Bank_Marketing_Raport

February 8, 2021

1 Bank Marketing Data Science Solution

In this notebok, will be presented step-by-step analysis, explaining each step for the best understanding problem and solution. The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. Benefit of creating model should be to increase the efficiency of the choice of customers to whom the offer will be presented what it means separating group of customers who can be targeted with specific offer.

1.1 Workflow stages

In this notebook, workflow goes through six stages:

1. Definition of problem.
2. Obtaining data.
3. Analyze, identify patterns and explore the data.
4. Prepare and cleanse the data.
5. Model, predict and solve the problem.
6. Visualize, report, and present the problem solving steps and final solution.

Additionally, we may combine workflow stage, for example analyze by visualizing data. Also stage may perform multiple times, for example visualizing data.

Based on dataset, our model, creating in this notebook, should predict if the client will subscribe a term deposit.

1.2 Workflow goals

To better undestanding proccess our analysis, below, main goals were presented. There are steps which we should always use.

There is a couple of main goals:

Classifying:

We may classify or categorize our sample.

Correlating:

Witch feature in our dataset significantly affect to our goal? Is there correlation amont feature and goal or features? This can be tested for numberical (function corr()) and categorical features

(function phik_matrix() - it will be explained during analysis). Correlating may help in creating or correcting features.

Converting:

All features have to be converted to numerical values. For example text variables converting to numeric.

Completing:

Data mostly have missing values so it require us to fill any missing value in features. Thanks to this, model may work best.

Correcting:

We may analyze dataset for possible error or outliers and remove samles. Also we can discard if this feature skew the results.

Creating:

We may create new features based on existing feature.

Charting:

Selecting the right visualisaton plots and charts.

1.3 Description of dataset

1.3.1 Data consists of 20 inputs which we can separate to:

Bank client data:

1. age
2. job : type of job
3. marital : marital status
4. education: type of education
5. default: has credit in default?
6. housing: has housing loan?
7. loan: has personal loan

Related with the last contact of the current campaign:

8. contact: contact communication type
9. month: last contact month of year
10. day_of_week: last contact day of the week
11. duration: last contact duration, in seconds

Other attributes:

12. campaign: number of contacts performed during this campaign and for this client)
13. pdays: number of days that passed by after the client was last contacted from a previous campaign
14. previous: number of contacts performed before this campaign and for this client
15. poutcome: outcome of the previous marketing campaign

Social and economic context attributes

16. emp.var.rate: employment variation rate
17. cons.price.idx: consumer price index
18. cons.conf.idx: consumer confidence index
19. euribor3m: euribor 3 month rate
20. nr.employed: number of employees

1.3.2 1 output (target)

21. y - has the client subscribed a term deposit?

1.4 Load Data

Thanks to Pandas packages, we load data. We create copy our data to work on copy dataset. Additionally we create dataframe for names to receive quick access during our analysis.

[5]:

	Column	Explanation
1	age	(numeric)
2	job	type of job (categorical: "admin.","blue-collar","entrepreneur","housemaid","management","retired","self-employed","services","student","technici…
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	
6	housing	
7	loan	
8	contact	
9	month	
10	day_of_week	
11	duration	
12	campaign	
13	pdays	
14	previous	
15	poutcome	
16	emp.var.rate	
17	cons.price.idx	
18	cons.conf.idx	
19	euribor3m	
20	nr.employed	

```
5
has credit in default? (categorical: "no","yes","unknown")
6
has housing loan? (categorical: "no","yes","unknown")
7
has personal loan? (categorical: "no","yes","unknown")
8
contact communication type (categorical: "cellular","telephone")
9                                                                    last
contact month of year (categorical: "jan", "feb", "mar", …, "nov", "dec")
10
last contact day of the week (categorical: "mon","tue","wed","thu","fri")
11   last contact duration, in seconds (numeric). Important note:  this
attribute highly affects the output target (e.g., if duration=0 then y="no").
…
12                                          number of contacts performed
during this campaign and for this client (numeric, includes last contact)
13    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                                                                    number
of contacts performed before this campaign and for this client (numeric)
15                                                 outcome of the
previous marketing campaign (categorical: "failure","nonexistent","success")
16
employment variation rate - quarterly indicator (numeric)
17
consumer price index - monthly indicator (numeric)
18
consumer confidence index - monthly indicator (numeric)
19
euribor 3 month rate - daily indicator (numeric)
20
number of employees - quarterly indicator (numeric)
```

1.5 Analyze by describing data

Pandas helps us decsribe dataset. Now we check all available feature in our dataset.

```
['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']
```

Fist of all, rename output y to obvious name

Check number of rows and columns in our data

```
Dataset consists of 41188 rows and 21 columns
```

Categorical features

These values classify the samples into sets of similar samples.Categorical features are the values nominal, ordinal, ratio or interval.

- Categorcal: job, marital, housing, loan, contact, month, day of week, term deposit, poutcome
- Ordinal: education

Numerical features

These values change from sample to sample. Within numerical features are the values discrete, continuous or timeseries.

- Continous: age, duration, campaign, pdays, previos, nr.employed
- Discrete: emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m

```
Number of categorical columns: 11,
list this columns: ['job', 'marital', 'education', 'default', 'housing', 'loan',
'contact', 'month', 'day_of_week', 'poutcome', 'term deposit'].

Number of numeric columns: 10,
list this columns: ['age', 'duration', 'campaign', 'pdays', 'previous',
'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'].
```

We can conclude from the above results that half of our data is categorical (object), for this reason, during data cleaning and feature engineering we can check exactly this columns and tranform them to predict model.

[10]:

	age	job	marital	education	default	housing	loan	contact	\
0	56	housemaid	married	basic.4y	no	no	no	telephone	
1	57	services	married	high.school	unknown	no	no	telephone	
2	37	services	married	high.school	no	yes	no	telephone	
3	40	admin.	married	basic.6y	no	no	no	telephone	
4	56	services	married	high.school	no	no	yes	telephone	

	month	day_of_week	…	campaign	pdays	previous	poutcome	emp.var.rate	\
0	may	mon	…	1	999	0	nonexistent	1.1	
1	may	mon	…	1	999	0	nonexistent	1.1	
2	may	mon	…	1	999	0	nonexistent	1.1	
3	may	mon	…	1	999	0	nonexistent	1.1	
4	may	mon	…	1	999	0	nonexistent	1.1	

	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	term deposit
0	93.994	-36.4	4.857	5191.0	no
1	93.994	-36.4	4.857	5191.0	no
2	93.994	-36.4	4.857	5191.0	no
3	93.994	-36.4	4.857	5191.0	no
4	93.994	-36.4	4.857	5191.0	no

```
[5 rows x 21 columns]
```

Any of features are mixed data types, all features are numeric and alphanumeric.

We can conclude base on describe of numerical values:

- mean of age is 40, as we can see, age has propably outliers (98 old),
- contact in this campaign also has outliers,
- most of the clients in this campaign had had no contact with the bank previous.
- consumer confidence index is value above 0. This can be mistake because all of values in this feature is below 0

[11]:

	count	mean	std	min	25%	\
age	41188.0	40.024060	10.421250	17.000	32.000	
duration	41188.0	258.285010	259.279249	0.000	102.000	
campaign	41188.0	2.567593	2.770014	1.000	1.000	
pdays	41188.0	962.475454	186.910907	0.000	999.000	
previous	41188.0	0.172963	0.494901	0.000	0.000	
emp.var.rate	41188.0	0.081886	1.570960	-3.400	-1.800	
cons.price.idx	41188.0	93.575664	0.578840	92.201	93.075	
cons.conf.idx	41188.0	-40.502600	4.628198	-50.800	-42.700	
euribor3m	41188.0	3.621291	1.734447	0.634	1.344	
nr.employed	41188.0	5167.035911	72.251528	4963.600	5099.100	

	50%	75%	max
age	38.000	47.000	98.000
duration	180.000	319.000	4918.000
campaign	2.000	3.000	56.000
pdays	999.000	999.000	999.000
previous	0.000	0.000	7.000
emp.var.rate	1.100	1.400	1.400
cons.price.idx	93.749	93.994	94.767
cons.conf.idx	-41.800	-36.400	-26.900
euribor3m	4.857	4.961	5.045
nr.employed	5191.000	5228.100	5228.100

We can conclude base on describe of categorical values:

- feature job has the most unique values from categorical
- freature default, housing, loan and poutcome have 3 unique values, yes, no and unknown
- term deposit is categorical with yes or no values

[12]:

	count	unique	top	freq
job	41188	12	admin.	10422
marital	41188	4	married	24928
education	41188	8	university.degree	12168
default	41188	3	no	32588
housing	41188	3	yes	21576
loan	41188	3	no	33950
contact	41188	2	cellular	26144
month	41188	10	may	13769
day_of_week	41188	5	thu	8623

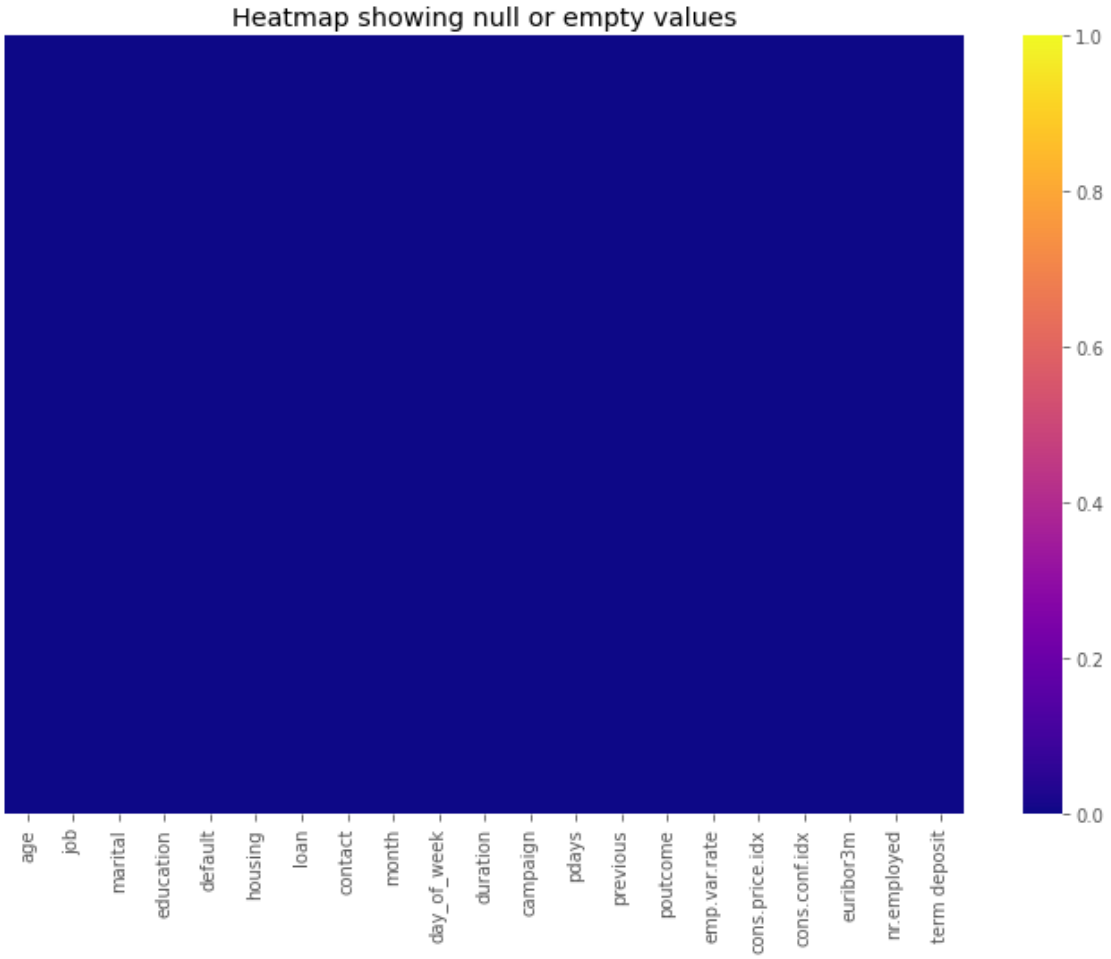
```
poutcome      41188      3        nonexistent  35563
term deposit  41188      2                 no  36548
```

```
job: ['housemaid' 'services' 'admin.' 'blue-collar' 'technician' 'retired'
 'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
 'student']
marital: ['married' 'single' 'divorced' 'unknown']
education: ['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professional.course'
 'unknown' 'university.degree' 'illiterate']
default: ['no' 'unknown' 'yes']
housing: ['no' 'yes' 'unknown']
loan: ['no' 'yes' 'unknown']
contact: ['telephone' 'cellular']
month: ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar' 'apr' 'sep']
day_of_week: ['mon' 'tue' 'wed' 'thu' 'fri']
poutcome: ['nonexistent' 'failure' 'success']
term deposit: ['no' 'yes']
```

As we can see, categorical variables doesn't have much variation in variables so, at this stage we can't reject some of features.

Check information about missing values (NaN) in specific columns

[14]:
```
Text(0.5, 1.0, 'Heatmap showing null or empty values')
```




```
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
```

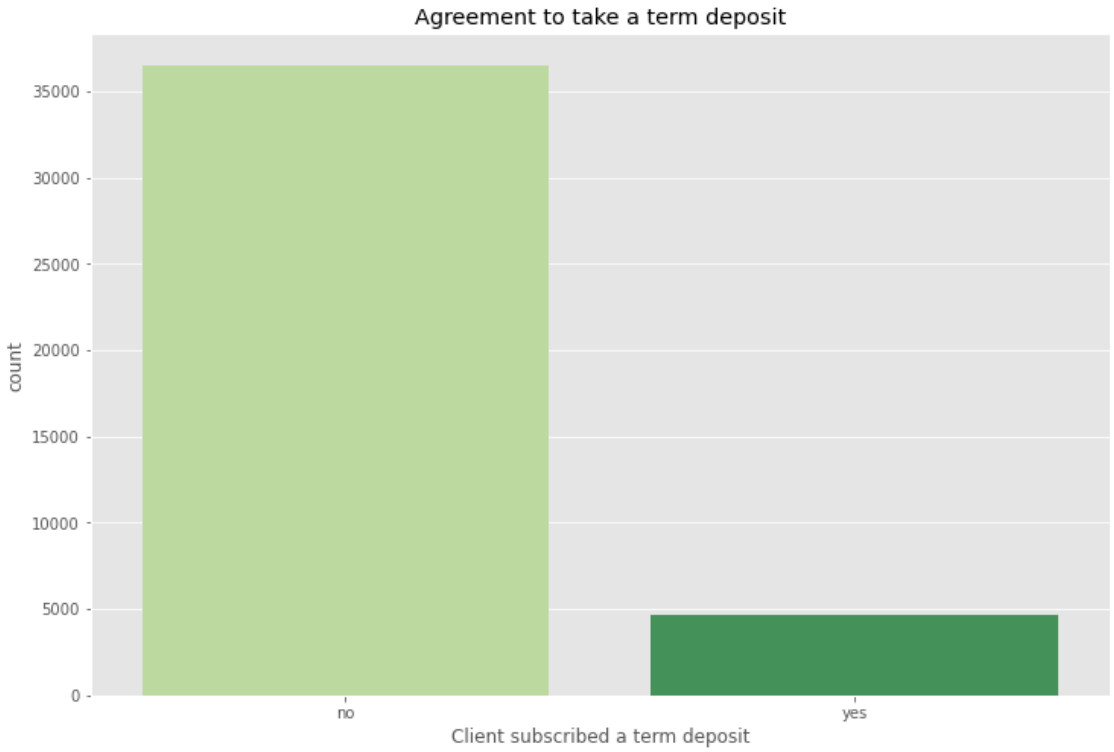
```
cons.conf.idx     0
euribor3m         0
nr.employed       0
term deposit      0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  term deposit    41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 4.9+ MB
```

There we can see, our data not include missing value (NaN). Our data has missing value coded by string, during analysis we will fill it.

1.6 Analyze by pivoting and vizualization

Make a hypothesis that overwhelming majority of answers if client subscribed a term deposit will be 'no'.

[16]:
```
Text(0.5, 1.0, 'Agreement to take a term deposit')
```



[17]:
```
term deposit
no              0.887346
yes             0.112654
dtype: float64
```

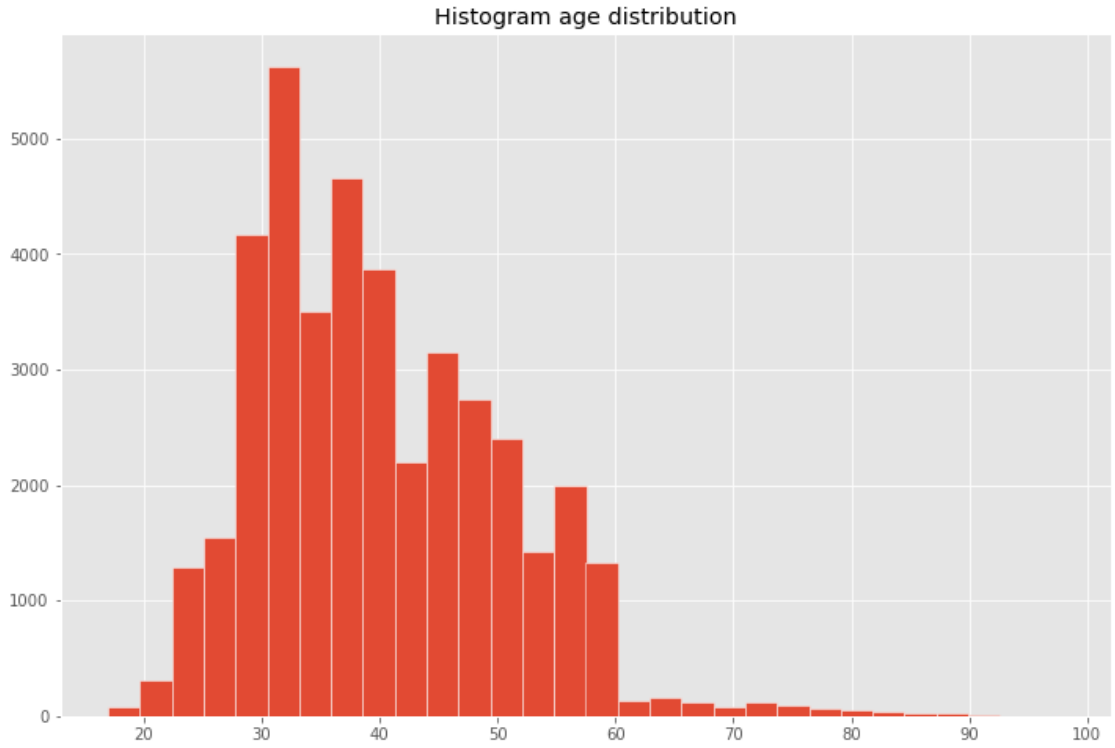
Countplot confirm our hypothesis. Here is not balance, so accuracy not will be great measure, looking at accuracy, our score will should above 88%. Better option will be for example F-score, which is calculated:

$$F = 2 * \frac{precision * recall}{precision + recall}$$

1.6.1 Age

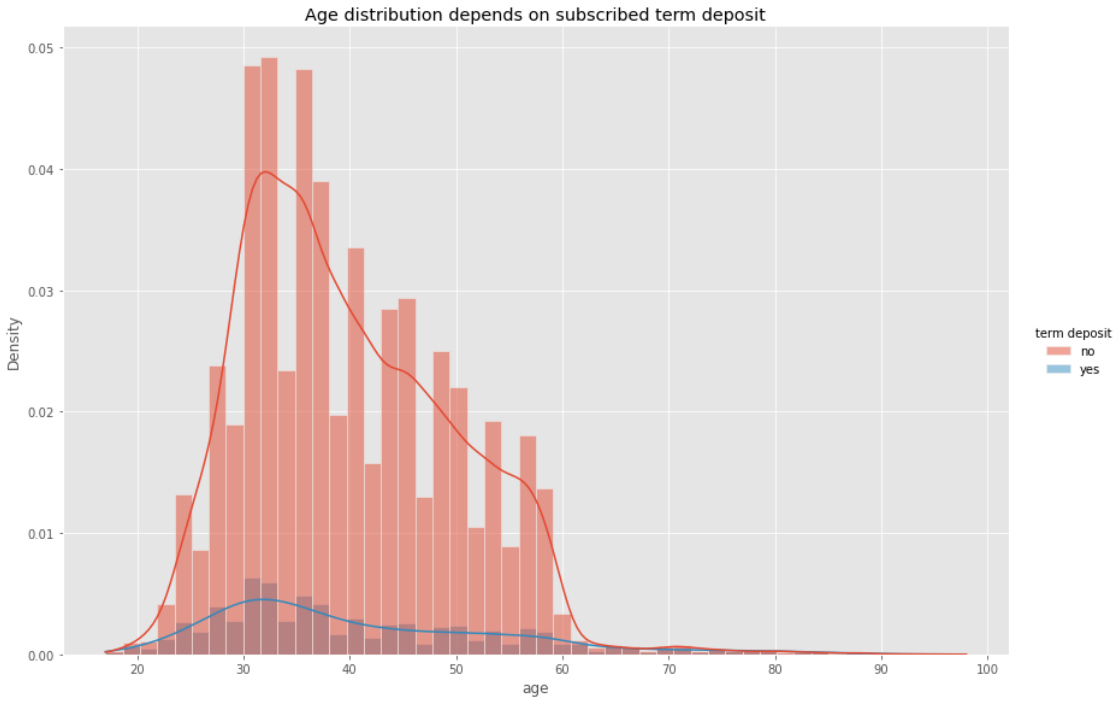
In this visualization will be presented feature age, means the age of the people who were called by bank representatives.

