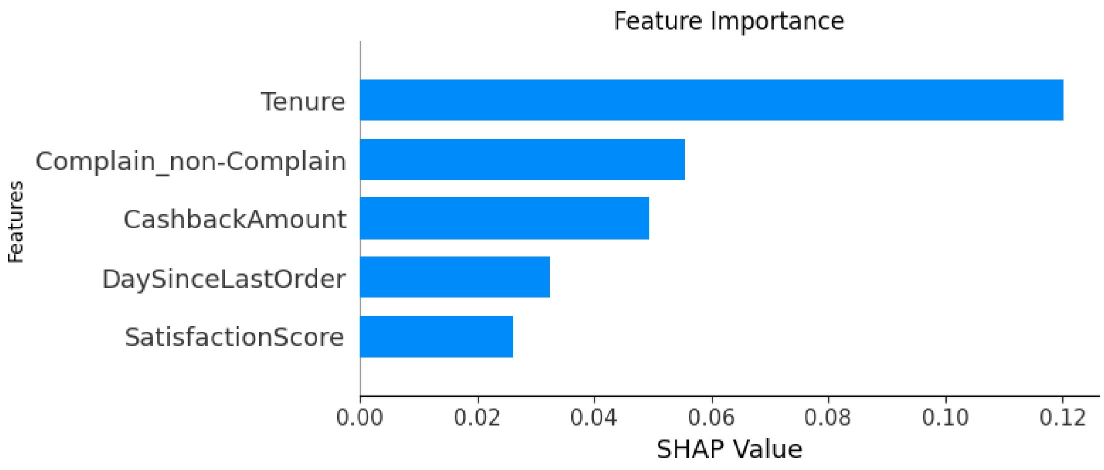
```
Recall Score XGB Default  tanpa feature selection:  0.9154929577464789
Recall Score XGB Tuned  tanpa feature selection:  0.8873239436619719
```

Final Report



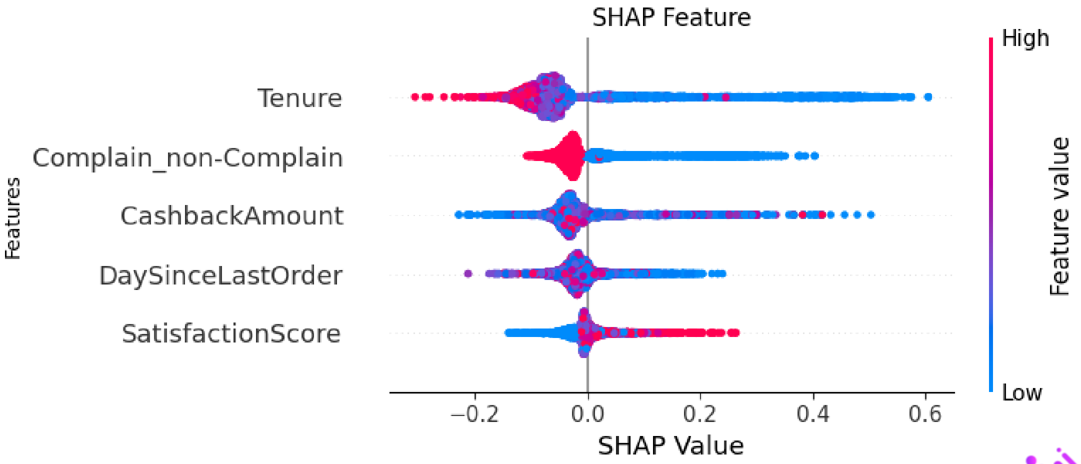
Classification Report Default XGB tanpa feature selection :

	precision	recall	f1-score	support
0	0.98	0.97	0.98	699
1	0.86	0.92	0.88	142
accuracy			0.96	841
macro avg	0.92	0.94	0.93	841
weighted avg	0.96	0.96	0.96	841



