Banking Marketing Targets

Dokumen Laporan Final Project (Stage 3)

By: Group 3 DS Batch 21 aka **Jump-start**

Data Scientist Team

- Maghfira Amalia
- Mahendra Prabaswara
- Ridwan Donovan
- Windy Nurfikri

Problem Statement

Term deposits are one of the major income sources of a bank. These days, selling deposits through telemarketing is still the main of various plans to reach out to the clients. However, the bank still needs to pay attention to the effectiveness of the direct campaign considering the high cost and the time it takes. Based on previous campaign data of 45,211 clients, **only 3.3% of them actually subscribed to a term deposit**. Despite the fact that **36,959 clients were never contacted**. Therefore, determining the right potential depositors needs to be done to save resources. Meanwhile, we also need to review the efficiency and effectiveness of telemarketing as a direct campaign method.

Goal

Increase the efficiency and effectiveness of direct campaign

Objectives

- Predict the potential depositor by classifying them (based on their background and financial history)
- Find out the success rate of telemarketing as direct campaign method

Business Metric

Conversion Rate

Descriptive Statistic

A. Apakah ada kolom dengan tipe data kurang sesuai, atau nama kolom dan isinya kurang sesuai? Pada kolom `job` ada value 'admin.' yang mana seharusnya penulisan tidak perlu menggunakan tanda titik (.) seperti pekerjaan lainnya. Selebihnya, semua tipe data sudah sesuai.

```
data = pd.read_csv('train.csv', sep = ';')
df = data.rename(columns={'y':'subscribed'})
df.sample(5)
```

	age	job	marital	education	default	balance	housing	loan	contact	day	mont
33026	26	admin.	single	secondary	no	1375	yes	no	cellular	17	ap
19393	49	blue-collar	married	secondary	no	141	no	no	cellular	6	au
16347	36	management	divorced	tertiary	no	14930	no	no	cellular	23	j
2820	42	admin.	married	secondary	no	2656	yes	no	unknown	14	ma
41162	89	retired	married	tertiary	no	553	no	no	telephone	19	au

B. Apakah ada kolom yang memiliki nilai kosong? Jika ada, apa saja?
Tidak ada kolom dengan nilai kosong.

```
[6] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 18 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
 17  y2         45211 non-null  int64
dtypes: int64(8), object(10)
memory usage: 6.2+ MB
```

Rakamin Academy

C. Apakah ada kolom yang memiliki nilai summary agak aneh? (min/mean/median/max/unique/top/freq)

- Pada kolom **`previous`** nampak issue pada nilai maksimalnya, dimana salah satu customer dihubungi pada campaign sebelumnya sebanyak 275 kali. Kemungkinan akan di drop pada saat pre-processing.
- kolom **`balance`**, **`duration`**, dan **`pdays`** tampak right-skewed (median < mean).

```
df.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

Keanehan ditemukan pada summary data numerical

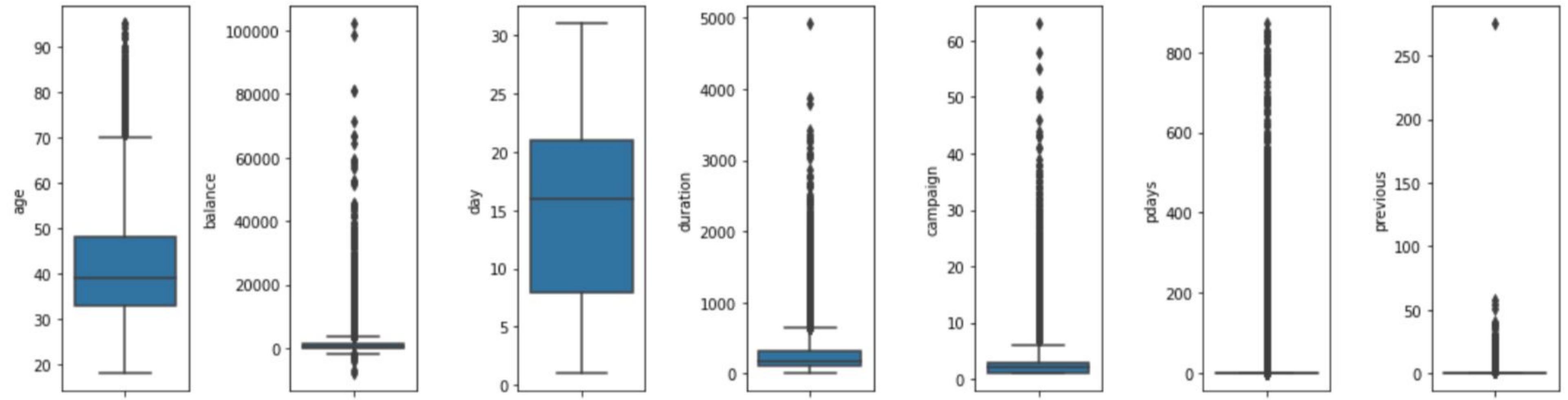
```
df[cats].describe()
```

	marital	education	default	housing	loan	contact	month	poutcome	y
count	45211	45211	45211	45211	45211	45211	45211	45211	45211
unique	3	4	2	2	2	3	12	4	2
top	married	secondary	no	yes	no	cellular	may	unknown	no
freq	27214	23202	44396	25130	37967	29285	13766	36959	39922

- Data didominasi oleh customer yang sudah menikah (**`marital`**) dan/atau tidak memiliki tunggakkan (**`default`**) ataupun pinjaman (**`loan`**).
- lebih dari 75% data customer tidak diketahui hasil/output dari campaign sebelumnya (**`poutcome`**).

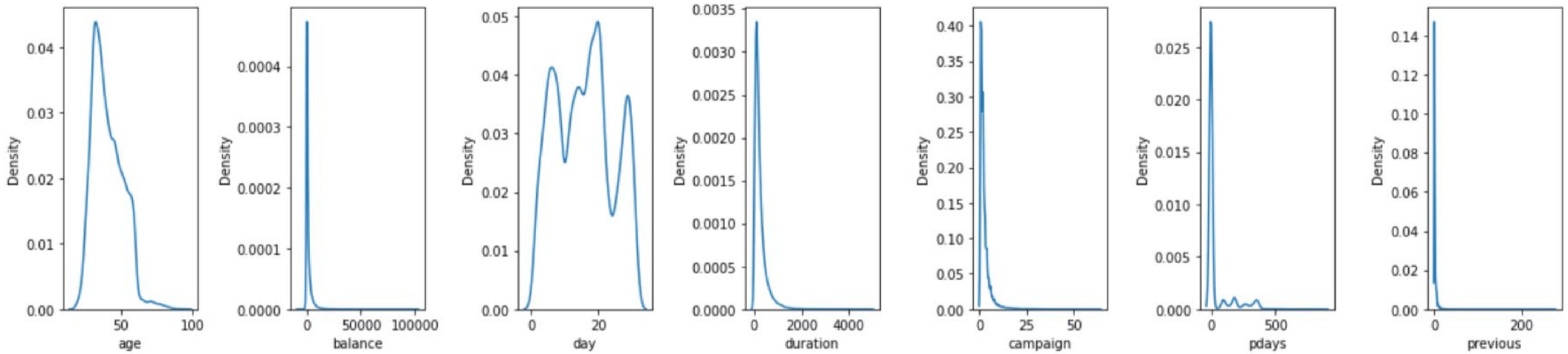
Univariate Analysis