[18]:
```
Text(0.5, 1.0, 'Histogram age distribution')
```



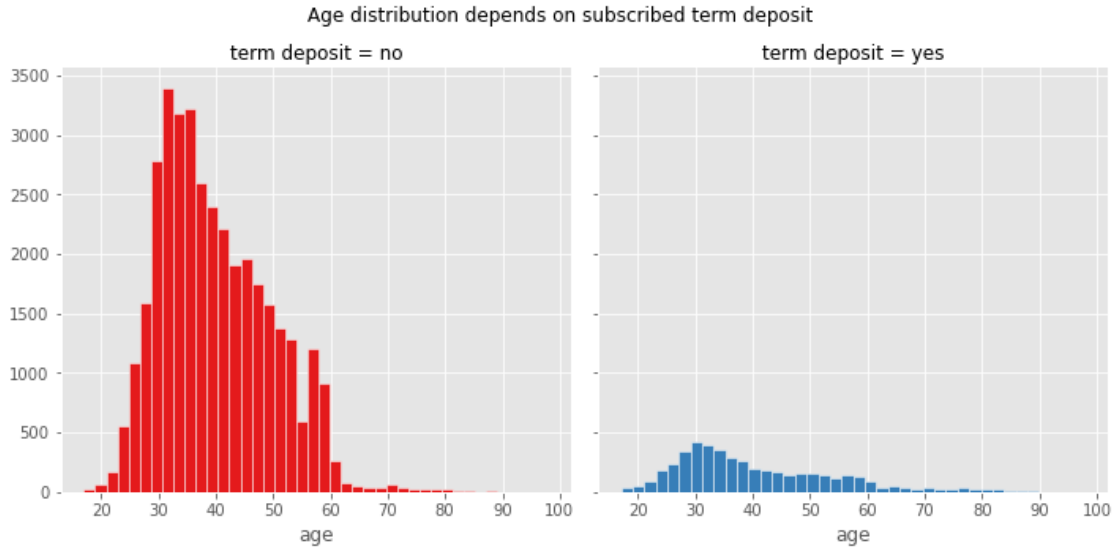
[19]:
```
75       41
83       49
88       49
129      41
139      45
         ..
41174    62
41178    62
41181    37
41183    73
41186    44
Name: age, Length: 4640, dtype: int64
```

[20]:
```
Text(0.5, 1.0, 'Age distribution depends on subscribed term deposit')
```

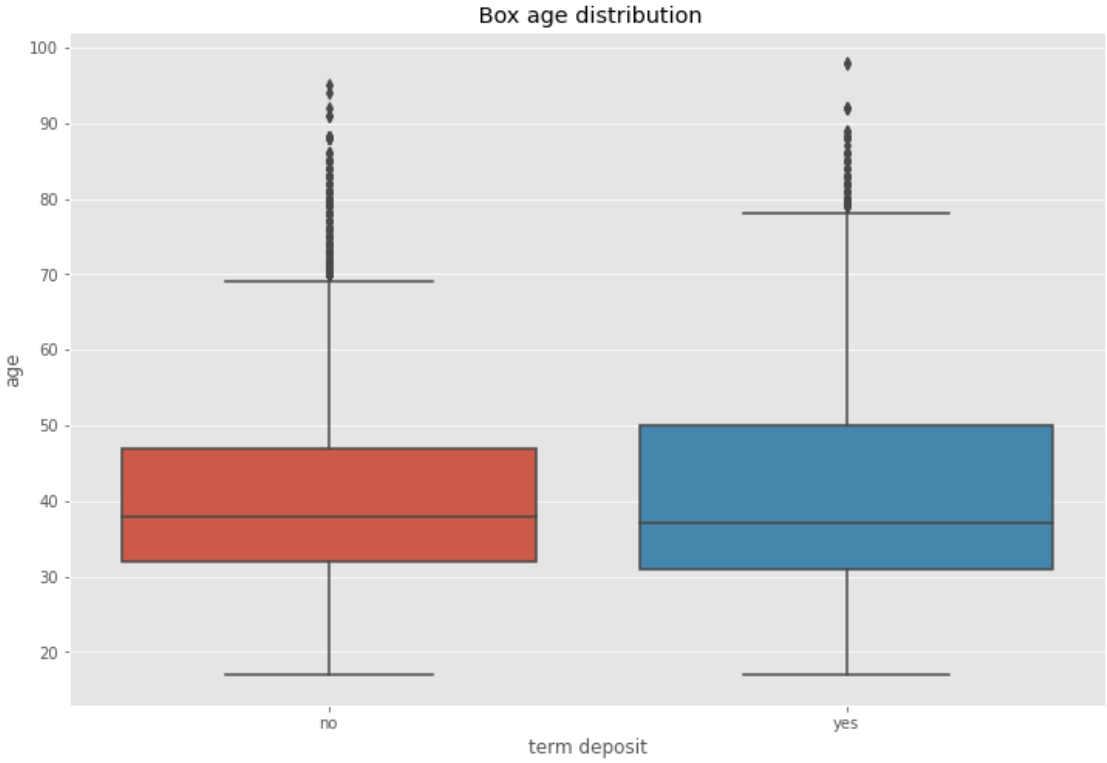


To see exactly how to unfolds age above 60, we create 2 plots depends on term deposit

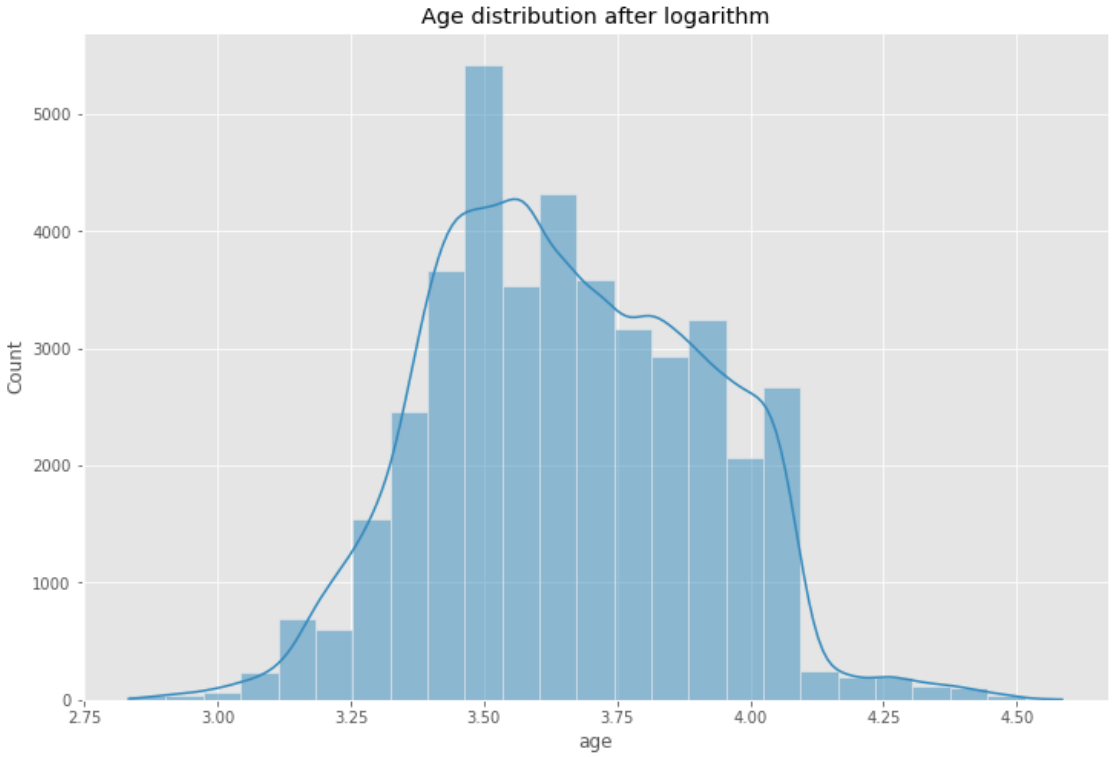
[21]: `Text(0.5, 0.98, 'Age distribution depends on subscribed term deposit')`



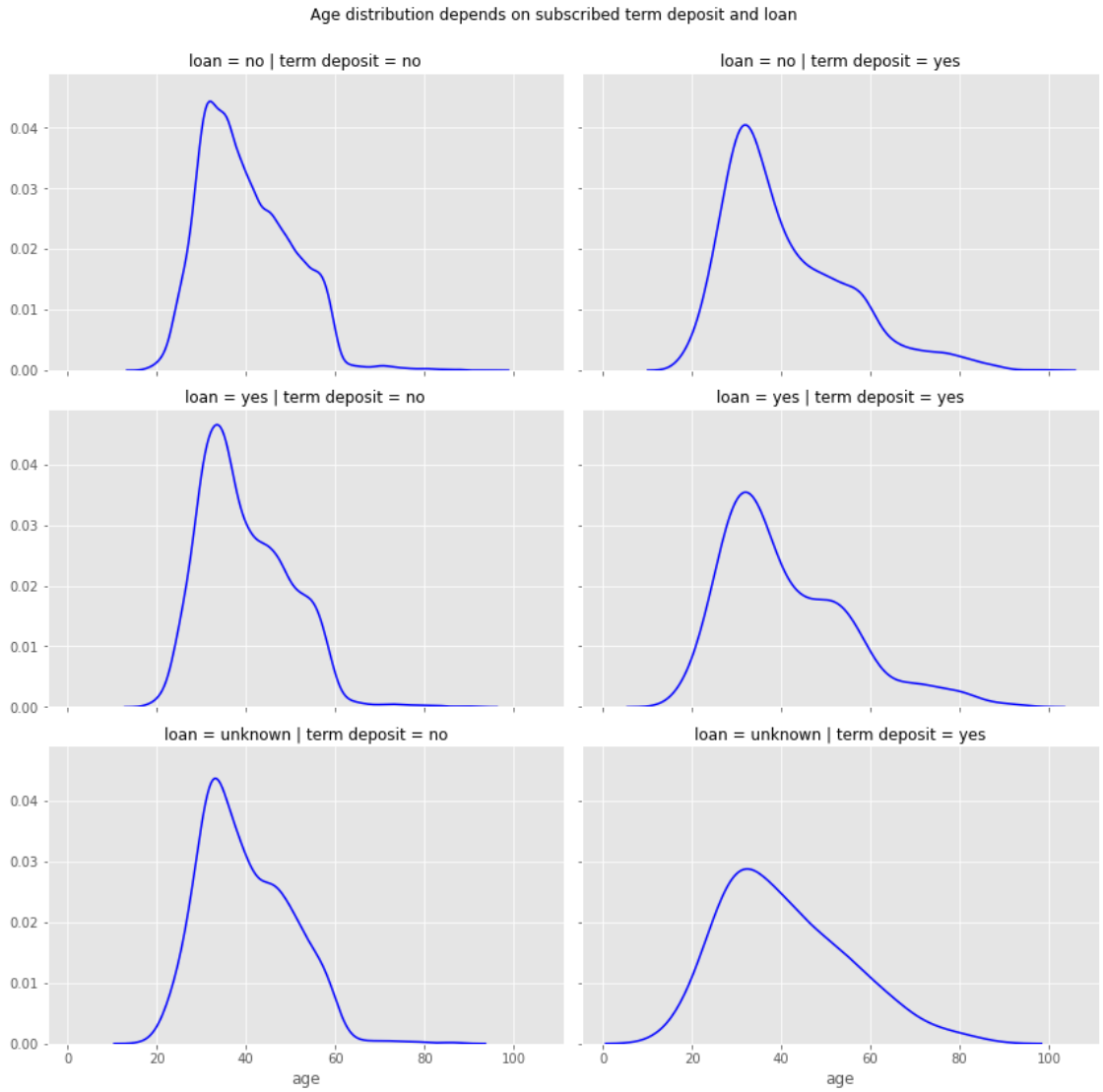
[22]: `Text(0.5, 1.0, ' Box age distribution')`



[23]: Text(0.5, 1.0, 'Age distribution after logarithm')



[24]: Text(0.5, 0.98, 'Age distribution depends on subscribed term deposit and loan')



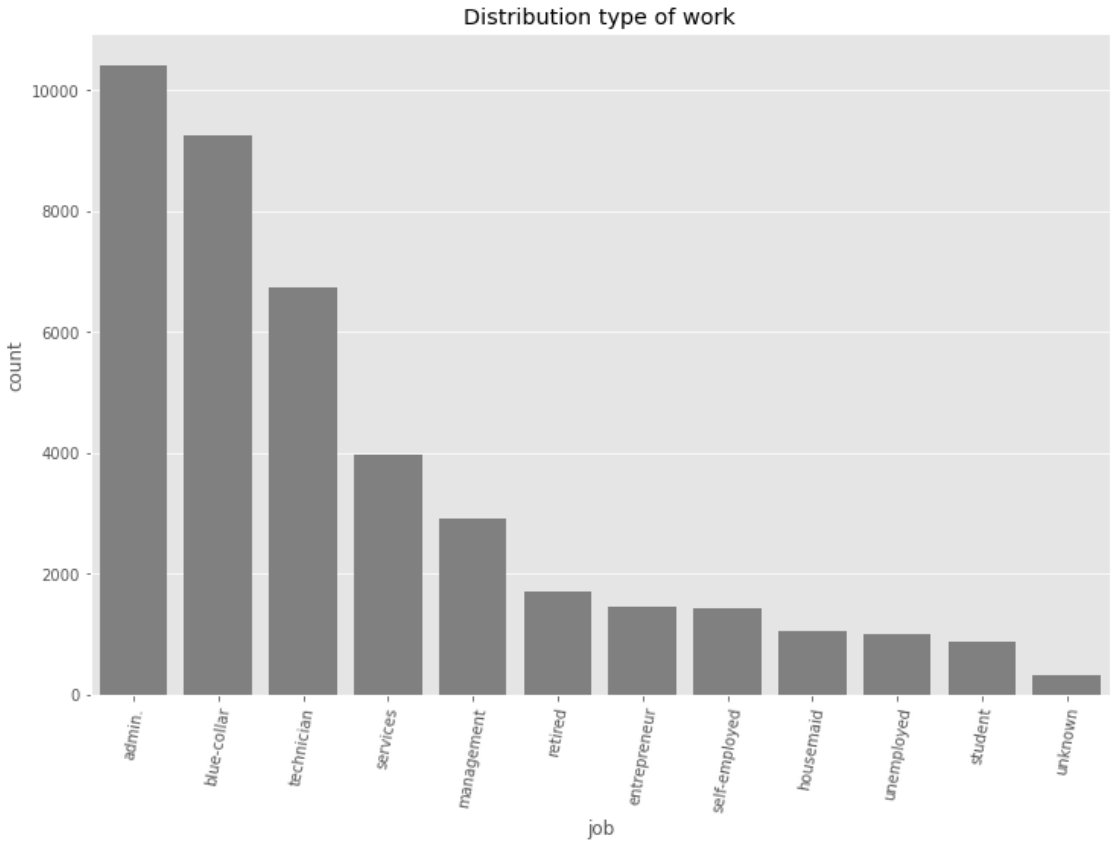
Observations:

- Ratio resposne yes/no is highest below 20 and above 80 age.
- Histplot of age show us that it is skewed right. Here is also a lot of outliers
- Great decision will be transofrm age by log, then data will be almost in normal distribution
- In the boxplot we can see also a lot of outliers. Additionaly, this plot shows similarity mean of age between response 'no' and 'yes'.
- Feature loan rather no affect to age and response

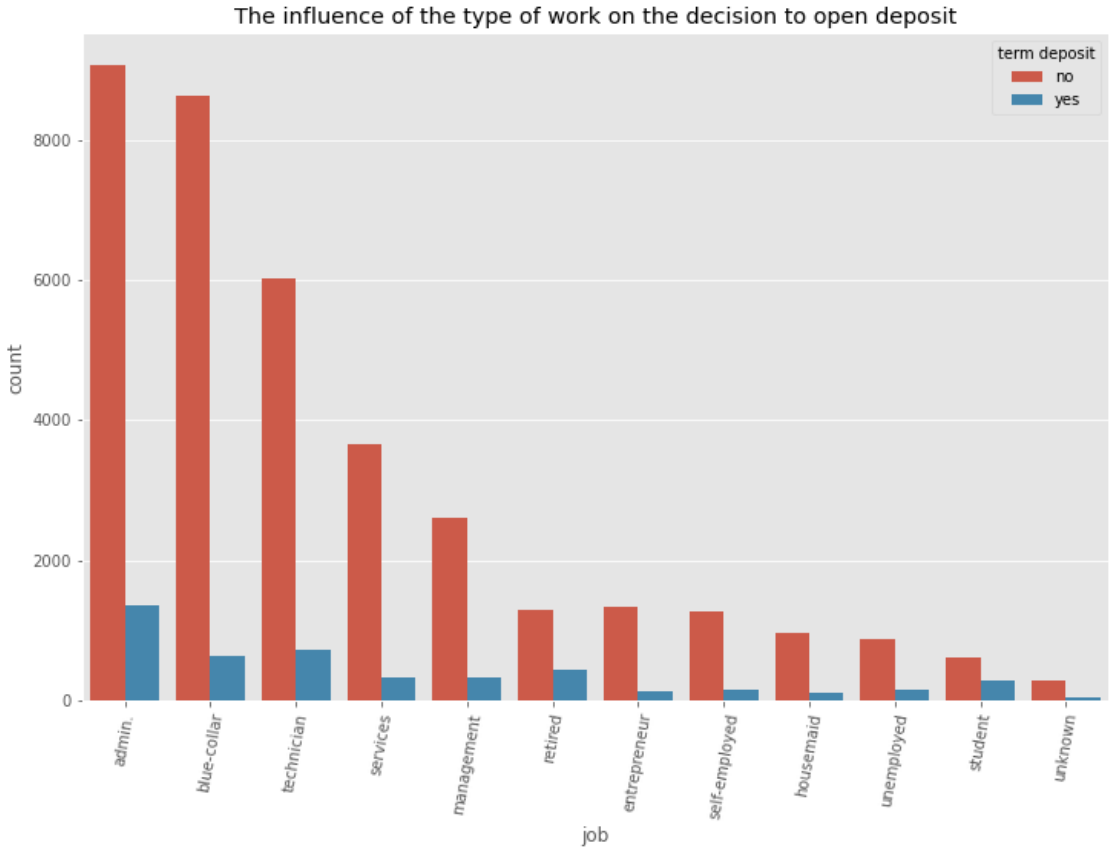
1.6.2 Job

In this section will be presented feature job. Feature means the type of work of the people who were called by bank representatives. Value of this feature: admin., blue-collar, entrepreneur, housemaid, management, retired, selfemployed, services, student, technician, unemployed, unknown.

[25]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'admin.'),
  Text(1, 0, 'blue-collar'),
  Text(2, 0, 'technician'),
  Text(3, 0, 'services'),
  Text(4, 0, 'management'),
  Text(5, 0, 'retired'),
  Text(6, 0, 'entrepreneur'),
  Text(7, 0, 'self-employed'),
  Text(8, 0, 'housemaid'),
  Text(9, 0, 'unemployed'),
  Text(10, 0, 'student'),
  Text(11, 0, 'unknown')])
```

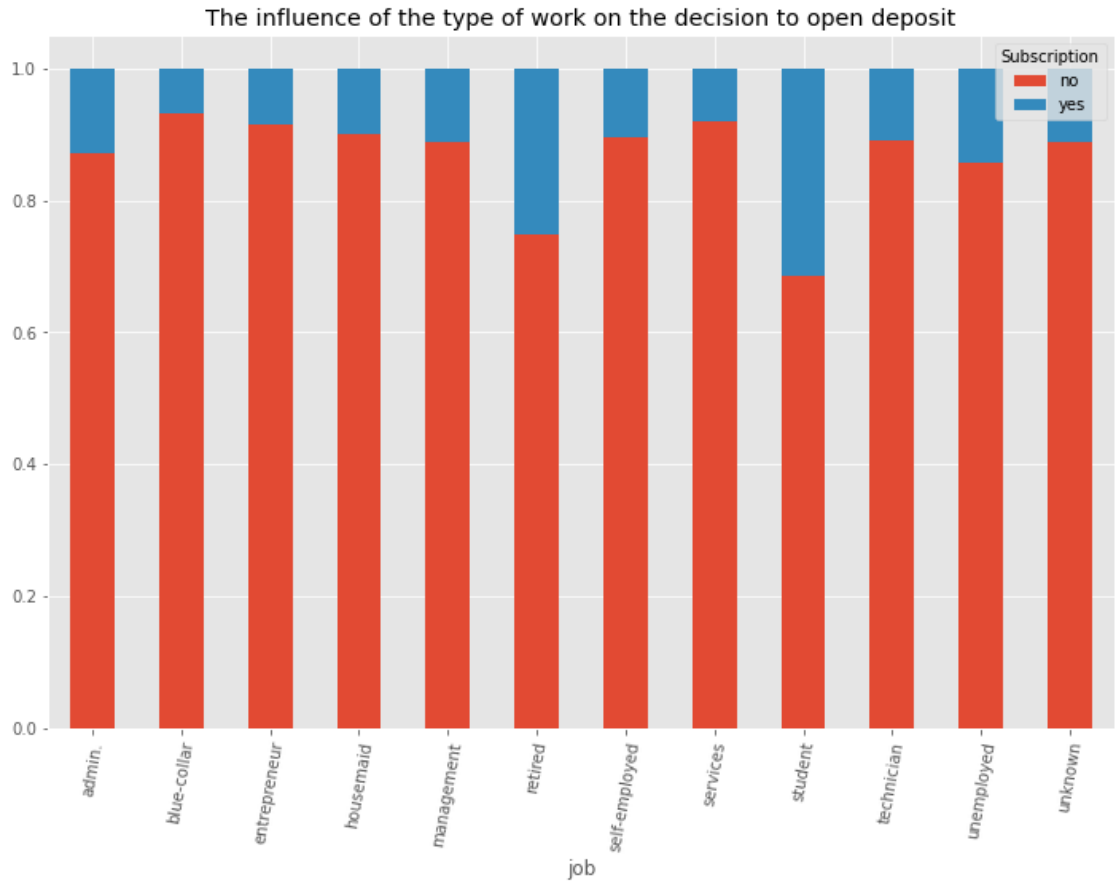


[26]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]),
 [Text(0, 0, 'admin.'),
  Text(1, 0, 'blue-collar'),
  Text(2, 0, 'technician'),
  Text(3, 0, 'services'),
  Text(4, 0, 'management'),
  Text(5, 0, 'retired'),
  Text(6, 0, 'entrepreneur'),
  Text(7, 0, 'self-employed'),
  Text(8, 0, 'housemaid'),
  Text(9, 0, 'unemployed'),
  Text(10, 0, 'student'),
  Text(11, 0, 'unknown')])

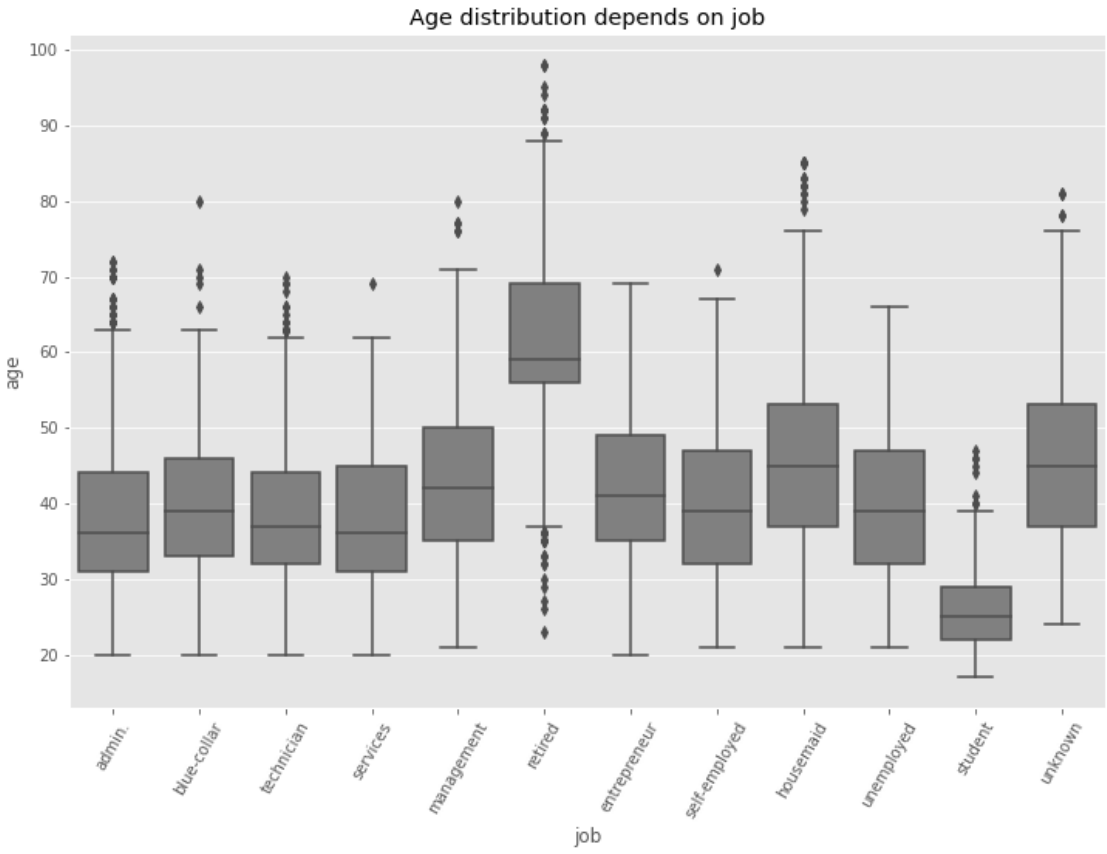


[27]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]),
 [Text(0, 0, 'admin.'),
  Text(1, 0, 'blue-collar'),
  Text(2, 0, 'entrepreneur'),
  Text(3, 0, 'housemaid'),
  Text(4, 0, 'management'),

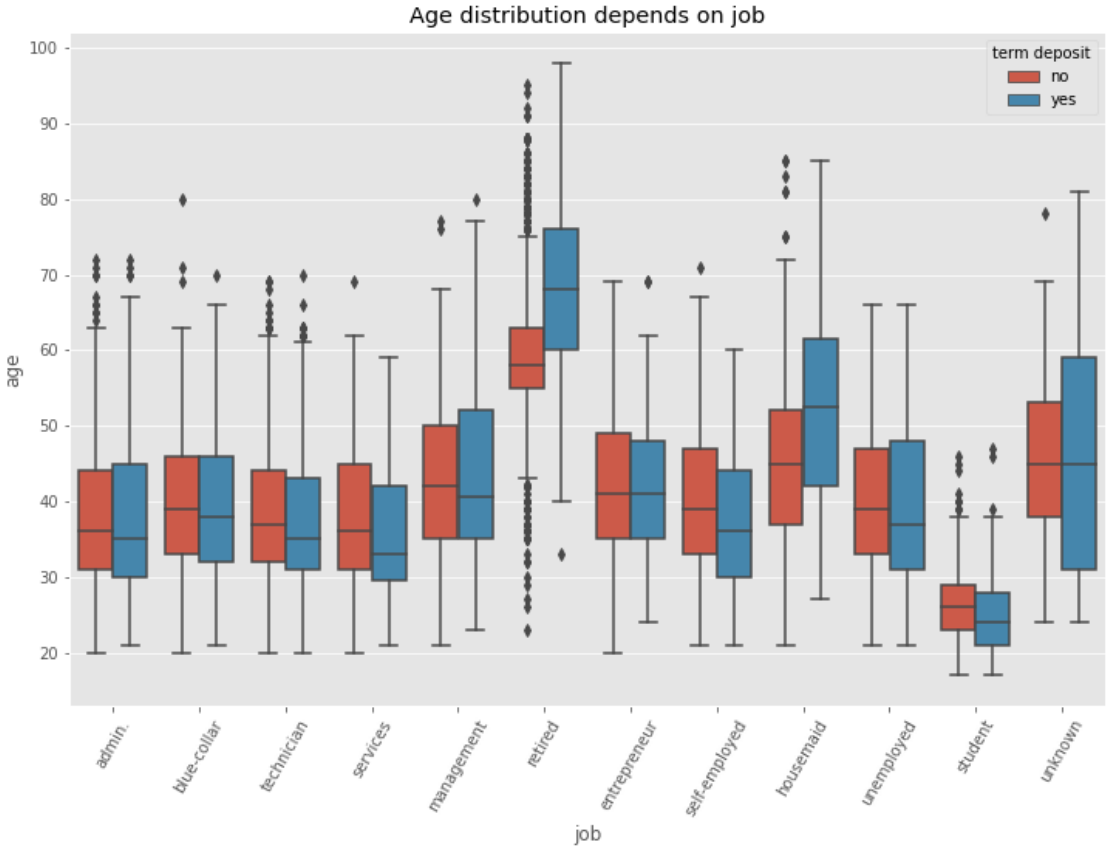
```
Text(5, 0, 'retired'),
Text(6, 0, 'self-employed'),
Text(7, 0, 'services'),
Text(8, 0, 'student'),
Text(9, 0, 'technician'),
Text(10, 0, 'unemployed'),
Text(11, 0, 'unknown')])
```



[28]: `Text(0.5, 1.0, 'Age distribution depends on job')`



[29]: Text(0.5, 1.0, 'Age distribution depends on job')



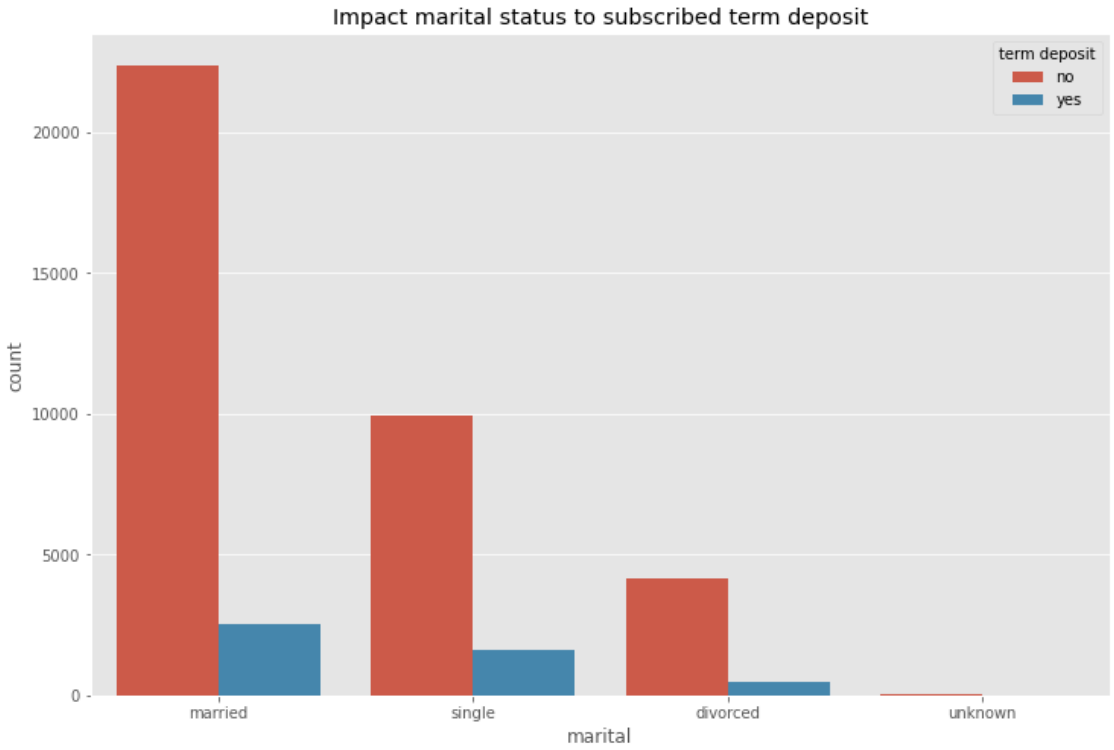
Observations:

- The most frequency job is administator
- In the plot we can notice assumed correctness as students have about 25 years old also retired people have more years than other.
- We can notice wired outliers in retired. Of course, outliers above upper limit is understanding. Outliers below 35 years old may be error.

1.6.3 Marital

Marital is feature which present us marital status people. Value of this feature: divorced, married, single, unknown.

[30]: `Text(0.5, 1.0, 'Impact marital status to subscribed term deposit')`




