Rakamin Academy

INFINITY

Anggota Kelompok 8:

- Andre Adeputra S
- Gilang Muhammad Risky
- Jomas Sekar Pawestri
- M Nurkholis Fauzi
- Muhammad Syarif U
- Naomi Florenata Damanik
- Sakinah Nurul R
- Vanesa

Latar Belakang Masalah

Problem statement :

Perusahaan Trips&Travel di bidang Travel ingin membangun model bisnis yang layak untuk memperluas basis pelanggan. Salah satu cara yang akan dilakukan adalah dengan menawarkan jenis paket liburan yang baru. Namun, biaya pemasaran yang dikeluarkan cukup tinggi karena pelanggan dihubungi secara acak. Sedangkan berdasarkan data tahun lalu, conversion rate pelanggan yang membeli produk dari perusahaan hanya 18%. Oleh karena itu, perusahaan ini ingin menggunakan data yang tersedia untuk menyasar pelanggan yang potensial untuk menurunkan biaya pemasaran (marketing cost)

Role :

Kami dari Infinity Solutions merupakan agensi yang bergerak di bidang data untuk memberi solusi bisnis dari perusahaan klien. Peran kami di sini adalah untuk memberi rekomendasi kepada tim marketing dari klien melalui penggalian data-data marketing yang telah mereka lakukan sebelumnya.

Latar Belakang Masalah

Goal :

Efisiensi target marketing

Objective :

Analisis data customer yang memungkinkan membeli paket perjalanan yang baru diperkenalkan

Menentukan fitur mana saja yang paling signifikan

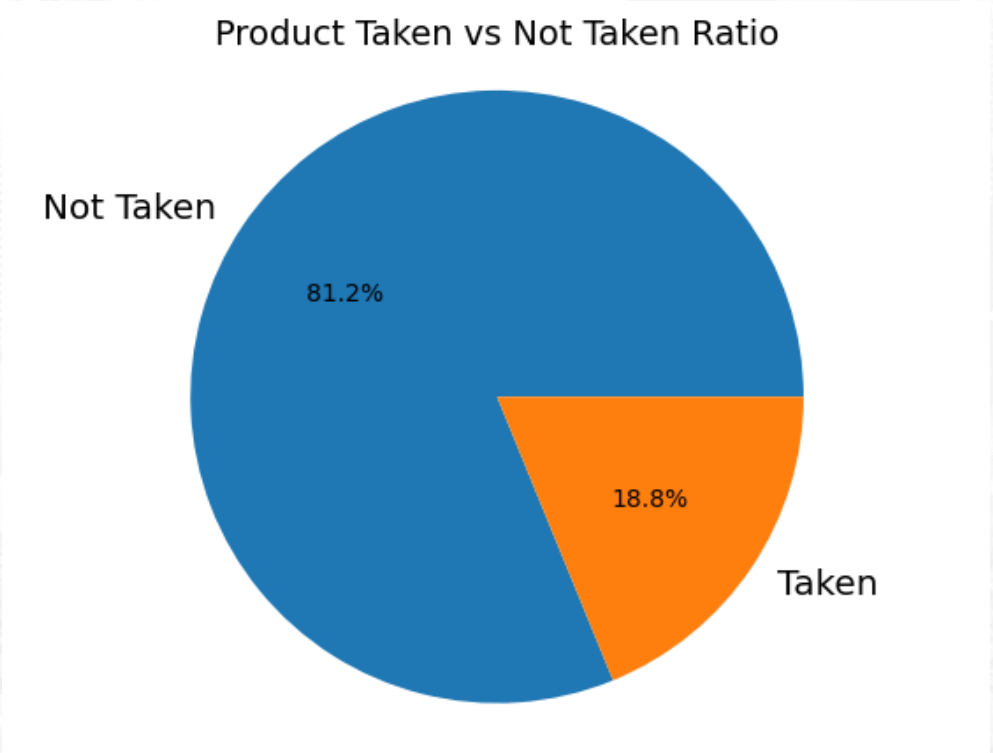
Membuat machine learning untuk memprediksi pelanggan potensial

Membuat rekomendasi target konsumen kepada tim marketing untuk diprioritaskan

Business Metrics :

