Rakamin Academy

Online Shoppers Purchasing Intention

Dokumen Laporan Final Project Stage 2 - Anaconda

Data Cleansing

Pada tahap data cleansing, akan diterapkan pembersihan data melalui metode-metode berikut:

1. Handle missing values
2. Handle duplicated data
3. Handle outliers
4. Feature transformation
5. Feature encoding
6. Handle class imbalance

Data Cleansing

1. Handle Missing Value

```
[ ]  df.isnull().sum().sum()

     0
```

Pengamatan

Sesuai hasil pengamatan di Stage sebelumnya,
Tidak ada missing values di setiap kolom.
Jumlah non-null baris sama dengan jumlah observasi.

Data Cleansing

2 Handle Duplicated Data

```
[3]: df.duplicated().sum()
```

```
[3]: 125
```

```
[5]: df[df.duplicated(keep=False)].reset_index(drop=True).head()    # show all duplicated data
```

[5]:

	administrative	administrative_duration	informational	informational_duration	product_related	product_related_duration	bounce_rates	exit_rates	page_values	sp
0	0	0.0	0	0.0	1	0.0	0.2	0.2	0.0	
1	0	0.0	0	0.0	1	0.0	0.2	0.2	0.0	
2	0	0.0	0	0.0	1	0.0	0.2	0.2	0.0	
3	0	0.0	0	0.0	1	0.0	0.2	0.2	0.0	
4	0	0.0	0	0.0	1	0.0	0.2	0.2	0.0	

```
[4]: df[df.duplicated(keep='last')]        # show last duplicated data
```

[4]:

	administrative	administrative_duration	informational	informational_duration	product_related	product_related_duration	bounce_rates	exit_rates	page_value
85	0	0.0	0	0.0	1	0.0	0.2	0.2	0.
132	0	0.0	0	0.0	1	0.0	0.2	0.2	0.
159	0	0.0	0	0.0	1	0.0	0.2	0.2	0.
252	0	0.0	0	0.0	2	0.0	0.2	0.2	0.
286	0	0.0	0	0.0	1	0.0	0.2	0.2	0.

Setelah diperiksa, dataset ini terdapat duplicated data sebanyak 125. Walaupun seluruh data merupakan data yang berbeda di setiap visitor, duplicated data harus tetap di drop karena model machine learning tidak dapat mendeteksi perbedaan setiap rows.

Split Dataset

```
[ ]  # set up the validation framework
     full_train, test = split_dataset(df, target='revenue', test_size=0.2)
```

```
[ ]  full_train[target].value_counts(normalize=True)
```

```
False    0.845296
True     0.154704
Name: revenue, dtype: float64
```

```
[ ]  test[target].value_counts(normalize=True)
```

```
False    0.845093
True     0.154907
Name: revenue, dtype: float64
```

Sebelum data preprocessing dan modelling, data di-split menjadi dua yaitu train set dan test set agar tidak terjadi data leakage. Rasio split yang digunakan 80% train set dan 20% test set. Kemudian agar distribusi target antara train dan test set tetap sama maka menggunakan stratified sampling.

Data Cleansing

3. Handling Outliers & Feature Transformation

Feature-feature numerik di dataset ini memiliki distribusi right-skewed sehingga terlihat jelas outliers yang ada. Namun, karena mayoritas data banyak yang bernilai 0, outliers tidak akan dihapus tetapi akan dikurangi jumlahnya dengan menggunakan log-transformer atau power-transformer. Berikut ini transformer yang akan digunakan.