```python
plt.figure(figsize=(15, 4))
for i in range(0, len(nums)):
  plt.subplot(1, len(nums), i+1)
  sns.boxplot(y=df[nums[i]], orient='v')
  plt.tight_layout()
```



- Pada **`previous`** dan **`duration`** terdapat outlier yang berbeda sangat signifikan.

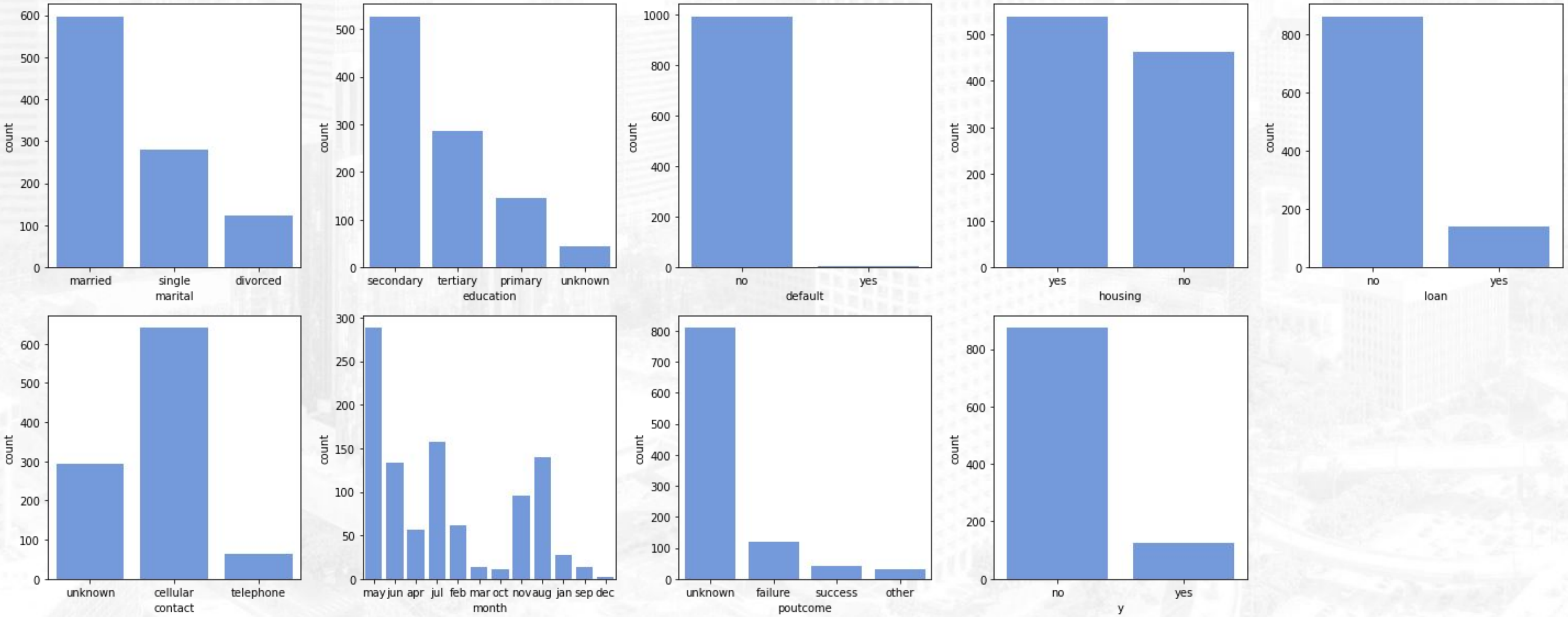
```
plt.figure(figsize=(17, 4))
for i in range(0, len(nums)):
  plt.subplot(1, len(nums), i+1)
  sns.kdeplot(x=df[nums[i]])
  plt.tight_layout()
```



- hampir semua fiture numerical, distribusi datanya (sangat) right-skewed.

Sedangkan untuk data kategorikal, dapat terlihat hampir seluruh fitur data memiliki ketimpangan, kecuali **`housing`**.

```python
cat_sample = df.sample(1000, random_state=42)
plt.figure(figsize=(20, 8))
for i in range(0, len(cats)):
  plt.subplot(2, 5, i+1)
  sns.countplot(x=cat_sample[cats[i]], color = 'cornflowerblue')
  plt.tight_layout()
```



Multivariate Analysis

Rakamin Academy

Encoding **y** untuk untuk kebutuhan cek korelasi fitur dengan target output.