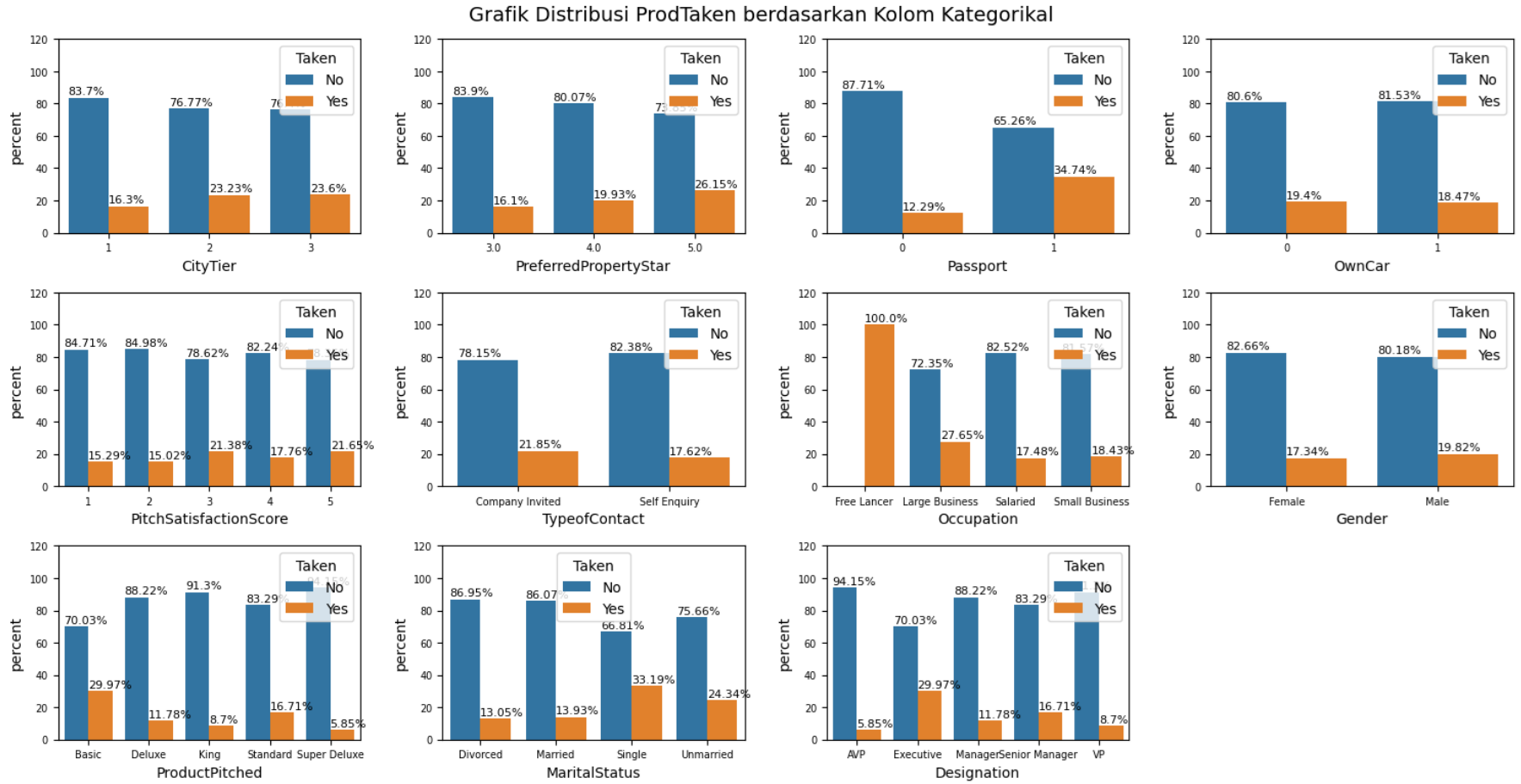
Conversion Rate meningkat hingga 25%

Data Eksplorasi

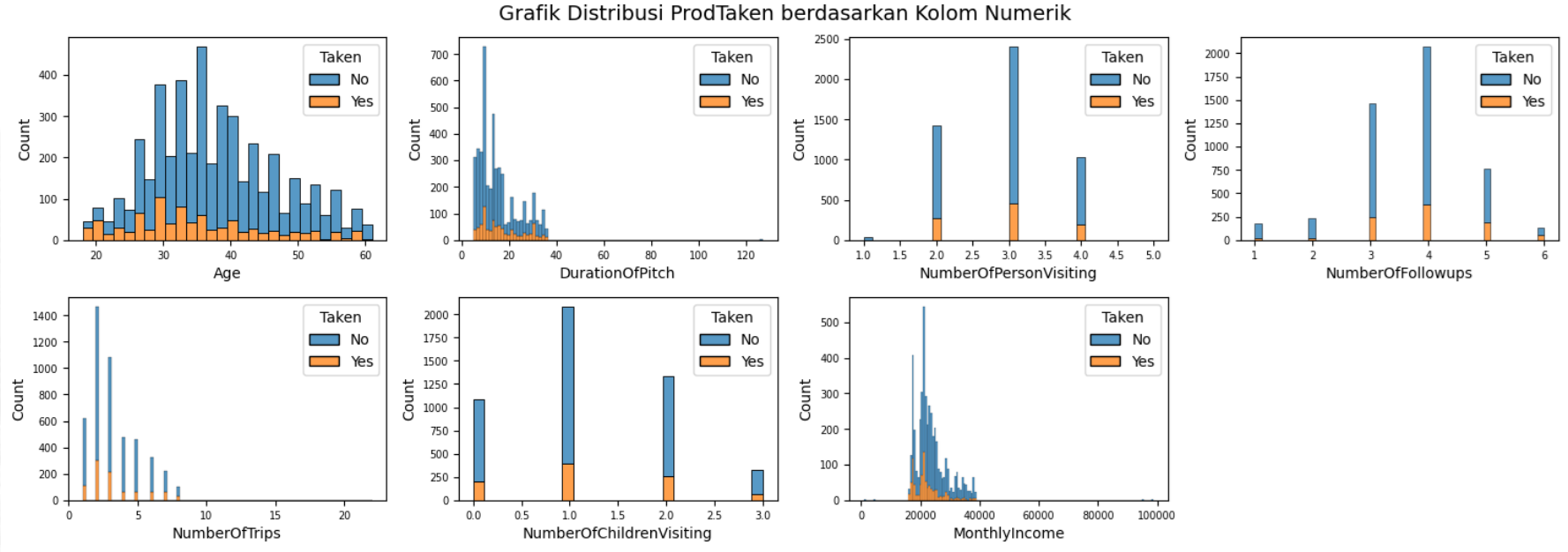


Dari hasil observasi data ditemukan hanya 18.8 % pelanggan yang tertarik untuk membeli produk liburan yang ditawarkan

Data Eksplorasi



Data Eksplorasi



Data Eksplorasi

Insight:

- Pada fitur Passport, sebesar 34% pelanggan dengan paspor memutuskan untuk membeli produk, dan hanya 12% pelanggan tanpa paspor yang memutuskan untuk membeli produk yang ditawarkan.
- Pada fitur PreferredPropertyStar, sebesar 26% pelanggan yang memilih properti bintang 5 memutuskan untuk membeli produk yang ditawarkan.
- Pada fitur ProductPitched, sebesar hampir 30% pelanggan yang memilih Basic memutuskan untuk membeli produk yang ditawarkan.
- Pada fitur MaritalStatus, 33% dari pelanggan yang masih single dan 24% pelanggan Unmarried memutuskan untuk membeli produk yang ditawarkan
- Pada fitur Designation, conversion rate tertinggi pada jabatan executive sebesar hampir 30%
- Pada fitur Age, pelanggan yang memutuskan untuk membeli produk yang ditawarkan cukup banyak yang berumur sekitar 26-40 tahun
- Pada fitur DurationOfPitch, pelanggan yang membeli produk lebih banyak berasal dari pelanggan yang diberikan penawaran selama kurang dari 20 menit
- Pada fitur MonthlyIncome, pelanggan yang membeli produk lebih banyak berasal dari pelanggan dengan kisaran gaji 15,000–25,000 USD

Rekomendasi Bisnis

Kami dari Infinity Solutions ingin merekomendasikan perusahaan Trips&Travel untuk melakukan efisiensi target marketing dengan berfokus pada calon pelanggan yang berpotensi untuk membeli paket perjalanan baru yang ditawarkan, yaitu pelanggan yang memiliki paspor, memilih properti bintang 5, memilih paket Basic, pelanggan yang masih single dan berstatus unmarried, memiliki jabatan executive, berusia sekitar 26-40 tahun, dan pelanggan yang diberikan penawaran selama kurang dari 20 menit. Pelanggan yang membeli produk lebih banyak berasal dari pelanggan dengan kisaran gaji 15,000–25,000 USD. Karakteristik pelanggan tersebut dapat terlebih dahulu diberikan penawaran (diprioritaskan) sebelum ditawarkan kepada pelanggan lain.

Rekomendasi Pre-Processing untuk EDA

1. Descriptive Analysis

- Handling missing value
- Cek klasifikasi di fitur gender
- Persentase klasifikasi

2. Univariate Analysis

- Handling outlier dari fitur DurationOfPitch, NumberOfTrips, dan MonthlyIncome
- Handling imbalance data fitur ProdTaken
- Handling redundant pada fitur Gender dan MaritalStatus

3. Multivariate Analysis

- Kolom NumberOfPersonVisiting dan NumberOfChildrenVisiting memiliki korelasi positif cukup kuat sehingga bisa digabung menjadi fitur kolom baru
- Fitur yang dipertahankan yaitu Passport, PreferredPropertyStar, ProductPiched, MaritalStatus, Designation, Age, DurationOfPitch

Github

Link repository github:
https://github.com/andreadeputra/Rakamin_Final_Project

Data Cleansing

```
df1 = df.copy()

# Conditional to avoid errors on repeated cell runs
if 'CustomerID' in df1.columns:
    df1.drop('CustomerID', axis=1, inplace=True)
if 'CustomerID' in nums:
    nums.remove('CustomerID')
df1.replace(['Divorced', 'Unmarried'], 'Single', inplace=True)
df1.replace(['Senior Manager', 'AVP'], ['Manager', 'VP'], inplace=True)
df1.drop(df1[df1.Occupation == 'Free Lancer'].index, inplace=True)
```

```
fill_modus = ['TypeofContact', 'PreferredPropertyStar', 'NumberOfFollowups',
              'NumberOfTrips', 'NumberOfChildrenVisiting']
fill_median = ['DurationOfPitch', 'MonthlyIncome']
fill_mean = ['Age']

for col in fill_modus:
    df1[col].fillna(df[col].mode()[0], inplace=True)

for col in fill_median:
    df1[col].fillna(df[col].median(), inplace=True)

for col in fill_mean:
    df1[col].fillna(df[col].mean(), inplace=True)

df1.info()
```

Handling missing data terdiri dari 3:

- Fill modus untuk data tipe kontak, preferred property star, number of follow ups, number of trips dan number of children visiting
- Fill median untuk duration of pitch dan monthly income karena datanya skewed
- Fill mean untuk data umur karena distribusinya normal

Data Cleansing

Data yang akan di handle sebelumnya dilakukan sebagai berikut:

- Kolom CustomerID didrop terlebih dahulu karena data berupa index dan bukan fitur
- Kolom Gender yang memiliki 3 nilai karena typo Female dan Fe Male, telah diubah menjadi bernilai Male atau Female saja
- Kolom Marital Status disederhanakan menjadi Single dan Married, dengan Single mencakup Divorced dan Unmarried
- Kolom Designation akan disederhanakan dengan meleburkan Senior Manager ke dalam Manager dan meleburkan AVP ke dalam VP
- Baris customer yang memiliki Occupation 'Free Lancer' didrop karena hanya terdapat 2 baris data

Data Cleansing

```
df1.drop_duplicates(inplace=True)
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4745 entries, 0 to 4887
Data columns (total 19 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ProdTaken                 4745 non-null   int64
 1   Age                       4745 non-null   float64
 2   TypeofContact             4745 non-null   object
 3   CityTier                  4745 non-null   int64
 4   DurationOfPitch           4745 non-null   float64
 5   Occupation                4745 non-null   object
 6   Gender                    4745 non-null   object
 7   NumberOfPersonVisiting    4745 non-null   int64
 8   NumberOfFollowups         4745 non-null   float64
 9   ProductPitched            4745 non-null   object
 10  PreferredPropertyStar     4745 non-null   float64
 11  MaritalStatus             4745 non-null   object
 12  NumberOfTrips             4745 non-null   float64
 13  Passport                  4745 non-null   int64
 14  PitchSatisfactionScore    4745 non-null   int64
 15  OwnCar                    4745 non-null   int64
 16  NumberOfChildrenVisiting  4745 non-null   float64
 17  Designation               4745 non-null   object
 18  MonthlyIncome             4745 non-null   float64
dtypes: float64(7), int64(6), object(6)
```

```
[ ]  dupes = df1.duplicated()
     print(f'Total Duplicate Rows: {dupes.sum()}')
     dupes_percent = round(df1[dupes].shape[0] / df1.shape[0] * 100, 1)
     print(f'Rasio baris data duplikat adalah {dupes_percent}% dari total baris')

     Total Duplicate Rows: 141
     Rasio baris data duplikat adalah 2.9% dari total baris
```