```
Comparison ratio term deposit in each group in marital
```

[32]:
```
marital
unknown     0.176471
single      0.162847
divorced    0.115087
married     0.113056
Name: term deposit, dtype: float64
```

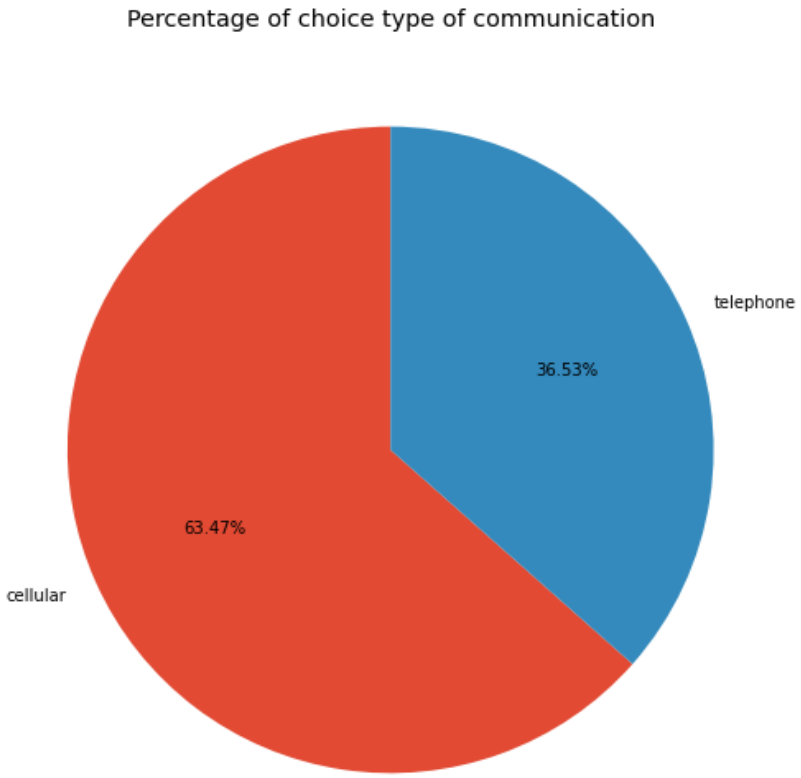
Observations:

- Base on more accurate calculate, single marital status has infuence to output.

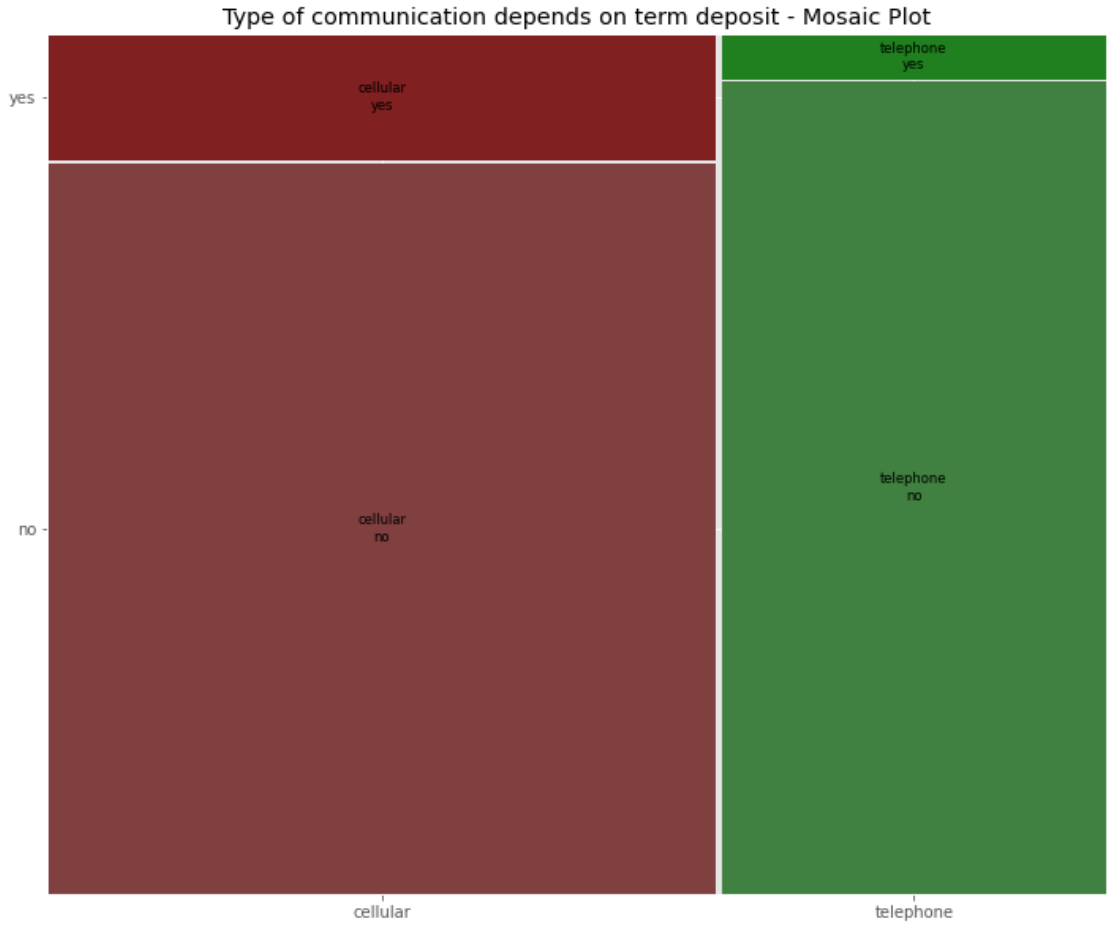
1.6.4 Contact

Feature contact consist of information about type of communication. Value of this feature: cellular, telephone.

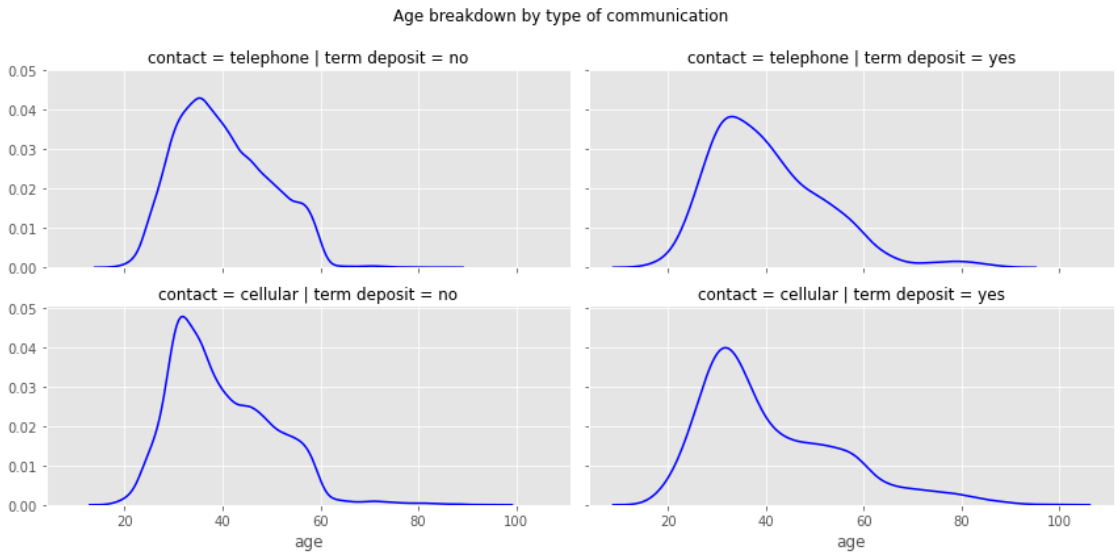
[33]:
```
array([26144, 15044], dtype=int64)
```



As pie plot show, about 64% use cellular so we can conclude there are young people, below 40 years.



[37]: Text(0.5, 0.98, 'Age breakdown by type of communication')



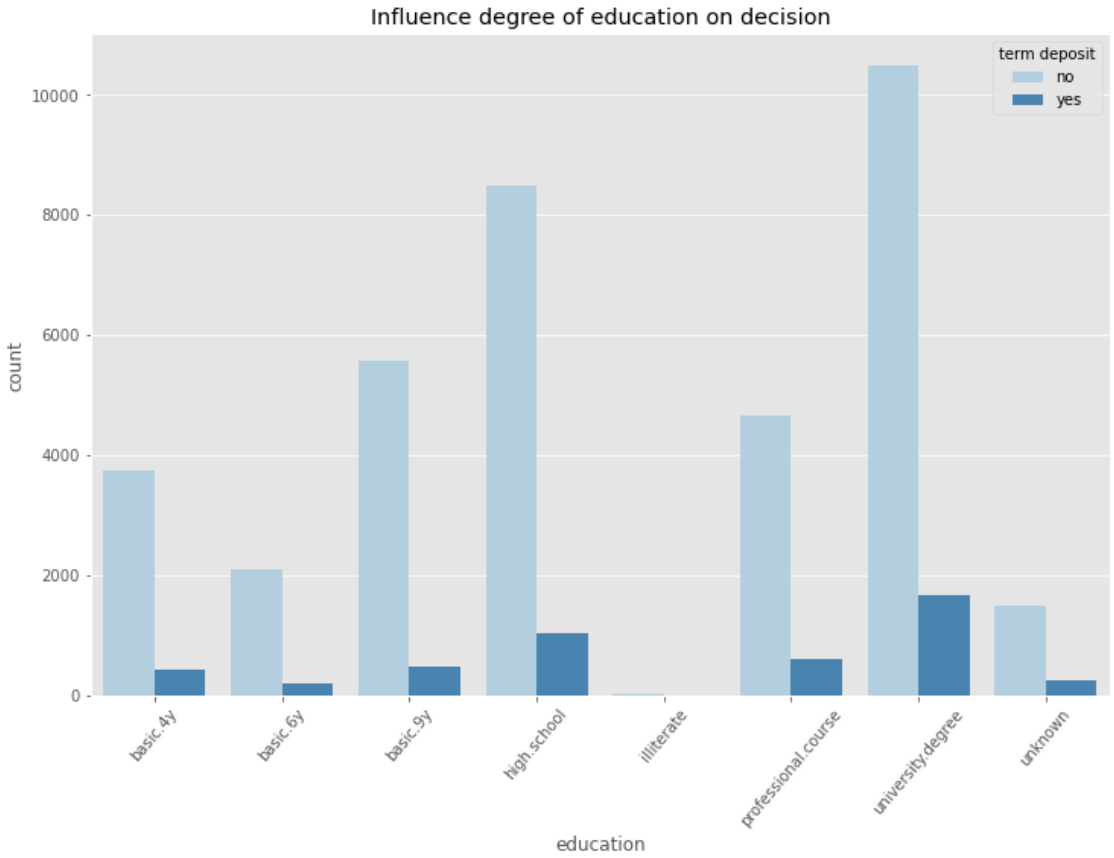
Observation:

- Our assumption about infuence type of contact to age was wrong. Both in contact cellular and telephone dominate pepole about 40 years.
- Only old people contact by cellular (retired people). The older it is, the more often it subscribes

1.6.5 Education

This feature includes information about degree of education people. Value of this feature: basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown.

[38]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7]),
 [Text(0, 0, 'basic.4y'),
  Text(1, 0, 'basic.6y'),
  Text(2, 0, 'basic.9y'),
  Text(3, 0, 'high.school'),
  Text(4, 0, 'illiterate'),
  Text(5, 0, 'professional.course'),
  Text(6, 0, 'university.degree'),
  Text(7, 0, 'unknown')])
```



[39]:
```
education
illiterate             0.285714
unknown                0.169595
university.degree      0.159078
professional.course    0.128012
high.school            0.121523
basic.4y               0.114194
basic.6y               0.089354
basic.9y               0.084889
Name: term deposit, dtype: float64
```

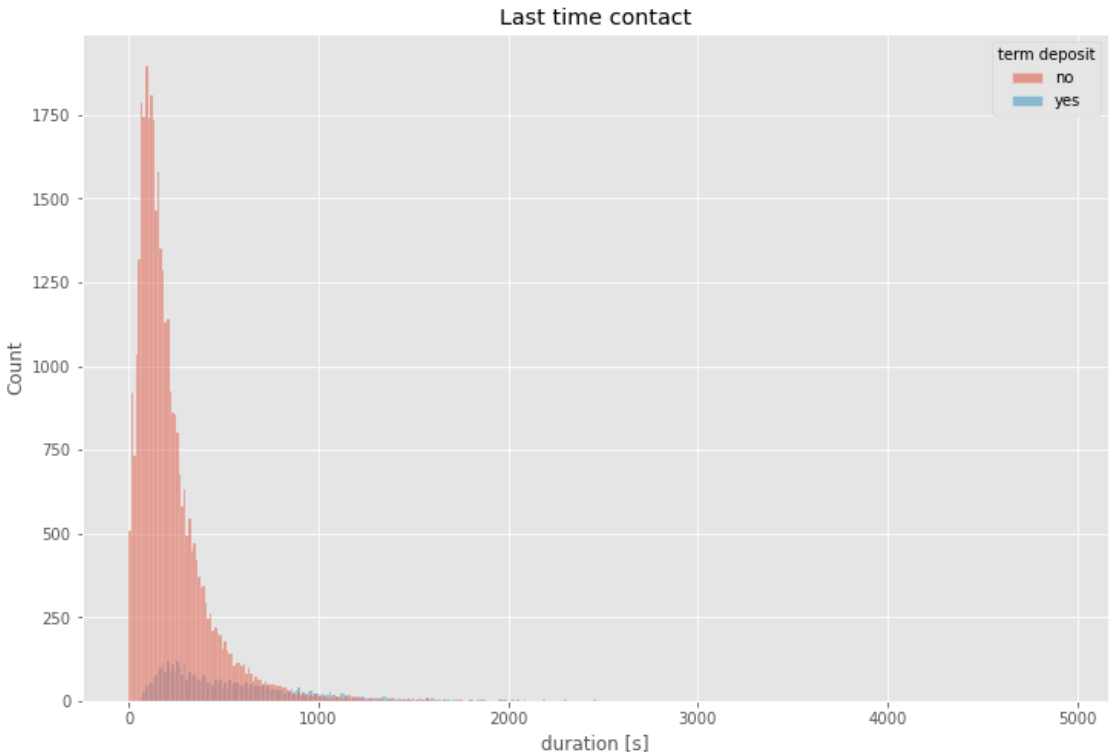
Observations:

- That most of the people who were called were people with higher education (more chance to agree subscribed term deposit due to higher earnings)
- during our conclusion, we have to reject variable illiterate due to insufficient data, so the higher degree of education, the heigher propability to subscribe term deposit

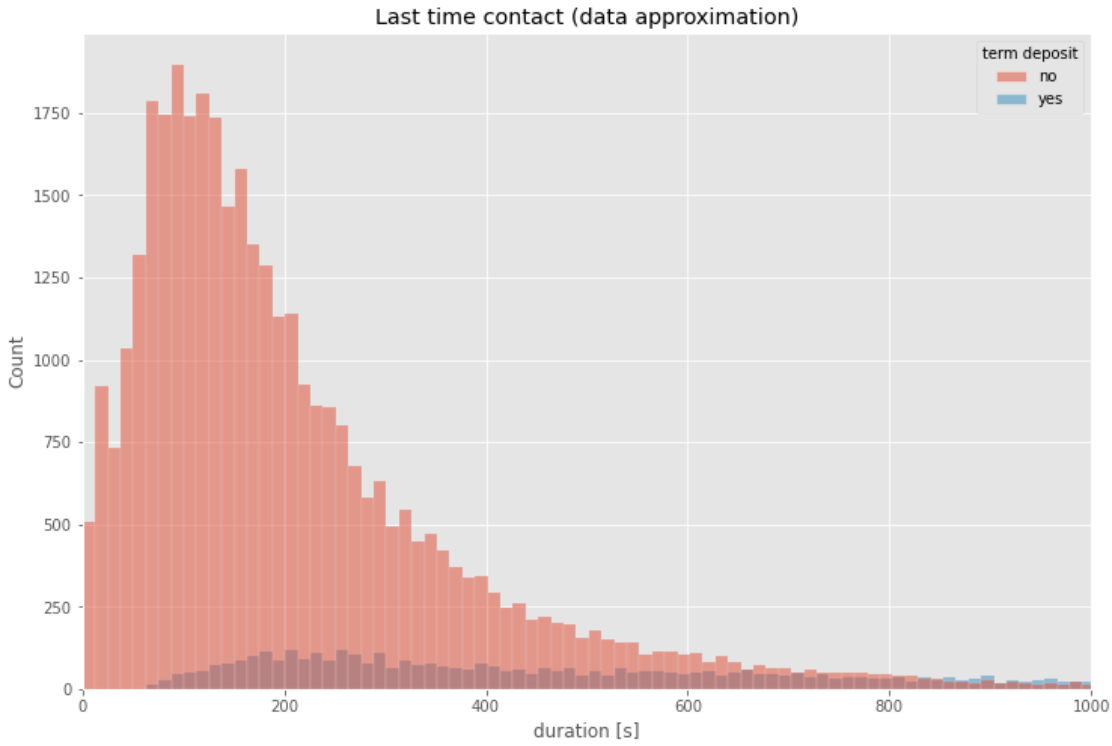
1.6.6 Duration

This feature includes information about last contact duration in second.