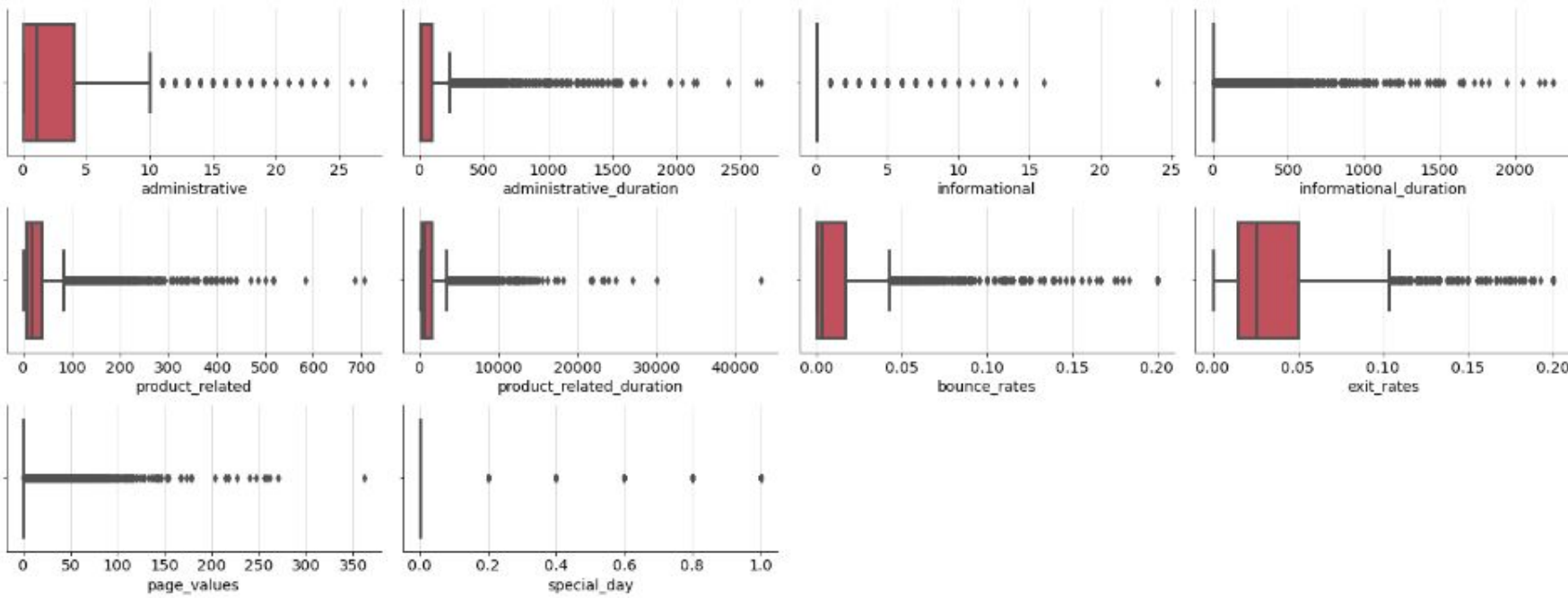
- np.log1p: log(x+1)
- yeo-johnson transformation

Referensi:
https://statisticaloddsandends.wordpress.com/2021/02/19/the-box-cox-and-yeo-johnson-transformations-for-continuous-variables/

Data Cleansing

3. Handling Outliers & Feature Transformation

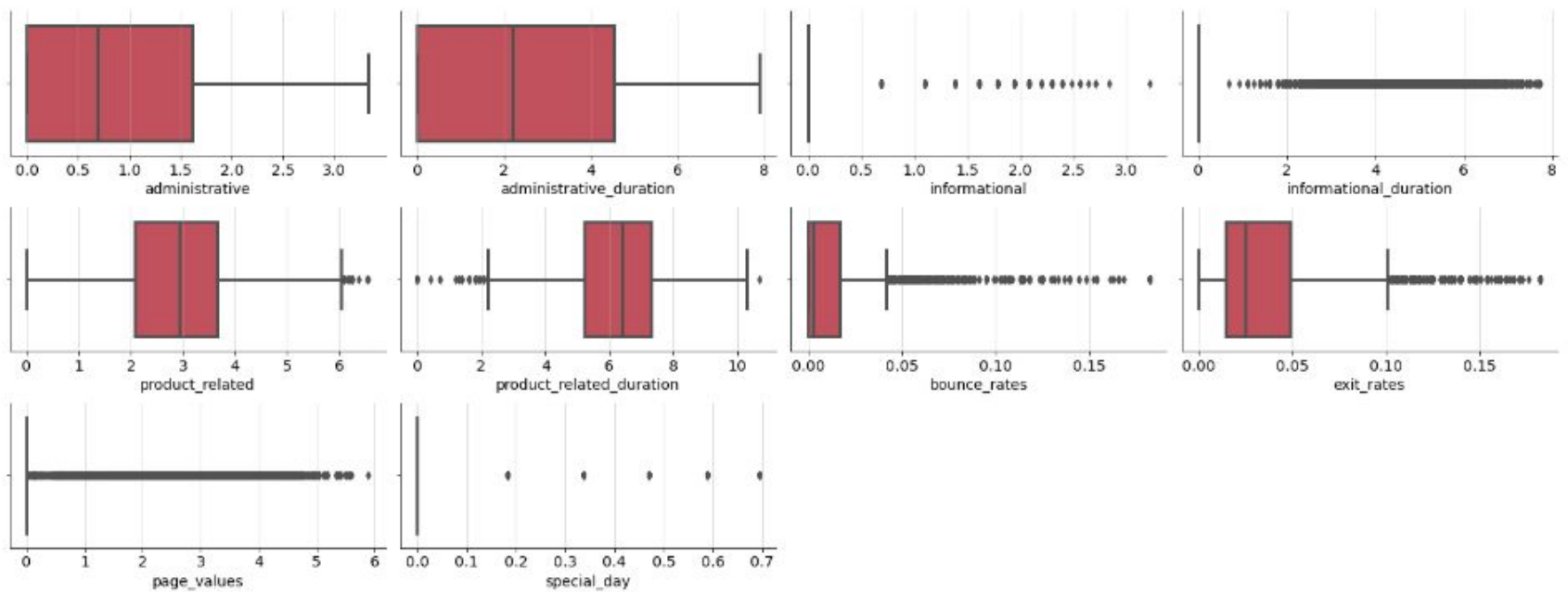
Berikut distribusi kolom numerik sebelum transformation.