Handling duplicate data:

Duplikat data yang ditemukan sebanyak 141 rows dan didrop dengan df.drop_duplicate (inplace=True)

Data Cleansing

```
# untuk drop outlier berdasar zscore
df1.drop(df_zscore.dropna(how='all').index, inplace=True)
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4735 entries, 0 to 4887
Data columns (total 19 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ProdTaken                 4735 non-null   int64
 1   Age                       4735 non-null   float64
 2   TypeofContact             4735 non-null   object
 3   CityTier                  4735 non-null   int64
 4   DurationOfPitch           4735 non-null   float64
 5   Occupation                4735 non-null   object
 6   Gender                    4735 non-null   object
 7   NumberOfPersonVisiting    4735 non-null   int64
 8   NumberOfFollowups         4735 non-null   float64
 9   ProductPitched            4735 non-null   object
 10  PreferredPropertyStar     4735 non-null   float64
 11  MaritalStatus             4735 non-null   object
 12  NumberOfTrips             4735 non-null   float64
 13  Passport                  4735 non-null   int64
 14  PitchSatisfactionScore    4735 non-null   int64
 15  OwnCar                    4735 non-null   int64
 16  NumberOfChildrenVisiting  4735 non-null   float64
 17  Designation               4735 non-null   object
 18  MonthlyIncome             4735 non-null   float64
dtypes: float64(7), int64(6), object(6)
```

```
[ ]  # Untuk melihat nilai apa saja yang dianggap outlier oleh zscore
     df_zscore.dropna(how='all').dropna(how='all', axis=1).replace(np.nan, '-')
```

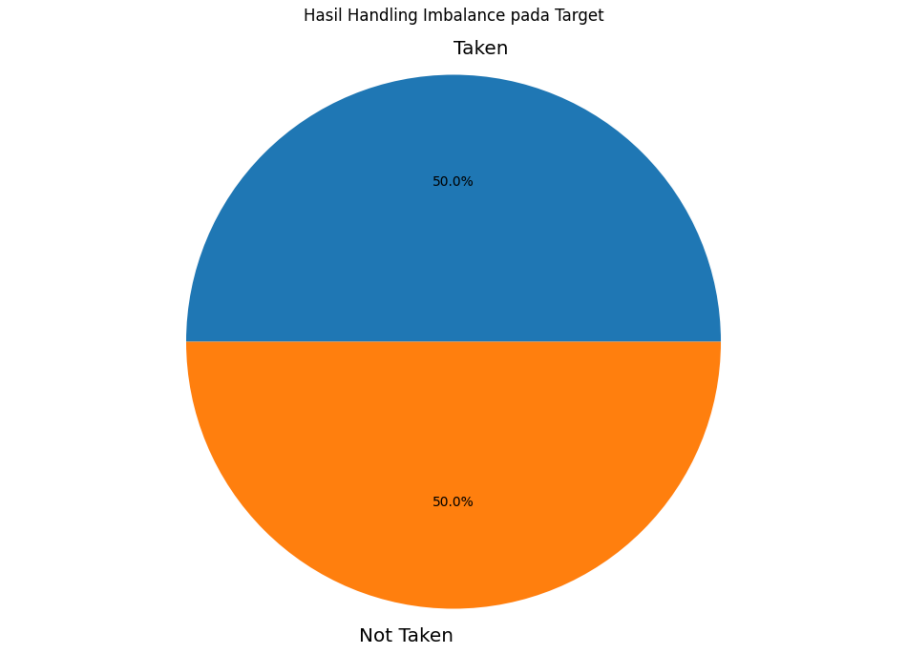
	DurationOfPitch	NumberOfTrips	MonthlyIncome
38	-	-	95000.0
142	-	-	1000.0
385	-	19.0	-
816	-	21.0	-
1434	126.0	-	-
2482	-	-	98678.0
2586	-	-	4678.0
2829	-	20.0	-
3260	-	22.0	-
3878	127.0	-	-

Handling outlier:

Data outlier yang ekstrim dihandle dengan metode IQR dan Z-Score. Namun, yang digunakan adalah metode z-score yang dianggap sudah cukup untuk mendeteksi nilai yang sangat ekstrim sesuai yang ditunjukkan oleh tabel di atas. Karena, dengan metode IQR dikhawatirkan akan membuang terlalu banyak data.

Data Cleansing

```
[ ]  # pemisahan features vs target
     X = df1.drop(['ProdTaken'], axis = 1)
     y = df1[['ProdTaken']]
```



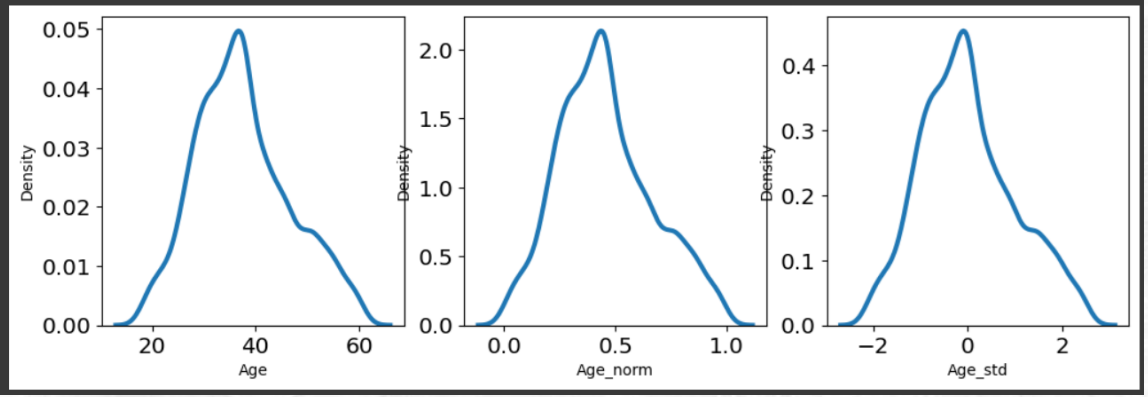
Handling Imbalance data:

- Handling imbalance data menggunakan SMOTE oversampler untuk memperkenalkan data baru secara sintetik.
- Karena keterbatasan jumlah data, undersampling tidak dilakukan untuk menangani ketimpangan data.

Feature Engineering

```
[ ]  df1['Age_norm'] = MinMaxScaler().fit_transform(df1['Age'].values.reshape(len(df1), 1))
     df1['Age_std'] = StandardScaler().fit_transform(df1['Age'].values.reshape(len(df1), 1))
     Age1 = ['Age','Age_norm','Age_std']
     df1[Age1].describe()
```

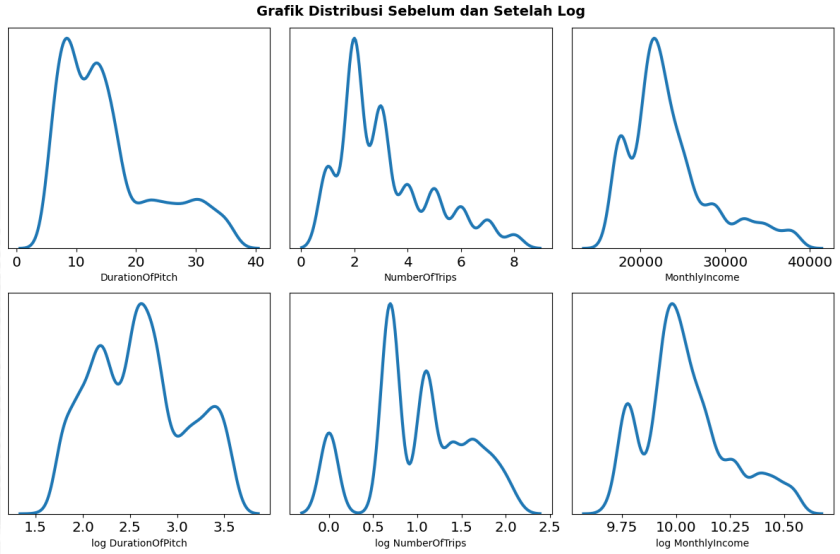
```
[ ]  for i in range(len(Age1)):
         plt.subplot(2,3,i+1)
         sns.kdeplot(df1[Age1[i]])
```



