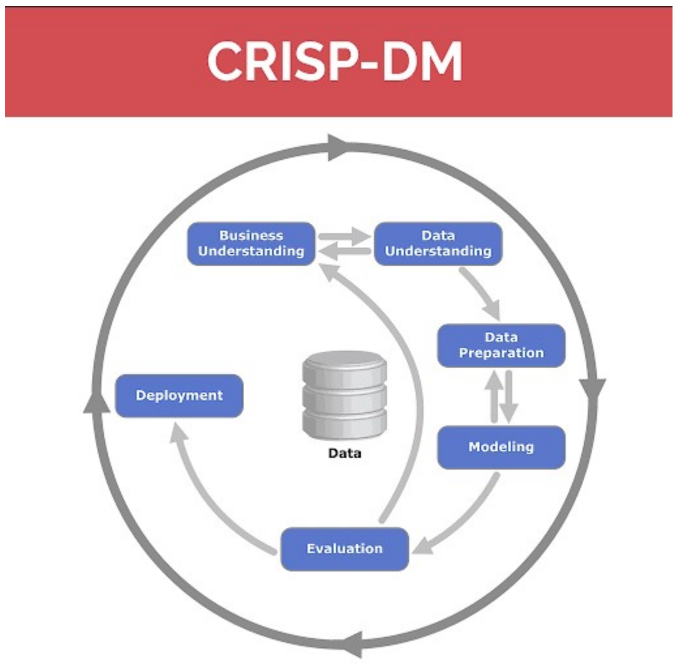
IMPLEMENTASI CRISP-DM UNTUK BANK MARKETING DATA SET



Fase dari CRISP-DM:

- Business Understanding
- Data Understanding
- Data Preparation
- Modeling
- Evaluation
- Deployment

1. Business Understanding

Business Understanding

Berdasarkan data yang tersedia, tujuan dari analisis ini yaitu untuk meningkatkan efisiensi pemasaran dari deposito berjangka untuk klien potensial berdasarkan dari data mereka. Data yang akan digunakan dibuat model prediksinya berdasarkan kategori di dalam data, apakah klien akan berinvestasi untuk deposito berjangka. Bentuk pendekatan yang digunakan adalah dengan membuat model prediksi dalam bentuk classifier berdasarkan kategori tersebut.

2. Data Understanding

Data Understanding

Berdasarkan informasi yang ada pada Bank Marketing Data Set, data ini memiliki 21 variabel yang terdiri dari 20 variabel input dan 1 variabel output.

Input variable:

 # Bank client data:

1. age (numeric)
2. job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3. marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4. education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5. default: has credit in default? (categorical: 'no','yes','unknown')
6. housing: has housing loan? (categorical: 'no','yes','unknown')
7. loan: has personal loan? (categorical: 'no','yes','unknown')

 # Related with the last contact of the current campaign:

8. contact: contact communication type (categorical: 'cellular','telephone')
9. month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10. day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11. duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

 # Other attributes:

12. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14. previous: number of contacts performed before this campaign and for this client (numeric)
15. poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

 # Social and economic context attributes

16. emp.var.rate: employment variation rate - quarterly indicator (numeric)
17. cons.price.idx: consumer price index - monthly indicator (numeric)
18. cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19. euribor3m: euribor 3 month rate - daily indicator (numeric)
20. nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

21. y - has the client subscribed a term deposit? (binary: 'yes','no')

Import library yang diperlukan

```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn import metrics
```

Load dataset

```
data = pd.read_csv("bank-additional-full.csv", sep=";")

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
age               41188 non-null int64
job               41188 non-null object
marital           41188 non-null object
education         41188 non-null object
default           41188 non-null object
housing           41188 non-null object
loan              41188 non-null object
contact           41188 non-null object
month             41188 non-null object
day_of_week       41188 non-null object
duration          41188 non-null int64
campaign          41188 non-null int64
pdays             41188 non-null int64
previous          41188 non-null int64
poutcome          41188 non-null object
emp.var.rate      41188 non-null float64
cons.price.idx    41188 non-null float64
cons.conf.idx     41188 non-null float64
euribor3m         41188 non-null float64
nr.employed       41188 non-null float64
y                 41188 non-null object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

Describe data

```
data.describe()
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

Data shape

```
data.shape
```

```
(41188, 21)
```

Cek missing value secara keseluruhan

```
#Missing Value
data.isna().sum()
```

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

Didapat bahwa pada dataset tidak terdapat missing value

Cek imbalance data

```
#Imbalance
data['y'].value_counts()
```

```
no     36548
yes     4640
Name: y, dtype: int64
```

Dataset termasuk imbalance karena terdapat 88.7% yang tidak melakukan investasi untuk deposito berjangka sedangkan yang berinvestasi hanya 11.3%