Data Cleansing

3. Handling Outliers & Feature Transformation

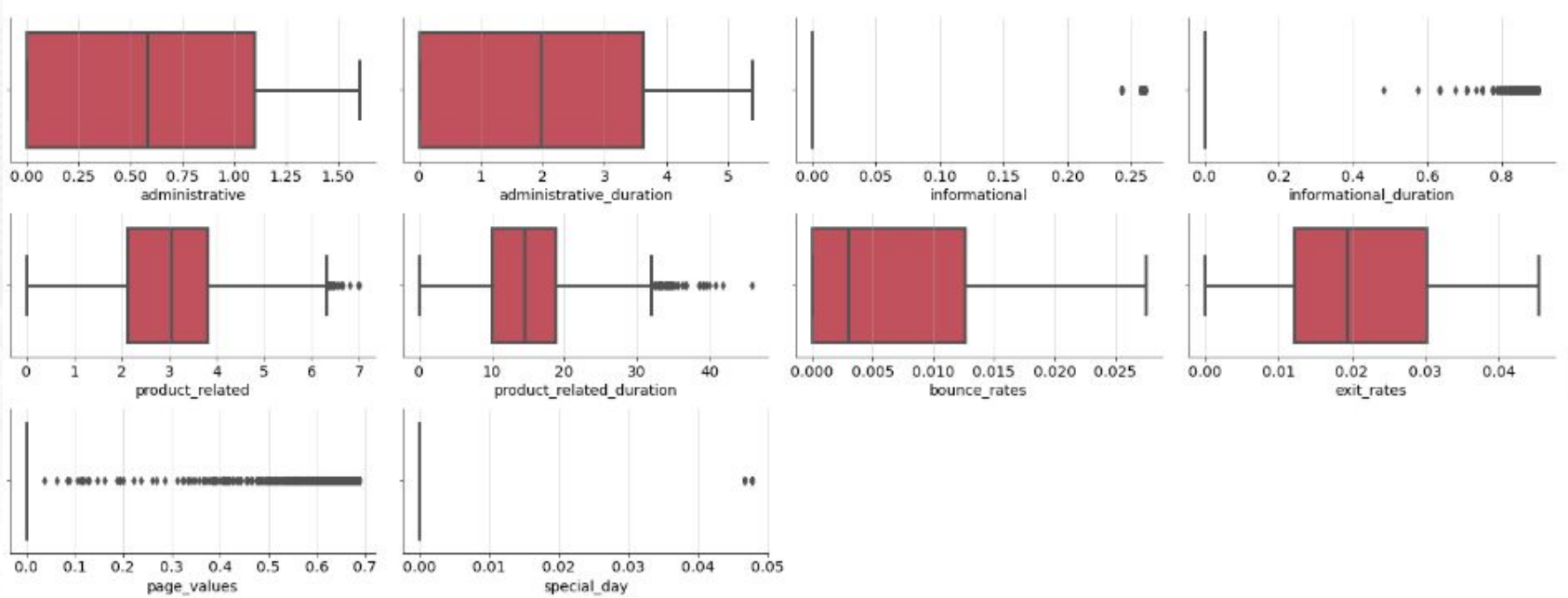
Berikut distribusi kolom numerik setelah diterapkannya log-transformation, Kolom yang outliernya berkurang atau bahkan hilang yaitu **administrative, administrative_duration, product_related, dan product_related_duration**. selain keempat kolom tersebut np.log1p transformer tidak memberi dampak apa-apa terhadap distribusi.