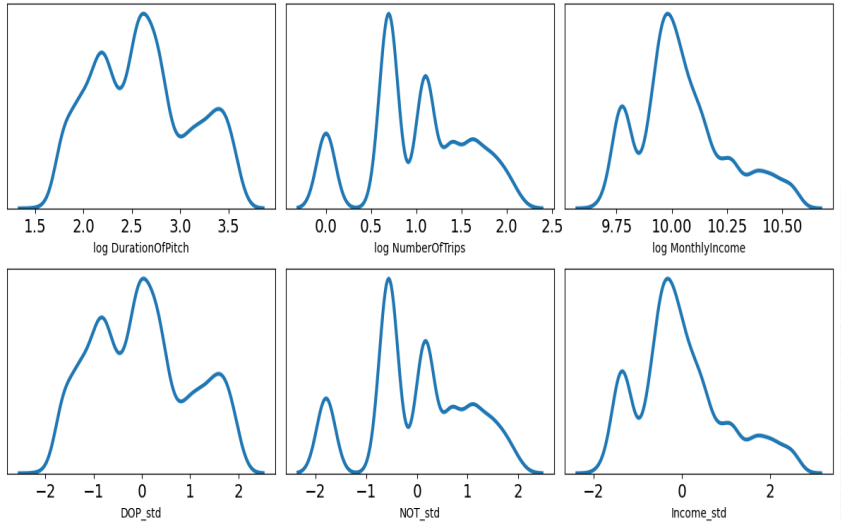
Feature Transformation:

- Age menggunakan transformasi standardisasi, karena hasil transformasi memiliki bentuk sebaran yang relatif sama.

Feature Engineering



Log transformation



Lanjut transformasi standardisasi

Feature Transformation:

- Kolom dengan distribusi bersifat skew, dilakukan transformasi log terlebih dahulu sebelum standardisasi

Feature Engineering

```
[ ]  le = LabelEncoder()
     binary_encode = ['TypeofContact', 'Gender', 'MaritalStatus']

     for col in binary_encode:
         df1[col] = le.fit_transform(df1[col])
         mapping = dict(zip(le.classes_, le.transform(le.classes_)))
         print(f'Hasil encode pada kolom {col}: {mapping}')
```

```
Hasil encode pada kolom TypeofContact: {'Company Invited': 0, 'Self Enquiry': 1}
Hasil encode pada kolom Gender: {'Female': 0, 'Male': 1}
Hasil encode pada kolom MaritalStatus: {'Married': 0, 'Single': 1}
```

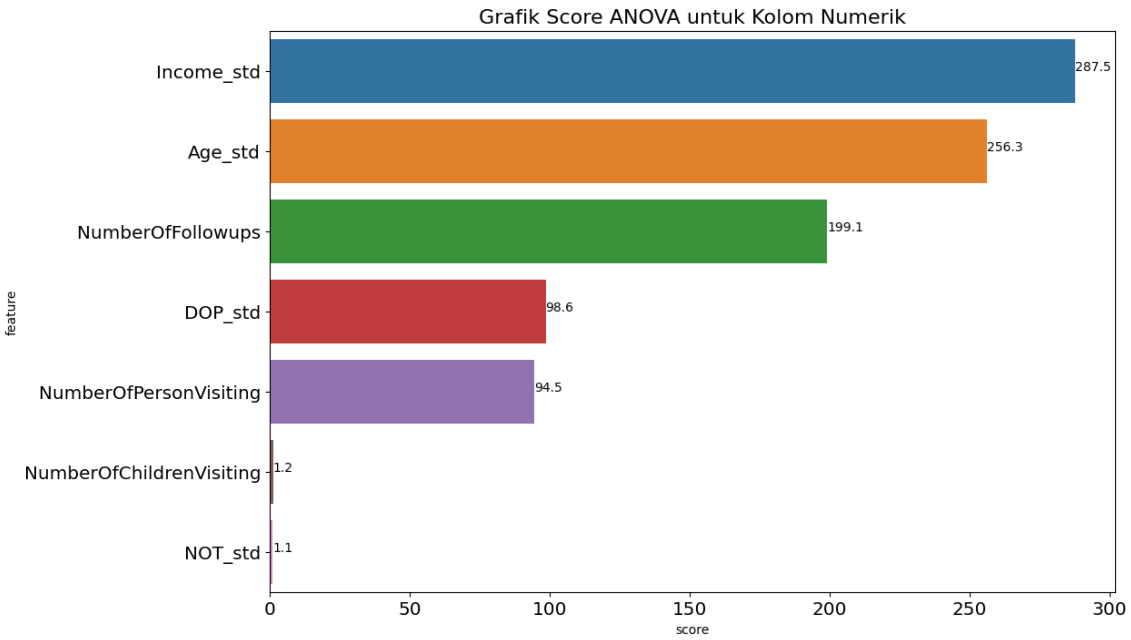
```
[ ]  map_encode = {'ProductPitched' : {'Basic': 0, 'Standard': 1, 'Deluxe': 2, 'Super Deluxe': 3, 'King': 4}}

     for col, mapping in map_encode.items():
         df1[col] = df1[col].map(mapping)
```

```
[ ]  df1 = pd.get_dummies(df1, columns=['Occupation','Designation'])
     df1.head(5)
```

- **Feature Encoding:**

  - TypeofContact, gender, maritalstatus, productpitched menggunakan **Label Encoding**
  - Occupation dan designation menggunakan **One-Hot Encoding**

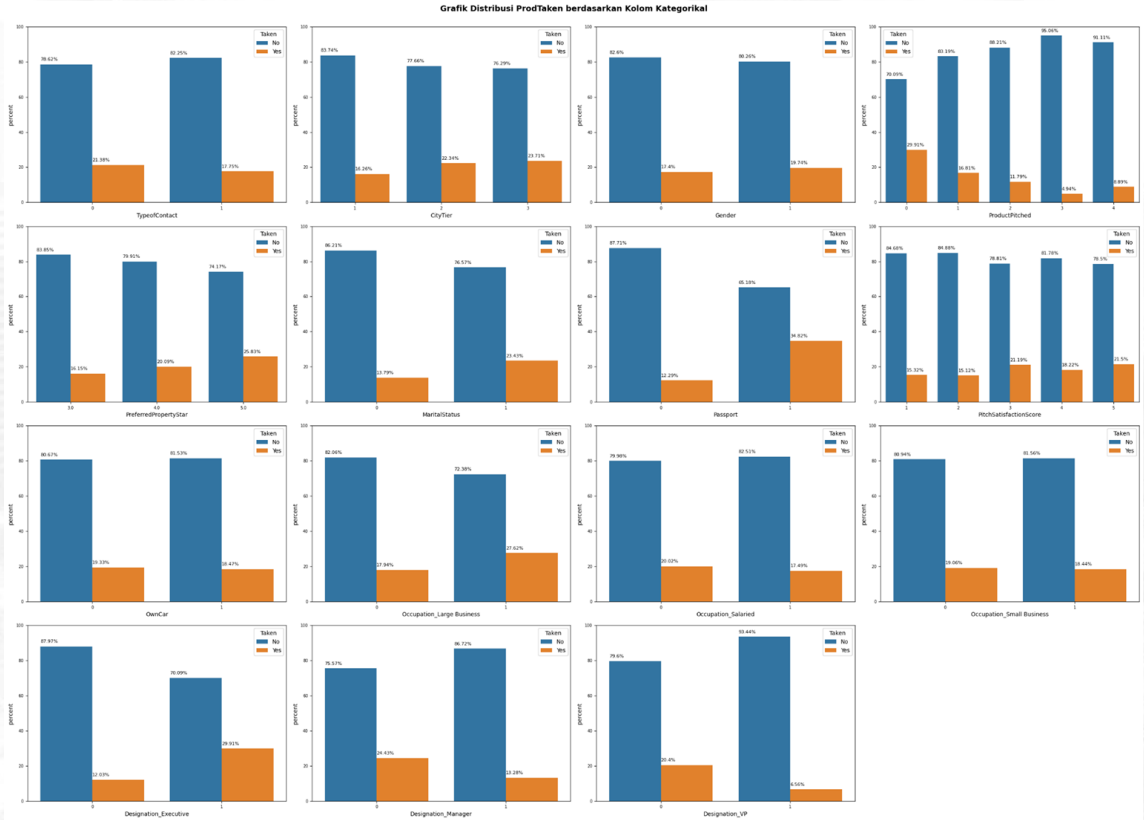
Feature Engineering



Feature Selection:

- Diamati kembali pengaruh fitur terhadap target pada dataframe yang telah dibersihkan
- **Pada fitur numerikal** digunakan test nilai ANOVA untuk melihat fitur mana yang kurang relevan sehingga nilai **NumberOfTrips yang lemah didrop**
- Fitur **NumberOfChildren** akan digunakan untuk membuat fitur baru untuk melihat perilaku customer berdasarkan adanya anak-anak atau tidak

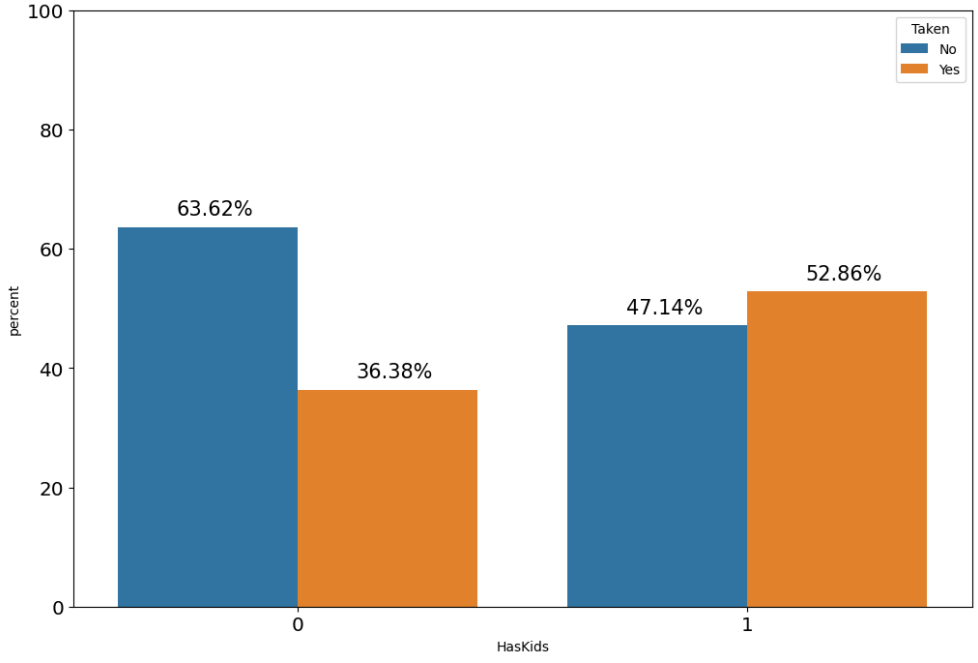
Feature Engineering



Feature Selection:

- **Data kategorikal** semuanya dipakai

Feature Engineering



Feature Extraction:

- NumberOfChildrenVisiting didrop dan digantikan oleh fitur baru HasKids, apakah mereka membawa anak atau tidak
- Visualisasi fitur HasKids terhadap target ProdTaken

Machine Learning Modelling and Evaluation

Model Klasifikasi:

Model yang akan digunakan adalah model klasifikasi untuk memprediksi customer yang akan membeli dan yang tidak. Jenis model klasifikasi yang digunakan antara lain:

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. XGBoost
5. AdaBoost
6. Bagging

Machine Learning Modelling and Evaluation

Evaluation Metrics:

Sebelum penentuan metrics dianalisis arti dari nilai positif dan negatif dari hasil prediksi yaitu:

- True Positive: terprediksi akan membeli, terbukti akan membeli
- True Negative: terprediksi tidak membeli, terbukti tidak membeli
- False Positive: terprediksi akan membeli, padahal tidak membeli, sehingga ada usaha marketing yang terbuang sia-sia (tidak tepat sasaran)
- False Negative: terprediksi tidak membeli, padahal akan membeli, sehingga ada kesempatan bisnis yang hilang (revenue loss)

Dengan tujuan untuk meningkatkan efisiensi marketing, fokus dari model prediksi adalah **menekan False Positive**, sehingga metrics yang digunakan metrics **Precision**, semakin tinggi nilai Precision maka semakin tepat sasaran marketing yang dilakukan sesuai prediksi model.

Modelling

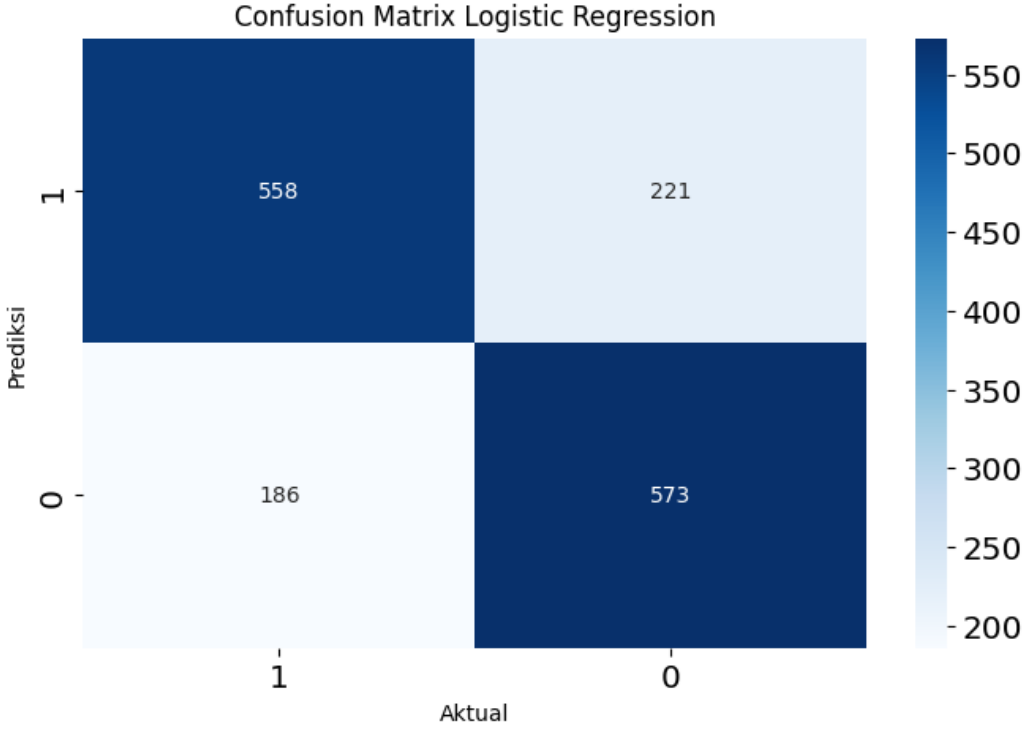
Logistic Regression : sebelum dan setelah tuning tidak berbeda

```
4]: # Sebelum tuning
    logreg = LogisticRegression(random_state=random_state)
    logreg.fit(X_train, y_train)
    eval_classification(logreg)
```

```
Accuracy (Test Set): 0.74
Precision (Test Set): 0.72
Recall (Test Set): 0.75
F1-Score (Test Set): 0.73
roc_auc (test-proba): 0.81
roc_auc (train-proba): 0.82

5-fold cross validation
precision (crossval train): 0.76
precision (crossval test): 0.71
precision (train-test gap): 0.05
```

```
7]: # Tuning sesuai hyperparameter terbaik
    logreg1 = LogisticRegression(random_state=random_state, **rs.best_params_)
    logreg1.fit(X_train, y_train)
    eval_classification(logreg1)
```

```
Accuracy (Test Set): 0.74
Precision (Test Set): 0.72
Recall (Test Set): 0.75
F1-Score (Test Set): 0.73
roc_auc (test-proba): 0.81
roc_auc (train-proba): 0.82

5-fold cross validation
precision (crossval train): 0.76
precision (crossval test): 0.71
precision (train-test gap): 0.05
```