Feature Importance

SHAP Feature

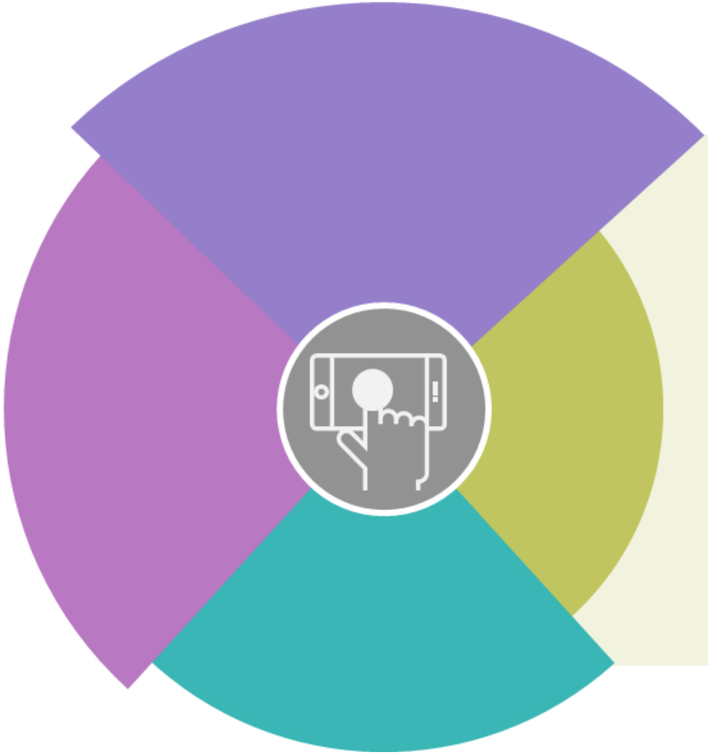


Feature Limitation

	min	max
Tenure	0.00	31.00
DaySinceLastOrder	0.00	18.00
CashbackAmount	110.09	324.99

	dataFeatures	uniqueSample
0	PreferredLoginDevice	[Mobile Phone, Computer]
1	CityTier	[Tier_1, Tier_3, Tier_2]
2	PreferredPaymentMode	[Debit-C & eWallet, Credit-C & UPI, COD]
3	NumberOfDeviceRegistered	[3-4_Devices, 5_Devices, 1-2_Devices, 6_Devices]
4	PreferedOrderCat	[Grocery, Fashion, Laptop, ACC & Others, Mobile Phone]
5	SatisfactionScore	[Neutral, Dissatisfied, Satisfied & more_than, Highly_Dissatisfied]
6	MaritalStatus	[Single, Married, Divorced]
7	Complain	[non-Complain, Complain]

Rule-Based vs Model



Churn Cost

40%

Increase

→

Model can predict

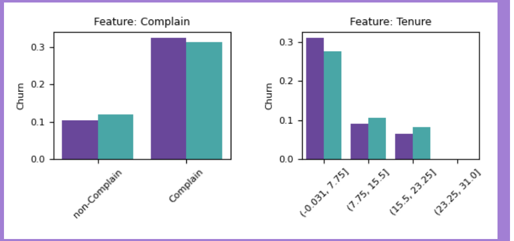
2.3x

More Churn than Rule-Based

Rule-Based and Model Simulation

Based on Data Test (*841 data*)

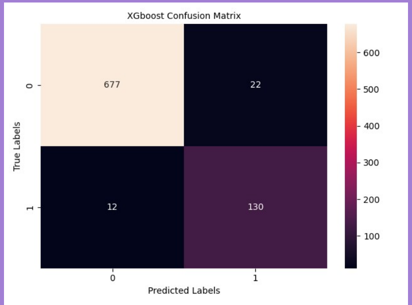
Rule-Based Simulation




```
# Cek logika dari data nilai negatif menggunakan csv
rule_based = data_test[(data_test['Tenure']<7.75) &
                       (data_test['Complain']=='Complain')]

print(f'Metode Pemberian Cashback pada Customer Churn dengan Metode RULE-BASED:\n')
print(f'Rule-Based Churn Cost: Rp{(rule_based.shape[0]*budget_perCust):,} atau sekitar {rule_based.shape[0]} Customer.')
print(f'Jumlah customer yang benar-benar churn dan tercover cashback: {rule_based["Churn"].value_counts()[1]} Customer')
print(f'Biaya yang terbuang karena salah memperkirakan customer churn: Rp{(rule_based.shape[0]-rule_based["Churn"].value_counts()
```

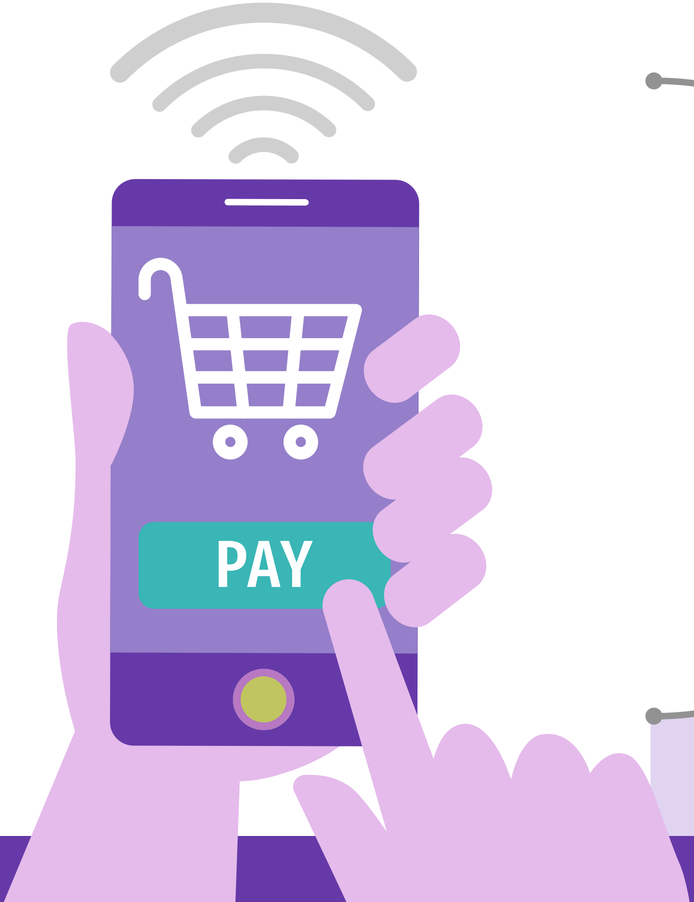
```
Metode Pemberian Cashback pada Customer Churn dengan Metode RULE-BASED:

Rule-Based Churn Cost: Rp5,400,000 atau sekitar 108 Customer.
Jumlah customer yang benar-benar churn dan tercover cashback: 56 Customer
Biaya yang terbuang karena salah memperkirakan customer churn: Rp2,600,000
```