```
[9] def segment(x):
        if x['y'] == 'yes':
            segment = 1
        else:
            segment = 0
        return segment
```

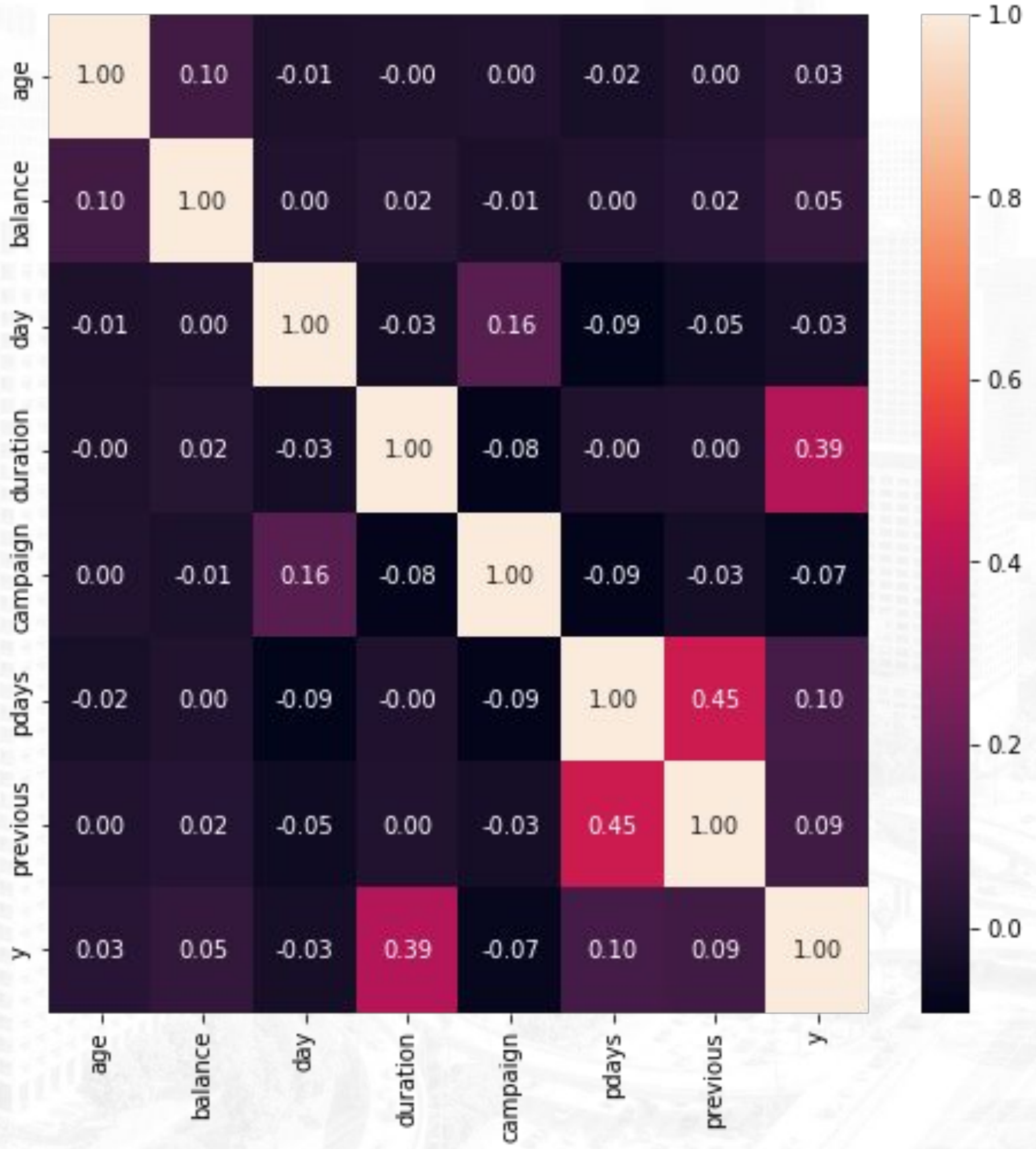
```
[10] df['y'] = df.apply(lambda x: segment(x), axis= 1)
     df.sample(10)
```

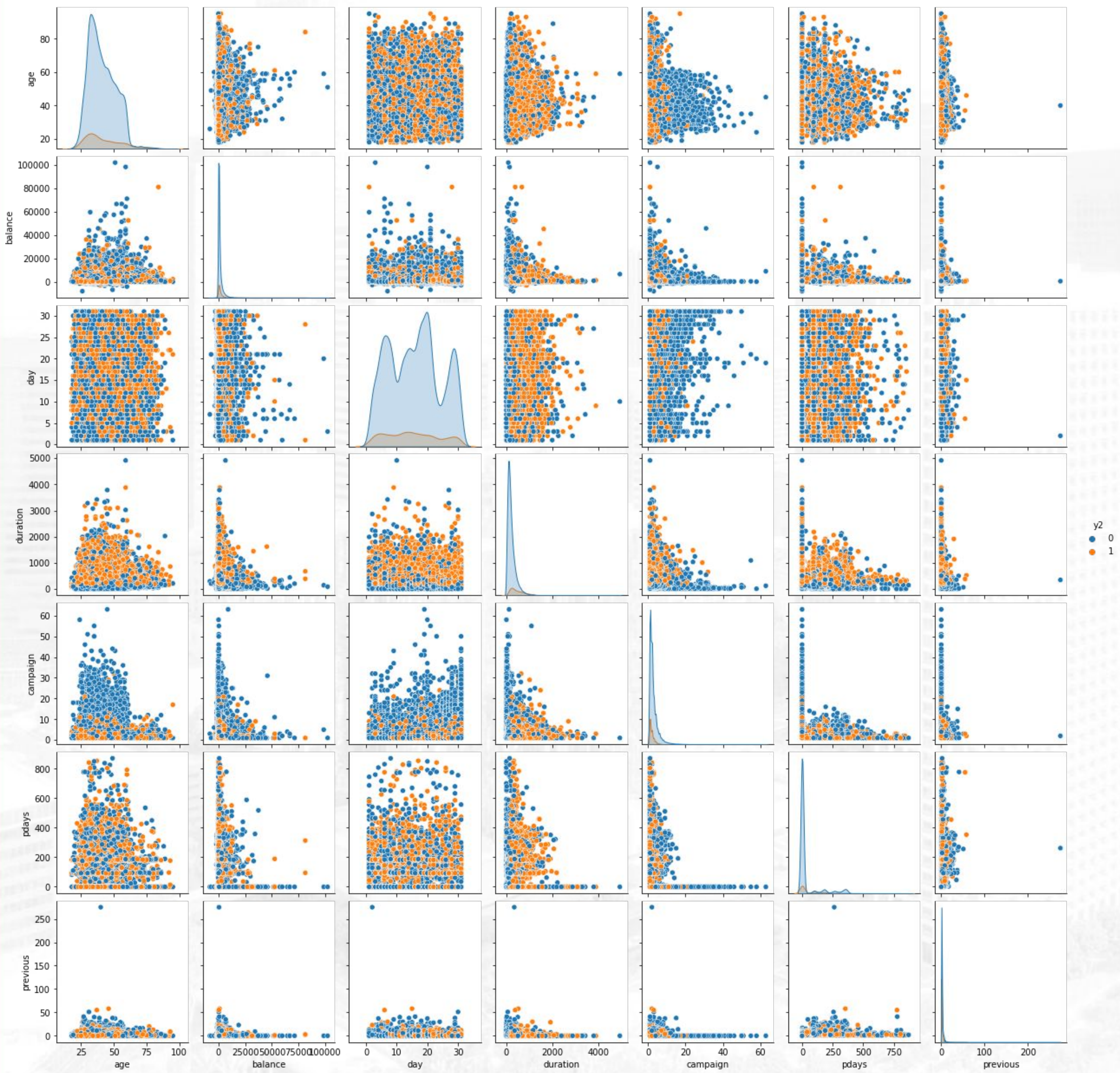
	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
13195	57	technician	married	primary	no	4442	no	no	cellular	8	jul	97	6	-1	0	unknown	0
20666	37	technician	single	secondary	no	3665	no	no	cellular	12	aug	664	3	-1	0	unknown	1
7142	38	retired	single	secondary	no	44	no	no	unknown	29	may	148	1	-1	0	unknown	0
592	41	admin.	divorced	primary	no	4070	yes	no	unknown	6	may	140	2	-1	0	unknown	0
15482	31	entrepreneur	single	secondary	no	379	yes	no	cellular	18	jul	570	2	-1	0	unknown	0
8457	38	services	married	secondary	no	823	yes	no	unknown	3	jun	132	5	-1	0	unknown	0
37178	39	management	married	tertiary	no	141	yes	no	cellular	13	may	788	2	331	6	other	0
40357	59	self-employed	married	tertiary	no	185	no	no	cellular	22	jun	177	5	138	1	failure	0
36342	46	blue-collar	married	secondary	no	-27	yes	no	cellular	11	may	254	1	-1	0	unknown	0

Korelasi Data Numerik

Dari correlation heatmap di atas dapat disimpulkan bahwa:

- **`y`** memiliki potensi korelasi yang tinggi dengan *duration* (strong potential correlation)
- **`y`** juga memiliki korelasi yang lemah dengan **`previous`** dan **`pdays`** (decent potential feature)