Modelling

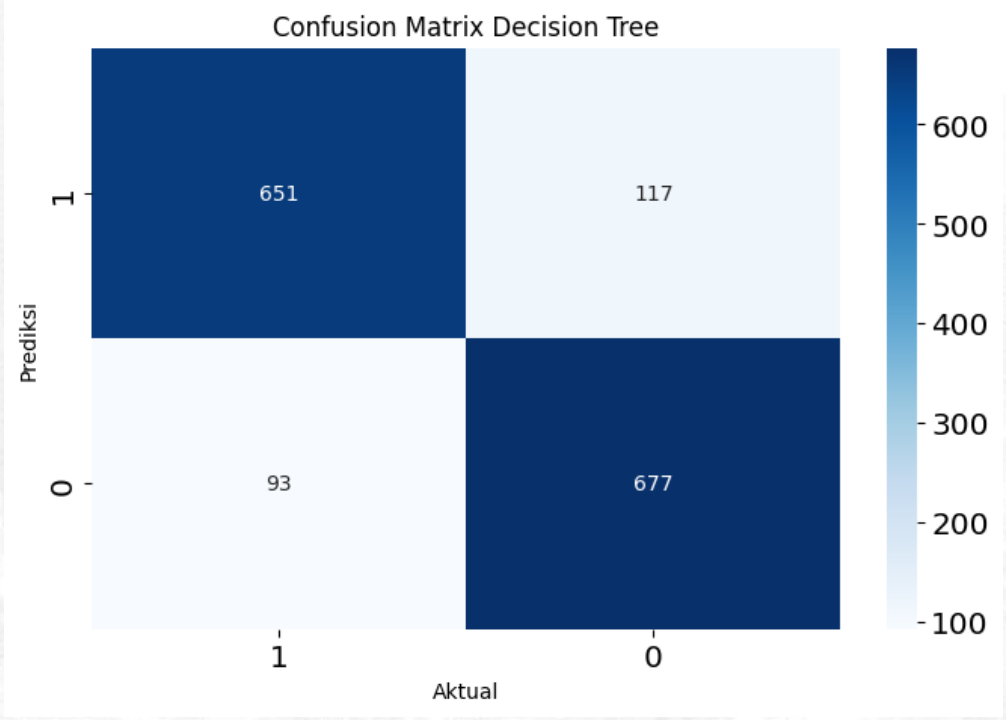
Decision Tree

```
# Sebelum tuning
dt = DecisionTreeClassifier(random_state=random_state)
dt.fit(X_train, y_train)
eval_classification(dt)
```

```
Accuracy (Test Set): 0.90
Precision (Test Set): 0.88
Recall (Test Set): 0.92
F1-Score (Test Set): 0.90
roc_auc (test-proba): 0.90
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 1.00
precision (crossval test): 0.89
precision (train-test gap): 0.11
```

```
# Tuning sesuai hyperparameter terbaik
dt1 = DecisionTreeClassifier(random_state=random_state, **rs.best_params_)
dt1.fit(X_train, y_train)
eval_classification(dt1)
```

```
Accuracy (Test Set): 0.86
Precision (Test Set): 0.85
Recall (Test Set): 0.88
F1-Score (Test Set): 0.86
roc_auc (test-proba): 0.90
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 0.97
precision (crossval test): 0.84
precision (train-test gap): 0.13
```



Modelling

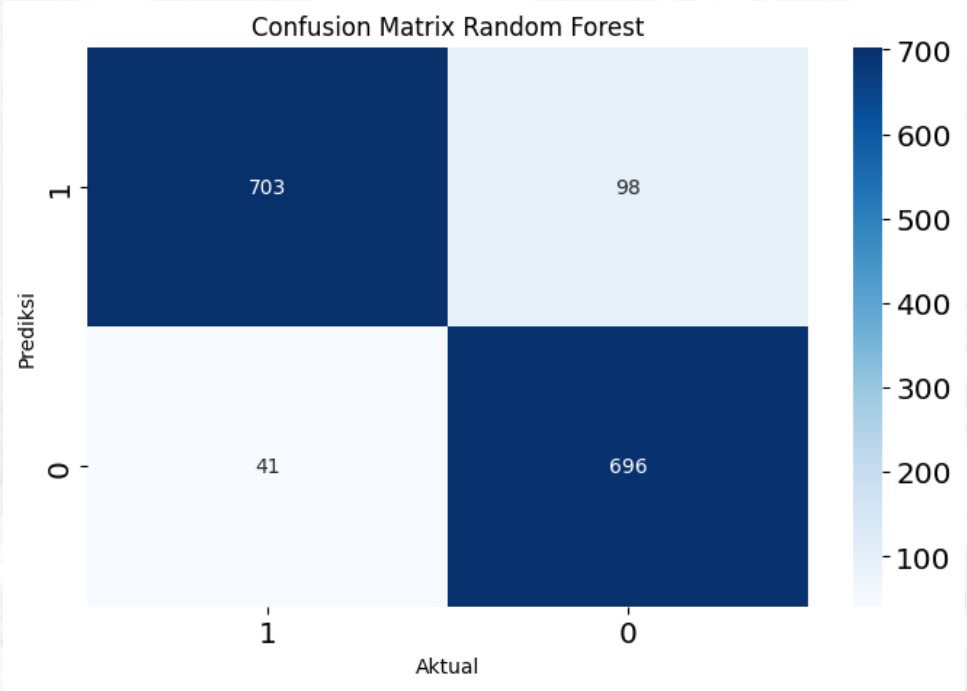
Random Forest

```
# Algoritma Random Forest tanpa hyperparameter tuning
rf = RandomForestClassifier(random_state=random_state)
rf.fit(X_train, y_train)
eval_classification(rf)
```

```
Accuracy (Test Set): 0.96
Precision (Test Set): 0.94
Recall (Test Set): 0.98
F1-Score (Test Set): 0.96
roc_auc (test-proba): 0.99
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 1.00
precision (crossval test): 0.95
precision (train-test gap): 0.05
```

```
# Melakukan tuning berdasarkan parameter terbaik hasil random search
rf1 = RandomForestClassifier(random_state=random_state, **rs.best_params_)
rf1.fit(X_train, y_train)
eval_classification(rf1)
```

```
Accuracy (Test Set): 0.91
Precision (Test Set): 0.88
Recall (Test Set): 0.94
F1-Score (Test Set): 0.91
roc_auc (test-proba): 0.97
roc_auc (train-proba): 0.99

5-fold cross validation
precision (crossval train): 0.94
precision (crossval test): 0.86
precision (train-test gap): 0.08
```



Modelling

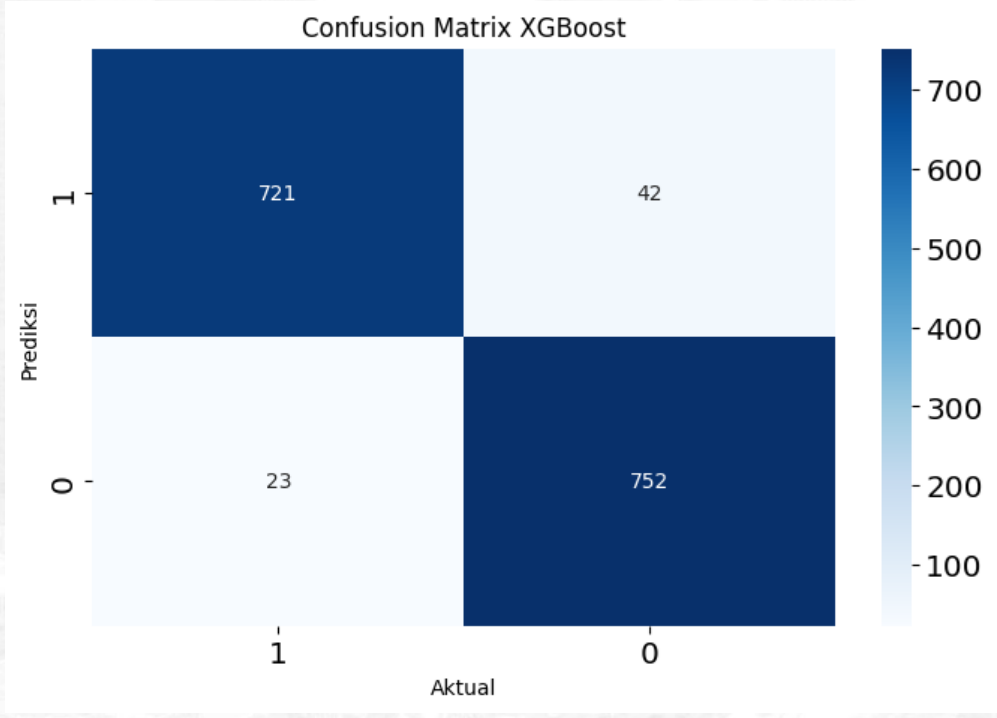
XGBoost

```
xg = XGBClassifier(random_state=random_state)
xg.fit(X_train, y_train)
eval_classification(xg)
```

```
Accuracy (Test Set): 0.96
Precision (Test Set): 0.95
Recall (Test Set): 0.96
F1-Score (Test Set): 0.96
roc_auc (test-proba): 0.99
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 1.00
precision (crossval test): 0.96
precision (train-test gap): 0.04
```

```
xg1 = XGBClassifier(random_state=random_state, **rs.best_params_)
xg1.fit(X_train, y_train)
eval_classification(xg1)
```

```
Accuracy (Test Set): 0.96
Precision (Test Set): 0.94
Recall (Test Set): 0.97
F1-Score (Test Set): 0.96
roc_auc (test-proba): 0.99
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 1.00
precision (crossval test): 0.95
precision (train-test gap): 0.05
```