Cek duplicated data

```
#Duplicated data
data[data.duplicated()]
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.
1266	39	blue-collar	married	basic.6y	no	no	no	telephone	may	thu	...	1	999	0	nonexistent	
12261	36	retired	married	unknown	no	no	no	telephone	jul	thu	...	1	999	0	nonexistent	
14234	27	technician	single	professional.course	no	no	no	cellular	jul	mon	...	2	999	0	nonexistent	
16956	47	technician	divorced	high.school	no	yes	no	cellular	jul	thu	...	3	999	0	nonexistent	
18465	32	technician	single	professional.course	no	yes	no	cellular	jul	thu	...	1	999	0	nonexistent	
20216	55	services	married	high.school	unknown	no	no	cellular	aug	mon	...	1	999	0	nonexistent	
20534	41	technician	married	professional.course	no	yes	no	cellular	aug	tue	...	1	999	0	nonexistent	
25217	39	admin.	married	university.degree	no	no	no	cellular	nov	tue	...	2	999	0	nonexistent	
28477	24	services	single	high.school	no	yes	no	cellular	apr	tue	...	1	999	0	nonexistent	
32516	35	admin.	married	university.degree	no	yes	no	cellular	may	fri	...	4	999	0	nonexistent	
36951	45	admin.	married	university.degree	no	no	no	cellular	jul	thu	...	1	999	0	nonexistent	
38281	71	retired	single	university.degree	no	no	no	telephone	oct	tue	...	1	999	0	nonexistent	

12 rows × 21 columns

Dari hasil diatas, diketahui bahwa terdapat 12 **Duplicated Data** pada data

Mencari unique value pada variabel dengan tipe data object

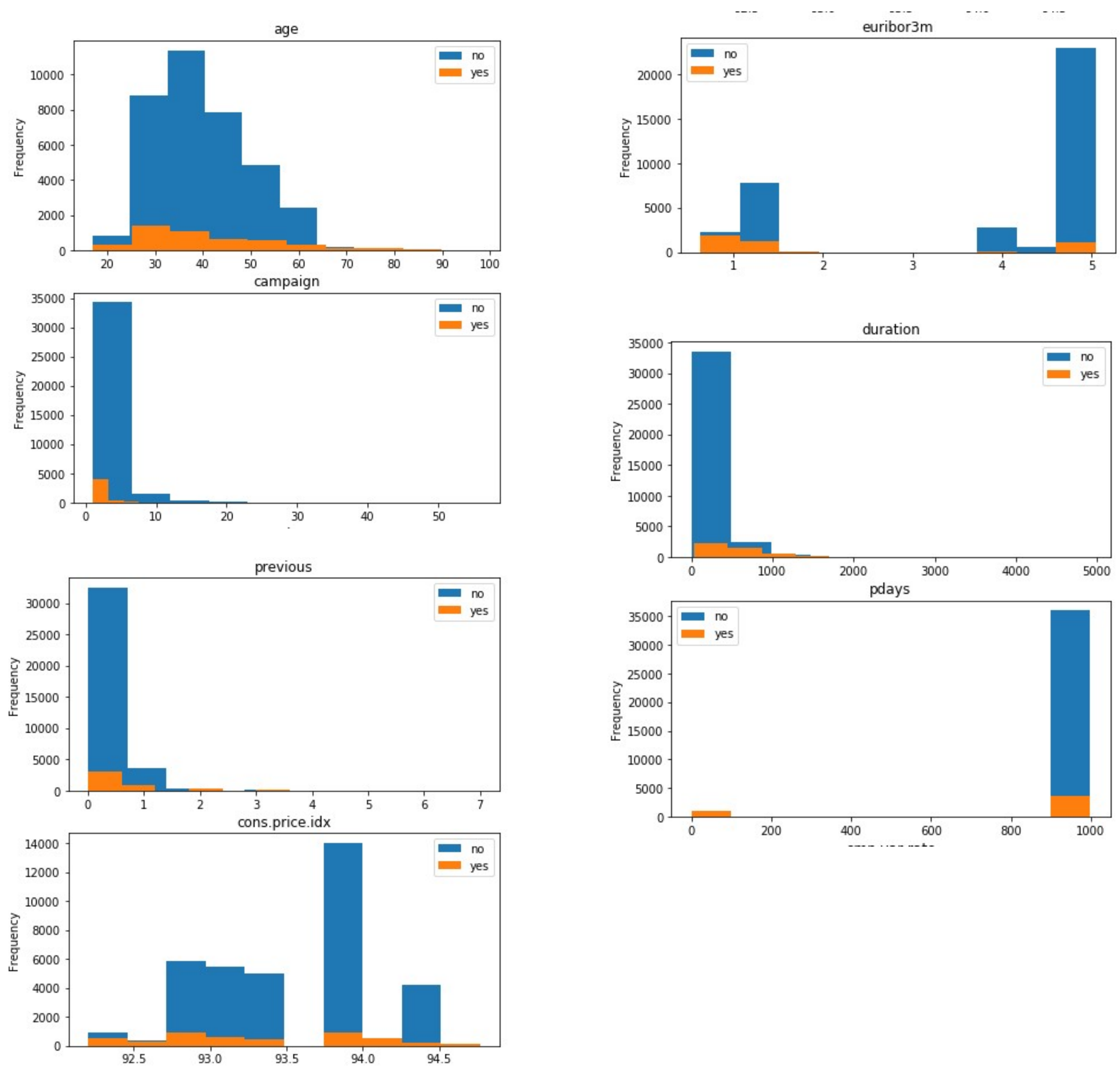
```
#Mencari unique value

for i in data.select_dtypes(include='object'):
    print('\nUnique value pada ' + i + ': ' + str(data[i].unique()))

print ('\nJumlah unique value pada setiap object')
print(data.select_dtypes(include='object').nunique())
```

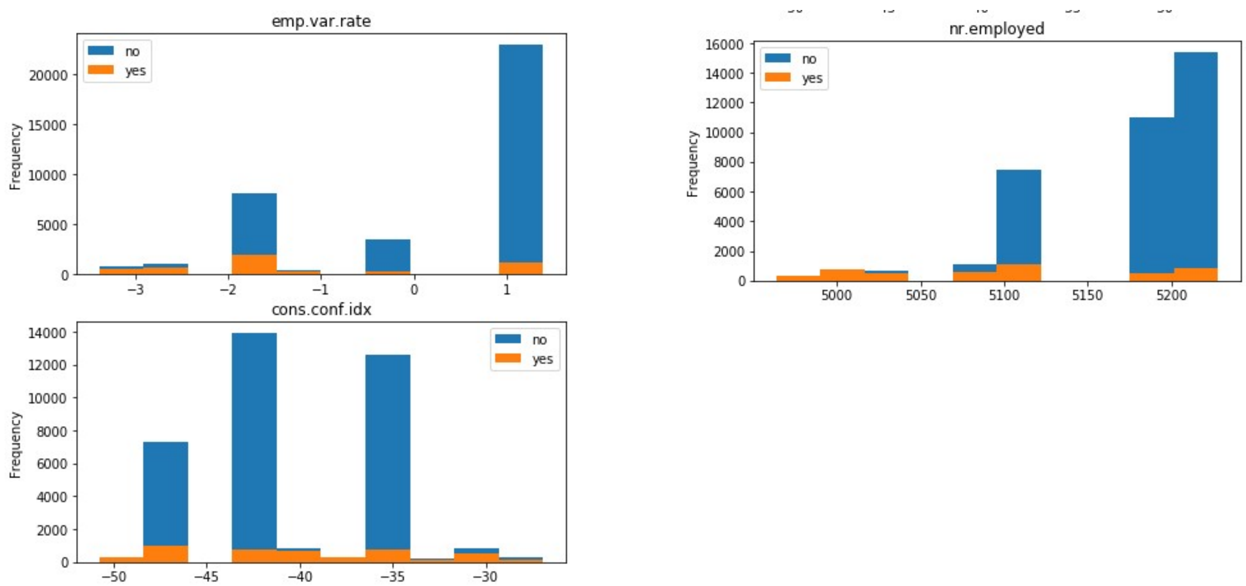
```
Unique value pada job: ['housemaid' 'services' 'admin.' 'blue-collar' 'technician' 'retired'
 'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
 'student']

Unique value pada marital: ['married' 'single' 'divorced' 'unknown']

Unique value pada education: ['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professional.course'
 'unknown' 'university.degree' 'illiterate']

Unique value pada default: ['no' 'unknown' 'yes']

Unique value pada housing: ['no' 'yes' 'unknown']

Unique value pada loan: ['no' 'yes' 'unknown']

Unique value pada contact: ['telephone' 'cellular']

Unique value pada month: ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar' 'apr' 'sep']

Unique value pada day_of_week: ['mon' 'tue' 'wed' 'thu' 'fri']

Unique value pada poutcome: ['nonexistent' 'failure' 'success']