Data Cleansing

3. Handling Outliers & Feature Transformation

Berikut kondisi feature numerik sesudah transformation dengan yeo-johnson. Setelah diterapkan, kolom **bounce_rates** dan **exit_rates** tidak memiliki outlier lagi setelah yeo-johnson transformer.



Data Cleansing

3. Handling Outliers & Feature Transformation

Kesimpulan log-transformation

1. Kolom **administrative** dan a**dministrative_duration** tidak memiliki outliers setelah ditransformasi dengan **np.log1p** dan **yeo-johnson.**
2. Kolom **bounce_rates** dan **exit_rates** lebih cocok ditransformasi dengan **yeo-johnson** karena setelah transformasi outliers tidak ada.
3. Untuk kolom **product_related** dan **product_related_duration**, menggunakan kedua **transformer** tersebut dapat mengurangi outliers secara signifikan.

Data Cleansing

3. Handling Outliers & Feature Transformation

Kesimpulan transformation

4. Kolom **informational**, **informational_duration**, **page_values**, dan **special_day** tidak berubah distribusinya setelah transformation. Hal ini terjadi karena
    - Kolom **informational**, **informational_duration**, **page_values** memiliki data yang mayoritas tersebar di dekat nol sehingga saat ditransform dengan np.log1p atau yeo-johnson, nilai-nilai yang besar tetap jauh dari nilai 0.
    - Kolom **special_day** hanya memiliki 6 unique values yang mana lebih tepat di-treat sebagai kolom kategorikal. Namun, kolom special_day kemungkinan tidak memberi impak ke model karena 89% data memiliki nilai 0.
5. Jadi, karena setelah mencoba mentransform dengan kedua transformer tersebut hanya ada **empat feature yang outliers hilang**, maka untuk tahap modelling akan fokus menggunakan algoritma machine learning yang robust terhadap outlier seperti tree-based classifier dan ensemble classifier.

Data Cleansing

3. Handling Outliers & Feature Transformation

Normalisasi atau scaling

Apabila model yang digunakan linear seperti Logistic Regression, maka semua feature harus berada salah satu kondisi di bawah ini

1. **range yang sama** dengan menggunakan MinMaxScaler untuk menskala ulang feature numerik menjadi di range 0 hingga 1.
2. **memiliki mean 0 dan standar deviasi 1**. Kondisi ini dicapai dengan menggunakan StandardScaler.
3. jika terdapat banyak outliers, ada kemungkinan scaling menggunakan mean dan variance tidak terlalu membantu meningkatkan performa model. Oleh karena itu, scaling harus menggunakan **median dan IQR** dengan RobustScaler

Referensi:

https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler

Data Cleansing

3. Handling Outliers & Feature Transformation

Statistik deskriptif feature numerik sebelum normalisasi

	count	mean	std	min	25%	50%	75%	max
administrative	9764.0	2.348	3.328	0.0	0.000	1.000	4.000	27.000
administrative_duration	9764.0	82.420	180.052	0.0	0.000	10.000	94.835	3398.750
informational	9764.0	0.505	1.269	0.0	0.000	0.000	0.000	24.000
informational_duration	9764.0	35.634	144.572	0.0	0.000	0.000	0.000	2549.375
product_related	9764.0	32.235	44.923	0.0	8.000	19.000	39.000	705.000
product_related_duration	9764.0	1218.681	1970.936	0.0	197.200	619.783	1500.078	63973.522
bounce_rates	9764.0	0.020	0.045	0.0	0.000	0.003	0.016	0.200
exit_rates	9764.0	0.041	0.046	0.0	0.014	0.025	0.048	0.200
page_values	9764.0	5.893	18.413	0.0	0.000	0.000	0.000	361.764
special_day	9764.0	0.061	0.198	0.0	0.000	0.000	0.000	1.000

Data Cleansing

3. Handling Outliers & Feature Transformation

	count	mean	std	min	25%	50%	75%	max
administrative	9764.0	0.335	0.335	0.0	0.000	0.358	0.678	1.0
administrative_duration	9764.0	0.333	0.330	0.0	0.000	0.377	0.652	1.0
informational	9764.0	0.208	0.397	0.0	0.000	0.000	0.000	1.0
informational_duration	9764.0	0.193	0.389	0.0	0.000	0.000	0.000	1.0
product_related	9764.0	0.431	0.168	0.0	0.321	0.441	0.547	1.0
product_related_duration	9764.0	0.289	0.133	0.0	0.205	0.292	0.377	1.0
bounce_rates	9764.0	0.265	0.330	0.0	0.000	0.110	0.472	1.0
exit_rates	9764.0	0.473	0.261	0.0	0.272	0.431	0.657	1.0
page_values	9764.0	0.213	0.399	0.0	0.000	0.000	0.000	1.0
special_day	9764.0	0.101	0.300	0.0	0.000	0.000	0.000	1.0

Feature numerik setelah di-rescale dengan MinMaxScaler

Setelah ditransformasi dan dinormalisasi, seluruh feature berada di range 0 - 1 dan memiliki nilai tengah di antara 0 - 0.5.