- **`campaign`** memiliki korelasi positif dengan **`day`**
- Sedangkan korelasi **`campaign`** dengan **`age`** sangat lemah, kemungkinan bukan fitur yang potensial (decent potential feature)
- ada korelasi antar **`previous`** dengan **`pdays`**, namun tidak cukup kuat untuk dikatakan redundant



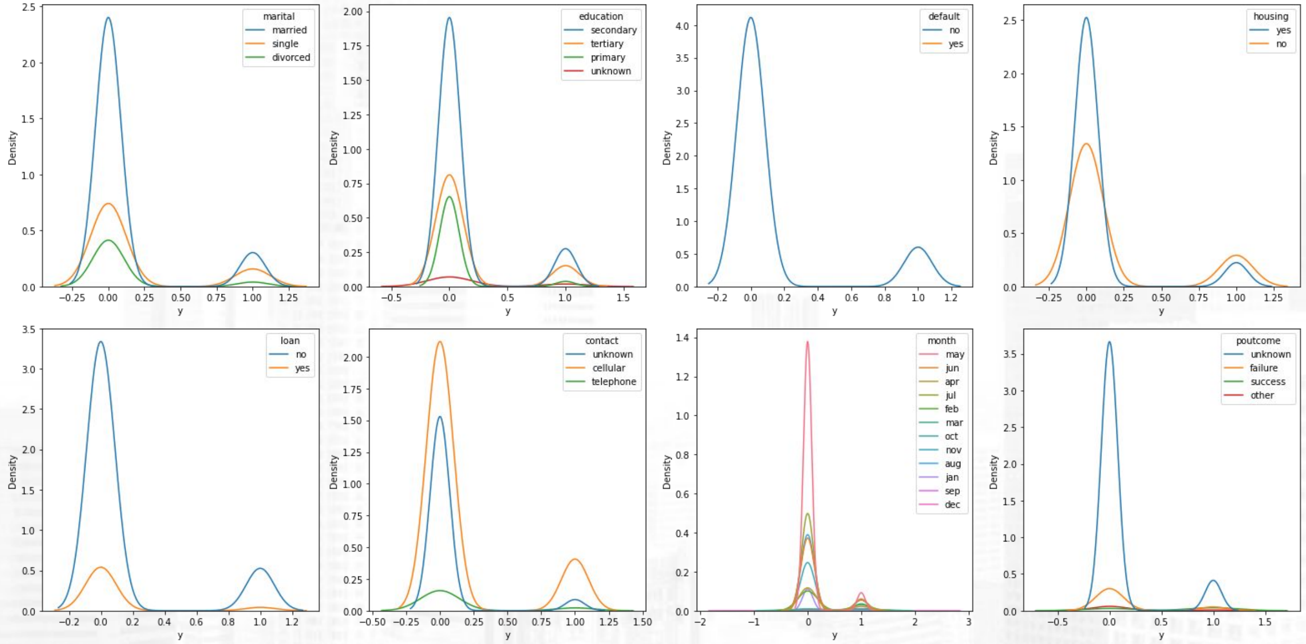


belum ada fitur yg memiliki korelasi linear yg cukup kuat.

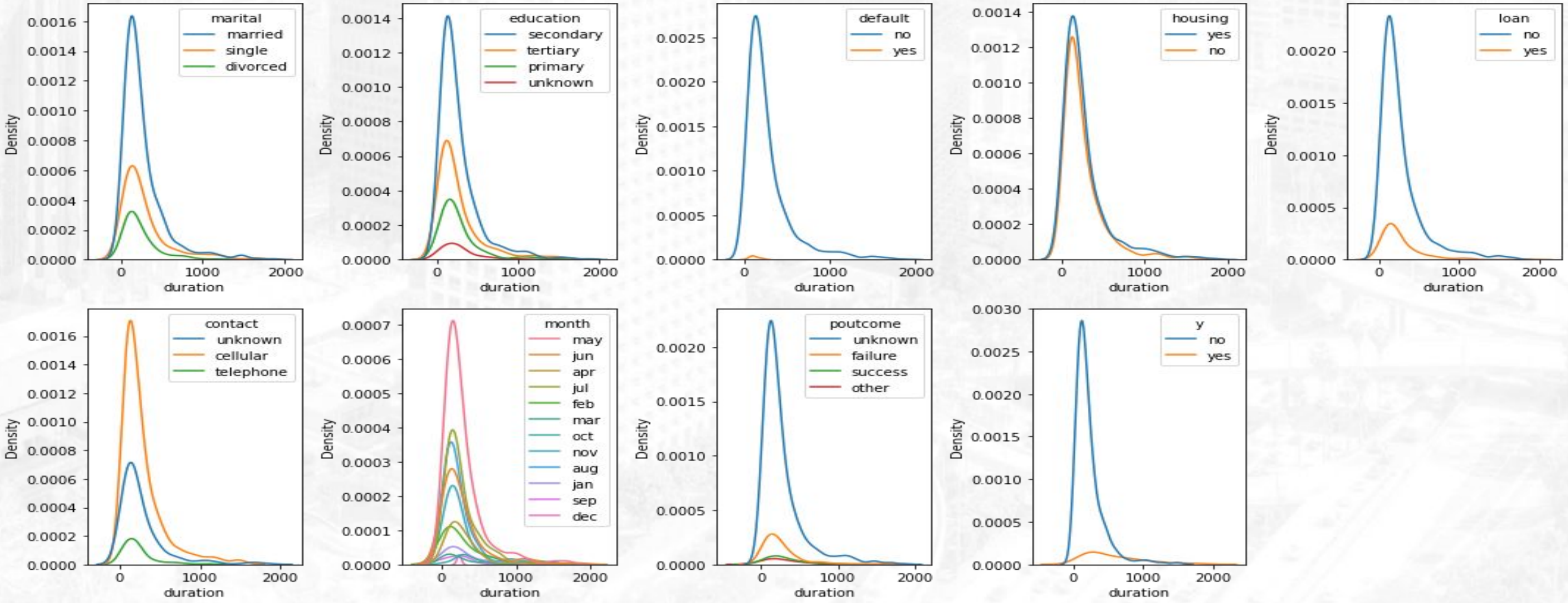
Rakamin Academy

Korelasi Data Kategorik terhadap `y`

Belum nampak ada korelasi yg kuat pada data kategorik terhadap output target.



Korelasi data kategorik terhadap `duration`

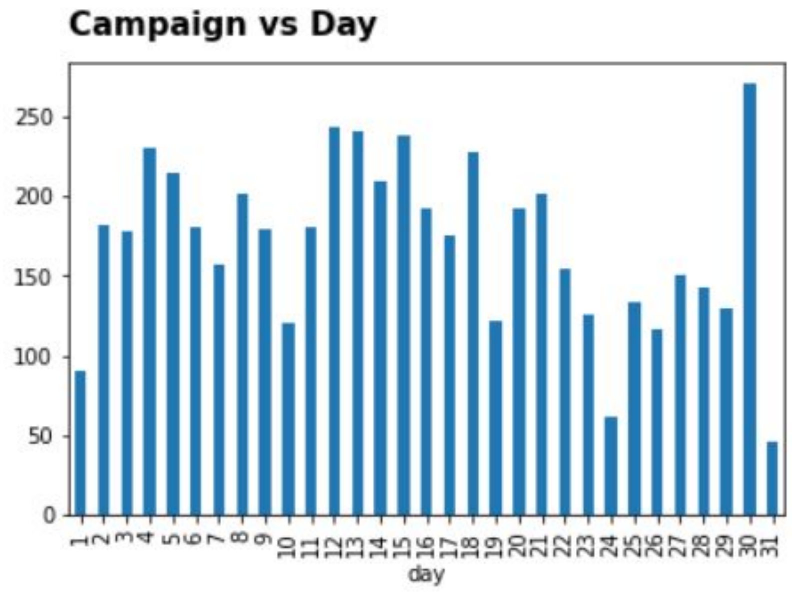


namun customer yang melakukan subsribed deposito (`y` = 'yes') cenderung memiliki durasi telepon yg lebih lama dibandingkan yang tidak subscribed.

Business Insight

- Dilihat pada heatmap, terdapat korelasi yg cukup tinggi antara target output (`y`) dengan `duration.` Jadi, untuk meningkatkan jumlah nasabah yang melakukan deposito dengan menambahkan durasi telepon kepada nasabah. Tetapi perlu diperhatikan juga semakin lama durasi telepon semakin besar biaya yang diperlukan.
- Pada `previous` dan `pdays` walaupun memiliki korelasi tapi sepertinya tidak ada kausalitas.
- Terdapat korelasi `campaign` terhadap `day`. Sehingga dapat dipilih hari-hari tententu yang memiliki tingkat keberhasilan campaign yang tinggi, agar campaign dapat lebih efektif.