Unique value pada y: ['no' 'yes']

Jumlah unique value pada setiap object
```

```
Jumlah unique value pada setiap object
job            12
marital         4
education       8
default         3
housing         3
loan            3
contact         2
month          10
day_of_week     5
poutcome        3
y               2
dtype: int64
```

Menampilkan histogram untuk setiap variabel numerik

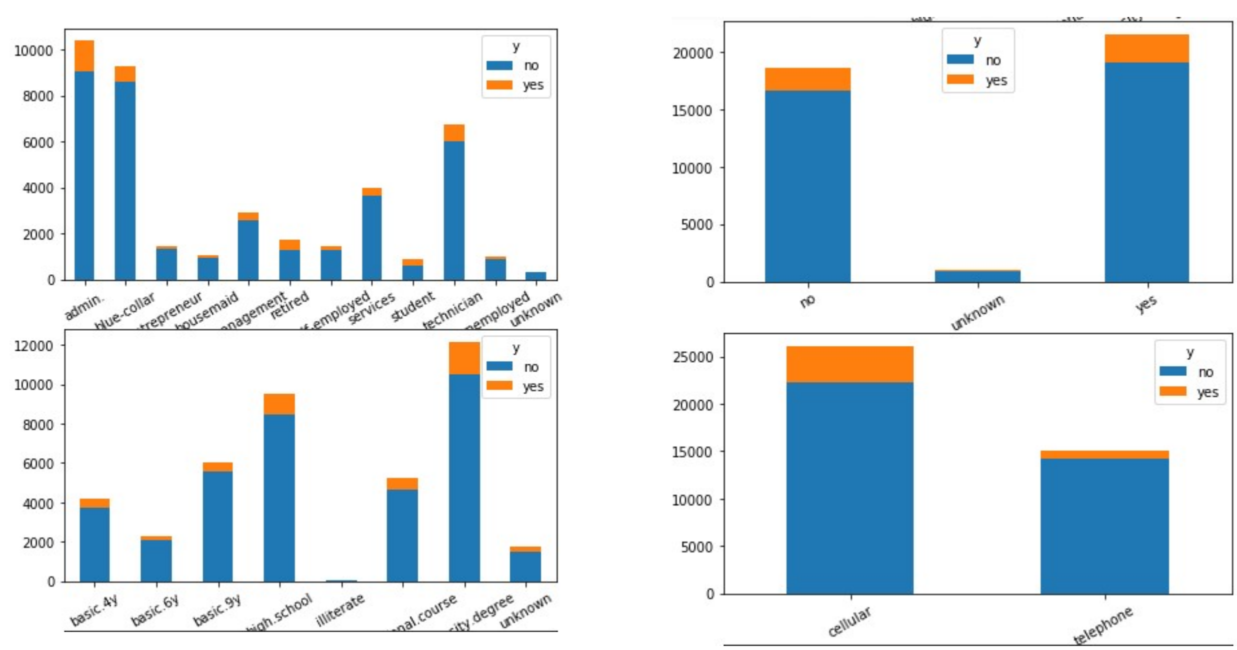
```python
import matplotlib.pyplot as plt
%matplotlib inline

n = len(data.select_dtypes(include='number').columns)
i = 0
for col in data.select_dtypes(include='number').columns.values:
    plt.subplot(n//2,2,i+1)
    data.groupby('y')[col].plot(kind='hist',stacked=True,figsize=(15,20))
    plt.gca().set_title(col)
    plt.legend()
    i += 1
plt.show()
```

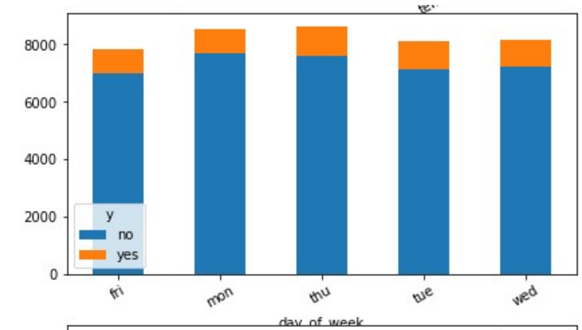




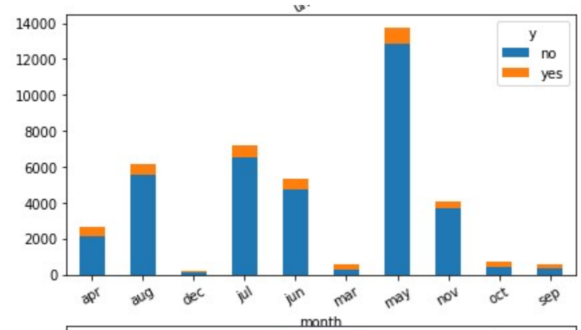
Menampilkan histogram untuk setiap kolom dengan data kategori