Model Simulation




```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred_default_xgb2).ravel()

print(f'Metode Pemberian Cashback pada Customer Churn dengan MODELLING:\n')
print(f'Modelling Churn Cost: Rp{(budget_perCust*tp)+(budget_perCust*fp):,}')
print(f'Jumlah customer yang benar-benar churn dan tercover cashback: {tp} (recall 1 = {tp/(fn+tp):.2f})')
print(f'Total customer churn yang gagal diprediksi oleh model: {fn} (false negatif)')
print(f'Biaya yang terbuang Rp{budget_perCust*fp:,} karena salah memprediksi sebanyak {fp} customer (false positif)')
```

```
Metode Pemberian Cashback pada Customer Churn dengan MODELLING:

Modelling Churn Cost: Rp7,600,000
Jumlah customer yang benar-benar churn dan tercover cashback: 130 (recall 1 = 0.92)
Total customer churn yang gagal diprediksi oleh model: 12 (false negatif)
Biaya yang terbuang Rp1,100,000 karena salah memprediksi sebanyak 22 customer (false positif)
```



04

Recommendation

Apa rekomendasi bisnis yang dapat diberikan pada Stakeholder?

Business Recommendation

1. Implementing Machine Learning Models

2. Offering cashback/voucher

3. Other Promo Types

4. loyalty level scheme

5. Service improvements (CRM & Operational)

Business Recommendation

6.

Other category optimizations

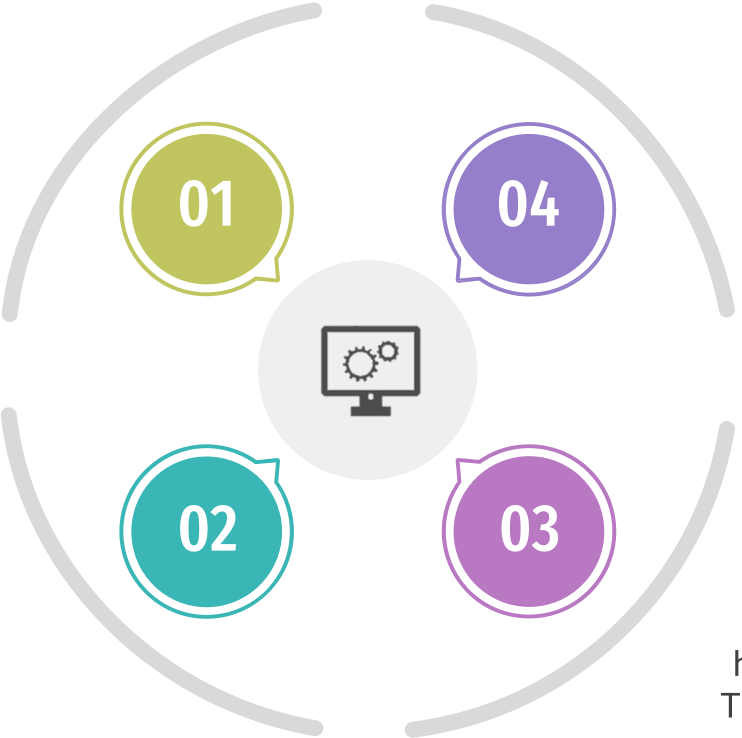
7.

Encourage investment to City Tier 3

8.

User Experience Improvements

Model Recommendation



01

Performa model mungkin dapat diperbaiki melalui pengujian kembali *hyperparam_space*

04

Perbaikan performa model mungkin dapat coba dilakukan melalui *tuning* pada 2 model teratas lainnya (*Decision Tree* dan *Adabost*)

02

Menambahkan beberapa fitur lain untuk meningkatkan akurasi model seperti *Last_Login, Total_purchase, Total_product_type_purchased*

03

Perbanyak data untuk kualitas hasil modelling yang lebih baik. Termasuk perbaiki kualitas data seperti mengurangi *missing values dan error labels*.

Thank you!