```python
dy = df[df['y'] == 1]

dy_cpg = dy.groupby('day')['campaign'].count()

dy_cpg.plot(x='day', y='campaign', kind='bar')
plt.title("Campaign vs Day", loc='left', y=1.03, fontsize=15, weight='bold')
plt.show()
```



campaign yg menghasilkan output positif (customer subscribed to deposit) jarang terjadi pada akhir bulan.

Langkah Meningkatkan Model

a. Encode Data Yes dan No pada `y` menjadi boolean agar bisa lebih mudah melihat
b. Ambil insight dari korelasi antara `y` dengan data kategorik serta korelasi `duration` dengan data kategorik.
c. Handling outlier menggunakan metode manual filtering.
d. Melakukan box-cox transformation pada `balance`, `duration`, `campaign`, dan `previous` kemudian lakukan normalisasi data.
e.

Data Cleansing

A. Handle missing values

Tidak ada data yang null jadi tidak perlu dlakukan handle missing values

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

age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

Data tidak ada yang null

B. Handle duplicated data

Tidak ditemukan data duplikat jadi tidak perlu dilakukan handle duplicate data

```
[ ]  1 df.duplicated().sum()

0
```

Data tidak ada yang duplikat

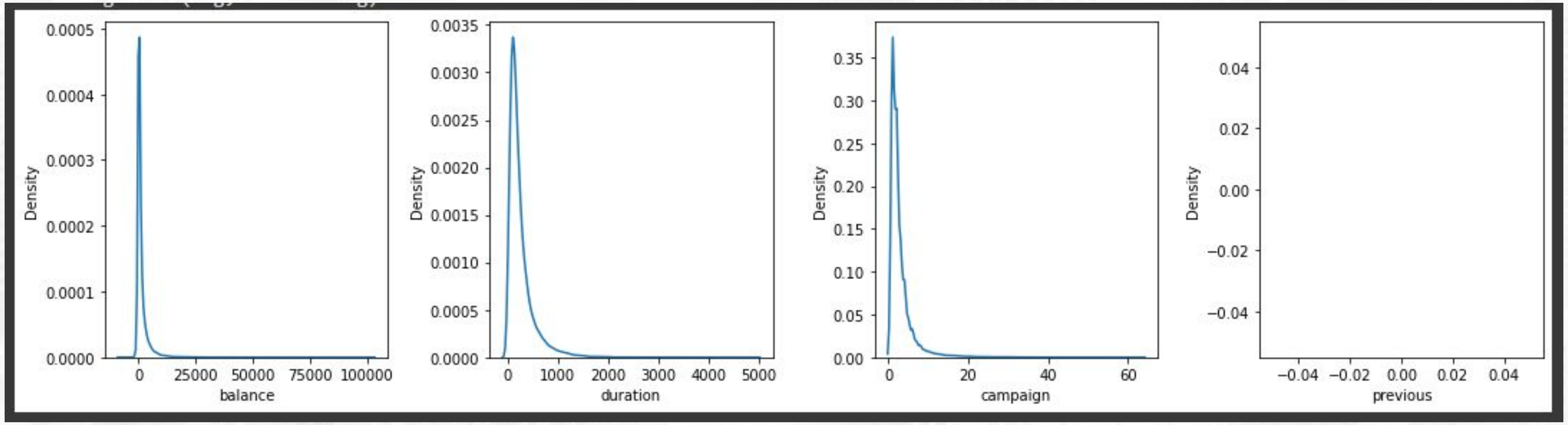
C. Handle outliers

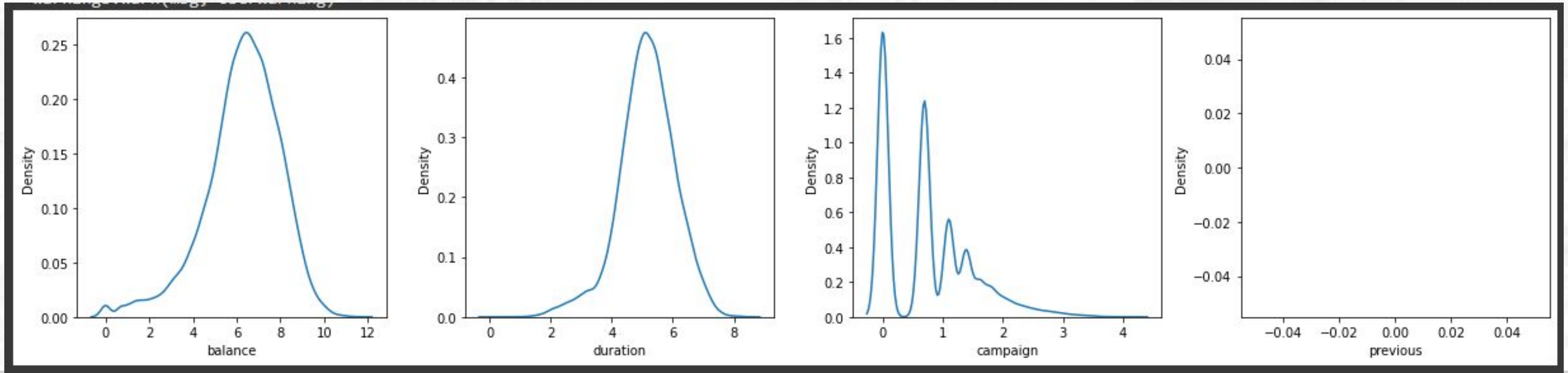
```python
from sklearn.preprocessing import MinMaxScaler, StandardScaler

