Imports

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from scipy import stats
import collections
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, acc
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
import warnings

#importing packages for modeling
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline


%matplotlib inline
warnings.filterwarnings('ignore')
```

Dataset

In [5]:

```
bank_df = pd.read_csv('bank-additional-full.csv',sep=';')
```

In [6]:

```
bank_df.head()
```

Out[6]:

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

5 rows × 21 columns

Data Exploratory

In [7]:

```
bank_df.isna().sum()
```

Out[7]:

```
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
```

In [8]:

```
bank_df.columns
```

Out[8]:

```
Index(['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'],
      dtype='object')
```

In [9]:

```
bank_df.values
```

Out[9]:

```
array([[56, 'housemaid', 'married', ..., 4.857, 5191.0, 'no'],
       [57, 'services', 'married', ..., 4.857, 5191.0, 'no'],
       [37, 'services', 'married', ..., 4.857, 5191.0, 'no'],
       ...,
       [56, 'retired', 'married', ..., 1.028, 4963.6, 'no'],
       [44, 'technician', 'married', ..., 1.028, 4963.6, 'yes'],
       [74, 'retired', 'married', ..., 1.028, 4963.6, 'no']], dtype=object)
```

In [10]:

```
bank_df.dtypes
```

Out[10]:

```
age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
```

```
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
```

In [11]:

```
bank_df.info()
```

```
<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  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

In [12]:

```
bank_df.describe()
```

Out[12]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.id
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.00000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.57566
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.57884
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.20100
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.07500
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.74900
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.99400
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.76700

In [13]:

```
bank_df.nunique()
```

Out[13]:

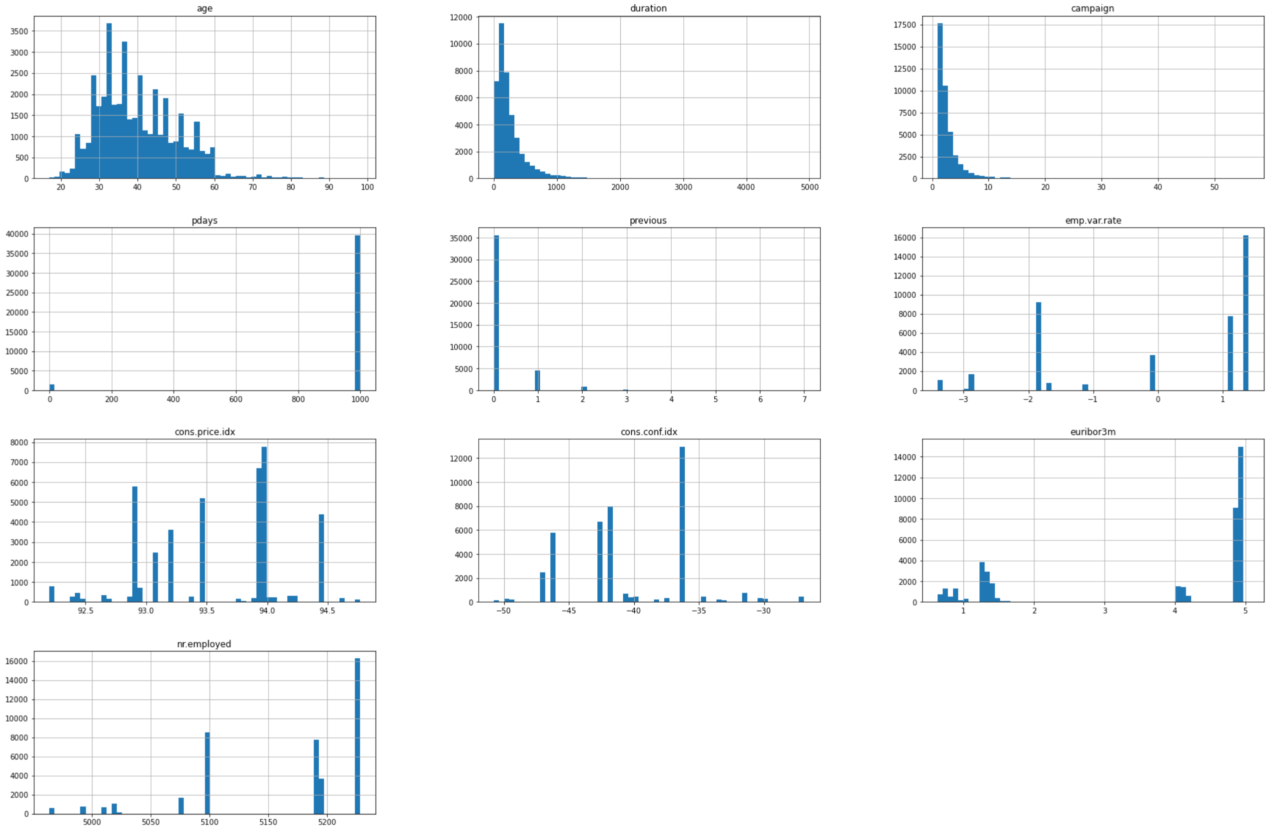
```
age                 78
job                 12
marital              4
education            8
default              3
housing              3
loan                 3
contact              2
month               10
day_of_week          5
duration          1544
campaign            42
pdays               27
previous             8
poutcome             3
emp.var.rate        10
cons.price.idx      26
cons.conf.idx       26
euribor3m          316
nr.employed         11
y                    2
dtype: int64
```