Data Cleansing

3. Handling Outliers & Feature Transformation

Feature numerik setelah di-rescale dengan StandardScaler

	count	mean	std	min	25%	50%	75%	max
administrative	9764.0	-0.0	1.0	-1.002	-1.002	0.067	1.022	1.984
administrative_duration	9764.0	0.0	1.0	-1.010	-1.010	0.134	0.968	2.024
informational	9764.0	0.0	1.0	-0.524	-0.524	-0.524	-0.524	1.993
informational_duration	9764.0	-0.0	1.0	-0.495	-0.495	-0.495	-0.495	2.077
product_related	9764.0	-0.0	1.0	-2.570	-0.655	0.062	0.693	3.399
product_related_duration	9764.0	-0.0	1.0	-2.166	-0.626	0.025	0.657	5.332
bounce_rates	9764.0	-0.0	1.0	-0.804	-0.804	-0.471	0.627	2.226
exit_rates	9764.0	-0.0	1.0	-1.809	-0.768	-0.161	0.705	2.016
page_values	9764.0	-0.0	1.0	-0.535	-0.535	-0.535	-0.535	1.973
special_day	9764.0	-0.0	1.0	-0.335	-0.335	-0.335	-0.335	2.994

Data Cleansing

3. Handling Outliers & Feature Transformation

Feature numerik setelah di-rescale dengan RobustScaler

	count	mean	std	min	25%	50%	75%	max
administrative	9764.0	-0.033	0.494	-0.528	-0.528	0.0	0.472	0.948
administrative_duration	9764.0	-0.068	0.505	-0.578	-0.578	0.0	0.422	0.955
informational	9764.0	0.055	0.105	-0.000	-0.000	-0.0	0.000	0.265
informational_duration	9764.0	0.177	0.357	-0.000	-0.000	-0.0	0.000	0.918
product_related	9764.0	-0.046	0.742	-1.953	-0.532	0.0	0.468	2.476
product_related_duration	9764.0	-0.019	0.779	-1.707	-0.507	0.0	0.493	4.135
bounce_rates	9764.0	0.329	0.699	-0.232	-0.232	0.0	0.768	1.885
exit_rates	9764.0	0.109	0.679	-1.120	-0.413	0.0	0.587	1.478
page_values	9764.0	0.150	0.279	-0.000	-0.000	-0.0	0.000	0.701
special_day	9764.0	0.005	0.014	-0.000	-0.000	-0.0	0.000	0.047

Data Cleansing

4. Feature Encoding

Berikut encoding per masing-masing kolom kategorikal yang tepat digunakan yaitu.

1. **month**
   - **Sin-cos transformation**. Dengan transformer ini, hubungan siklus dapat diketahui oleh algoritma machine learning.
   - **Quarter binning**. Kolom month di-group berdasarkan quarter.
2. **operating_systems, browser, region, dan traffic_type**

   Keempat kolom tersebut memiliki jumlah unique value > 5 sehingga kurang tepat menggunakan one-hot encoder secara langsung. Berikut ini feature encoding yang tepat digunakan.
   - **Count encoder**. Jumlah observasi setiap unique value pada setiap kolom kategorikal
   - **Frequency encoder**. Persentase jumlah observasi setiap unique value pada setiap kolom kategorikal
   - **Target encoder**. Seluruh unique value dari kolom ini diubah menjadi fraksi jumlah visitor yang menghasilkan revenue per jumlah seluruh visitor.
   - **Rare-label dan one-hot encoder**. Sebagai contoh penggunaan rare-label, kategori 1, 2, dan 3 akan tetap dipertahankan dan selain ketiga kategori itu akan dijadikan satu kategori yaitu 'rare'. Kemudian keempat kategori tersebut di one-hot encoded.
3. **visitor_type, dan weekend**
   - karena unique value hanya 2-3 saja, maka **one-hot encoder** yang paling tepat diterapkan.