```
%matplotlib inline
import seaborn as sns
import matplotlib.pyplot as plt

n = len(data.select_dtypes(include='object').columns)
i = 0
for obj in data.select_dtypes(include='object'):
    ax = plt.subplot(6,2,i+1)
    data.groupby([obj, 'y']).size().unstack().plot(kind='bar', stacked=True, ax=ax,figsize=(15,25))
    plt.xticks(rotation=30)
    i += 1
plt.show()
```

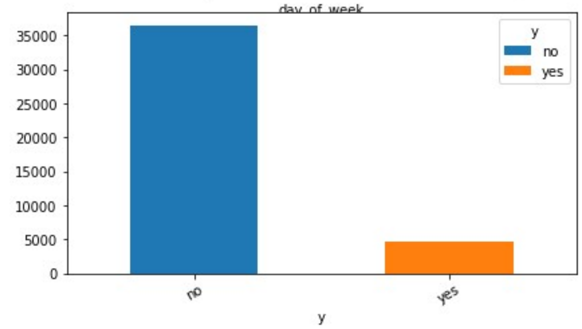




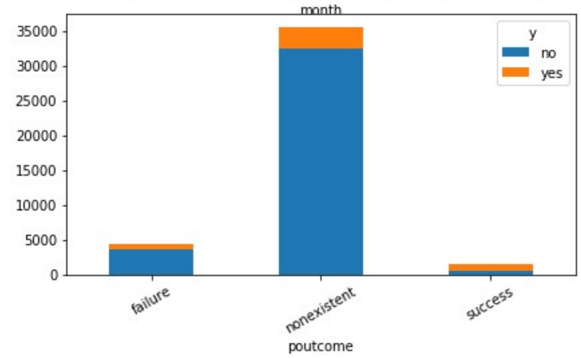
8000
6000
4000
2000
0
y
no
yes
fri
mon
thu
tue
wed
day_of_week



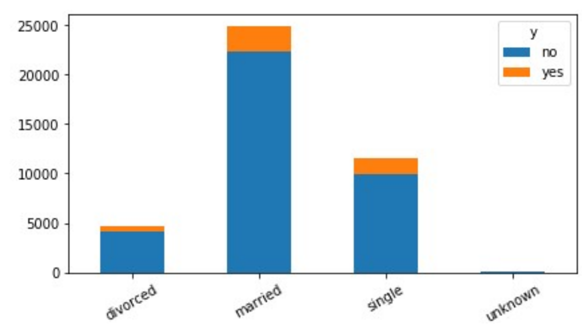
14000
12000
10000
8000
6000
4000
2000
0
y
no
yes
apr
aug
dec
jul
jun
mar
may
nov
oct
sep
month



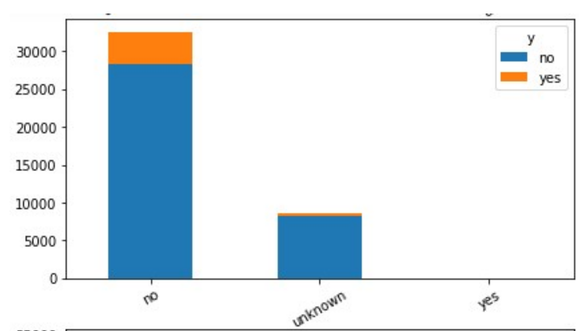
35000
30000
25000
20000
15000
10000
5000
0
y
no
yes
no
yes
y



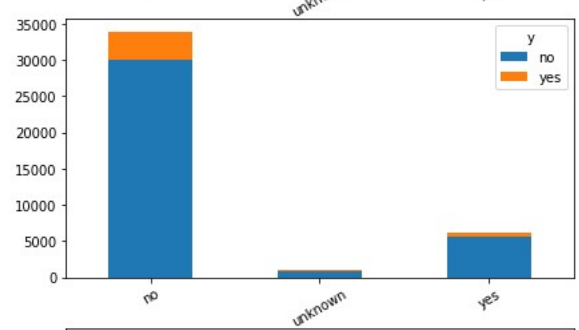
35000
30000
25000
20000
15000
10000
5000
0
y
no
yes
failure
nonexistent
success
poutcome



25000
20000
15000
10000
5000
0
y
no
yes
divorced
married
single
unknown



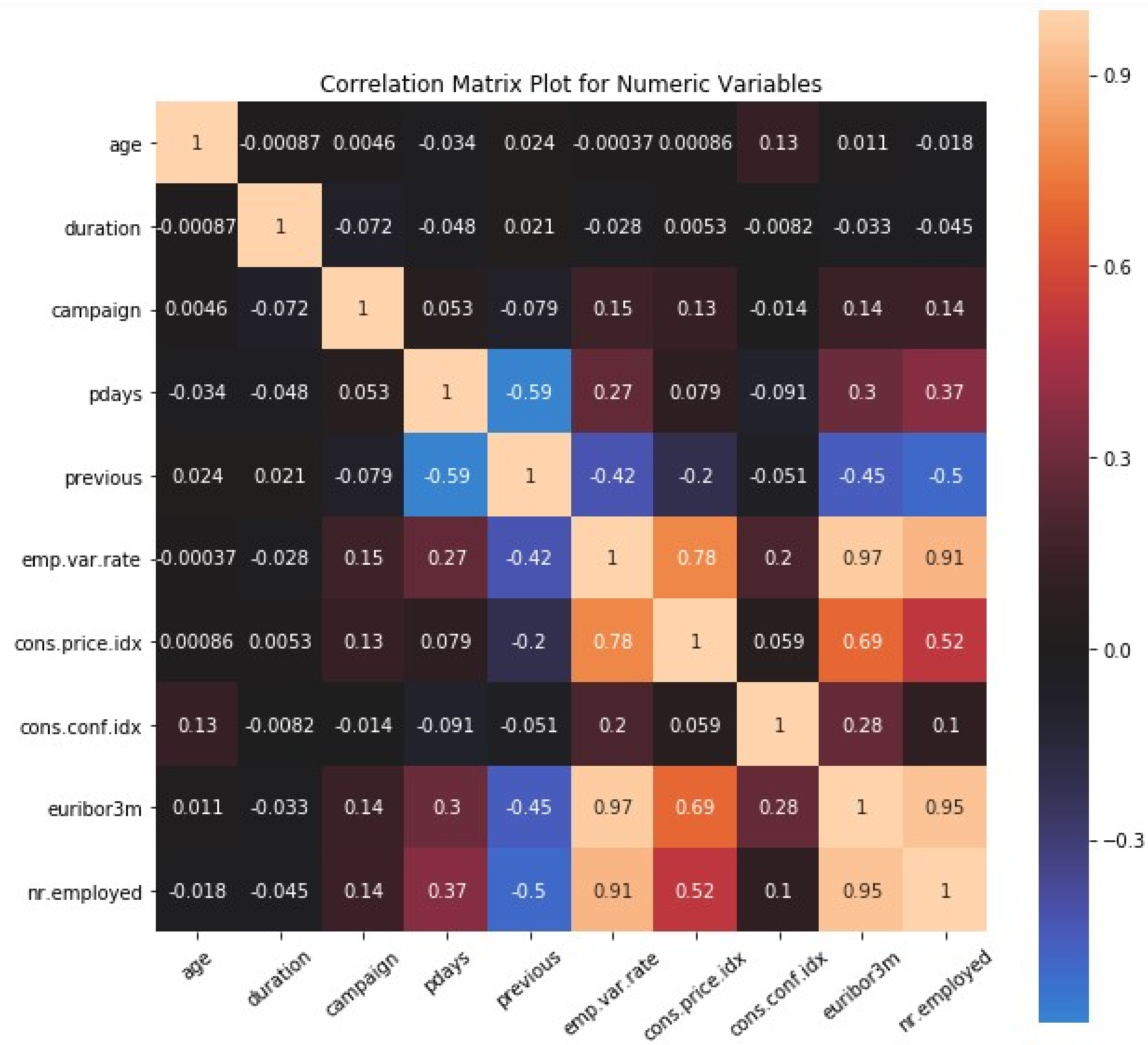
30000
25000
20000
15000
10000
5000
0
y
no
yes
no
unknown
yes



35000
30000
25000
20000
15000
10000
5000
0
y
no
yes
no
unknown
yes

Correlation Matrix

```
%matplotlib inline
plt.figure(figsize=(10, 10))
sns.heatmap(data.select_dtypes(include='number').corr(), cbar=True, square=True, annot=True, center=0)
plt.xticks(rotation=40)
plt.title('Correlation Matrix Plot for Numeric Variables')
plt.show()
```



Mencari Outliers