Modelling

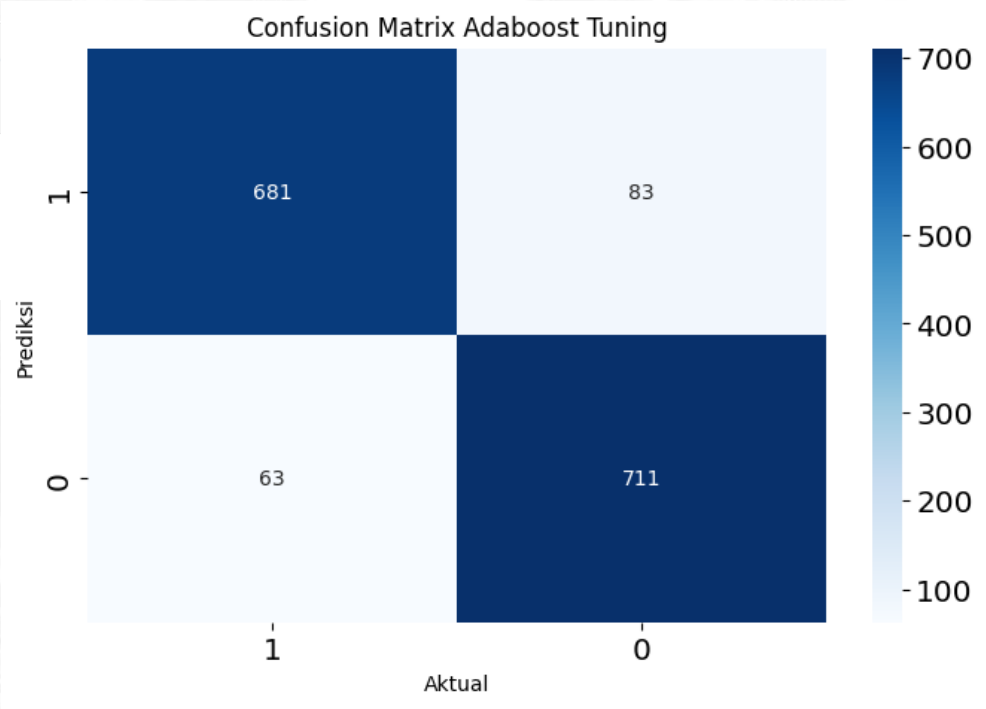
AdaBoost

```
# Adaboost
ab = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=16, min_samples_leaf=2, min_samples_split=4,
                        random_state=random_state, criterion='entropy'), random_state=random_state)

ab.fit(X_train, y_train)
eval_classification(ab)
```

```
Accuracy (Test Set): 0.97
Precision (Test Set): 0.96
Recall (Test Set): 0.98
F1-Score (Test Set): 0.97
roc_auc (test-proba): 1.00
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 1.00
precision (crossval test): 0.97
precision (train-test gap): 0.03
```

```
ab1.fit(X_train, y_train)
eval_classification(ab1)
```

```
Accuracy (Test Set): 0.91
Precision (Test Set): 0.89
Recall (Test Set): 0.92
F1-Score (Test Set): 0.90
roc_auc (test-proba): 0.91
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 1.00
precision (crossval test): 0.89
precision (train-test gap): 0.11
```



Modelling

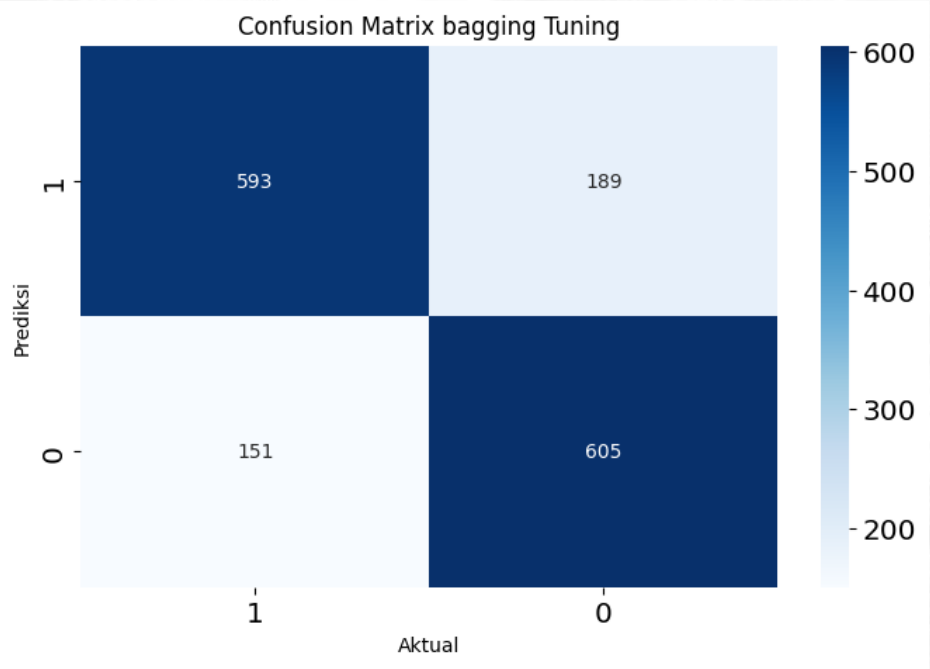
Bagging

```
# Bagging non tuning
bagging = BaggingClassifier(base_estimator=DecisionTreeClassifier(max_depth=16, min_samples_leaf=2, min_samples_split=4,
                            random_state=random_state, criterion='entropy'), random_state=random_state)

bagging.fit(X_train, y_train)
eval_classification(bagging)
```

```
Accuracy (Test Set): 0.94
Precision (Test Set): 0.92
Recall (Test Set): 0.96
F1-Score (Test Set): 0.94
roc_auc (test-proba): 0.99
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 0.99
precision (crossval test): 0.92
precision (train-test gap): 0.07
```

```
bagging_tuning = bagging_tuning.set_params(**BT.best_params_)
bagging_tuning.fit(X_train, y_train)
eval_classification(bagging_tuning)
```

```
Accuracy (Test Set): 0.78
Precision (Test Set): 0.76
Recall (Test Set): 0.80
F1-Score (Test Set): 0.78
roc_auc (test-proba): 0.85
roc_auc (train-proba): 0.86

5-fold cross validation
precision (crossval train): 0.78
precision (crossval test): 0.71
precision (train-test gap): 0.07
```



Model Selection

Evaluasi model

```
eval_table = pd.DataFrame(eval_score)
eval_table['model name'] = ['LogisticRegression', 'LogisticRegression tuned',
                            'DecisionTree', 'DecisionTree tuned',
                            'RandomForest', 'RandomForest tuned',
                            'XGBoost', 'XGBoost tuned',
                            'AdaBoost', 'AdaBoost tuned',
                            'Bagging', 'Bagging tuned',]
eval_table
```

	model name	precision	cv train	cv test	cv gap
8	AdaBoost	0.96	1.00	0.97	0.03
6	XGBoost	0.95	1.00	0.96	0.04
4	RandomForest	0.94	1.00	0.95	0.05
7	XGBoost tuned	0.94	1.00	0.95	0.05
10	Bagging	0.92	0.99	0.92	0.07
9	AdaBoost tuned	0.89	1.00	0.89	0.11
2	DecisionTree	0.88	1.00	0.89	0.11
5	RandomForest tuned	0.88	0.94	0.86	0.08
3	DecisionTree tuned	0.85	0.97	0.84	0.13
11	Bagging tuned	0.76	0.78	0.71	0.07
0	LogisticRegression	0.72	0.76	0.71	0.05
1	LogisticRegression tuned	0.72	0.76	0.71	0.05

Pemilihan model berdasarkan evaluasi model sebelum dan setelah tuning dan diurutkan berdasarkan nilai precision untuk mempermudah pemilihan model.