Data Cleansing

4. Feature Encoding (kolom month)

Sin-cos transformer

```
from feature_engine.creation import CyclicalFeatures

def cyclic_transformation(df):
    df = df.copy()

    sin_cos_transformer = CyclicalFeatures(variables=['month'])
    return sin_cos_transformer.fit_transform(df)[['month', 'month_sin', 'month_cos']].head()

cyclic_transformation(full_train)
```

	month	month_sin	month_cos
5551	6	1.224647e-16	-1.000000
8471	12	-2.449294e-16	1.000000
6563	11	-5.000000e-01	0.866025
5054	5	5.000000e-01	-0.866025
2730	5	5.000000e-01	-0.866025

Data Cleansing

4. Feature Encoding (kolom month)

Quarter Binning

```
from feature_engine.discretisation import ArbitraryDiscretiser

def quarter_binning(df):
    df = df.copy()
    month = df[['month']].copy()

    user_dict = {'month': [1, 3, 6, 9, 12]}

    transformer = ArbitraryDiscretiser(
        binning_dict=user_dict, return_object=False, return_boundaries=False)

    df = transformer.fit_transform(df)
    month['month_quarter'] = df['month'].copy() + 1

    return month

display(
    quarter_binning(full_train).head(),
    quarter_binning(full_train).describe().round(1)
)
```

	month	month_quarter
5551	6	2
8471	12	4
6563	11	4
5054	5	2
2730	5	2

	month	month_quarter
count	9764.0	9764.0
mean	7.7	2.8
std	3.4	1.2
min	2.0	1.0
25%	5.0	2.0
50%	8.0	3.0
75%	11.0	4.0
max	12.0	4.0

Data Cleansing

4. Feature Encoding (kolom operating_systems, browser, region, traffic_type)

Count Encoder

```
from feature_engine.encoding import CountFrequencyEncoder

def count_freq_encoder(df, cols, method='count'):
    df = df.copy()

    encoder = CountFrequencyEncoder(encoding_method=method, variables=cols)
    return encoder.fit_transform(df)

count_freq_encoder(full_train, cols, method='count')[cols].sample(5, random_state=SEED)
```

	operating_systems	browser	region	traffic_type
5738	5234	6365	928	1905
11816	5234	567	3802	275
10550	5234	567	599	3125
8626	5234	567	3802	3125
5562	2027	6365	633	1905

Data Cleansing

4. Feature Encoding (kolom operating_systems, browser, region, traffic_type)

Frequency Encoder

```
count_freq_encoder(full_train, cols, method='frequency')[cols].sample(5, random_state=SEED)
```

	operating_systems	browser	region	traffic_type
5738	0.536051	0.651884	0.095043	0.195104
11816	0.536051	0.058070	0.389390	0.028165
10550	0.536051	0.058070	0.061348	0.320053
8626	0.536051	0.058070	0.389390	0.320053
5562	0.207599	0.651884	0.064830	0.195104

Data Cleansing

4. Feature Encoding (kolom operating_systems, browser, region, traffic_type)

Target Encoder

```
import category_encoders
from category_encoders.target_encoder import TargetEncoder

def target_encoding(df, cols, target='revenue'):
    df = df.copy()

    target_encoder = TargetEncoder(
        cols=cols
    )
    return target_encoder.fit_transform(
        df[cols],
        df[target]
    )

target_encoding(full_train, cols)[cols].sample(5, random_state=SEED)
```

	operating_systems	browser	region	traffic_type
5738	0.178258	0.154753	0.149784	0.111286
11816	0.178258	0.181658	0.161494	0.287273
10550	0.178258	0.181658	0.166945	0.214720
8626	0.178258	0.181658	0.161494	0.214720
5562	0.102615	0.154753	0.139021	0.111286

Data Cleansing

4. Feature Encoding (kolom operating_systems, browser, region, traffic_type)

Rare-label encoder

```
from feature_engine.encoding import RareLabelEncoder

def rare_label(df, cols):
    df = df.copy()
    df[cols] = df[cols].astype(str)

    encoder = RareLabelEncoder(
        replace_with='rare',
        n_categories=3,
        max_n_categories=5,
        variables=cols
    )
    return encoder.fit_transform(df)

rare_label(full_train, cols)[cols].sample(5, random_state=SEED)
```

	operating_systems	browser	region	traffic_type
5738	2	2	4	1
11816	2	4	1	rare
10550	2	4	rare	2
8626	2	4	1	2
5562	3	2	6	1

Applying one-hot encoder after rare-label encoding

```
from feature_engine.encoding import OneHotEncoder

def one_hot(df, cols):
    df = df.copy()

    encoder = OneHotEncoder(variables=cols)
    return encoder.fit_transform(df)

temp_df = rare_label(full_train, cols).copy()
one_hot(temp_df, cols).iloc[:, 14:].head()
```

	operating_systems_1	operating_systems_2	operating_systems_3	operating_systems_rare	
5551	1	0	0	0	
8471	0	1	0	0	
6563	0	1	0	0	
5054	0	1	0	0	
2730	0	1	0	0	