[40]:
```
Text(0.5, 0, 'duration [s]')
```



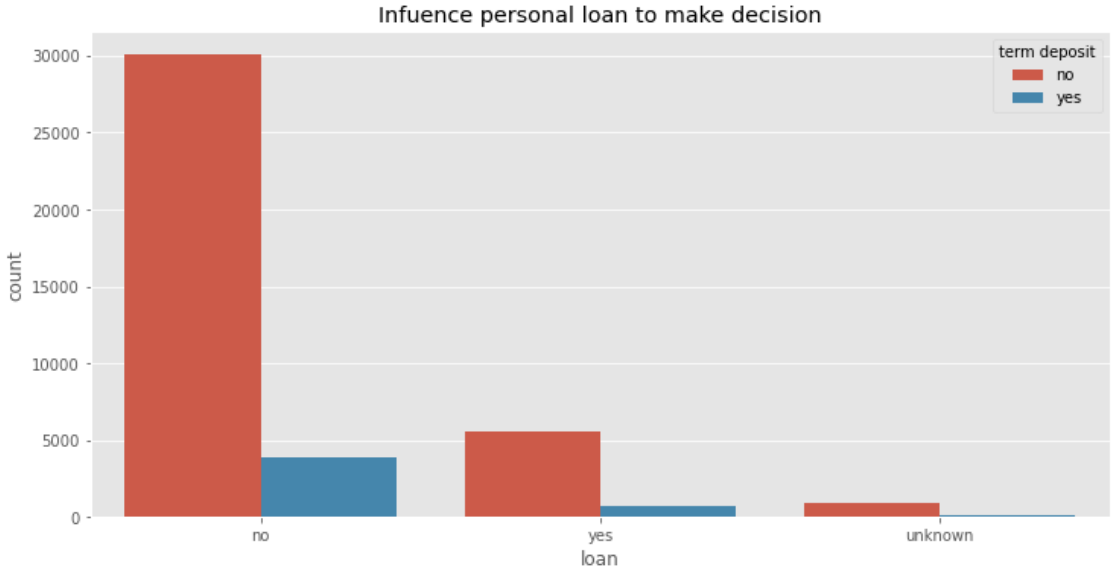
[41]: `Text(0.5, 0, 'duration [s]')`

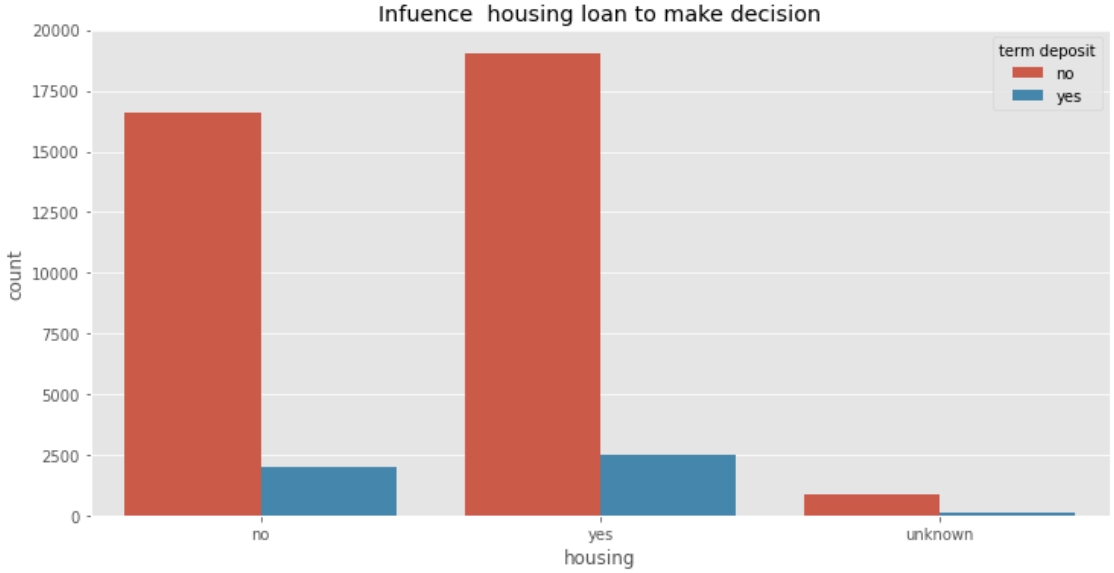


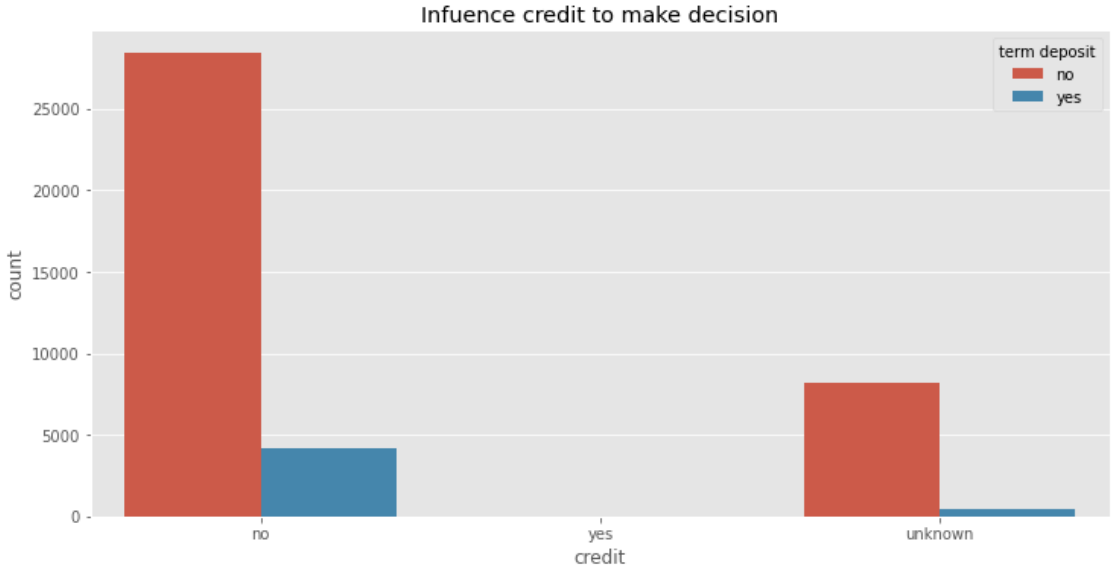
Distribution of last contact duration for response term deposit = 0 is skewed right. The most comon value of contact is value about 200 s. Is a lot outliers that contact leasts > 1000s. Then probably people subscribed term deposit. Also we can see that all contact < 50s, ending response 'no', so like as in description, this variables highly affect to output.

1.6.7 All type of credits

[42]: `Text(0.5, 1.0, 'Infuence credit to make decision')`


Infuence personal loan to make decision
term deposit
no
yes
count
30000
25000
20000
15000
10000
5000
0
no
yes
unknown
loan


Infuence housing loan to make decision
term deposit
no
yes
count
20000
17500
15000
12500
10000
7500
5000
2500
0
no
yes
unknown
housing


Infuence credit to make decision
term deposit
no
yes
count
25000
20000
15000
10000
5000
0
no
yes
unknown
credit

In plot we can see few response yes in column credit, so we can reject this variable

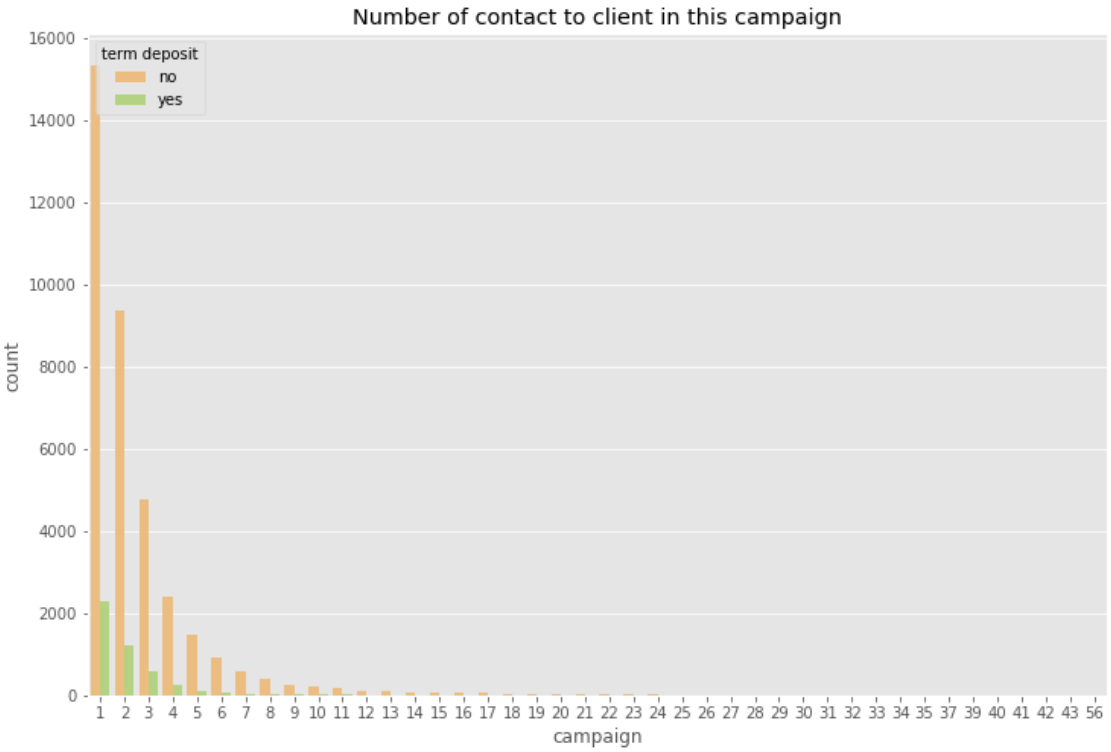
```
Comparison ratio term deposit in each group in loan
loan
no         0.127907
yes        0.122731
unknown    0.121178
Name: term deposit, dtype: float64

Comparison ratio term deposit in each group in housing
housing
yes        0.131470
no         0.122078
unknown    0.121178
Name: term deposit, dtype: float64
```

None of this variables has no significant on output.

1.6.8 Campaign

This feature includes information about number of contacts performed during this campaign and for client.

[44]: `Text(0.5, 1.0, 'Number of contact to client in this campaign')`



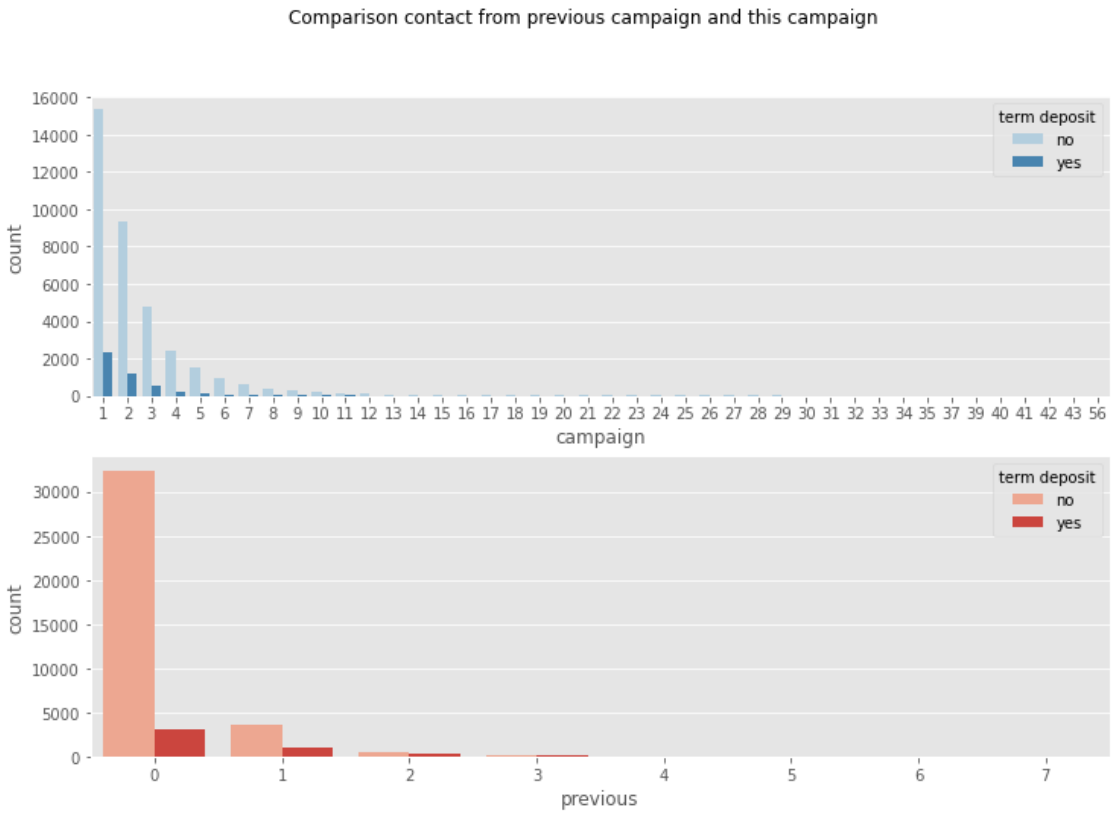
[45]:
```
no     399
yes      7
Name: term deposit, dtype: int64
```

Observations:

- The most frequency number of contact is 1.
- All this people no subscribed term deposit. only 7 people from 408 subscribed. Propably they were undecided or they like talk.

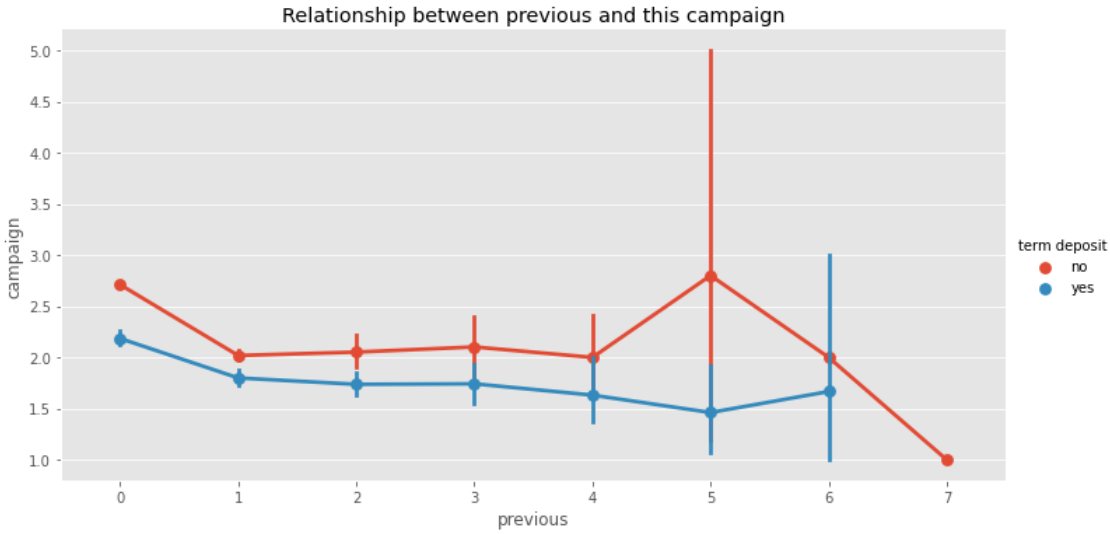
1.6.9 Previous and campaign

Previous feature includes information about number of contacts performed before this campaign and for client.



[47]:
```
  term deposit  previous
0           no         8
1          yes        16
```

[48]:
```
Text(0.5, 1.0, 'Relationship between previous and this campaign')
```



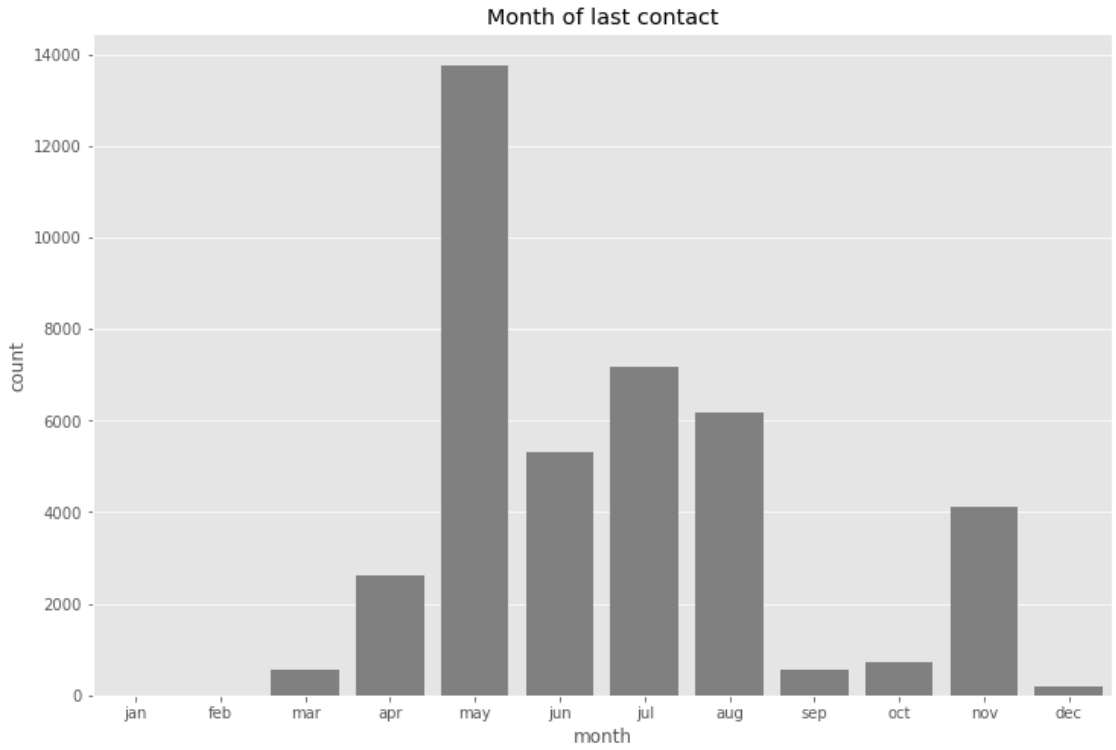
Observations

- This plot shows that in this campaign ratio resposne yes/no is heigher than in a previous campaign. Also, base on previous and this plot, we can notice that max number of contact in previous campaign was 7, in this 56.
- This is no relationship between present and previous campaign.

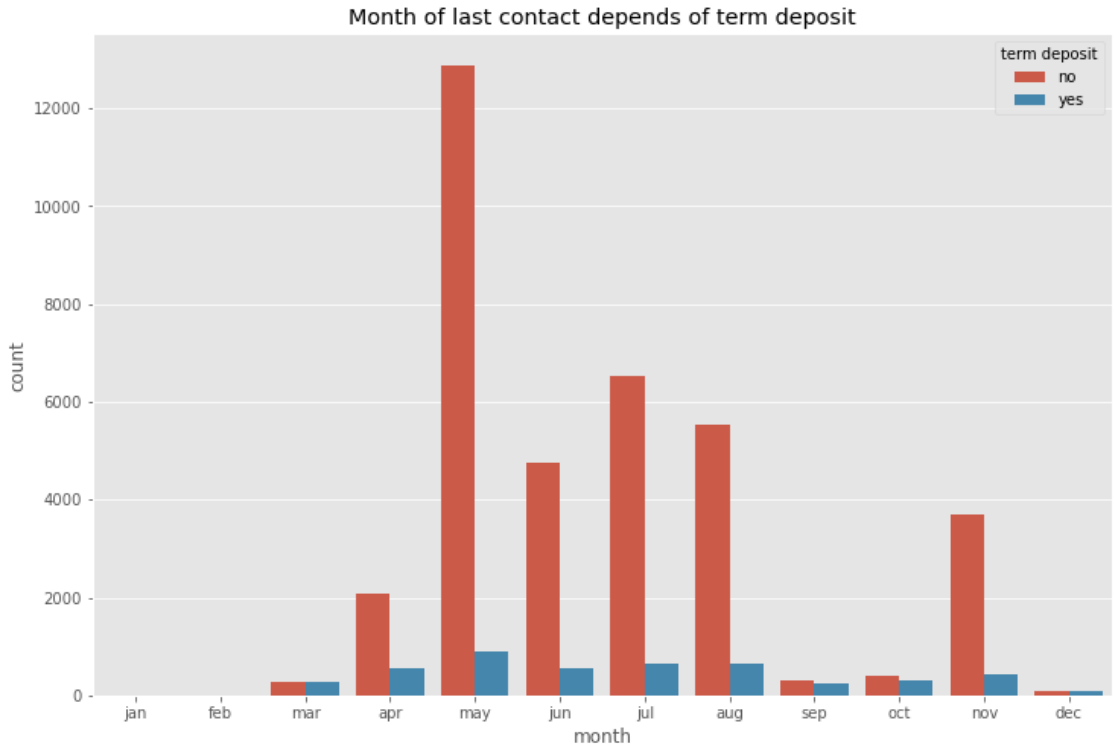
1.6.10 Month

This feature includes information about which month was last contacted. Value of this feature: January, February, March, April, May, June, July, August, September, October, November, December.

[49]: `Text(0.5, 1.0, 'Month of last contact')`



[50]: `Text(0.5, 1.0, 'Month of last contact depends of term deposit')`



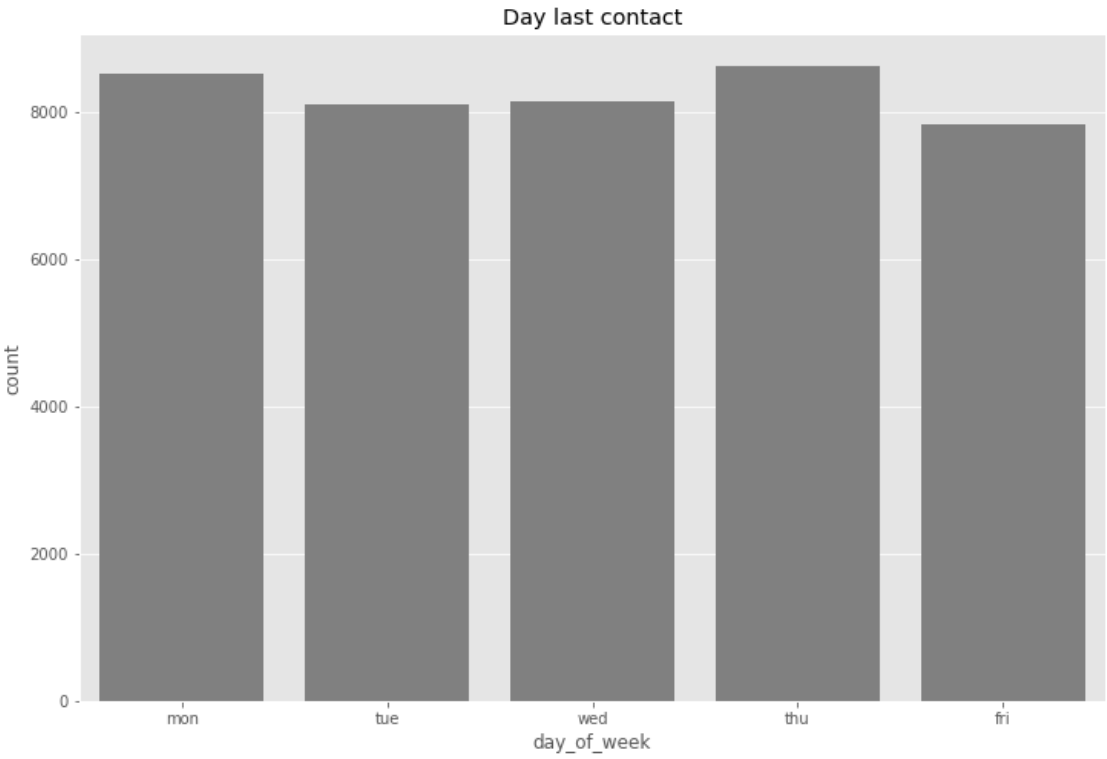
Observations

- The most month of contact is may and summer months, due to higher probability of agree to subscribe
- In january and februrary, there was no contact, probably closing cases from last year

1.6.11 Day of week

This feature includes information about which month was last contacted. Value of this feature: Monday, Tuesday, Wednesday, Thursday, Friday.