```
#Outliers

for i in ['age','job','marital','education','default',
          'housing','loan','contact','month','day_of_week',
          'duration','campaign','pdays','previous','poutcome',
          'emp.var.rate','cons.price.idx','cons.conf.idx',
          'euribor3m','nr.employed','y']:

    if (data[i].dtypes in ['int64','float64']):
        print('\nAttribute-',[i],':',data[i].dtypes)
        Q1=data[i].quantile(0.25)
        print('Q1',Q1)
        Q3=data[i].quantile(0.75)
        print('Q3',Q3)
        IQR=Q3-Q1
        print('IQR',IQR)
        min=data[i].min()
        max=data[i].max()
        min_IQR=Q1-1.5*IQR
        max_IQR=Q3+1.5*IQR
        if (min<min_IQR):
            print('Low outlier is found',min_IQR)
        if (max>max_IQR):
            print('High outlier is found',max_IQR)
```

```
Attribute- ['age'] : int64
Q1 32.0
Q3 47.0
IQR 15.0
High outlier is found 69.5

Attribute- ['duration'] : int64
Q1 102.0
Q3 319.0
IQR 217.0
High outlier is found 644.5

Attribute- ['campaign'] : int64
Q1 1.0
Q3 3.0
IQR 2.0
High outlier is found 6.0

Attribute- ['pdays'] : int64
Q1 999.0
Q3 999.0
IQR 0.0
Low outlier is found 999.0

Attribute- ['previous'] : int64
Q1 0.0
Q3 0.0
IQR 0.0
High outlier is found 0.0
```

```
Attribute- ['emp.var.rate'] : float64
Q1 -1.8
Q3 1.4
IQR 3.2

Attribute- ['cons.price.idx'] : float64
Q1 93.075
Q3 93.994
IQR 0.9189999999999969

Attribute- ['cons.conf.idx'] : float64
Q1 -42.7
Q3 -36.4
IQR 6.300000000000004
High outlier is found -26.949999999999992

Attribute- ['euribor3m'] : float64
Q1 1.344
Q3 4.961
IQR 3.617

Attribute- ['nr.employed'] : float64
Q1 5099.1
Q3 5228.1
IQR 129.0
```

Hasil Data Understanding:

- Memiliki 21 variabel (20 variabel input dan 1 variabel output) dan 41188 entri.
- Terdapat 12 Duplicated Data
- Dataset pada variabel prediksi (y) termasuk imbalance karena terdapat 88.7% yang tidak melakukan investasi untuk deposito berjangka sedangkan yang berinvestasi hanya 11.3%
- Variabel euribor3m dengan variabel emp.var.rate memiliki correlation value yang tinggi (0.97)

3. Data Preparation

Menghapus duplicated data

```
#Menghapus duplicated data

data_prep = data.drop_duplicates()
data_prep.shape
```

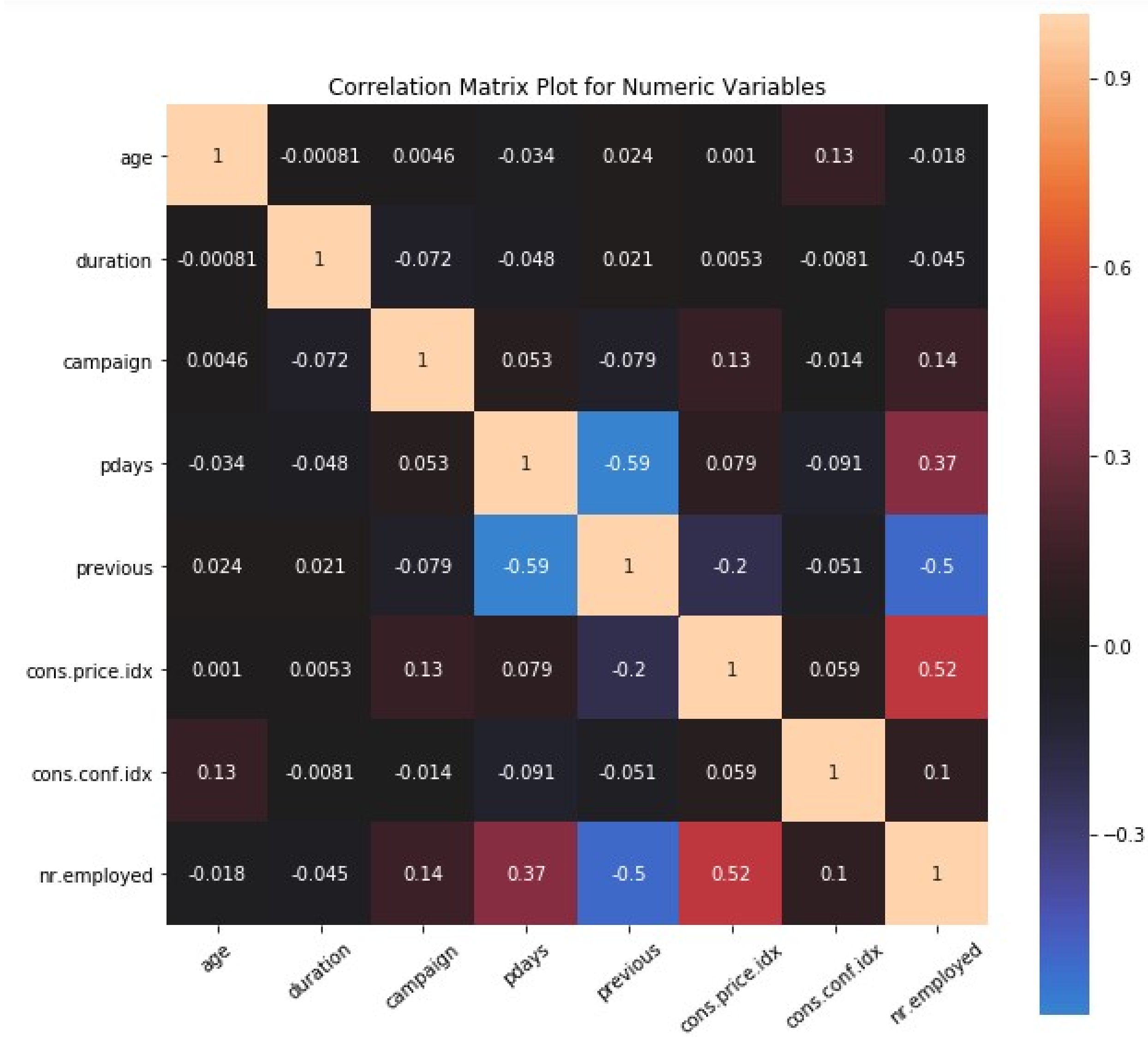
```
(41176, 21)
```

Berdasarkan hasil dari correlation matrix sebelumnya, dilakukan penghapusan dua variabel yang memiliki nilai correlation matrix yang tinggi

Berdasarkan hasil correlation matrix, variabel *emp.var.rate* dan *euribor3m* dihapus karena memiliki nilai correlation matrix yang tinggi