Model yang dipilih berdasarkan precision terbaik adalah **AdaBoost**

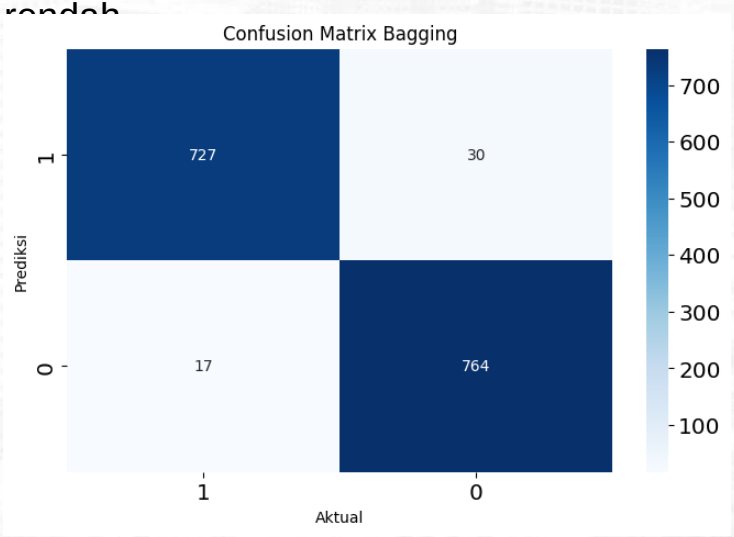
Model Selection

Hasil evaluasi model yang dipilih:
AdaBoost

```
# model yang dipilih
selected = ab
eval_classification(selected)
```

```
Accuracy (Test Set): 0.97
Precision (Test Set): 0.96
Recall (Test Set): 0.98
F1-Score (Test Set): 0.97
roc_auc (test-proba): 1.00
roc_auc (train-proba): 1.00

5-fold cross validation
precision (crossval train): 1.00
precision (crossval test): 0.97
precision (train-test gap): 0.03
```

Model AdaBoost non tuning yang dipilih menunjukkan tingkat precision yang tinggi dengan gap antara data train dan test yang rendah

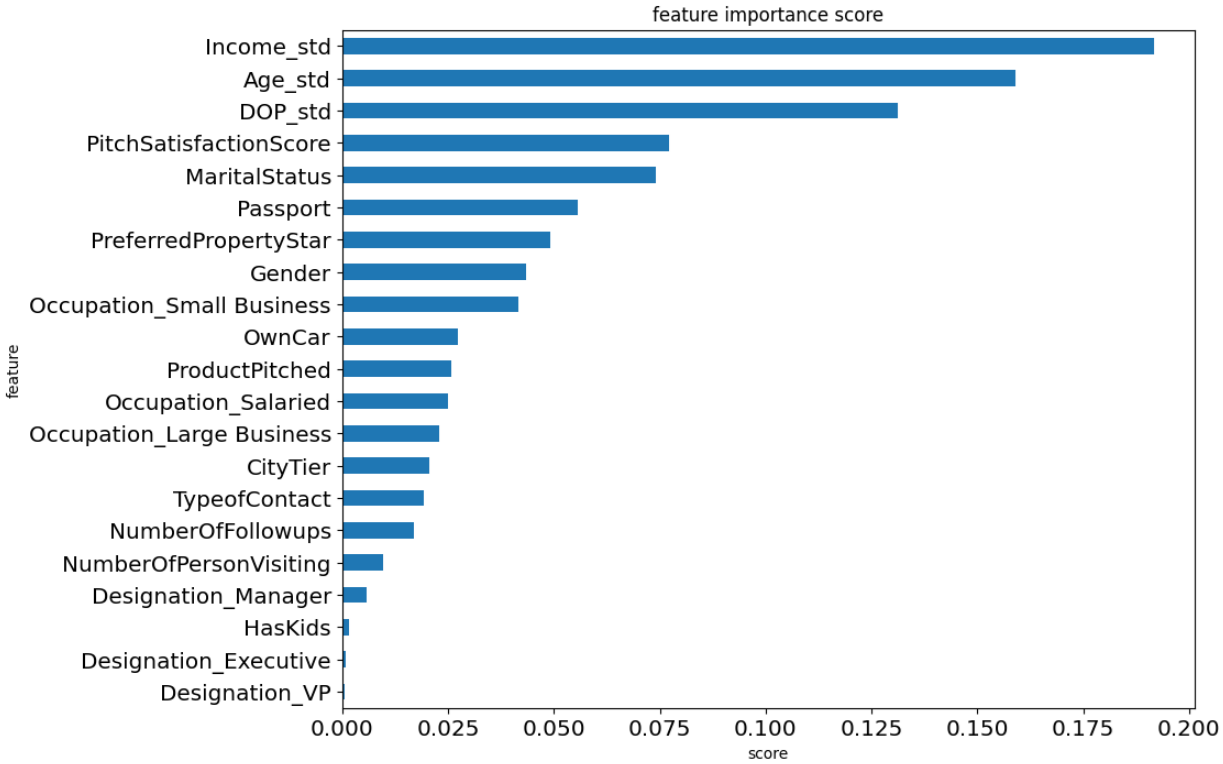


Feature Importance

Hasil uji feature importance

```
show_feature_importance(selected)
```

Berdasarkan model AdaBoost yang dipilih, feature yang paling penting untuk memprediksi paket baru adalah Income_std, Age, dan duration of pitch_std.



Business Recommendation

Rekomendasi yang dapat diberikan antara lain:

1. Monthly Income
    - Harga produk baru disarankan pada range harga Basic dan Standard campaign sebelumnya. Harga yang sesuai dengan selera customer berdasar pendapatan akan mendorong mereka untuk membeli.
2. Age
    - Buat jenis paket yang fleksibel dari segi durasi. Sampaikan penawaran dengan lugas. Range umur customer yang banyak membeli berada di usia aktif bekerja dan usia melek teknologi, sehingga butuh fleksibilitas.
3. Duration Of Pitch
    - Gunakan 10-minutes rule dalam melakukan penawaran. Sampaikan poin utama produk dan jawab pertanyaan mengenai produk secara padat sehingga customer lebih yakin dalam memutuskan untuk membeli.

Pembagian Tugas

Andre Adeputra S:
- Stage0: Goals, presenter mentoring
- Stage1: Multivariate Analysis
- Stage2: Handling Outlier, Feature Extraction, notulensi
- Stage3: RandomForest, notulensi
- Stage4: Business Understanding dan Problem Statement, presenter mentoring

Gilang Muhammad Rizky:
- Stage0: Objective
- Stage1: Univariate Analysis, notulensi
- Stage2: Feature encoding
- Stage3: AdaBoost, Bagging, fungsi visualisasi confusion matrix, presenter mentoring

Jomas Sekar:
- Stage0: Problem Statement dan notulensi
- Stage1: Descriptive Analysis
- Stage2: Feature Transformation, presenter mentoring
- Stage3: Decision Tree
- Stage4: Notulensi mentoring

M Nurkholis Fauzi:
- Stage0: Goals
- Stage1: Univariate Analysis, notulensi, presenter mentoring
- Stage2: Feature transformation
- Stage3: Logistic Regression

Pembagian Tugas

Naomi Damanik:
- Stage0: Objective
- Stage1: Descriptive Analysis
- Stage2: Feature encoding, notulensi
- Stage3: Logistic Regression
- Stage4: Model evaluation dan Feature Importance

Vanesa:
- Stage0: Problem statement, notulensi
- Stage1: Business insight
- Stage2: Handling class imbalance
- Stage3: Decision Tree
- Stage4: Insight and recommendation, visual design presentasi, notulensi

Sakinah Nurul R:
- Stage0: Business metrics
- Stage1: Multivariate Analysis
- Stage2: Handling class imbalance, presenter mentoring
- Stage3: XGBoost, notulensi
- Stage4: EDA dan Pre-Processing

Muhammad Syarif U:
- Stage0: Business Metrics
- Stage1: Business Insight