[51]:
```
Text(0.5, 1.0, 'Day last contact')
```



[52]:
```
term deposit        no       yes       All
day_of_week
fri           0.169491  0.020540  0.190031
mon           0.186146  0.020564  0.206711
thu           0.183986  0.025371  0.209357
tue           0.173279  0.023138  0.196416
wed           0.174444  0.023041  0.197485
All           0.887346  0.112654  1.000000
```

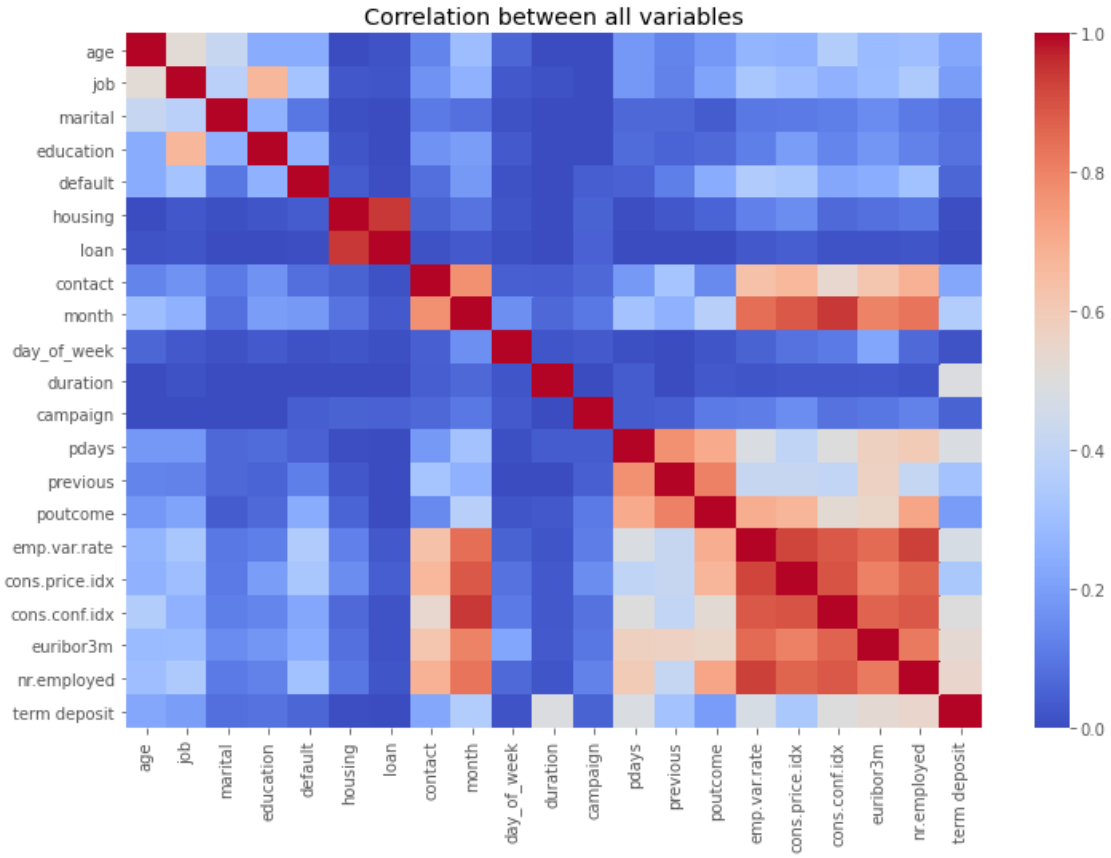
Observations:

- As we can see, in feature day_od_week is only five days, these are working days. Bank doesn't work in weekend, so we don't have error in this feature.
- Distribution all value in this feature is similar, so this feature doesn't have impact to output

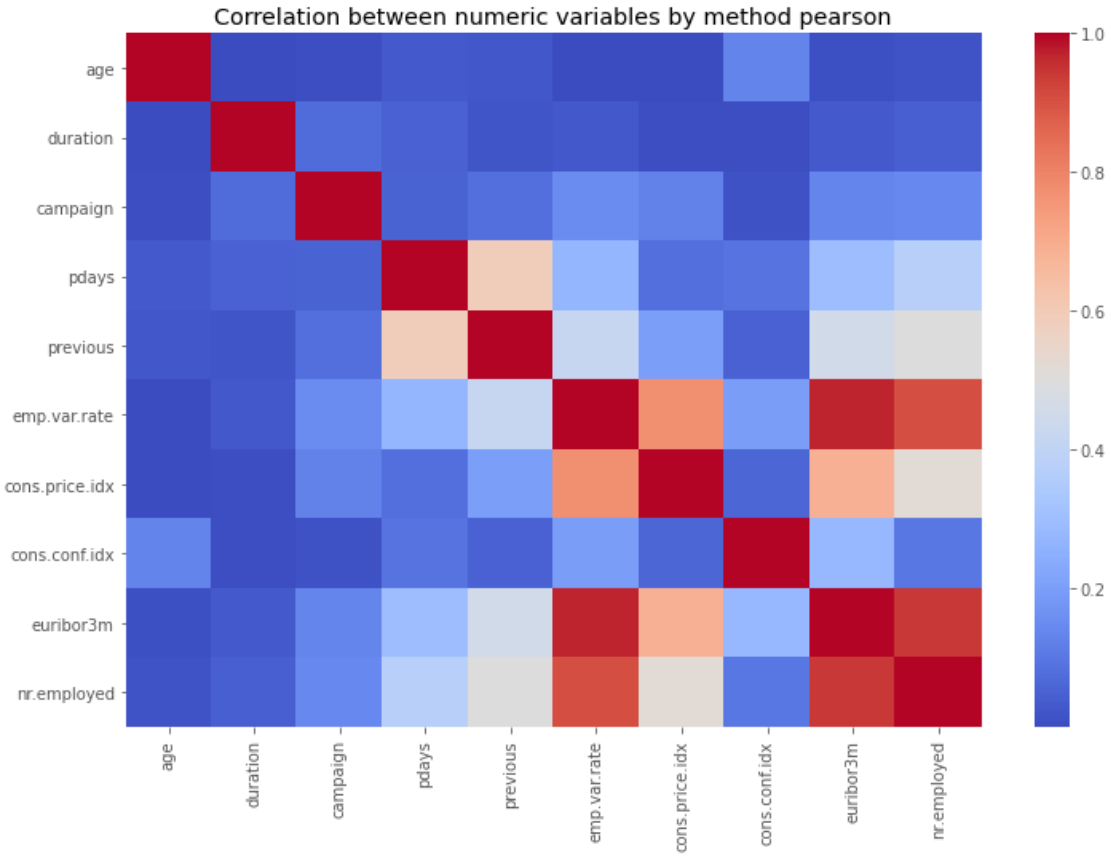
1.6.12 Correlation between features

For the first plot we use special function form libraries phik - phik_matrix. The combined features of Phi_K form an advantage over existing coefficients. First, it works consistently between categorical, ordinal and interval variables. Second, it captures non-linear dependency. Third, it reverts to the Pearson correlation coefficient in case of a bi-variate normal input distribution. These are useful features when studying the correlation matrix of variables with mixed types. It is orrelation matrix of bivariate gaussian derived from chi2-value.

[54]: `Text(0.5, 1.0, 'Correlation between all variables')`



[55]: `Text(0.5, 1.0, 'Correlation between numeric variables by method pearson')`



Observation:

- As we can see on both correlation plot, same variables are highly corelate each other. Later varaibles will be remove.

```
loan            housing          0.943147
cons.conf.idx   month            0.942669
nr.employed     emp.var.rate     0.929844
cons.price.idx  emp.var.rate     0.921977
dtype: float64
```

```
Data variables to drop due to high correlation: ['loan', 'cons.price.idx',
'cons.conf.idx', 'nr.employed']
```

Observation:

- We may remove one from highly correlated variables. During testing model we try predict with both features and with removed one of this freatures

[59]:
```
term deposit      1.000000
nr.employed       0.547178
euribor3m         0.529597
```

```
cons.conf.idx     0.501193
duration          0.489394
pdays             0.488128
emp.var.rate      0.474448
month             0.357098
cons.price.idx    0.336131
previous          0.314234
contact           0.225206
age               0.224248
job               0.195808
poutcome          0.195178
education         0.089572
marital           0.081401
default           0.059741
campaign          0.052260
day_of_week       0.018948
housing           0.005695
loan              0.000000
Name: term deposit, dtype: float64
```

Observations:

- There is couple of features that have very low correlation, so this feature will be removed and compare model with and without.

1.7 Feature engineering

In this section of this project, we filled our data (Nan) and transformed to numeric. Result of this part is dataframe:

[101]:

	age	education	campaign	previous	poutcome	emp.var.rate
0	4.025352	0	1	0	0	1.1
1	4.043051	3	1	0	0	1.1
2	3.610918	3	1	0	0	1.1
3	3.688879	1	1	0	0	1.1
4	4.025352	3	1	0	0	1.1

	cons.price.idx	cons.conf.idx	term deposit	month_jul	month_may
0	93.994	36.4	0	0.0	1.0
1	93.994	36.4	0	0.0	1.0
2	93.994	36.4	0	0.0	1.0
3	93.994	36.4	0	0.0	1.0
4	93.994	36.4	0	0.0	1.0

	month_nov	contact_telephone	marital_married	marital_single
0	0.0	1.0	1.0	0.0
1	0.0	1.0	1.0	0.0
2	0.0	1.0	1.0	0.0

```
3        0.0                1.0                1.0              0.0
4        0.0                1.0                1.0              0.0

   job fill_blue-collar  job fill_entrepreneur  job fill_retired  \
0                   0.0                    0.0               0.0
1                   0.0                    0.0               0.0
2                   0.0                    0.0               0.0
3                   0.0                    0.0               0.0
4                   0.0                    0.0               0.0

   job fill_services
0                0.0
1                1.0
2                1.0
3                0.0
4                1.0
```

2 Model predict and solve

Now we are ready to train a model and predict the required solution. Our problem is a classification problem. We want to identify relationship between output (term deposit) with other variables or features. We are also perfoming a category of machine learning which is called supervised learning as we are training our model with a given dataset. With these two criteria - Supervised Learning plus Classification, we can narrow down our choice of models to a few. These include:

- Logistic Regression
- Random Forest
- Support Vector Machines

First of all we should divide data to X, y to train and test data. We should do because our model is trained all time on train data. After training we can check our predict on test data. If we have only one data, our prediction will not be a reliable. So now we divide our data to train and test

We can standarize data. Without standarize model base on logistic regression has little worse results.

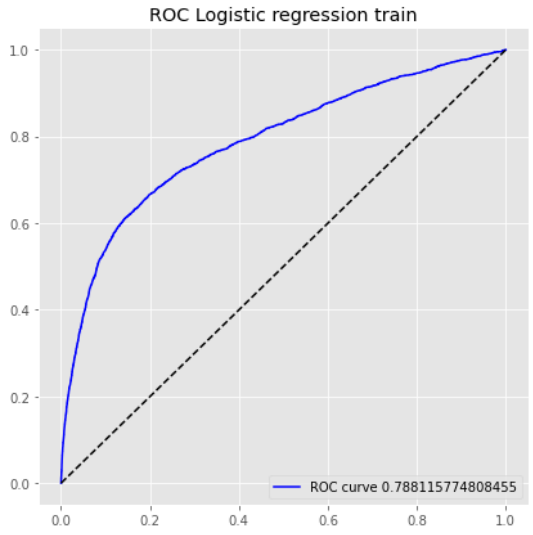
2.0.1 Logistic Regression

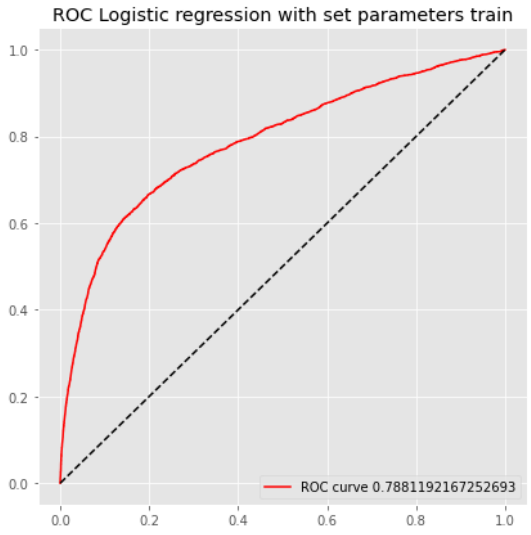
Logistic Regression is a useful model to run early in the workflow. Logistic regression measures the relationship between the categorical dependent variable (feature) and one or more independent variables (features) by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

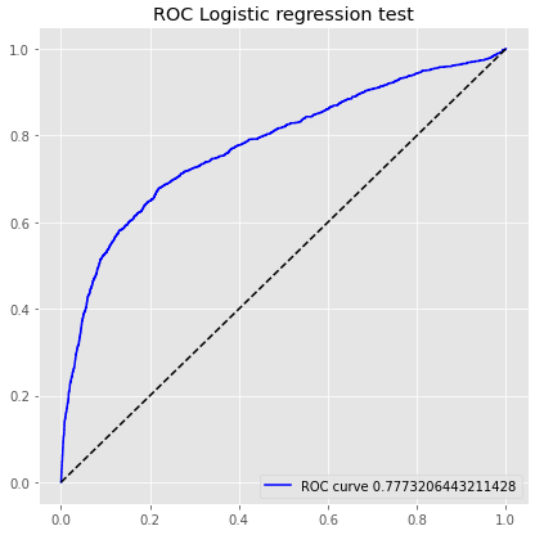
We can try, thanks to Grid Search CV we can choose the best parameters for model base on logistic regresion. After that, we also try model base on default parameters for logistic regression and we can choose better option.

```
The best parameters: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
```

```
Results for Logistic Regression:
Logistic Regression accuracy: 0.8954
Logistic Regression f-score: 0.2798
Logistic Regression recall: 0.182
Logistic Regression log_loss: 0.2855
---------------------------------------
Results for Logistic Regression with param:
Logistic Regression with param accuracy: 0.8954
Logistic Regression with param f-score: 0.2797
Logistic Regression with param recall: 0.182
Logistic Regression with param log_loss: 0.2855
```







Observations:

- Default setting for parameters give us the same result like the best choosen parameters.
- Base accuracy (all response the same like output == 'no') is 88%. In our model accuracy is 91%, what it means we have better model.
- As we can see, on train and test dataset, model works similar, so it isn't overfited or underfitted.
- F-score is the same for both model
- ROC curve (also great measure for imbalanced data) can efficiently give us the score that how our model is performing in classifing the labels. The best ROC is 1. We have 0.93 so it is great (all above black line is better)

We can use Logistic Regression to validate our assumptions and decisions for feature creating and completing goals. This can be done by calculating the coefficient of the features in the decision function.

Positive coefficients increase the log-odds of the response (and thus increase the probability), and negative coefficients decrease the log-odds of the response (and thus decrease the probability).

[109]:

	Feature	Correlation
6	cons.price.idx	0.692342
4	poutcome	0.277350
16	job fill_retired	0.089689
8	month_jul	0.038767
13	marital_single	0.032228
1	education	0.024581
12	marital_married	0.004611
15	job fill_entrepreneur	-0.017538
0	age	-0.052548
17	job fill_services	-0.083188
14	job fill_blue-collar	-0.089371
2	campaign	-0.124699
10	month_nov	-0.135021
3	previous	-0.137798
7	cons.conf.idx	-0.184723
9	month_may	-0.284131
11	contact_telephone	-0.356394
5	emp.var.rate	-1.179958

Observations:

- Cons.price.idx is highest positivie coefficient, the higher the value, the more likely the answer is yes.
- Inversely as emp.var.rate increases, probability of response yes decreases the most.

2.1 Resampling Technique

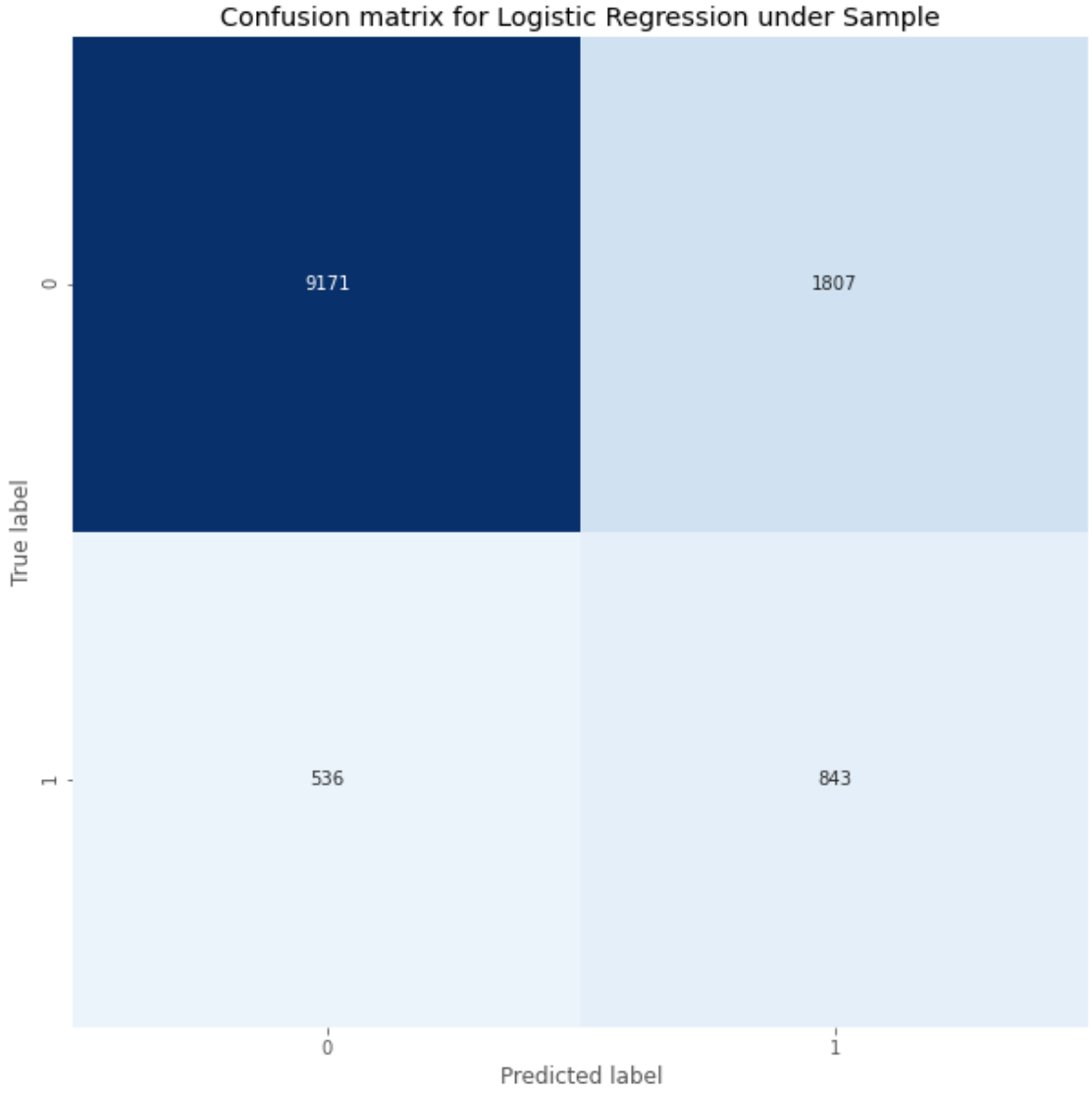
Due to highly imbalanced data, now we can try deal with them. One of possibilities is resampling technique.

Resampling consists of removing samples from the majority class (under-sampling) and adding more examples from the minority class (over-sampling). The simplest implementation of over-sampling is to duplicate random records from the minority class, which can cause overfishing. In under-sampling, the simplest technique involves removing random records from the majority class, which can cause loss of information.

2.1.1 Random Under-Sampling

Undersampling can be defined as removing some observations of the majority class. This is done until the majority and minority class is balanced out.

```
Results for Logistic Regression under Sample:
Logistic Regression under Sample accuracy: 0.8104
Logistic Regression under Sample f-score: 0.4185
Logistic Regression under Sample recall: 0.6113
Logistic Regression under Sample log_loss: 0.5306
```

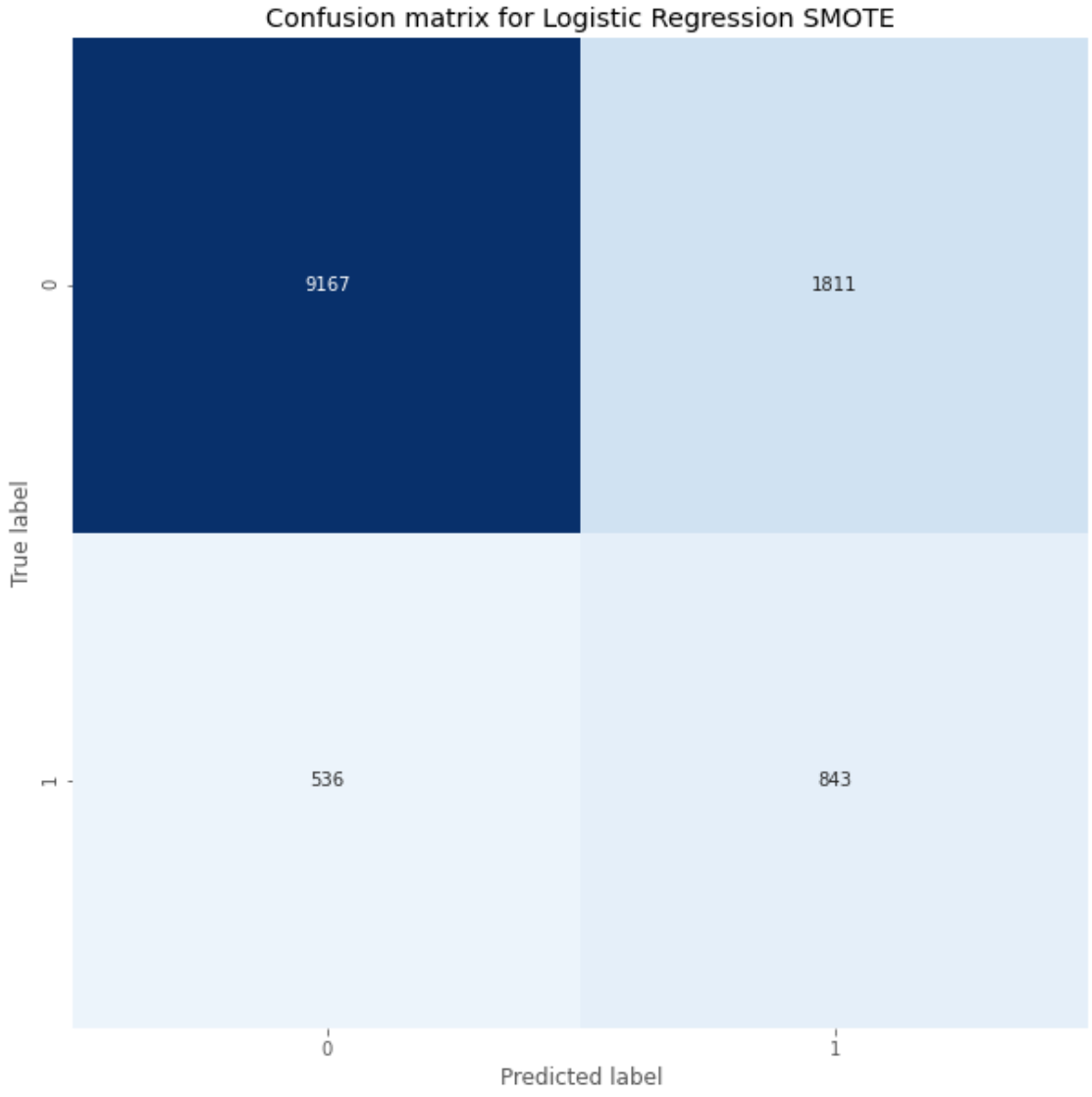


2.1.2 Synthetic Minority Oversampling Technique (SMOTE)

This technique generates synthetic data for the minority class.