```
pd.options.mode.chained_assignment = None
data_prep.drop(['emp.var.rate','euribor3m'], axis=1, inplace=True)
```

Correlation matrix baru



Mengubah variabel numeric kedalam skala 0-1

```
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()
col = data_prep.select_dtypes(include='number').columns.tolist()
data_prep[col] = sc.fit_transform(data_prep[col])
data_prep.hist(figsize=(10,10))
data_prep.describe()
```

	age	duration	campaign	pdays	previous	cons.price.idx	cons.conf.idx	nr.employed
count	41176.000000	41176.000000	41176.000000	41176.000000	41176.000000	41176.000000	41176.000000	41176.000000
mean	0.284244	0.052525	0.028507	0.963428	0.024716	0.535744	0.430843	0.769130
std	0.128650	0.052726	0.050369	0.187124	0.070709	0.225580	0.193634	0.273162
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.185185	0.020740	0.000000	1.000000	0.000000	0.340608	0.338912	0.512287
50%	0.259259	0.036600	0.018182	1.000000	0.000000	0.603274	0.376569	0.859735
75%	0.370370	0.064864	0.036364	1.000000	0.000000	0.698753	0.602510	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Mengubah variabel dengan tipe data object menjadi numerik

```
#Mengubah variabel dengan tipe data object menjadi numerik
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
import numpy as np

oe = OrdinalEncoder(dtype=np.uint8)
col = data_prep.drop(['y'],axis=1).select_dtypes(include='object').columns.tolist()
data_prep[col] = oe.fit_transform(data_prep[col])

le = LabelEncoder()
data_prep['y'] = le.fit_transform(data_prep['y'])

data_prep.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	cons.price.idx	cor
0	0.481481	3	1	0	0	0	0	1	6	1	0.053070	0.0	1.0	0.0	1	0.698753	
1	0.493827	7	1	3	1	0	0	1	6	1	0.030297	0.0	1.0	0.0	1	0.698753	
2	0.246914	7	1	3	0	2	0	1	6	1	0.045954	0.0	1.0	0.0	1	0.698753	
3	0.283951	0	1	1	0	0	0	1	6	1	0.030704	0.0	1.0	0.0	1	0.698753	
4	0.481481	7	1	3	0	0	2	1	6	1	0.062424	0.0	1.0	0.0	1	0.698753	

4. Modeling

Modeling menggunakan beberapa algoritma

```
#Membuat model dengan beberapa algoritma
models = {'knn':KNeighborsClassifier(n_neighbors=1),
          'naive_bayes':GaussianNB(),
          'logit':LogisticRegression(solver="lbfgs", multi_class="auto", dual=False, max_iter=1200000),
          'svm':SVC(kernel="rbf", gamma="auto"),
          'decision_tree':DecisionTreeClassifier(),
          'random_forest':RandomForestClassifier(n_estimators=100),
}
```

```
x_data = data_prep.iloc[:, :-1]
y_data = data_prep.iloc[:, -1]
```

```
#Pembagian jumlah data training
#Membagi dataset menjadi data train dan data test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=123, stratify=y_data)
```

5. Evaluation

k-NN

```
# train the modelprint("[INFO] using '{}' model".format(model_name))
print("[INFO] using '{}' model".format("knn"))
model = models['knn']
model.fit(x_train, y_train)

y_pred_KNN = model.predict(x_test)
akurasi_KNN = metrics.accuracy_score(y_test, y_pred_KNN)*100
cm_KNN = confusion_matrix(y_test, y_pred_KNN)

# make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictions = model.predict(x_test)
print(classification_report(y_test, predictions, y_data))
```

```
    accuracy                           0.92 406939545
   macro avg       0.85      0.87      0.86 406939545
weighted avg       0.90      0.93      0.92 406939545
```

```
print(y_pred_KNN)
```

```
[1 0 0 ... 1 0 0]
```

```
print(cm_KNN)
```

```
[[10277   684]
 [  980   412]]
```

```
print(akurasi_KNN)
```

```
86.52958795434309
```

Naïve Bayes

```
# train the modelprint("[INFO] using '{}' model".format(model_name))
print("[INFO] using '{}' model".format("naive_bayes"))
model = models['naive_bayes']
model.fit(x_train, y_train)

y_pred_NB = model.predict(x_test)
akurasi_NB = metrics.accuracy_score(y_test, y_pred_NB)*100
cm_NB = confusion_matrix(y_test, y_pred_NB)

# make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictions = model.predict(x_test)
print(classification_report(y_test, predictions, y_data))
```

```
    accuracy                           0.91 406939545
   macro avg       0.87      0.86      0.87 406939545
weighted avg       0.93      0.90      0.91 406939545
```

```
print(y_pred_NB)
```

```
[1 0 0 ... 1 0 0]
```

```
print(cm_NB)
```

```
[[9948 1013]
 [ 706  686]]
```

```
print(akurasi_NB)
```

```
86.08435197927629
```

Logistic Regression

```
# train the modelprint("[INFO] using '{}' model".format(model_name))
print("[INFO] using '{}' model".format("logit"))
model = models['logit']
model.fit(x_train, y_train)

y_pred_LR = model.predict(x_test)
akurasi_LR = metrics.accuracy_score(y_test, y_pred_LR)*100
cm_LR = confusion_matrix(y_test, y_pred_LR)

# make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictions = model.predict(x_test)
print(classification_report(y_test, predictions, y_data))
```

```
    accuracy                           0.94 406939545
   macro avg       0.90      0.91      0.90 406939545
weighted avg       0.92      0.97      0.94 406939545
```

```
print(y_pred_LR)
```

```
[0 0 0 ... 1 0 0]
```

```
print(cm_LR)
```

```
[[10706   255]
 [  877   515]]
```

```
print(akurasi_LR)
```

```
90.83623411317089
```

SVM

```
# train the modelprint("[INFO] using '{}' model".format(model_name))
print("[INFO] using '{}' model".format("svm"))
model = models['svm']
model.fit(x_train, y_train)

y_pred_SVM = model.predict(x_test)
akurasi_SVM = metrics.accuracy_score(y_test, y_pred_SVM)*100
cm_SVM = confusion_matrix(y_test, y_pred_SVM)

# make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictions = model.predict(x_test)
print(classification_report(y_test, predictions, y_data))
```

```
    accuracy                           0.94 406939545
   macro avg       0.88      0.90      0.87 406939545
weighted avg       0.90      0.98      0.93 406939545
```

```
print(y_pred_SVM)
```

```
[1 0 0 ... 0 0 0]
```

```
print(cm_SVM)
```

```
[[10830   131]
 [ 1129   263]]
```

```
print(akurasi_SVM)
```

```
89.80004857119728
```