df['age_norm'] = MinMaxScaler().fit_transform(df['age'].values.reshape(len(df), 1))
df['day_norm'] = MinMaxScaler().fit_transform(df['day'].values.reshape(len(df), 1))
df['campaign_norm'] = MinMaxScaler().fit_transform(df['campaign'].values.reshape(len(df), 1))
df['pdays_norm'] = MinMaxScaler().fit_transform(df['pdays'].values.reshape(len(df), 1))


df.describe()
```

	age	balance	day	duration	campaign	pdays	previous	y	log_balance	log_duration	age_norm	day_norm	campaign_norm	pdays_norm
count	36954.000000	36954.000000	36954.000000	36954.000000	36954.000000	36954.0	36954.0	36954.000000	3.367600e+04	3.695400e+04	36954.000000	36954.000000	36954.000000	36954.0
mean	40.932430	1318.788846	16.145424	257.726119	2.921957	-1.0	0.0	0.091573	-inf	-inf	0.297824	0.504847	0.030999	0.0
std	10.430218	3039.557077	8.372554	262.256406	3.325791	0.0	0.0	0.288427	NaN	NaN	0.135457	0.279085	0.053642	0.0
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.0	0.0	0.000000	-inf	-inf	0.000000	0.000000	0.000000	0.0
25%	33.000000	55.000000	9.000000	101.000000	1.000000	-1.0	0.0	0.000000	4.867534e+00	4.615121e+00	0.194805	0.266667	0.000000	0.0
50%	39.000000	414.000000	17.000000	177.000000	2.000000	-1.0	0.0	0.000000	6.240276e+00	5.176150e+00	0.272727	0.533333	0.016129	0.0
75%	49.000000	1358.000000	22.000000	318.000000	3.000000	-1.0	0.0	0.000000	7.333023e+00	5.762051e+00	0.402597	0.700000	0.032258	0.0
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	-1.0	0.0	1.000000	1.153397e+01	8.500657e+00	1.000000	1.000000	1.000000	0.0

D. Feature transformation

Berikut adalah perubahan dari grafik sebelum dan setelah dilakukan feature transformation

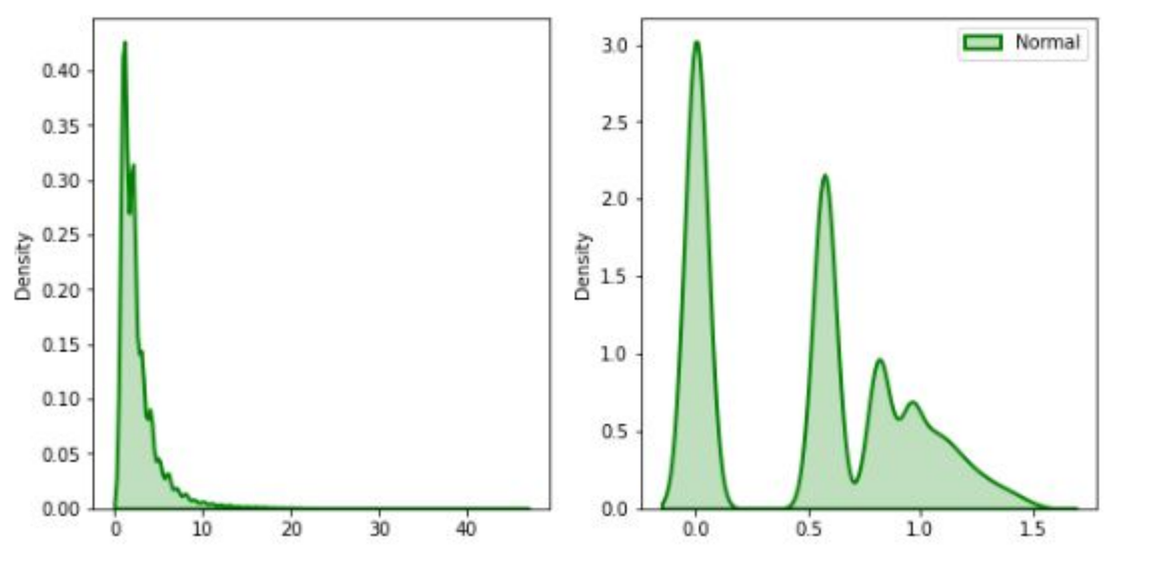




Feature Transformation menggunakan Box Cox

- Box cox tidak bisa digunakan pada data yang

```
for_box_cox = df['campaign'].values
fitted_data, fitted_lambda = stats.boxcox(for_box_cox)

# creating axes to draw plots
fig, ax = plt.subplots(1, 2)

# plotting the original data(non-normal) and
# fitted data (normal)
sns.distplot(for_box_cox, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Non-Normal", color ="green", ax = ax[0])
sns.distplot(fitted_data, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Normal", color ="green", ax = ax[1])