SMOTE (Synthetic Minority Oversampling Technique) works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.

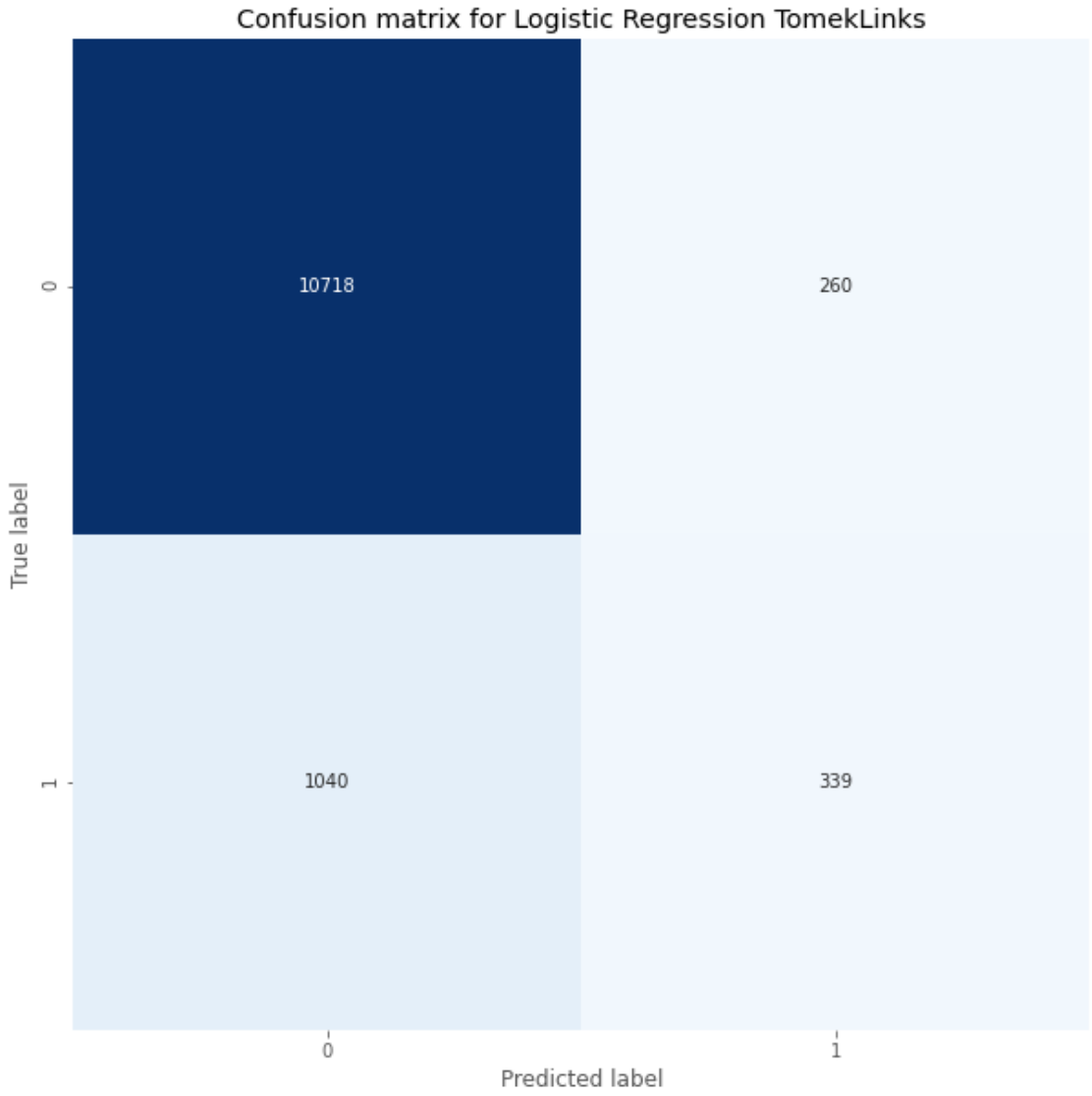
```
Results for Logistic Regression SMOTE:
Logistic Regression SMOTE accuracy: 0.8101
Logistic Regression SMOTE f-score: 0.4181
Logistic Regression SMOTE recall: 0.6113
Logistic Regression SMOTE log_loss: 0.5293
```



2.1.3 Under-sampling: Tomek links

Tomek links are pairs of very close instances, but of opposite classes. Removing the instances of the majority class of each pair increases the space between the two classes, facilitating the classification process.

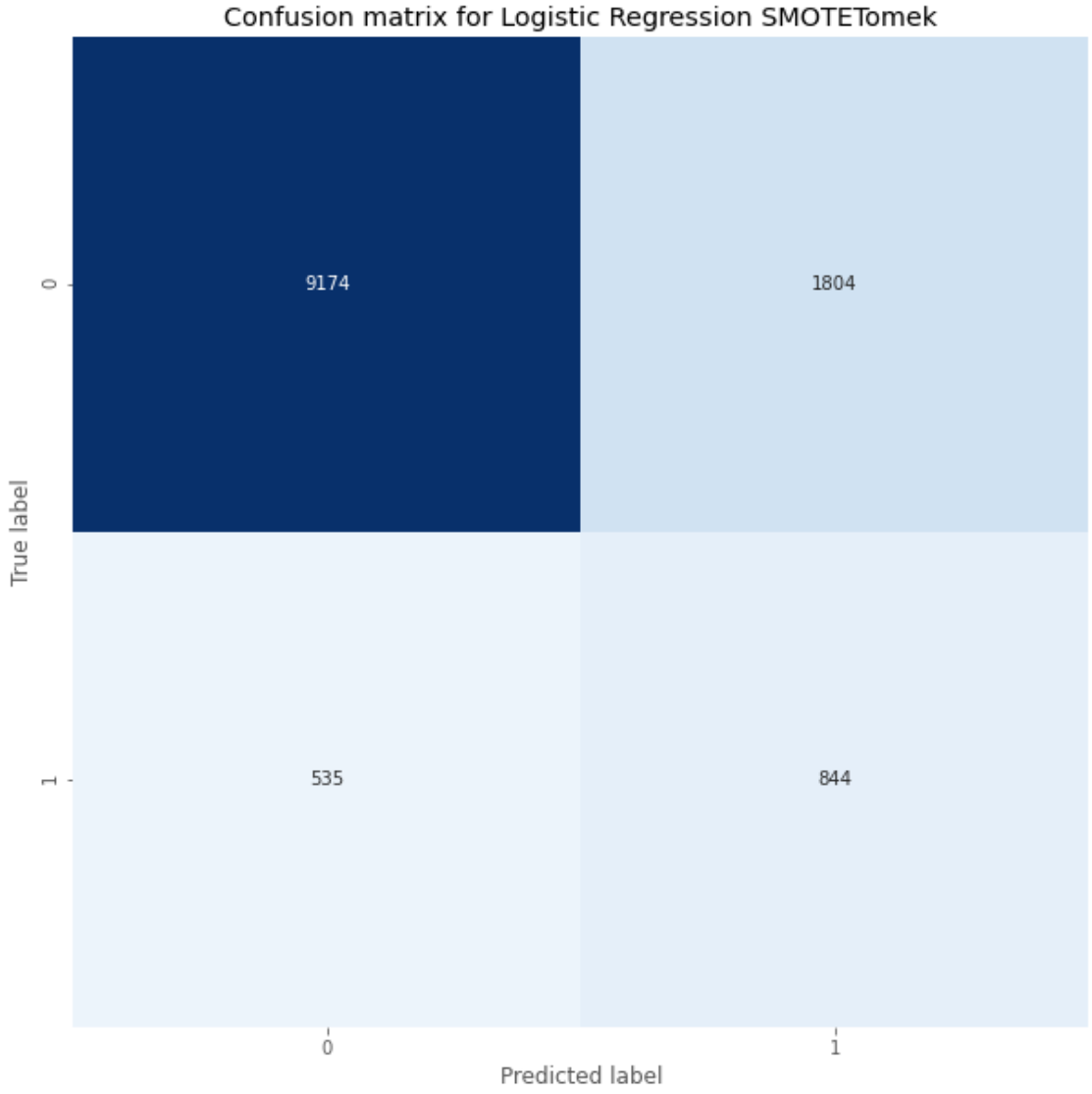
```
Results for Logistic Regression TomekLinks:
Logistic Regression TomekLinks accuracy: 0.8948
Logistic Regression TomekLinks f-score: 0.3428
Logistic Regression TomekLinks recall: 0.2458
Logistic Regression TomekLinks log_loss: 0.286
```



2.1.4 Over-sampling followed by under-sampling

Now, we try combination of over-sampling and under-sampling, using the SMOTE and Tomek links techniques.

```
Results for Logistic Regression SMOTETomek:
Logistic Regression SMOTETomek accuracy: 0.8107
Logistic Regression SMOTETomek f-score: 0.4192
Logistic Regression SMOTETomek recall: 0.612
Logistic Regression SMOTETomek log_loss: 0.53
```



[114]:

Classifier	Accuracy	F-score	Recall	Log_loss
Logistic Regression SMOTETomek	0.8107	0.4192	0.6120	0.5300
Logistic Regression under Sample	0.8104	0.4185	0.6113	0.5306

```
Logistic Regression SMOTE           0.8101   0.4181  0.6113    0.5293
Logistic Regression TomekLinks      0.8948   0.3428  0.2458    0.2860
Logistic Regression                 0.8954   0.2798  0.1820    0.2855
```

Observations:

- Used sampling method proved to be right, because all of these methods are better that without sampling
- Tree methods are similar: basic under sample, smote and combination smote and tomek links.

2.1.5 Principal Component Analysis

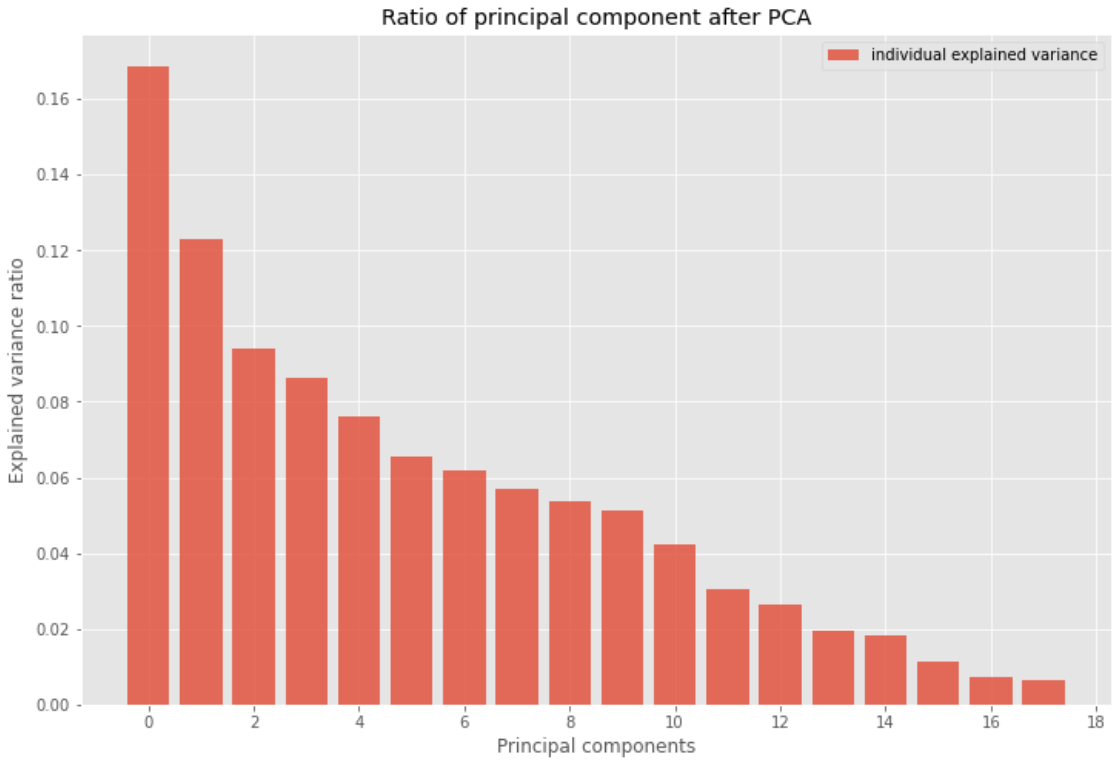
Pincipal component analysis is a technique to convert high dimensional data to low dimensional data by selecting the most important features that capture maximum information about the dataset. The features are selected on the basis of variance that they cause in the output. The feature that causes highest variance is the first principal component. The feature that is responsible for second highest variance is considered the second principal component, and so on. It is important to mention that principal components do not have any correlation with each other.

Now we can check PCA for our model.

After fitting and transforing our data, check principal component.

[116]:
```
array([0.16871878, 0.12298779, 0.09409119, 0.08636436, 0.0759825 ,
       0.06567283, 0.06200362, 0.05702128, 0.05358525, 0.0514317 ,
       0.04232106, 0.03050809, 0.02632848, 0.0195717 , 0.01838795,
       0.01137576, 0.00733059, 0.00631709])
```

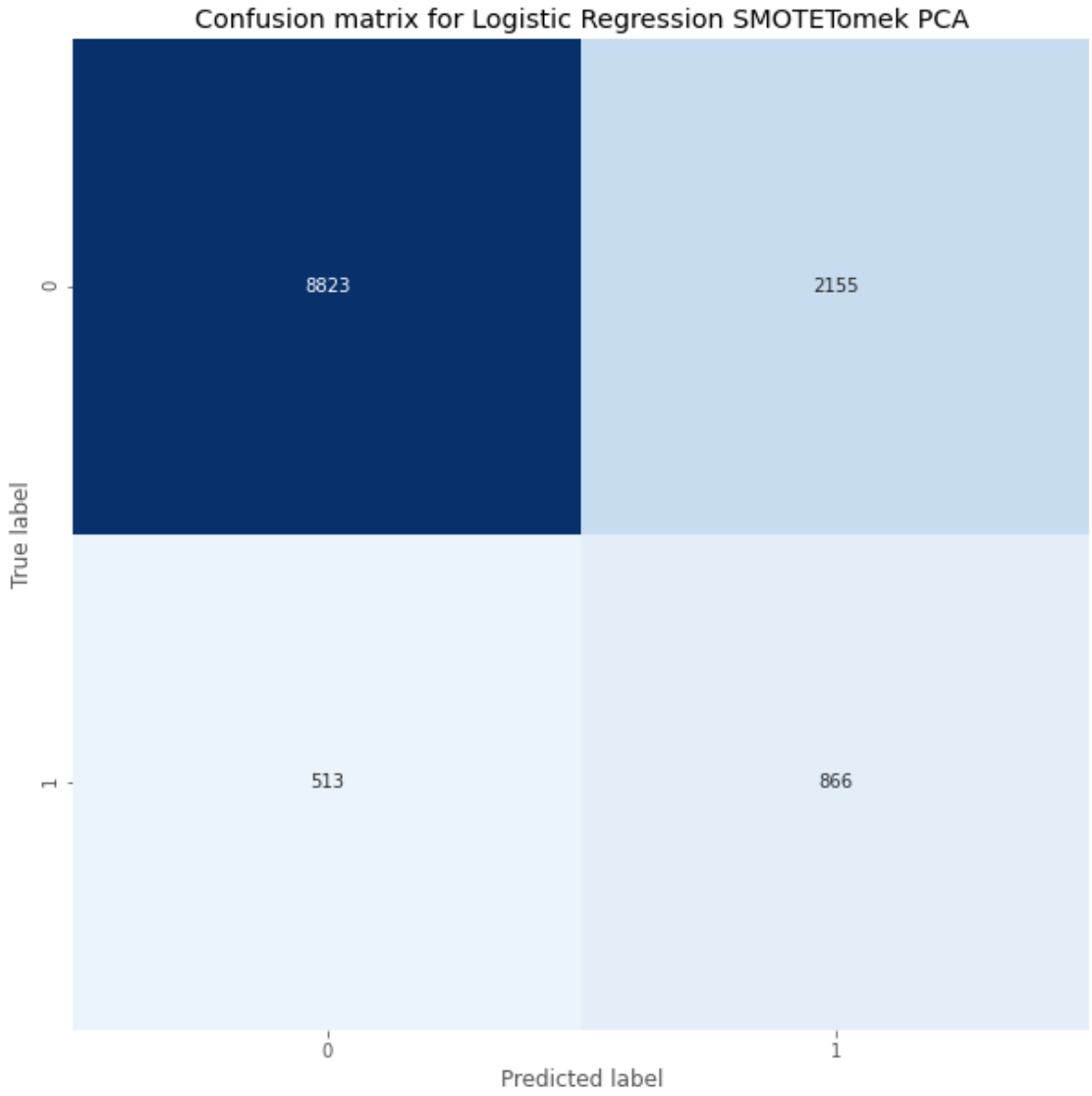
[117]:
```
<matplotlib.legend.Legend at 0x15ff898>
```



As we can see two last components have less than 0.01, so we can use 16 components.

Check our best result from logistic regression with PCA.

```
Results for Logistic Regression SMOTETomek PCA:
Logistic Regression SMOTETomek PCA accuracy: 0.7841
Logistic Regression SMOTETomek PCA f-score: 0.3936
Logistic Regression SMOTETomek PCA recall: 0.628
Logistic Regression SMOTETomek PCA log_loss: 0.5618
```



[120]:

Classifier	Accuracy	F-score	Recall	Log_loss
Logistic Regression SMOTETomek	0.8107	0.4192	0.6120	0.5300
Logistic Regression under Sample	0.8104	0.4185	0.6113	0.5306
Logistic Regression SMOTE	0.8101	0.4181	0.6113	0.5293
Logistic Regression SMOTETomek PCA	0.7841	0.3936	0.6280	0.5618
Logistic Regression TomekLinks	0.8948	0.3428	0.2458	0.2860
Logistic Regression	0.8954	0.2798	0.1820	0.2855

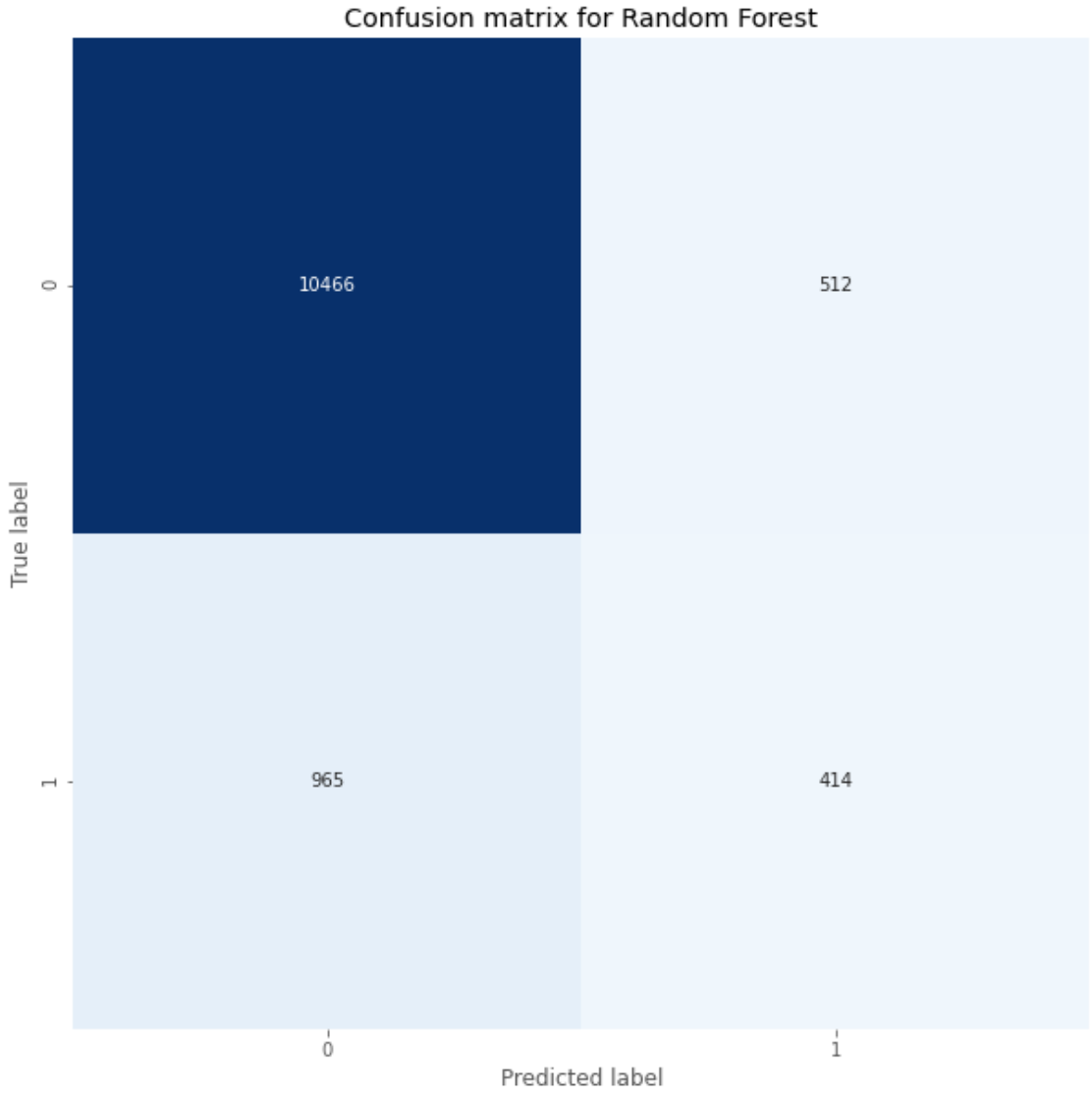
Observation:

- Model base on PCA is one of the worst so we reject this method for our model.