Data Cleansing

4. Feature Encoding (kolom visitor_type, weekend)

One Hot Encoder

```
from feature_engine.encoding import OneHotEncoder

cols = ['visitor_type', 'weekend']
one_hot(full_train, cols).iloc[:, -4:].head()
```

	visitor_type_New_Visitor	visitor_type_Other	weekend_True	weekend_False
5551	0	0	1	0
8471	0	0	0	1
6563	0	0	0	1
5054	0	0	0	1
2730	0	0	0	1

Catatan untuk data preprocessing

Rakamin Academy
School for Career Acceleration

Ada beberapa catatan untuk data preprocessing

1. Seluruh data preprocessing dari handle data duplikasi hingga feature encoding, data preprocessing hanya di-apply ke data train saja karena masih tahap eksperimen.
2. Agar tidak ada data leakage yaitu informasi dari data test ‘bocor’ ke model yang dibuat dari data train, maka harus menggunakan Pipeline untuk data preprocessing. Manfaat lain Pipeline, agar tambahan bias dalam penggunaan resampling dapat berkurang.
3. Data preprocessing seperti feature transformation dan feature scaling tidak perlu digunakan untuk semua model classifier karena hanya model linear dan berbasis jarak yang butuh feature transformation & scaling, sedangkan tree-based / ensemble tidak perlu karena variasi data range tidak berpengaruh dan robust terhadap outlier.
4. Oleh karena itu, untuk tahap modelling, pipeline untuk linear dan tree-based model akan dipisah.

Data Cleansing

5. Handle Class Imbalance

Karena terdapat class imbalance dengan ratio 85:15 (majority class: minority class) maka ada beberapa cara yang harus digunakan sebagai berikut

- Melakukan resampling (oversampling, undersampling, or over-undersampling) agar target class ratio menjadi balance yaitu 50:50. Namun, sebelum resampling, dataset di-split terlebih dahulu menjadi train dan test dengan ratio 80% train dan 20% test. Kemudian, resampling hanya diterapkan ke train set saja supaya performa model tidak over-optimistic.
- Memberi class weight lebih besar pada minority class dari kolom target.
- Menggunakan metrik evaluasi model yang tepat seperti ROC-AUC atau f1-score.

Referensi

Common pitfalls and recommended practice, imblearn

Fitting model on imbalanced datasets, imblearn

Data Cleansing

5. Handle Class Imbalance (Resampling)

Berikut ini algoritma resampler yang digunakan.

1. Oversampling dengan SMOTE

2. Undersampling dengan RandomUnderSampler

3. Over-undersampling dengan SMOTETomek

Untuk mencontohkan hasil resampling, menggunakan pipeline agar menghindari data leakage. Untuk kasus ini, pipeline yang digunakan untuk model linear (Logistic Regression). Pipeline terdiri dari feature transformation, feature scaling, feature encoding, resampling, dan model classifier. Sebagai contoh, berikut ini data preprocessing yang digunakan.

- Numerical feature transformation: log-transformation
- Numerical feature scaling: RobustScaler
- Categorical feature encoding: OneHotEncoder

Karena setiap algoritma resampling memiliki kekurangan seperti mengurangi informasi yang ada (undersampling) atau menambahkan data yang merubah distribusi (oversampling). Agar informasi dari dataset tidak ada yang di-drop maka untuk tahap modelling akan menggunakan SMOTE.

Data Cleansing

5. Handle Class Imbalance (Resampling)

Over-sampling by using SMOTE

```
handle_imbalance(df, res_name='SMOTE')
```

```
Train size before resampling:
False    8238
True     1526
Name: revenue, dtype: int64

Train size after resampling with SMOTE:
 True     8238
False    8238
Name: revenue, dtype: int64
```

Under-sampling by using RandomUnderSampler

```
handle_imbalance(df, res_name='RandomUnderSampler')
```

```
Train size before resampling:
False    8238
True     1526
Name: revenue, dtype: int64

Train size after resampling with RandomUnderSampler:
 True     1526
False    1526
Name: revenue, dtype: int64
```

Over-undersampling by using SMOTETomek

```
handle_imbalance(df, res_name='SMOTETomek')
```

```
Train size before resampling:
False    8238
True     1526
Name: revenue, dtype: int64

Train size after resampling with SMOTETomek:
 True     8222
False    8222
Name: revenue, dtype: int64
```

Feature Engineering

1. Feature Selection

Feature selection bisa berdasarkan:

1. Konteks

Memilih atau men-drop feature berdasarkan domain knowledge. Secara umum, seluruh feature di dataset dapat digunakan untuk modelling. Namun, karena keempat feature **operating_systems, browser, region, dan traffic_type** sudah di-encode menjadi integer tanpa ada informasi tambahan apapun, maka feature tersebut bisa di-drop agar tidak menyebabkan kesalahan interpretasi hasil modelling.