# adding legends to the subplots
plt.legend(loc = "upper right")

# rescaling the subplots
fig.set_figheight(5)
fig.set_figwidth(10)

print(f"Lambda value used for Transformation: {fitted_lambda}")
```



Berikut adalah perubahan dari datasebelum dan setelah dilakukan feature transformation

	age	balance	day	duration	campaign	pdays	previous	y
count	36954.000000	36954.000000	36954.000000	36954.000000	36954.000000	36954.0	36954.0	36954.000000
mean	40.932430	1318.788846	16.145424	257.726119	2.921957	-1.0	0.0	0.091573
std	10.430218	3039.557077	8.372554	262.256406	3.325791	0.0	0.0	0.288427
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.0	0.0	0.000000
25%	33.000000	55.000000	9.000000	101.000000	1.000000	-1.0	0.0	0.000000
50%	39.000000	414.000000	17.000000	177.000000	2.000000	-1.0	0.0	0.000000
75%	49.000000	1358.000000	22.000000	318.000000	3.000000	-1.0	0.0	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	-1.0	0.0	1.000000

	age	balance	day	duration	campaign	pdays	previous	y	log_balance	log_duration
count	36954.000000	36954.000000	36954.000000	36954.000000	36954.000000	36954.0	36954.0	36954.000000	3.367600e+04	3.695400e+04
mean	40.932430	1318.788846	16.145424	257.726119	2.921957	-1.0	0.0	0.091573	-inf	-inf
std	10.430218	3039.557077	8.372554	262.256406	3.325791	0.0	0.0	0.288427	NaN	NaN
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.0	0.0	0.000000	-inf	-inf
25%	33.000000	55.000000	9.000000	101.000000	1.000000	-1.0	0.0	0.000000	4.867534e+00	4.615121e+00
50%	39.000000	414.000000	17.000000	177.000000	2.000000	-1.0	0.0	0.000000	6.240276e+00	5.176150e+00
75%	49.000000	1358.000000	22.000000	318.000000	3.000000	-1.0	0.0	0.000000	7.333023e+00	5.762051e+00
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	-1.0	0.0	1.000000	1.153397e+01	8.500657e+00

E. Feature encoding

	age	job	education	default	balance	housing	loan	day	month	duration	...	day_norm	campaign_norm	pdays_norm	marital_divorced	marital_married	marital_single	contact_cellular	contact_telephone	contact_unknown	poutcome_unknown
0	58	management	3	0	2143	1	0	5	5	261	...	0.133333	0.0	0.0	0	1	0	0	0	1	1
1	44	technician	2	0	29	1	0	5	5	151	...	0.133333	0.0	0.0	0	0	1	0	0	1	1
2	33	entrepreneur	2	0	2	1	1	5	5	76	...	0.133333	0.0	0.0	0	1	0	0	0	1	1
3	47	blue-collar	0	0	1506	1	0	5	5	92	...	0.133333	0.0	0.0	0	1	0	0	0	1	1
4	33	unknown	0	0	1	0	0	5	5	198	...	0.133333	0.0	0.0	0	0	1	0	0	1	1

5 rows × 27 columns

F. Handle class imbalance menggunakan SMOTE

pertimbangan menggunakan SMOTE adalah menghindari adanya informasi yang hilang

```
[146] df1.y.value_counts()
```

```
0    39899
1     5284
Name: y, dtype: int64
```

```
[156] X = df1[[col for col in df1.columns if (str(df1[col].dtype) != 'object') and col not in ['y']]]
      y = df1['y'].values
      print(X.shape)
      print(y.shape)
```

```
(45183, 13)
(45183,)
```

```
[157] from imblearn import under_sampling, over_sampling
      X_over_SMOTE, y_over_SMOTE = over_sampling.SMOTE(0.5).fit_resample(X, y)
      print('SMOTE')
      print(pd.Series(y_over_SMOTE).value_counts())
```

```
SMOTE
0    39899
1    19949
dtype: int64
/usr/local/lib/python3.7/dist-packages/imblearn/utils/_validation.py:591: FutureWarning: Pass sampling
  FutureWarning,
```

2. Feature Engineering

A. Feature selection

Beberapa feature yang di drop menggunakan korelasi

```
1 abs(df.corr()['y'][abs(df.corr()['y'])>0.05].drop('y')).index.tolist()
```

```
['housing',
 'duration',
 'campaign',
 'log_balance',
 'log_duration',
 'campaign_norm',
 'marital_married',
 'marital_single',
 'contact_cellular',
 'contact_unknown']
```

B. Feature extraction

duration_minute : Mengubah duration pada satuan detik menjadi menit agar lebih mudah dipahami

```
cost_perminute = 0.01
df3['cost'] = (df['duration'] * cost_perminute).round(2)
df3.head()
```

	age	job	education	default	balance	housing	loan	day	month	duration	...	pdays_norm	marital_divorced	marital_married	marital_single	contact_cellular	contact_telephone	contact_unknown	poutcome_unknown	duration_minute
0	58	management	3	0	2143	1	0	5	5	261	...	0.0	0	1	0	0	0	1	1	4.4
1	44	technician	2	0	29	1	0	5	5	151	...	0.0	0	0	1	0	0	1	1	2.5
2	33	entrepreneur	2	0	2	1	1	5	5	76	...	0.0	0	1	0	0	0	1	1	1.3
3	47	blue-collar	0	0	1506	1	0	5	5	92	...	0.0	0	1	0	0	0	1	1	1.5
4	33	unknown	0	0	1	0	0	5	5	198	...	0.0	0	0	1	0	0	1	1	3.3

5 rows × 29 columns

C. 4 feature tambahan :

- duration_minute : Mengubah duration pada satuan detik menjadi menit agar lebih mudah dipahami
- cost : Menambahkan fitur tambahan cost yaitu biaya tambahan dengan tarif 0.01 euro per menit, sehingga menjadi bisa menjadi pertimbangan perusahaan untuk memperhatikan duration
- group_balance : Mengkategorikan nasabah sesuai dengan balancenya
- group_age : Mengkategorikan nasabah sesuai dengan umurnya, agar dapat diperhatikan usia produktif untuk bekerja

duration_minute	cost	group_balance	group_age
4.4	2.61	High	Adults
2.5	1.51	Low	Adults
1.3	0.76	Low	Adults
1.5	0.92	High	Adults
3.3	1.98	Low	Adults

Modeling

A. Split Data Train dan Test

Test size menggunakan default 0.2 dan random state 42. Untuk pembanding model agar mengetahui performance yang lebih baik, akan dilakukan percobaan dengan test size yang berbeda.