2.1.6 Random Forest

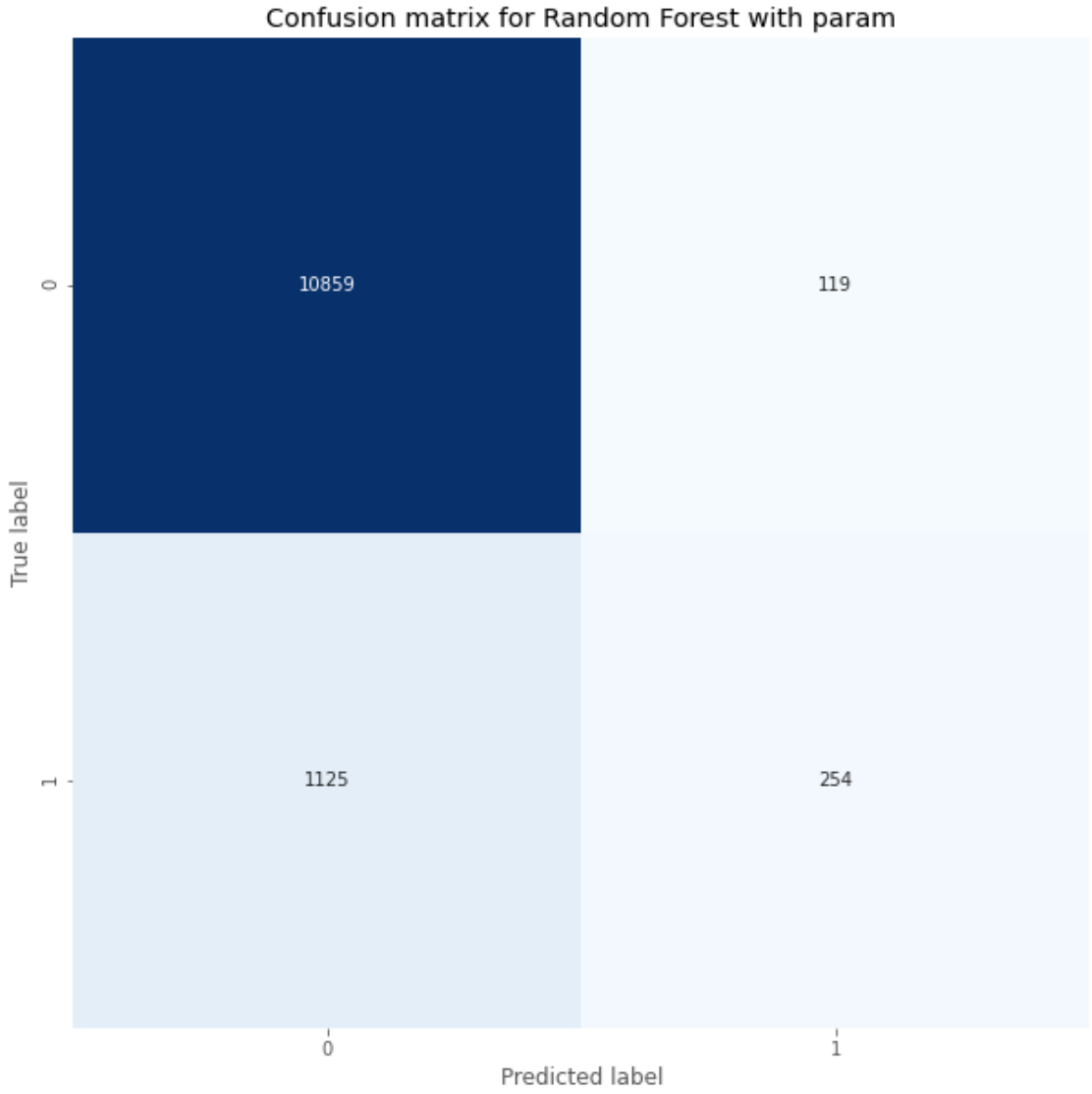
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees (n_estimators=50) at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees

```
Results for Random Forest:
Random Forest accuracy: 0.8805
Random Forest f-score: 0.3592
Random Forest recall: 0.3002
Random Forest log_loss: 1.0429
```




```
Results for Random Forest with param:
```

```
Random Forest with param accuracy: 0.8993
Random Forest with param f-score: 0.29
Random Forest with param recall: 0.1842
Random Forest with param log_loss: 0.2796
```



Observations:

- Random forest with set parameters (search by grid search cv) gave us worst results than random forest with default parameters so, we reject this method.

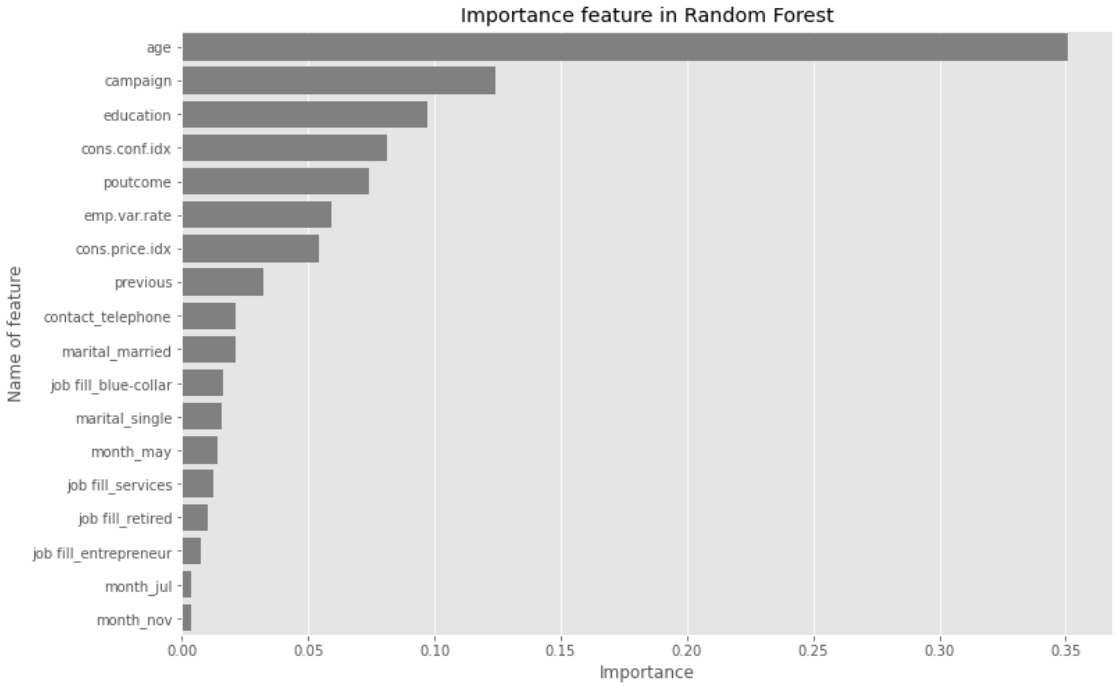
Now, we will use feature importance variable to see feature importance scores.

[124]:

	feature	importance
0	age	0.350850

```
2                campaign    0.124249
1               education    0.097101
7          cons.conf.idx     0.081282
4               poutcome     0.074050
5           emp.var.rate     0.059072
6          cons.price.idx    0.054234
3               previous     0.032492
11     contact_telephone     0.021389
12       marital_married     0.021149
14  job fill_blue-collar     0.016255
13       marital_single     0.015927
9              month_may     0.014067
17     job fill_services     0.012384
16      job fill_retired     0.010398
15  job fill_entrepreneur    0.007741
8              month_jul     0.003795
10             month_nov     0.003566
```

[125]: Text(0, 0.5, 'Name of feature')



[126]:

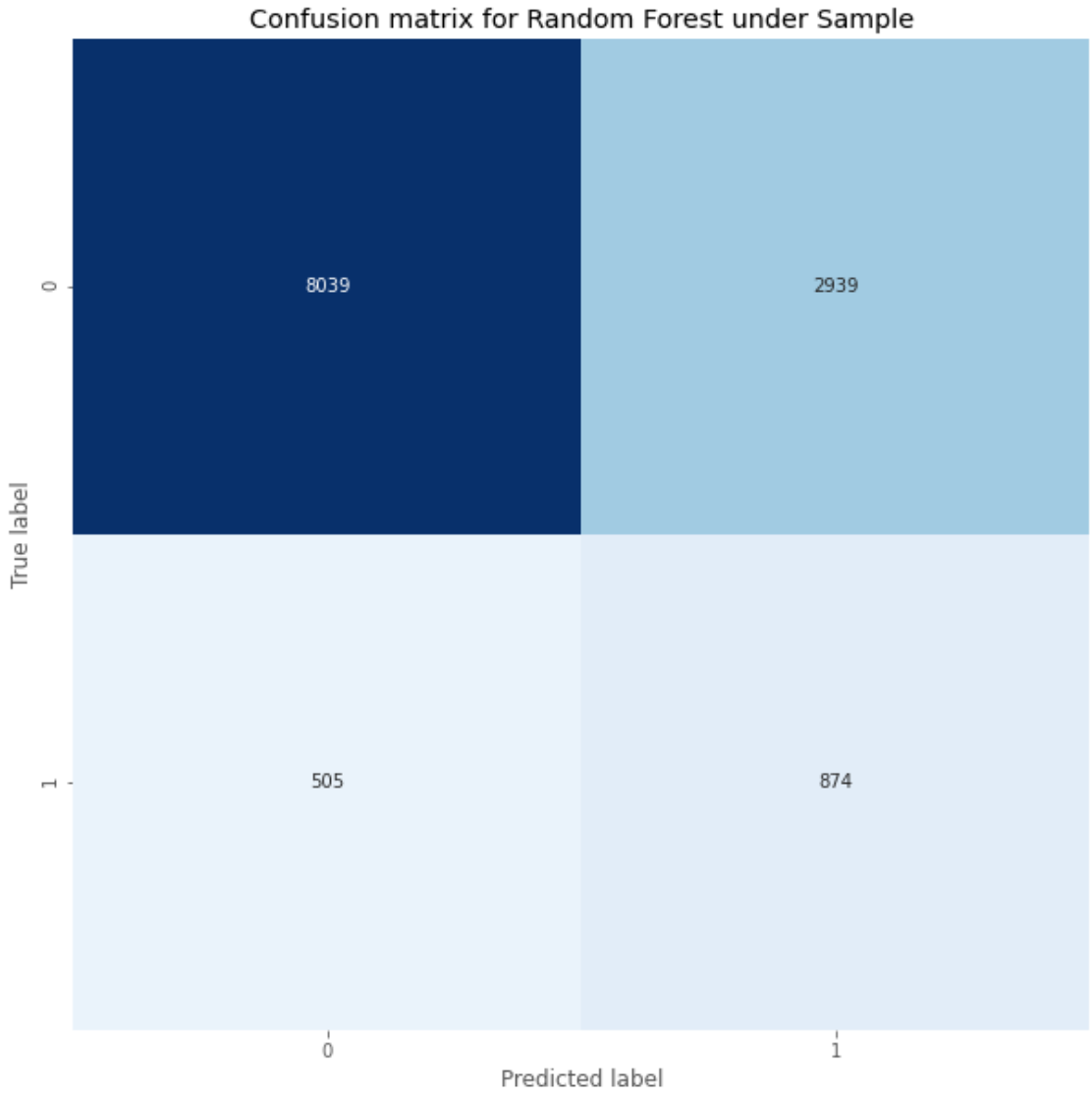
Classifier	Accuracy	F-score	Recall	Log_loss
Logistic Regression SMOTETomek	0.8107	0.4192	0.6120	0.5300
Logistic Regression under Sample	0.8104	0.4185	0.6113	0.5306
Logistic Regression SMOTE	0.8101	0.4181	0.6113	0.5293
Logistic Regression SMOTETomek PCA	0.7841	0.3936	0.6280	0.5618

Random Forest	0.8805	0.3592	0.3002	1.0429
Logistic Regression TomekLinks	0.8948	0.3428	0.2458	0.2860
Logistic Regression	0.8954	0.2798	0.1820	0.2855

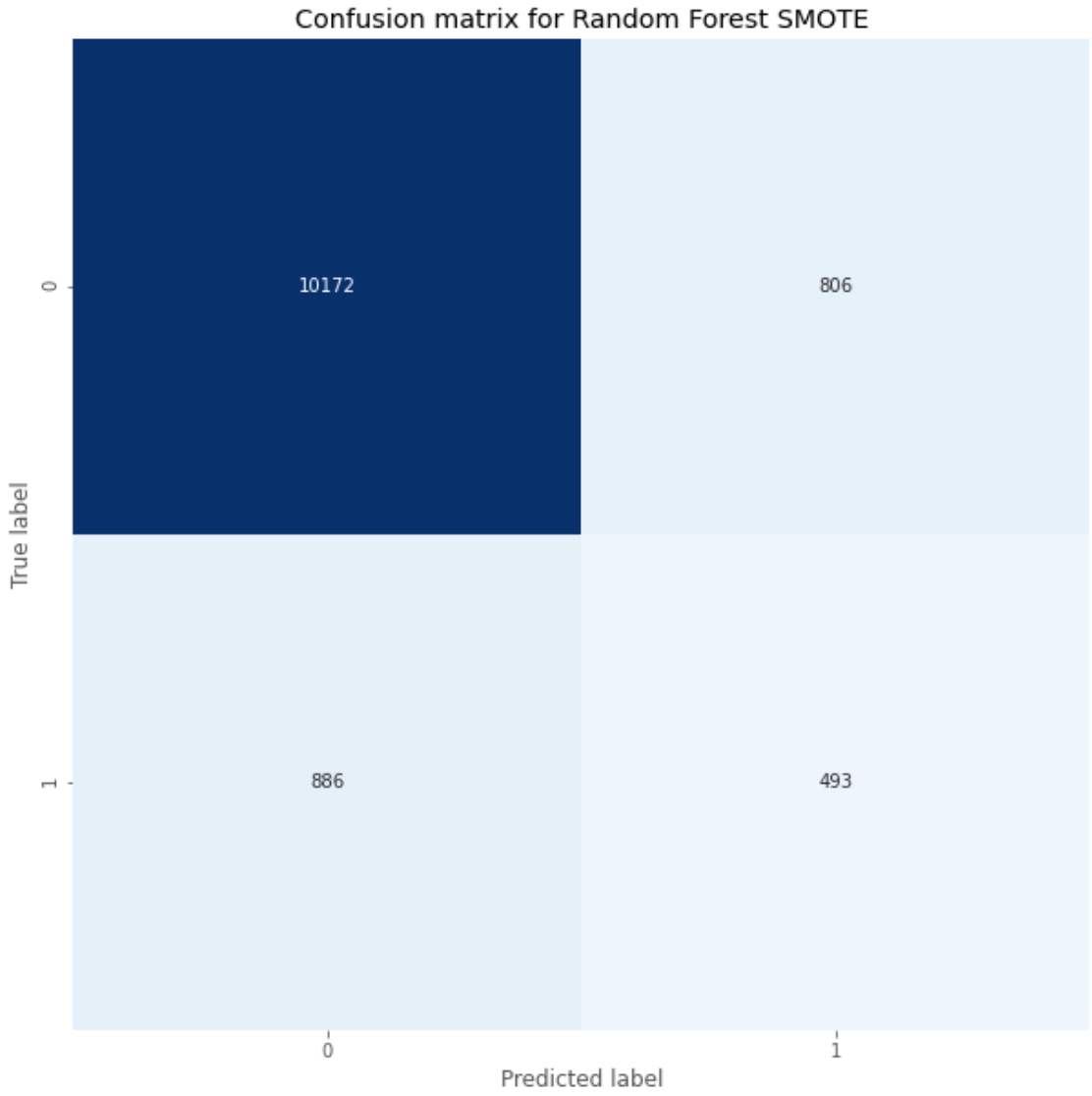
Observations:

- Random Forest (without sampling) gives us better results than Logistic Regression (F-score, recall), so now we can check this model with sampling
- Age is highest positivie coefficient, the higher the value, the more likely the answer is yes.
- Random Forest not requires standarize of data, but during testing model, better results (a little) our model achives with the standarization.

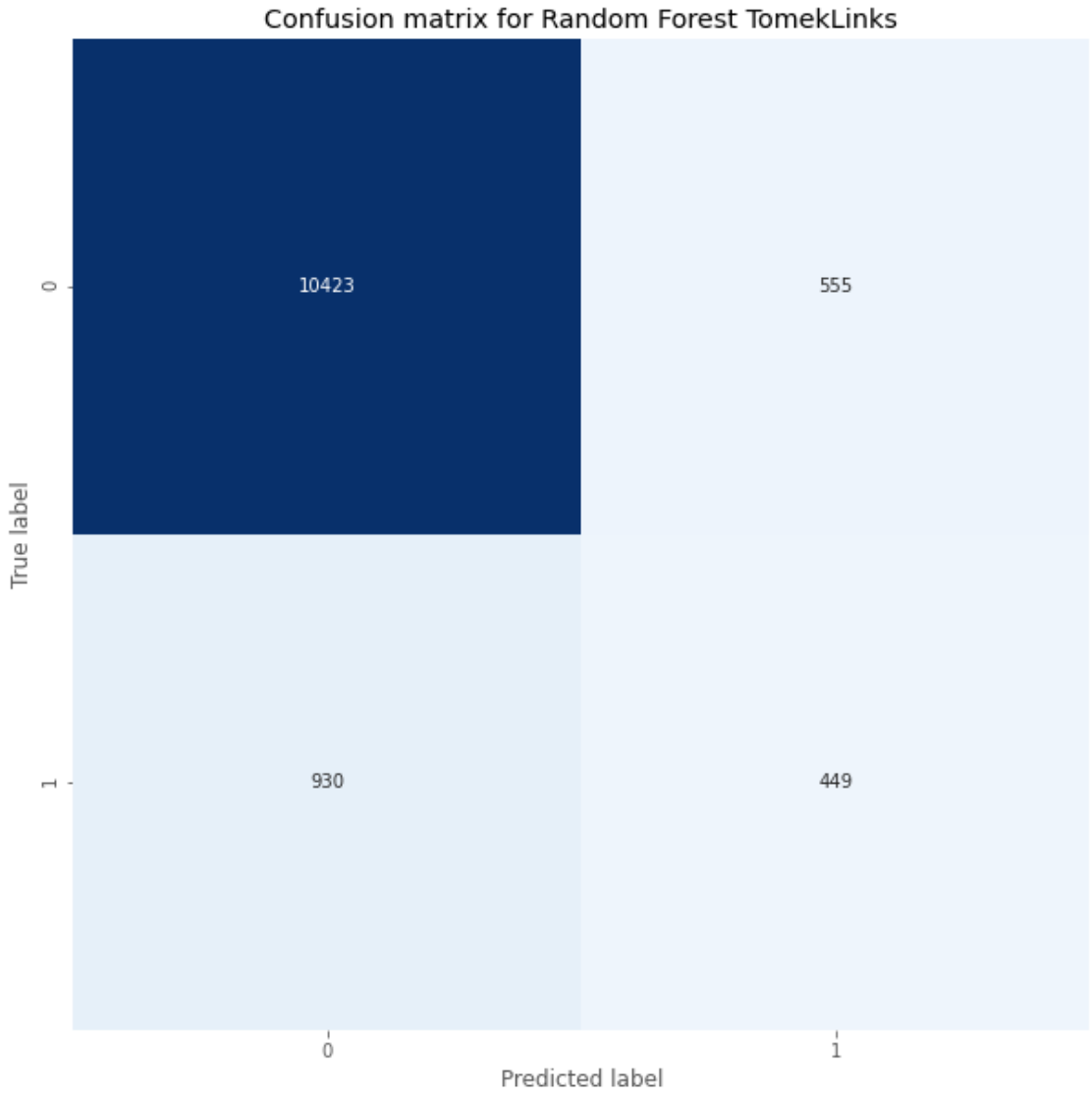
```
Results for Random Forest under Sample:
Random Forest under Sample accuracy: 0.7213
Random Forest under Sample f-score: 0.3367
Random Forest under Sample recall: 0.6338
Random Forest under Sample log_loss: 1.9856
```



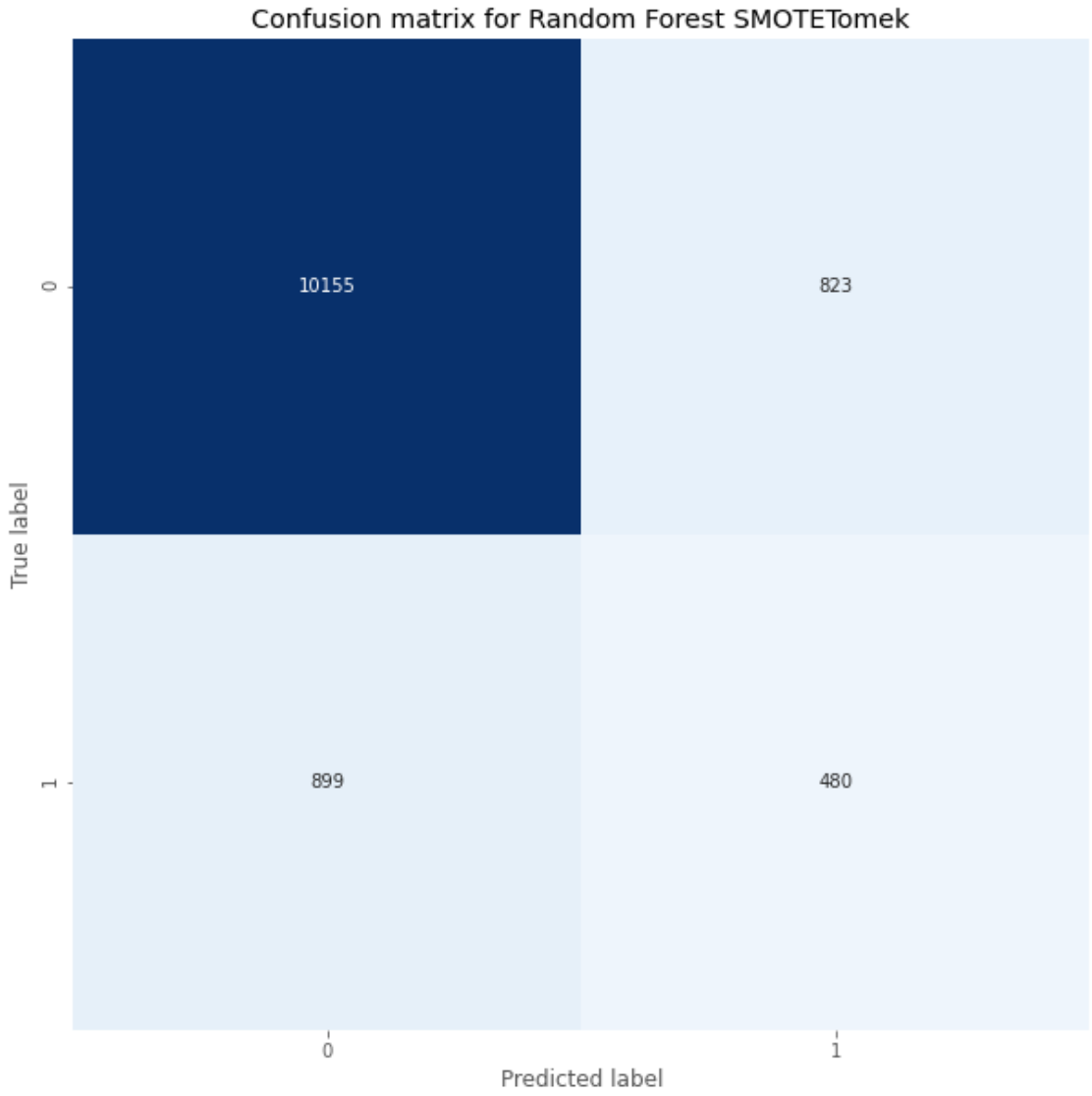

```
Results for Random Forest SMOTE:
Random Forest SMOTE accuracy: 0.8631
Random Forest SMOTE f-score: 0.3682
Random Forest SMOTE recall: 0.3575
Random Forest SMOTE log_loss: 1.6816
```




```
Results for Random Forest TomekLinks:
Random Forest TomekLinks accuracy: 0.8798
Random Forest TomekLinks f-score: 0.3768
Random Forest TomekLinks recall: 0.3256
Random Forest TomekLinks log_loss: 1.5544
```