2. Hasil EDA

- Berdasarkan descriptive statistics, kolom `visitor_type` dan `special_day` dapat di-drop karena >80% hanya untuk satu kategori saja.
- Berdasarkan analisis korelasi, kolom `region` dapat di-drop karena tidak berkorelasi dengan target (revenue)
- Antar feature yang berkorelasi tinggi seperi pasangan feature dibawah ini, akan di-drop salah satu feature yang berkolerasi tinggi dari setiap pasangan feature:
  - **`administrative`** dan **`administrative_duration`**
  - **`informational`** dan **`informational_duration`**
  - **`product_related`** dan **`product_related_duration`**

Feature Engineering

2. Feature Extraction

Ada tiga feature baru yang dapat dibuat dari feature-feature yang ada yaitu

1. rata-rata administrative duration per page: **administrative_duration** / **administrative**
2. rata-rata informational duration per page: **informational_duration** / **informational**
3. rata-rata product-related duration per page: **product_related_duration** / **product_related**

Beberapa feature tambahan yang dapat membuat performa model semakin bagus yaitu

- **Time to purchase**
  Sebuah metrics yang menunjukkan waktu yang dibutuhkan visitor sejak pertama kali masuk ke website kita sampai dilakukannya transaksi. Mengetahui metrics ini membantu perusahaan dalam pengambilan keputusan untuk menyusun strategi marketing, misalnya untuk set up email marketing journey map yang disesuaikan dengan umur visitor (sejak pertama kali visit).

- **Average transaction Value**
  Yaitu jumlah amount rata-rata transaksi yang dilakukan customer di ecommerce kita (satuan uang). Feature ini penting untuk menghitung dan mengestimasi ROI yang didapatkan terhadap beban marketing dan sales.

Feature Engineering

2. Feature Extraction

- **Cost per conversion**
  Yaitu biaya yang dibutuhkan untuk menghasilkan sebuah transaksi. Feature ini dapat dibilang cukup critical, digunakan agar kita dapat memaintain dengan baik biaya yang dikeluarkan sehingga menghasilkan ROI yang optimal.

- **Page per session**
  Yaitu jumlah page yang dikunjungi dalam satu sesi sebelum meninggalkan website kita. Metrics ini membantu sebagai acuan seberapa menarik dan engaging website/ecommerce kita. Dengan adanya feature ini, dapat melihat apakah semakin banyak page yang dilihat akan semakin berpotensi menghasilkan revenue atau tidak.

- **Average session duration**
  Adalah metrics yang memperlihatkan berapa waktu rata-rata yang dibutuhkan user dalam satu sesi. Metrics ini dapat membantu untuk melihat apakah ada aktivitas tidak normal. Misal dengan durasi per sesi yang cukup panjang hingga berjam-jam tapi tanpa ada transaksi. Hal ini juga bisa jadi salah satu acuan dalam mengambil keputusan dalam strategi marketing.

Feature Engineering

2. Feature Extraction

- **Shopping Cart Abandonment**
  Feature ini menunjukkan seberapa banyak atau sering seorang visitor melakukan proses memasukkan produk ke shopping cart tanpa benar-benar melakukan transaksi. Feature ini juga cukup penting untuk melihat apakah ada kesulitan atau interface yang kurang bersahabat pada proses checkout.

Reference:

https://guidingmetrics.com/content/ecommerce-industry-most-critical-metrics-kpis/

https://databox.com/what-is-pages-per-session-and-how-do-i-improve-it

Kontribusi

1. Handling duplikasi, handling outlier & feature scaling: Muhammad Irfan Fadhlurrahman & Ni Putu Tasya
2. Feature encoding: Ramado Dipradelana I & Bima Sandi
3. Handle imbalance: Muhammad Irfan Fadhlurrahman, Deni Indra Permana, Ni Putu Tasya
4. Feature selection: Deni Indra Permana
5. Feature extraction: Muhammad Irfan Fadhlurrahman
6. Additional features: Ni Putu Tasya
7. Laporan: Semuanya