```
[120] from sklearn.model_selection import train_test_split

[127] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

[128] X_train.shape

(36146, 13)

[129] X_test.shape

(9037, 13)
```

B. Modeling

Fokus utama adalah nilai F1 Score, dan RandomForest memiliki nilai F1 Score yang paling bagus diantara semuanya.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
```

```
139] models = {
        "DecisionTree"      : DecisionTreeClassifier(random_state=42),
        "RandomForest"      : RandomForestClassifier(random_state=42),
        "GradientBoosting"  : GradientBoostingClassifier(random_state=42)
    }

    for model_name, clf in models.items():
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)
        probs = clf.predict_proba(X_test)[:,1]

        print("\n")
        print("Evaluate model: {}".format(model_name))

        fpr, tpr, thresholds = metrics.roc_curve(y_test, probs, pos_label=1)
        auc = metrics.auc(fpr, tpr)
        print("AUC: "+str(round(auc*100,2))+'%')

        accuracy = metrics.accuracy_score(y_test, y_pred)
        print("accuracy: "+str(round(accuracy*100,2))+'%')

        precision = metrics.precision_score(y_test, y_pred)
        print("precision: "+str(round(precision*100,2))+'%')

        recall = metrics.recall_score(y_test, y_pred)
        print("recall: "+str(round(recall*100,2))+'%')

        f1_score = ((2 * precision * recall)/(precision + recall))
        print("F1 score: "+str(round(f1_score*100,2))+'%')

        train_score = clf.score(X_train, y_train)
        print("train_score: "+str(round(train_score*100,2))+'%')
```

```
Evaluate model: DecisionTree
AUC: 88.83%
accuracy: 88.82%
precision: 87.93%
recall: 90.16%
F1 score: 89.03%
train_score: 99.99%
test_score: 88.82%


Evaluate model: RandomForest
AUC: 97.24%
accuracy: 91.38%
precision: 90.4%
recall: 92.7%
F1 score: 91.54%
train_score: 99.99%
test_score: 91.38%


Evaluate model: GradientBoosting
AUC: 95.99%
accuracy: 89.23%
precision: 88.85%
recall: 89.86%
F1 score: 89.35%
train_score: 89.85%
test_score: 89.23%
```

C. Hyperparameter tuning

```
[159] from sklearn.model_selection import RandomizedSearchCV
      from sklearn.ensemble import RandomForestClassifier

      #List of Hyper-parameters will be tested
      hyperparameters = dict(
                             n_estimators = [int(x) for x in np.linspace(start = 100, stop = 200, num = 5)], # Number of subtrees
                             max_depth = [int(x) for x in np.linspace(10, 110, num = 4)]+[None],  # Maximum depth of each subtree
                             min_samples_split = [2, 5, 10]
                            )
```

```
[160] # Inisialiasi model
      clf = RandomForestClassifier(random_state=42)
      clf_tuned = RandomizedSearchCV(clf, hyperparameters, cv=5, random_state=12, scoring='precision', n_iter=15)
      clf_tuned.fit(X_train, y_train)
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
                   n_iter=15,
                   param_distributions={'max_depth': [10, 43, 76, 110, None],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [100, 125, 150, 175,
                                                         200]},
                   random_state=12, scoring='precision')
```

```
[ ] # Predict & Evaluation
    y_pred = clf_tuned.predict(X_test)
    probs = clf_tuned.predict_proba(X_test)[:,1]
```

```
print(y_pred)
print(probs)
```

```
[0 0 0 ... 1 1 1]
[0.49113152 0.44390641 0.         ... 0.94742857 0.96710884 0.90795692]
```

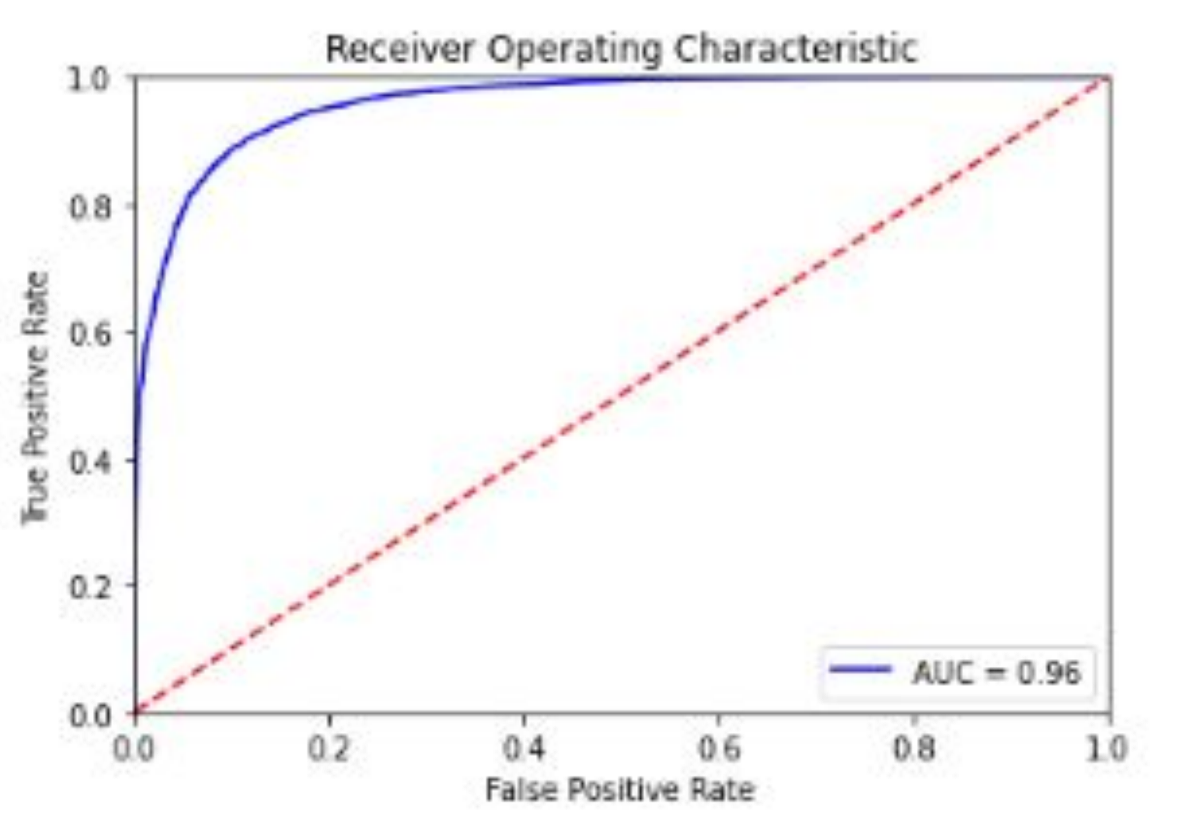
Check the best hyperparameter:

```
for key, value in hyperparameters.items() :
        print(key+':', clf_tuned.best_estimator_.get_params()[key])
```

```
n_estimators: 175
max_depth: 43
min_samples_split: 5
```

D. Model Evaluation
Mencari model yang paling bagus

```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



E. Feature Importance

```
clf_tuned.best_estimator_.feature_importances_
```

```
array([0.02276521, 0.11402032, 0.11434977, 0.0285804 , 0.35914542,
       0.03926539, 0.03577036, 0.05391773, 0.03172553, 0.01760321,
       0.07111136, 0.00567571, 0.10606958])
```

```
feat_importances = pd.Series(clf_tuned.best_estimator_.feature_importances_, index=X.columns)
ax = feat_importances.nlargest(25).plot(kind='barh', figsize=(10, 8))
ax.invert_yaxis()

plt.xlabel('score')
plt.ylabel('feature')
plt.title('feature importance score')
```