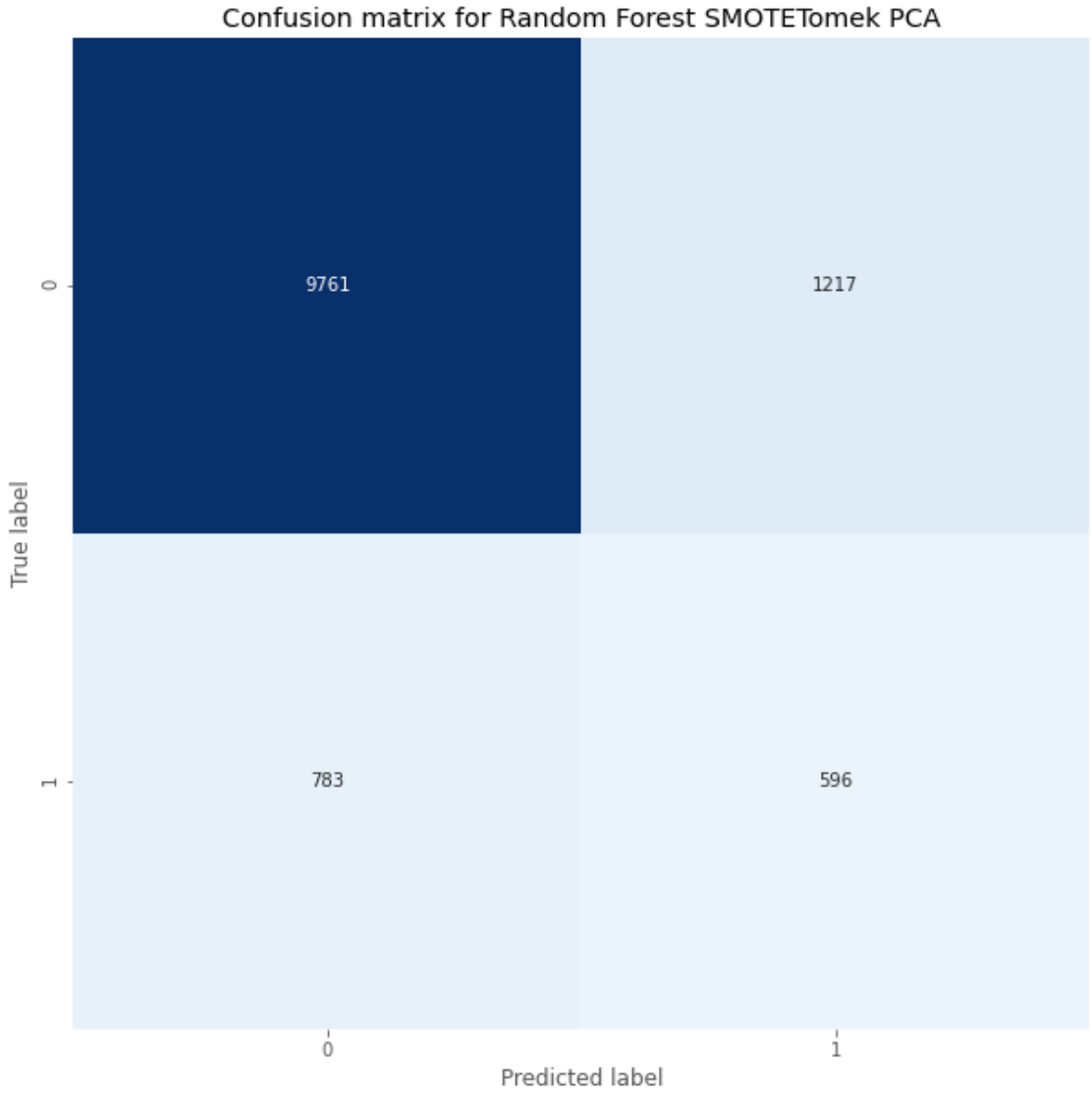

```
Results for Random Forest SMOTETomek:
Random Forest SMOTETomek accuracy: 0.8606
Random Forest SMOTETomek f-score: 0.3579
Random Forest SMOTETomek recall: 0.3481
Random Forest SMOTETomek log_loss: 1.6868
```



[131]:

Classifier	Accuracy	F-score	Recall	Log_loss
Logistic Regression SMOTETomek	0.8107	0.4192	0.6120	0.5300
Logistic Regression under Sample	0.8104	0.4185	0.6113	0.5306
Logistic Regression SMOTE	0.8101	0.4181	0.6113	0.5293
Logistic Regression SMOTETomek PCA	0.7841	0.3936	0.6280	0.5618
Random Forest TomekLinks	0.8798	0.3768	0.3256	1.5544
Random Forest SMOTE	0.8631	0.3682	0.3575	1.6816
Random Forest	0.8805	0.3592	0.3002	1.0429
Random Forest SMOTETomek	0.8606	0.3579	0.3481	1.6868
Logistic Regression TomekLinks	0.8948	0.3428	0.2458	0.2860
Random Forest under Sample	0.7213	0.3367	0.6338	1.9856
Logistic Regression	0.8954	0.2798	0.1820	0.2855

```
Results for Random Forest SMOTETomek PCA:
Random Forest SMOTETomek PCA accuracy: 0.8381
Random Forest SMOTETomek PCA f-score: 0.3734
Random Forest SMOTETomek PCA recall: 0.4322
Random Forest SMOTETomek PCA log_loss: 1.8546
```



[133]:

Classifier	Accuracy	F-score	Recall	Log_loss
Logistic Regression SMOTETomek	0.8107	0.4192	0.6120	0.5300
Logistic Regression under Sample	0.8104	0.4185	0.6113	0.5306
Logistic Regression SMOTE	0.8101	0.4181	0.6113	0.5293
Logistic Regression SMOTETomek PCA	0.7841	0.3936	0.6280	0.5618
Random Forest TomekLinks	0.8798	0.3768	0.3256	1.5544
Random Forest SMOTETomek PCA	0.8381	0.3734	0.4322	1.8546

Random Forest SMOTE	0.8631	0.3682	0.3575	1.6816
Random Forest	0.8805	0.3592	0.3002	1.0429
Random Forest SMOTETomek	0.8606	0.3579	0.3481	1.6868
Logistic Regression TomekLinks	0.8948	0.3428	0.2458	0.2860
Random Forest under Sample	0.7213	0.3367	0.6338	1.9856
Logistic Regression	0.8954	0.2798	0.1820	0.2855

Observations:

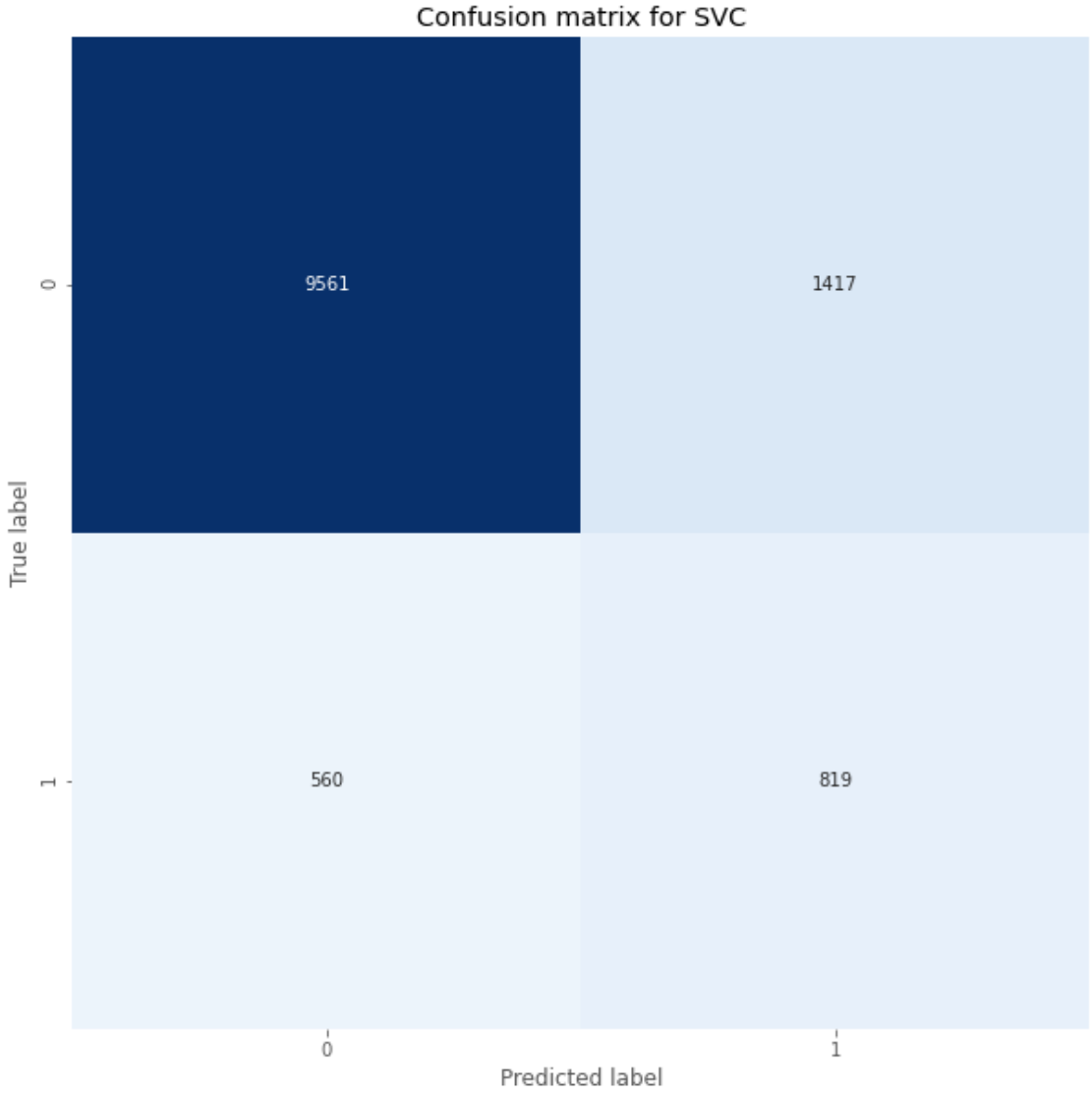
- Random Forest with resampling give us better results than without resampling.
- Random Forest with PCA give us a little better results than without but still Logistic Regression is better.
- Logistic Regression with resample SMOTETomek gives us the best result (almost the same like SMOTE).
- In table we can see that we have tree similar models. They have similar measure importat for us: F-score, also recall is important.
- In random forest standarize data also give us better results.

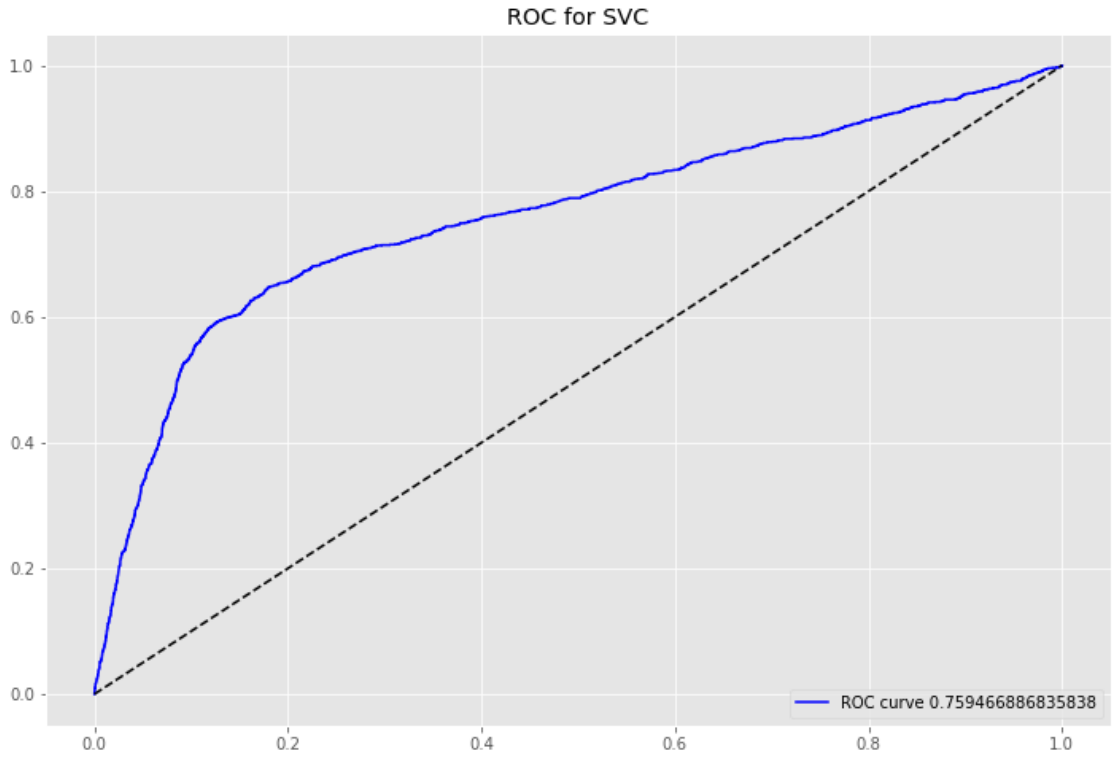
2.1.7 SVM

Support vector machines (SVM) are particular linear classifiers which are based on the margin maximization principle. They perform structural risk minimization, which improves the complexity of the classifier with the aim of achieving excellent generalization performance. The SVM accomplishes the classification task by constructing, in a higher dimensional space, the hyperplane that optimally separates the data into two categories.

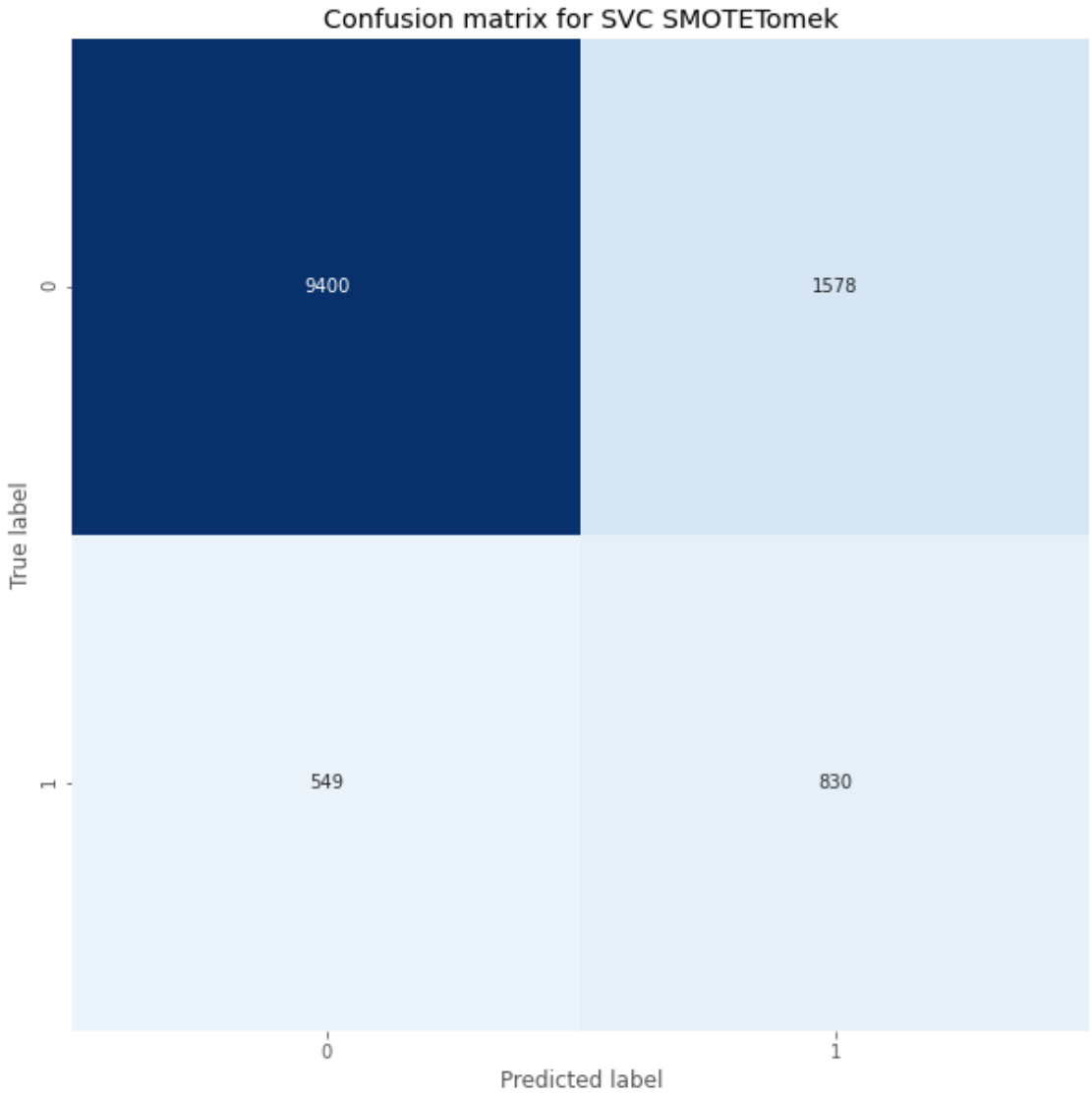
```
Results for SVC:
SVC accuracy: 0.84
SVC f-score: 0.4531
SVC recall: 0.5939
SVC log_loss: 0.2892
```

[134]: <matplotlib.legend.Legend at 0x5765b80>


Confusion matrix for SVC
9561
1417
560
819
0
1
True label
0
1
Predicted label




```
Results for SVC SMOTETomek:
SVC SMOTETomek accuracy: 0.8279
SVC SMOTETomek f-score: 0.4383
SVC SMOTETomek recall: 0.6019
SVC SMOTETomek log_loss: 0.5165
```



[136]:

Classifier	Accuracy	F-score	Recall	Log_loss
SVC	0.8400	0.4531	0.5939	0.2892
SVC SMOTETomek	0.8279	0.4383	0.6019	0.5165
Logistic Regression SMOTETomek	0.8107	0.4192	0.6120	0.5300
Logistic Regression under Sample	0.8104	0.4185	0.6113	0.5306
Logistic Regression SMOTE	0.8101	0.4181	0.6113	0.5293
Logistic Regression SMOTETomek PCA	0.7841	0.3936	0.6280	0.5618
Random Forest TomekLinks	0.8798	0.3768	0.3256	1.5544
Random Forest SMOTETomek PCA	0.8381	0.3734	0.4322	1.8546
Random Forest SMOTE	0.8631	0.3682	0.3575	1.6816
Random Forest	0.8805	0.3592	0.3002	1.0429
Random Forest SMOTETomek	0.8606	0.3579	0.3481	1.6868

```
Logistic Regression TomekLinks        0.8948   0.3428  0.2458    0.2860
Random Forest under Sample            0.7213   0.3367  0.6338    1.9856
Logistic Regression                   0.8954   0.2798  0.1820    0.2855
```

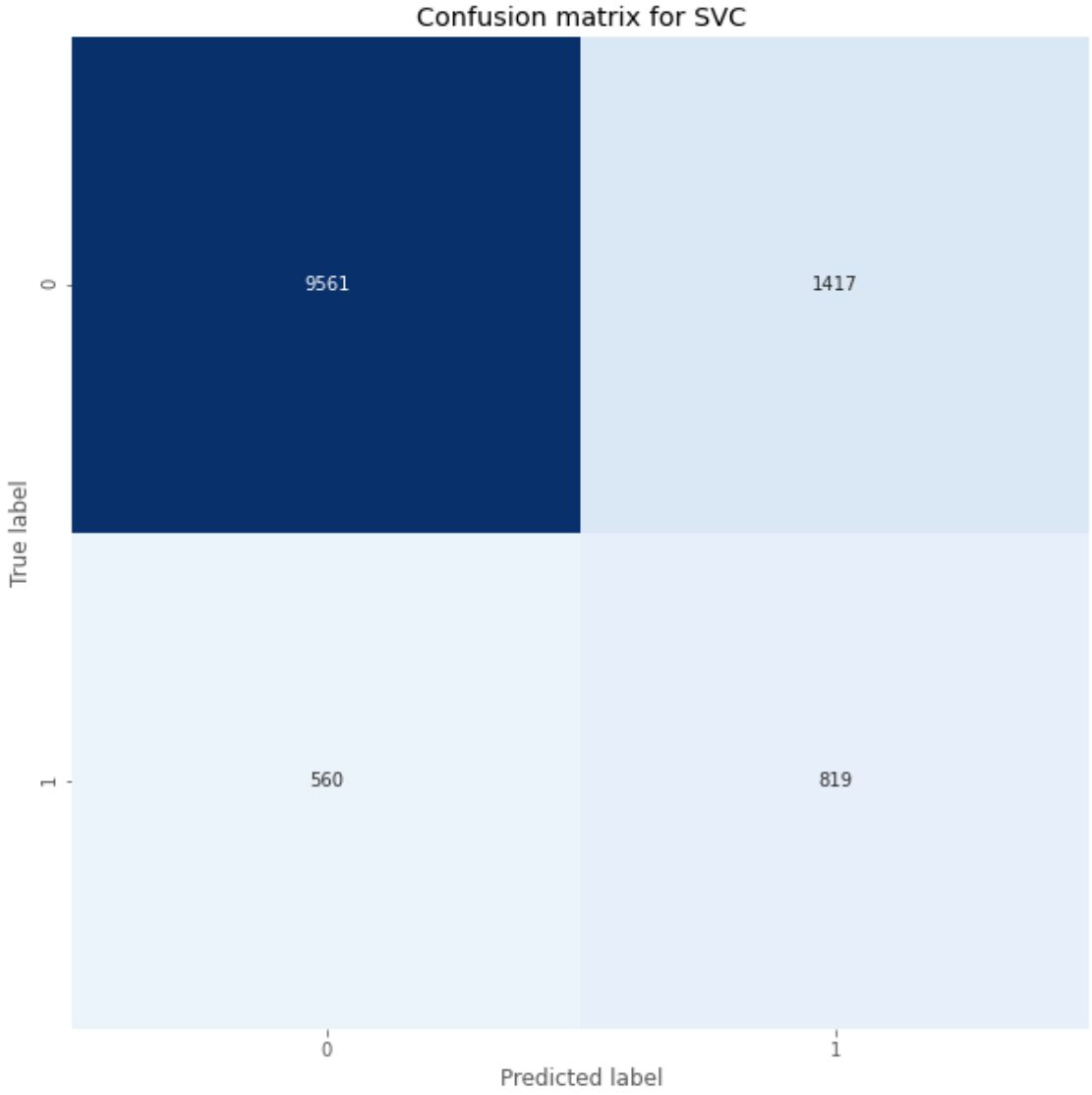
Conclusions:

- SVC model is the best results. As we can see in table, this method has the highest F-score. We receive F1-score about 0.45.
- SVC without resample works better tham with resample.

3 Summary

During creating of this analysis, we could discover information about correlation, affect to output and thanks to this we could do our model. Now, we may use our model to predict if client has subscribe term deposit base on delivered informations. It helps to bank to reduce number of employees that contact with people (greater precision in selecting customers). Our model is based on SVC. Below, one more time we can see our results for this model.

[137]:
```
Classifier  Accuracy  F-score  Recall  Log_loss
SVC             0.84   0.4531  0.5939    0.2892
```


Confusion matrix for SVC
9561
1417
560
819
0
1
True label
0
1
Predicted label