Decision Tree

```
# train the modelprint("[INFO] using '{}' model".format(model_name))
print("[INFO] using '{}' model".format("decision_tree"))
model = models['decision_tree']
model.fit(x_train, y_train)

y_pred_DT = model.predict(x_test)
akurasi_DT = metrics.accuracy_score(y_test, y_pred_DT)*100
cm_DT = confusion_matrix(y_test, y_pred_DT)

# make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictions = model.predict(x_test)
print(classification_report(y_test, predictions, y_data))
```

```
    accuracy                           0.93 406939545
   macro avg       0.89      0.88      0.88 406939545
weighted avg       0.93      0.92      0.93 406939545
```

```
print(y_pred_DT)
```

```
[0 0 0 ... 1 0 0]
```

```
print(cm_DT)
```

```
[[10203   758]
 [  686   706]]
```

```
print(akurasi_DT)
```

```
88.31053185461022
```

Random Forest

```
# train the modelprint("[INFO] using '{}' model".format(model_name))
print("[INFO] using '{}' model".format("random_forest"))
model = models['random_forest']
model.fit(x_train, y_train)

y_pred_RF = model.predict(x_test)
akurasi_RF = metrics.accuracy_score(y_test, y_pred_RF)*100
cm_RF = confusion_matrix(y_test, y_pred_RF)

# make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictions = model.predict(x_test)
print(classification_report(y_test, predictions, y_data))
```

```
    accuracy                        0.95 406939545
   macro avg       0.91      0.91      0.91 406939545
weighted avg       0.94      0.96      0.95 406939545
```

```
print(y_pred_RF)
```

```
[0 0 0 ... 1 0 0]
```

```
print(cm_RF)
```

```
[[10569   392]
 [  677   715]]
```

```
print(akurasi_RF)
```

```
91.34623168461103
```

Perbandingan akurasi dari setiap algoritma

```
print("KNN : ", akurasi_KNN)
print("Naïve Bayes : ", akurasi_NB)
print("Logistic Regression : ", akurasi_LR)
print("Support Vector Machines (SVMs) : ", akurasi_SVM)
print("Decision Trees : ", akurasi_DT)
print("Random Forests : ", akurasi_RF)
```

```
KNN :  86.52958795434309
Naïve Bayes :  86.08435197927629
Logistic Regression :  90.83623411317089
Support Vector Machines (SVMs) :  89.80004857119728
Decision Trees :  88.31053185461022
Random Forests :  91.34623168461103
```

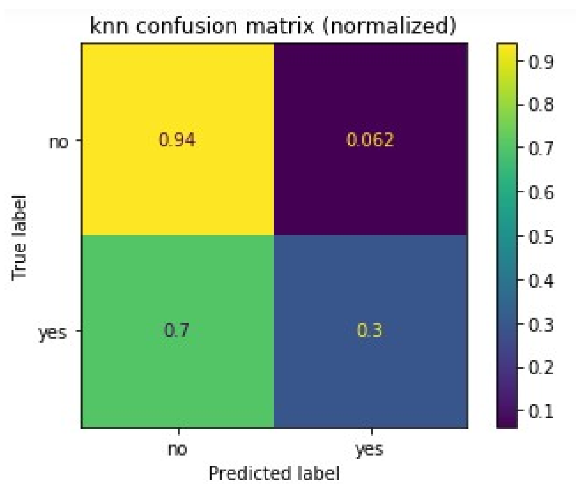
Didapatkan bahwa algoritma **Random Forest** memiliki akurasi yang paling tinggi

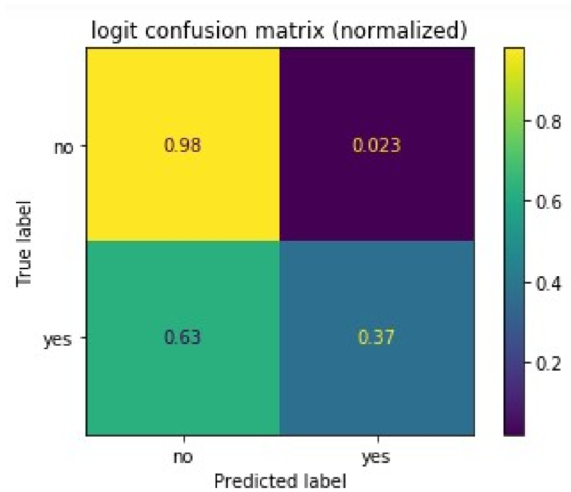
Mencari confusion matrix untuk setiap algoritma yang sudah di lakukan normalisasi

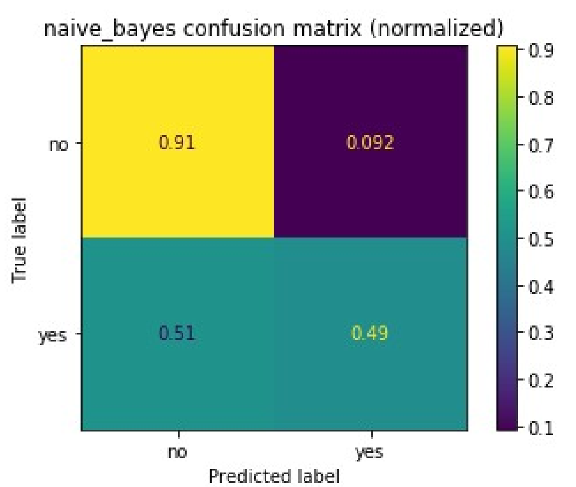
```
for model_name, model in models.items():
    model.fit(x_train, y_train)
    print(model_name,'model training: done!')
    models[model_name] = model
```

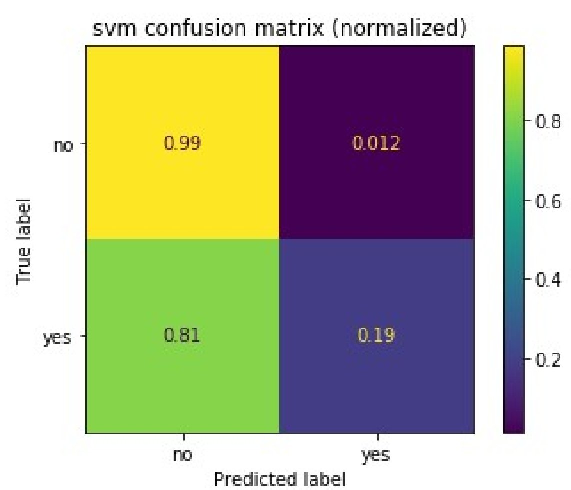
```
knn model training: done!
naive_bayes model training: done!
logit model training: done!
svm model training: done!
decision_tree model training: done!
random_forest model training: done!
```

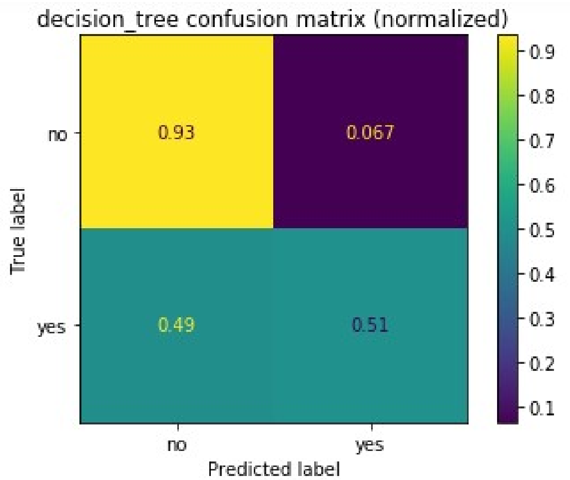
```
from sklearn.metrics import plot_confusion_matrix, balanced_accuracy_score

idx = []
bacc_scores = []

for model_name, model in models.items():
    # Normalisasi confusion matrix
    plot_confusion_matrix(model, x_test, y_test,
                          display_labels=['no','yes'],
                          normalize='true')
    plt.title(model_name + ' confusion matrix (normalized)')

    idx.append(model_name)
    bacc_scores.append(balanced_accuracy_score(y_test,model.predict(x_test)))
plt.show()
```

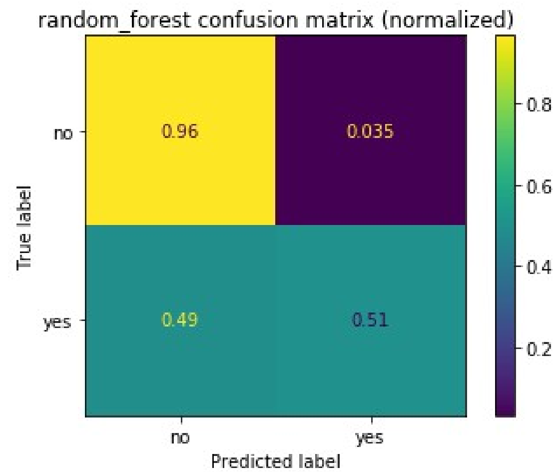












Perbandingan Balanced Accuracy untuk setiap algoritma

```
#Menampilkan balanced accuracy dari setiap algoritma
pd.DataFrame(list(zip(bacc_scores)),index=idx,columns=['Balanced Accuracy'])
```

	Balanced Accuracy
knn	0.616787
naive_bayes	0.700199
logit	0.673353
svm	0.588493
decision_tree	0.719437
random_forest	0.735174

Karena pada akurasi dan balanced accuracy didapatkan bahwa algoritma **Random Forest** memiliki nilai yang terbesar dari keduanya, maka algoritma ini yang akan digunakan untuk prediksi dataset pada bagian deployment

```
#Hyperparameter tuning untuk algoritma Random Forest
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold

cross_val = StratifiedKFold(n_splits=5, shuffle=True, random_state=123)
param = {'bootstrap': True,
              'criterion': 'mse',
              'max_depth': None,
              'max_features': 'auto',
              'max_leaf_nodes': None,
              'min_impurity_decrease': 0.0,
              'min_impurity_split': None,
              'min_samples_leaf': 1,
              'min_samples_split': 2,
              'min_weight_fraction_leaf': 0.0,
              'n_estimators': 10,
              'n_jobs': 1,
              'oob_score': False,
              'random_state': 42,
              'verbose': 0,
              'warm_start': False}

base_model = RandomForestRegressor(n_estimators = 10, random_state = 42)
base_model.fit(x_train, y_train)
y_pred=base_model.predict(x_test)
```

6. Deployment

Program sederhana untuk memprediksi apakah klien akan berinvestasi atau tidak sesuai dengan ketentuan variabel yang ada pada dataset

```python
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import OrdinalEncoder
import numpy as np

def prediksi_berinvestasi_atau_tidak(data):

    data = data.drop_duplicates()

    data.drop(['emp.var.rate','euribor3m'], axis=1, inplace=True)

    numeric = data.select_dtypes(include='number').columns.tolist()
    data[numeric] = sc.transform(data[numeric])

    obj = data.select_dtypes(include='object').columns.tolist()
    data[obj] = oe.transform(data[obj])

    # Prediksi
    idx = 0
    for result in base_model.predict(data):
        if result == 0:
            print('Klien ',data.index[idx],' diprediksi tidak akan berinvestasi.')
        else:
            print('Klien ',data.index[idx],' diprediksi akan berinvestasi')
        idx += 1
```

Dilakukan generate data sebanyak 10 untuk diuji

```python
new_data = data.drop(['y'],axis=1)

n_ppl = 10 # Jumlah data yang akan digenerate
data_sync = pd.DataFrame()
for col in new_data.columns:
    data_sync = pd.concat([data_sync, new_data[col].sample(n=n_ppl).to_frame().reset_index().drop(['index'],axis=1)],axis=1)
data_sync
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	e
0	47	blue-collar	married	professional.course	no	yes	no	cellular	nov	wed	21	10	999	0	nonexistent	
1	55	blue-collar	single	basic.9y	no	no	no	telephone	jul	thu	250	3	999	0	nonexistent	
2	45	admin.	married	basic.6y	no	yes	no	cellular	aug	thu	530	1	999	0	nonexistent	
3	30	entrepreneur	single	university.degree	no	no	no	cellular	jun	wed	709	3	999	0	nonexistent	
4	31	blue-collar	married	basic.9y	unknown	yes	no	telephone	may	thu	402	11	999	0	nonexistent	
5	34	admin.	single	professional.course	no	yes	no	cellular	jun	thu	692	6	999	0	nonexistent	
6	26	admin.	single	basic.4y	no	yes	no	cellular	jun	mon	368	1	999	0	nonexistent	
7	38	admin.	married	basic.6y	unknown	yes	no	cellular	may	tue	189	3	999	0	failure	
8	36	blue-collar	married	high.school	unknown	yes	yes	telephone	may	wed	155	1	999	2	nonexistent	
9	34	blue-collar	married	professional.course	no	no	yes	cellular	may	wed	1363	1	999	0	nonexistent	

Hasil prediksi dengan data yang sudah degenerate sebelumnya

```
prediksi_berinvestasi_atau_tidak(data_sync)
```

```
Klien  0  diprediksi tidak akan berinvestasi.
Klien  1  diprediksi akan berinvestasi
Klien  2  diprediksi akan berinvestasi
Klien  3  diprediksi akan berinvestasi
Klien  4  diprediksi akan berinvestasi
Klien  5  diprediksi akan berinvestasi
Klien  6  diprediksi tidak akan berinvestasi.
Klien  7  diprediksi tidak akan berinvestasi.
Klien  8  diprediksi tidak akan berinvestasi.
Klien  9  diprediksi akan berinvestasi
```