In [14]:

```
bank_df.hist(bins=60, figsize=(30,20))
```

Out[14]:

```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'duration'}>,
        <AxesSubplot:title={'center':'campaign'}>],
       [<AxesSubplot:title={'center':'pdays'}>,
        <AxesSubplot:title={'center':'previous'}>,
        <AxesSubplot:title={'center':'emp.var.rate'}>],
       [<AxesSubplot:title={'center':'cons.price.idx'}>,
        <AxesSubplot:title={'center':'cons.conf.idx'}>,
        <AxesSubplot:title={'center':'euribor3m'}>],
       [<AxesSubplot:title={'center':'nr.employed'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



In [15]:
```
prev_zero = bank_df[bank_df['previous'] == 0]
```

In [16]:
```
prev_zero['poutcome'].unique()
```

Out[16]:
```
array(['nonexistent'], dtype=object)
```

In [17]:
```
prev_one = bank_df[bank_df['previous'] > 0]
```

In [18]:
```
prev_one['poutcome'].unique()
```

Out[18]:
```
array(['failure', 'success'], dtype=object)
```

In [19]:
```
bank_default = bank_df.loc[(bank_df['housing'] == 'no') & (bank_df['loan'] == 'no') & (
```

In [20]:
```
bank_default
```

Out[20]:

	age	job	marital	education	default	housing	loan	contact	month	day_of
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	57	services	married	high.school	unknown	no	no	telephone	may	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	
5	45	services	married	basic.9y	unknown	no	no	telephone	may	

	age	job	marital	education	default	housing	loan	contact	month	day_of
6	59	admin.	married	professional.course	no	no	no	telephone	may	
...	...	...	...	...	...	...	...	...	...	
41065	29	technician	single	university.degree	no	no	no	telephone	nov	
41082	48	admin.	married	high.school	no	no	no	telephone	nov	
41129	61	admin.	married	high.school	no	no	no	telephone	nov	
41155	31	housemaid	single	university.degree	no	no	no	telephone	nov	
41166	32	admin.	married	university.degree	no	no	no	telephone	nov	

6566 rows × 21 columns

In [21]:

```
bank_default['default'].unique()
```

Out[21]:

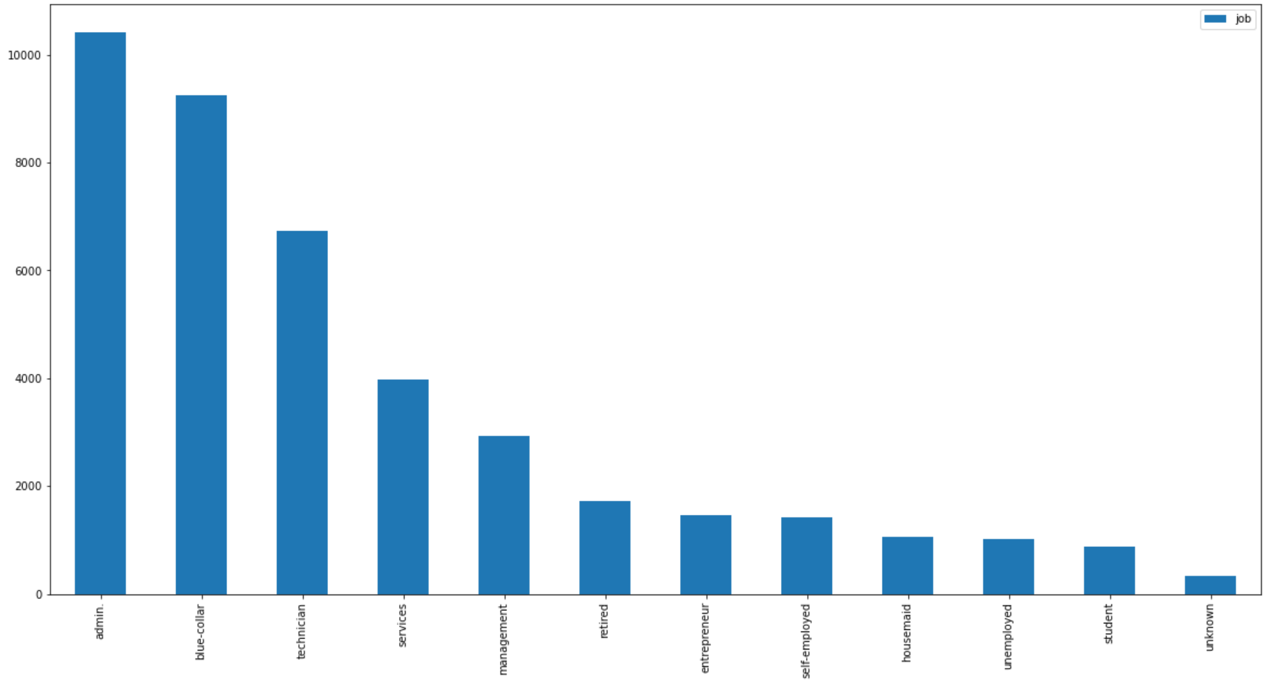
```
array(['no', 'unknown'], dtype=object)
```

In [22]:

```
pd.DataFrame(bank_df['job'].value_counts()).plot(kind='bar', figsize=(20,10))
pd.DataFrame(bank_df['job'].value_counts())
```

Out[22]:

	job
admin.	10422
blue-collar	9254
technician	6743
services	3969
management	2924
retired	1720
entrepreneur	1456
self-employed	1421
housemaid	1060
unemployed	1014
student	875
unknown	330

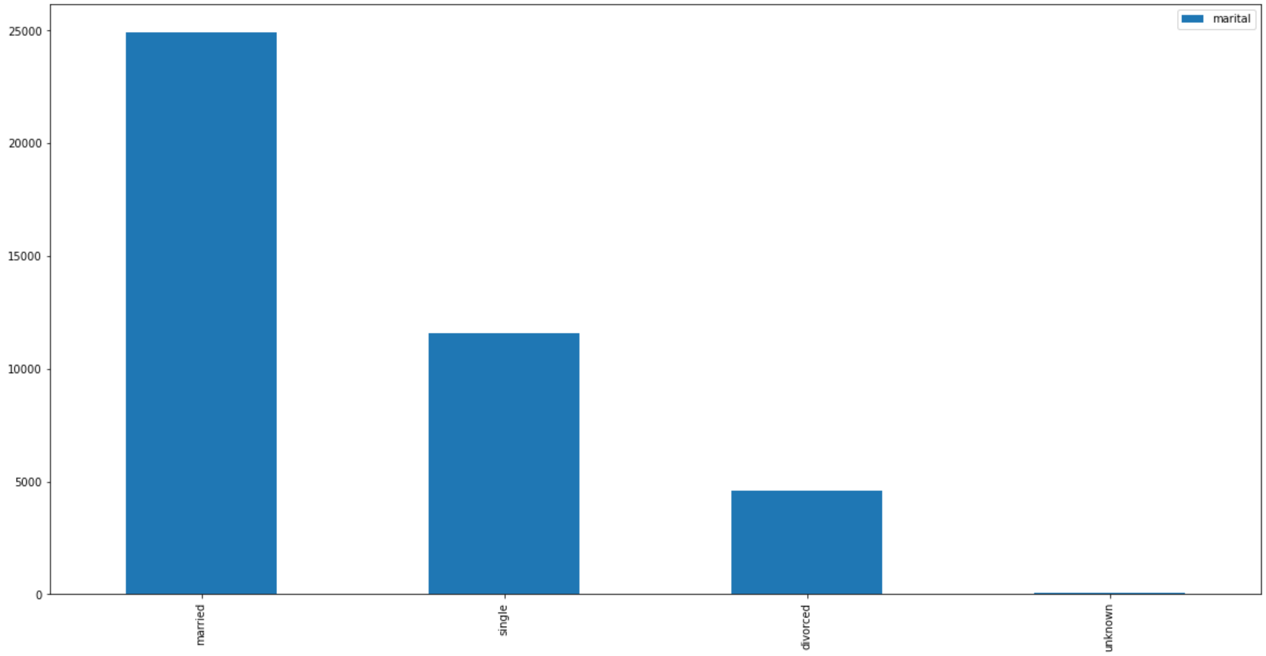


In [24]:

```
pd.DataFrame(bank_df['marital'].value_counts()).plot(kind='bar', figsize=(20,10))
pd.DataFrame(bank_df['marital'].value_counts())
```

Out[24]:

	marital
married	24928
single	11568
divorced	4612
unknown	80

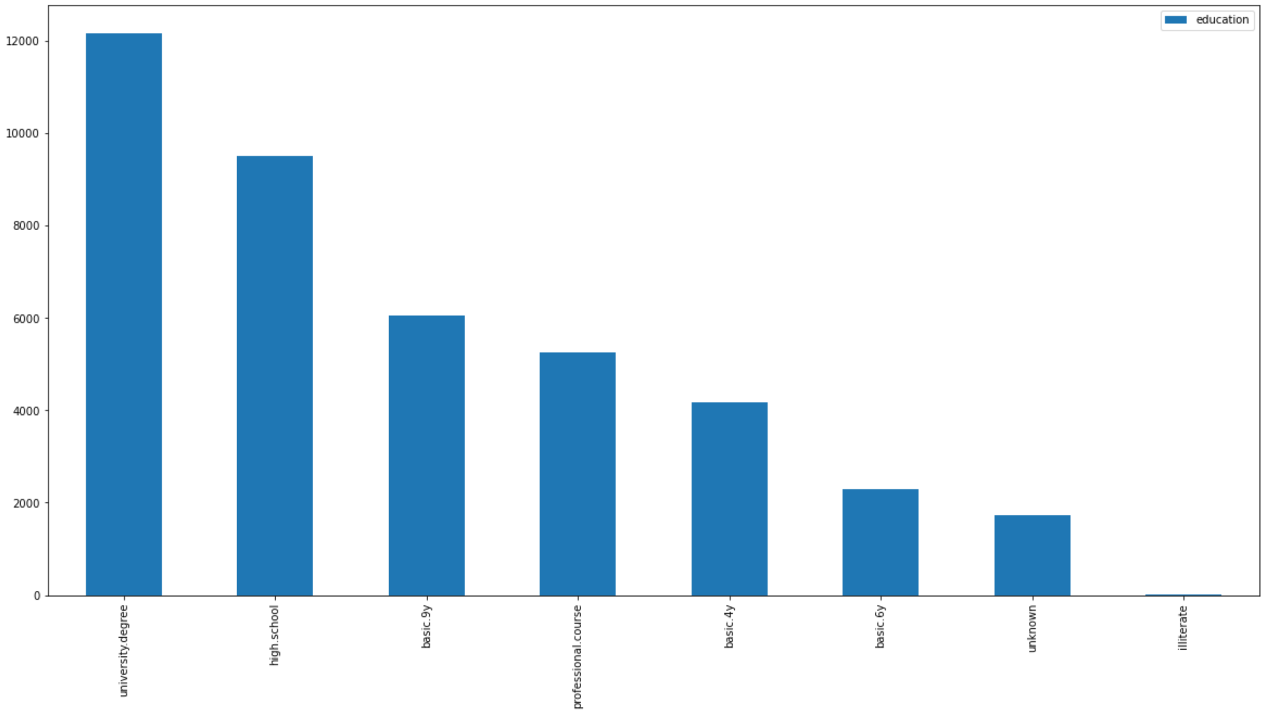


In [25]:

```
pd.DataFrame(bank_df['education'].value_counts()).plot(kind='bar', figsize=(20,10))
pd.DataFrame(bank_df['education'].value_counts())
```

Out[25]:

	education
university.degree	12168
high.school	9515
basic.9y	6045
professional.course	5243
basic.4y	4176
basic.6y	2292
unknown	1731
illiterate	18



In [27]:
```
nan_bank_df = bank_df
```

In [28]:
```
nan_bank_df['default'].replace('unknown', np.nan, inplace = True)
```

In [29]:
```
nan_bank_df['loan'].replace('unknown', np.nan, inplace = True)
```

In [30]:
```
nan_bank_df['housing'].replace('unknown', np.nan, inplace = True)
```

In [31]:
```
nan_bank_df['education'].replace('unknown', np.nan, inplace = True)
```

In [32]:
```
nan_bank_df['job'].replace('unknown', np.nan, inplace = True)
```

In [33]:

```
nan_bank_df['marital'].replace('unknown', np.nan, inplace = True)
```

In [34]:

```
nan_bank_df
```

Out[34]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	57	services	married	high.school	NaN	no	no	telephone	may	
2	37	services	married	high.school	no	yes	no	telephone	may	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	
4	56	services	married	high.school	no	no	yes	telephone	may	
...	...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	
41186	44	technician	married	professional.course	no	no	no	cellular	nov	
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	

41188 rows × 21 columns

In [35]:

```
nan_bank_df.isna().sum()
```

Out[35]:

```
age                  0
job                330
marital             80
education         1731
default           8597
housing            990
loan               990
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
```

In [36]:

```
new_bank_df = nan_bank_df.sort_values(by='age', ascending=True)
```

In [37]:
```
new_bank_df
```

Out[37]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	ca
38274	17	student	single	NaN	no	no	yes	cellular	oct	tue	...	
37579	17	student	single	basic.9y	no	NaN	NaN	cellular	aug	fri	...	
37539	17	student	single	basic.9y	no	yes	no	cellular	aug	fri	...	
37140	17	student	single	NaN	no	yes	no	cellular	aug	wed	...	
37558	17	student	single	basic.9y	no	yes	no	cellular	aug	fri	...	
...	...	...	...	...	...	...	...	...	...	...	...	
40450	92	retired	married	NaN	no	no	yes	cellular	aug	tue	...	
38921	94	retired	married	basic.9y	no	no	no	cellular	nov	wed	...	
27826	95	retired	divorced	basic.6y	no	no	no	cellular	mar	thu	...	
38455	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	fri	...	
38452	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	fri	...	

41188 rows × 21 columns

In [38]:
```
new_bank_df['job'].unique()
```

Out[38]:
```
array(['student', 'technician', 'services', 'entrepreneur', 'blue-collar',
       'admin.', 'unemployed', 'management', 'self-employed', 'housemaid',
       'retired', nan], dtype=object)
```

In [40]:
```
job_marital = pd.DataFrame()

job_marital['student'] = new_bank_df[new_bank_df['job'] == 'student']['marital'].value_
job_marital['housemaid'] = new_bank_df[new_bank_df['job'] == 'housemaid']['marital'].va
job_marital['services'] = new_bank_df[new_bank_df['job'] == 'services']['marital'].valu
job_marital['admin.'] = new_bank_df[new_bank_df['job'] == 'admin.']['marital'].value_co
job_marital['blue-collar'] = new_bank_df[new_bank_df['job'] == 'blue-collar']['marital'
job_marital['technician'] = new_bank_df[new_bank_df['job'] == 'technician']['marital'].
job_marital['retired'] = new_bank_df[new_bank_df['job'] == 'retired']['marital'].value_
job_marital['management'] = new_bank_df[new_bank_df['job'] == 'management']['marital'].
job_marital['unemployed'] = new_bank_df[new_bank_df['job'] == 'unemployed']['marital'].
job_marital['self-employed'] = new_bank_df[new_bank_df['job'] == 'self-employed']['mari
job_marital['entrepreneur'] = new_bank_df[new_bank_df['job'] == 'entrepreneur']['marita
job_marital['nan'] = new_bank_df[new_bank_df['job'] == 'nan']['marital'].value_counts()
```

In [41]:
```
job_marital
```

Out[41]:

	student	housemaid	services	admin.	blue-collar	technician	retired	management	unemployed
single	824	119	1137	3875	1825	2287	93	501	251
married	41	777	2294	5253	6687	3670	1274	2089	634

	student	housemaid	services	admin.	blue-collar	technician	retired	management	unemployed
divorced	9	161	532	1280	728	774	348	331	124

In [42]:

```
new_bank_df['marital'].unique()
```

Out[42]:

```
array(['single', 'married', 'divorced', nan], dtype=object)
```

In [43]:

```
age_marital = pd.DataFrame()

age_marital['married'] = new_bank_df[new_bank_df['marital'] == 'married']['age'].value_
age_marital['single'] = new_bank_df[new_bank_df['marital'] == 'single']['age'].value_co
age_marital['divorced'] = new_bank_df[new_bank_df['marital'] == 'divorced']['age'].valu
age_marital['nan'] = new_bank_df[new_bank_df['marital'] == 'nan']['age'].value_counts()
```

In [44]:

```
age_marital.sort_index()
```

Out[44]:

	married	single	divorced	nan
20	1	64.0	NaN	NaN
21	8	94.0	NaN	NaN
22	16	121.0	NaN	NaN
23	30	196.0	NaN	NaN
24	78	381.0	4.0	NaN
...	...	...	...	...
88	4	NaN	18.0	NaN
91	2	NaN	NaN	NaN
92	3	NaN	1.0	NaN
94	1	NaN	NaN	NaN
98	2	NaN	NaN	NaN

72 rows × 4 columns

In [45]:

```
new_bank_df['education'].unique()
```

Out[45]:

```
array([nan, 'basic.9y', 'high.school', 'basic.6y', 'basic.4y',
       'university.degree', 'professional.course', 'illiterate'],
      dtype=object)
```

In [46]:

```
age_education = pd.DataFrame()

age_education['basic.9y'] = new_bank_df[new_bank_df['education'] == 'basic.9y']['age'].
age_education['high.school'] = new_bank_df[new_bank_df['education'] == 'high.school']['
```

```
age_education['basic.6y'] = new_bank_df[new_bank_df['education'] == 'basic.6y']['age'].
age_education['basic.4y'] = new_bank_df[new_bank_df['education'] == 'basic.4y']['age'].
age_education['university.degree'] = new_bank_df[new_bank_df['education'] == 'universit
age_education['professional.course'] = new_bank_df[new_bank_df['education'] == 'profess
age_education['illiterate'] = new_bank_df[new_bank_df['education'] == 'illiterate']['ag
age_education['nan'] = new_bank_df[new_bank_df['education'] == 'nan']['age'].value_coun
```

In [47]:

```
age_education.sort_index()
```

Out[47]:

	basic.9y	high.school	basic.6y	basic.4y	university.degree	professional.course	illiterate	nan
17	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	4	5.0	2.0	4.0	NaN	NaN	NaN	NaN
19	14	4.0	4.0	3.0	NaN	NaN	NaN	NaN
20	6	33.0	NaN	7.0	2.0	2.0	NaN	NaN
21	24	52.0	NaN	2.0	8.0	4.0	NaN	NaN
...	...	...	...	...	...	...	...	...
81	1	2.0	1.0	10.0	1.0	2.0	NaN	NaN
83	2	2.0	NaN	8.0	2.0	1.0	NaN	NaN
84	1	1.0	NaN	4.0	NaN	NaN	NaN	NaN
86	1	NaN	NaN	4.0	NaN	1.0	NaN	NaN
94	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN

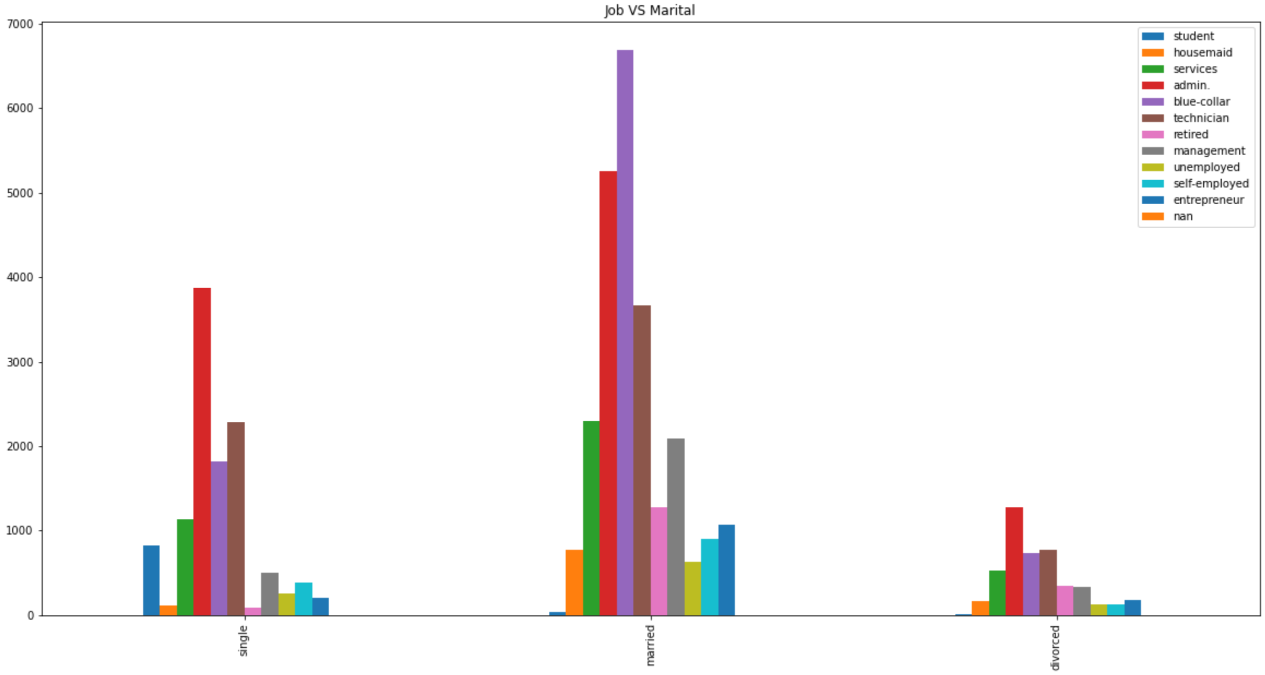
66 rows × 8 columns

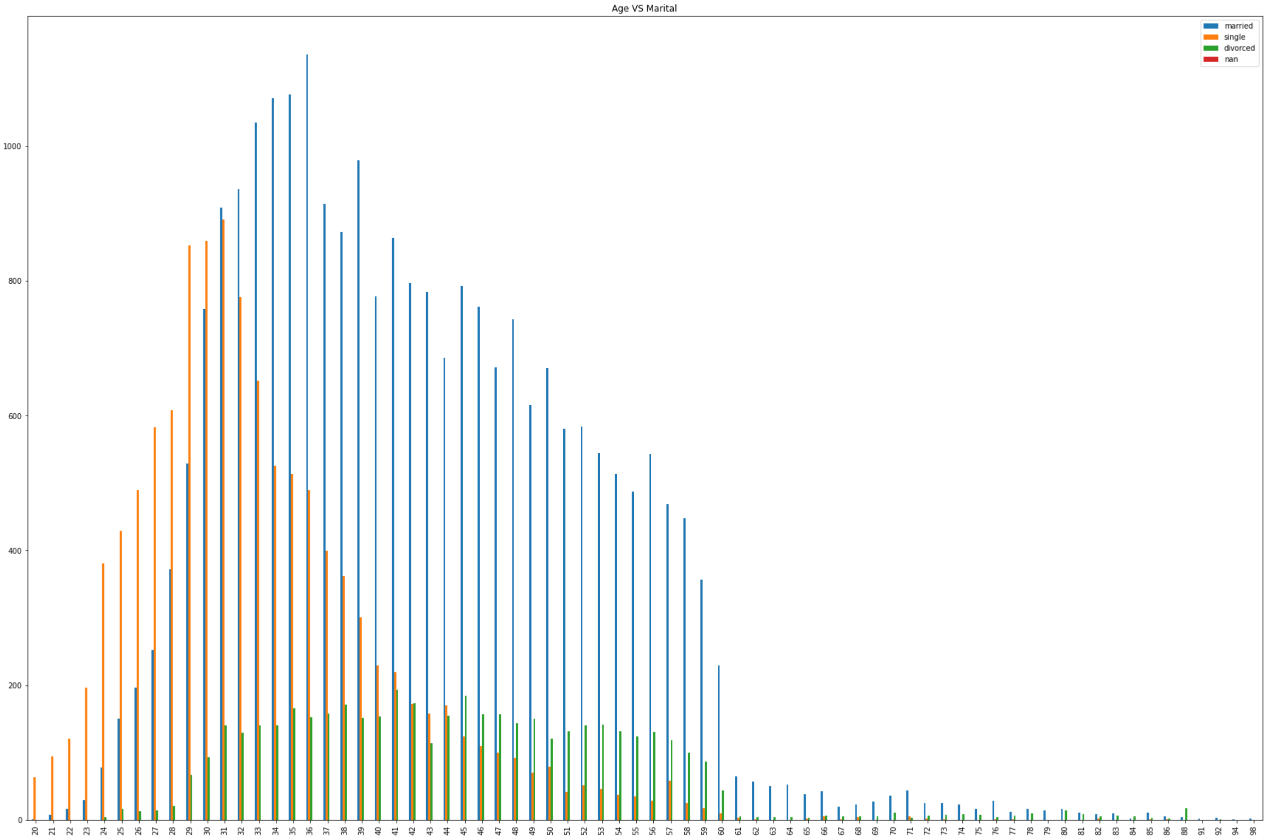
In [48]:

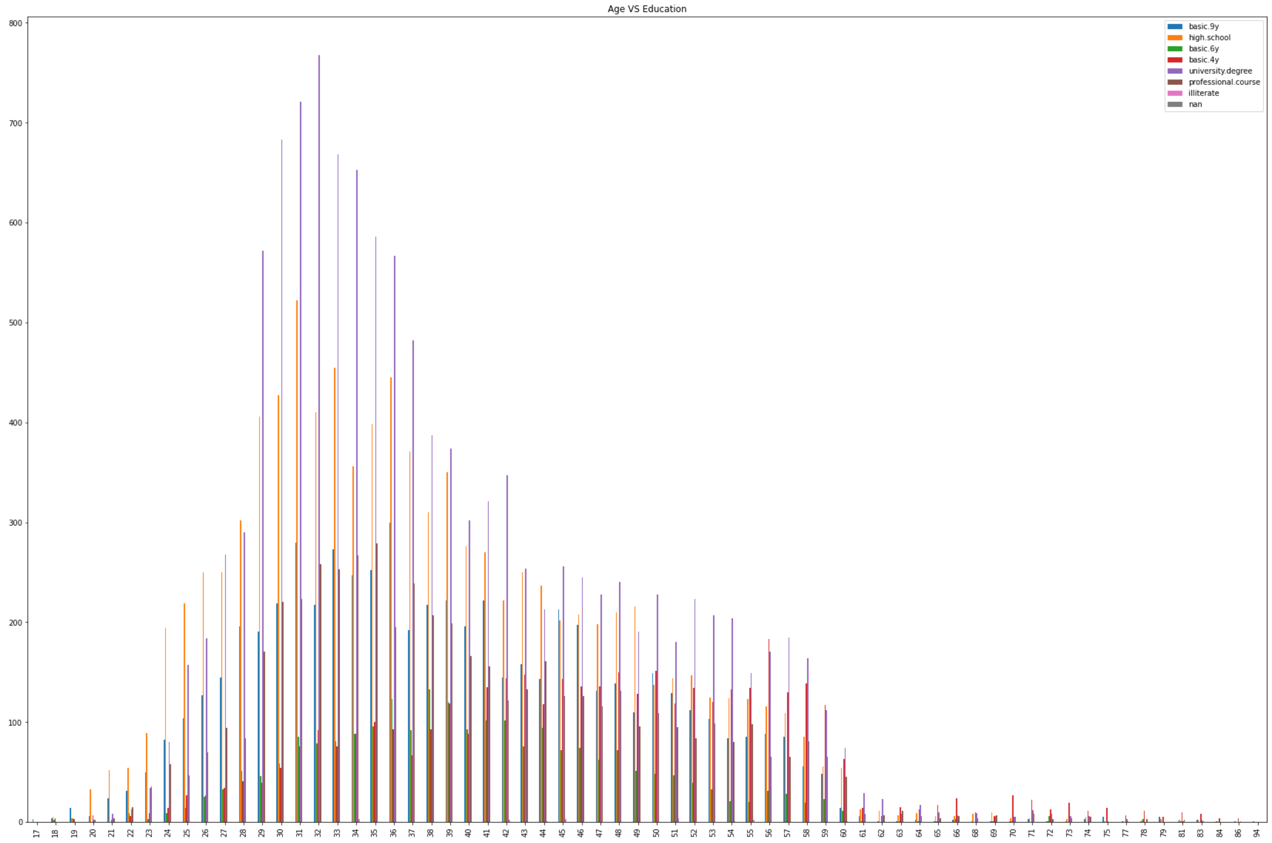
```
job_marital.plot.bar(title = "Job VS Marital", figsize=(20,10))
age_marital.sort_index().plot.bar(title = 'Age VS Marital', figsize = (30,20))
age_education.sort_index().plot.bar(title = 'Age VS Education', figsize = (30,20))
```

Out[48]:

```
<AxesSubplot:title={'center':'Age VS Education'}>
```


Job VS Marital
student
housemaid
services
admin.
blue-collar
technician
retired
management
unemployed
self-employed
entrepreneur
nan
0
1000
2000
3000
4000
5000
6000
7000
single
married
divorced


Age VS Marital
married
single
divorced
nan
0
200
400
600
800
1000
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
88
91
92
94
98

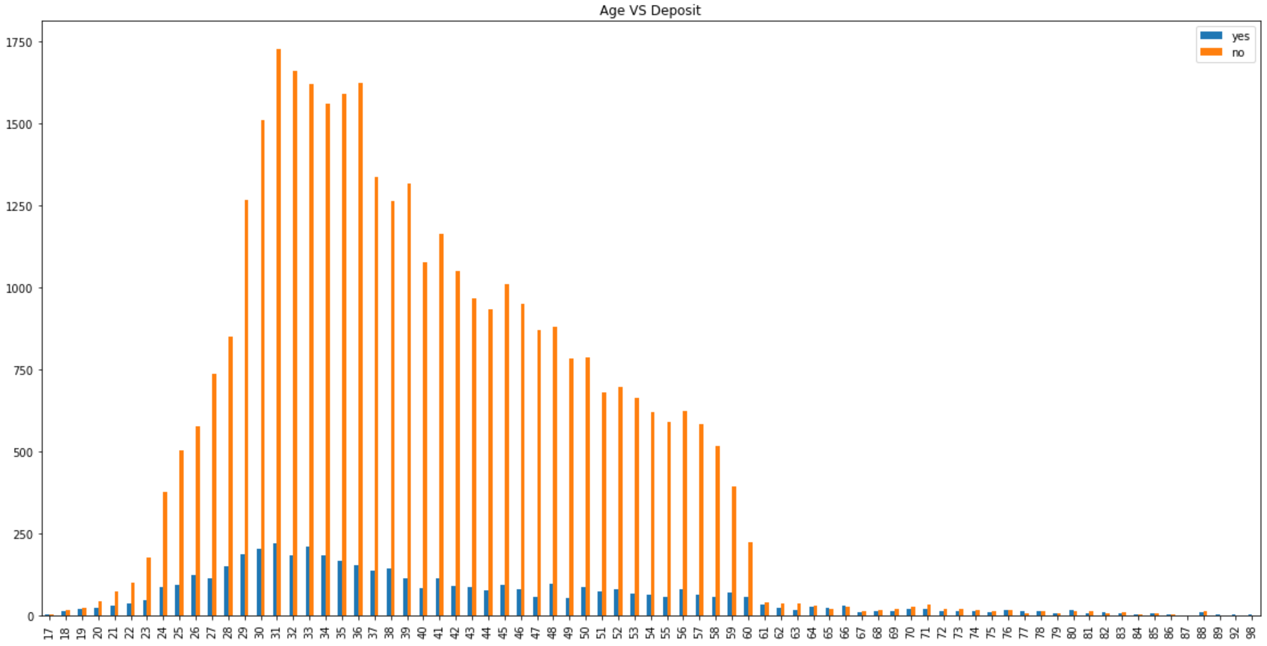


In [49]:

```
age_vs_deposit = pd.DataFrame()

age_vs_deposit['yes'] = new_bank_df[new_bank_df['y'] == 'yes']['age'].value_counts()
age_vs_deposit['no'] = new_bank_df[new_bank_df['y'] == 'no']['age'].value_counts()

age_vs_deposit.sort_index().plot.bar(title = "Age VS Deposit", figsize=(20,10))
```

Out[49]:

```
<AxesSubplot:title={'center':'Age VS Deposit'}>
```



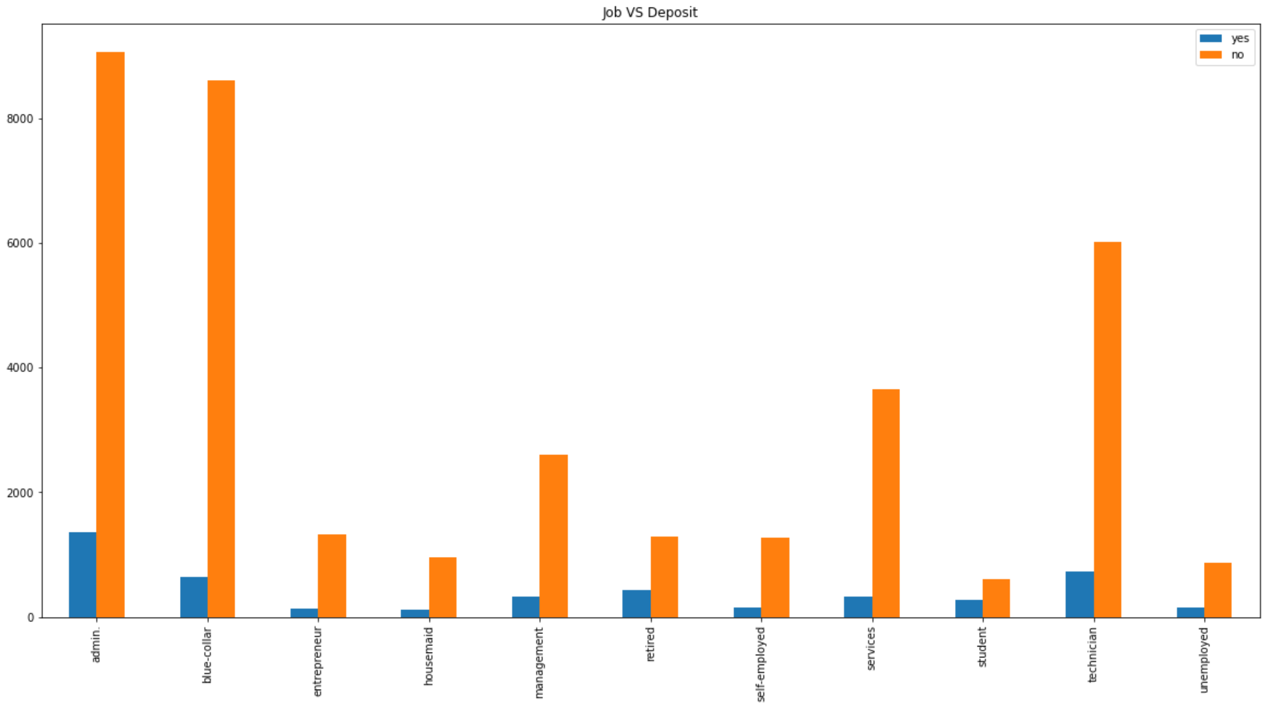
In [50]:

```
job_vs_deposit = pd.DataFrame()
```

```
job_vs_deposit['yes'] = new_bank_df[new_bank_df['y'] == 'yes']['job'].value_counts()
job_vs_deposit['no'] = new_bank_df[new_bank_df['y'] == 'no']['job'].value_counts()

job_vs_deposit.sort_index().plot.bar(title = "Job VS Deposit", figsize=(20,10))
```

Out[50]:

```
<AxesSubplot:title={'center':'Job VS Deposit'}>
```

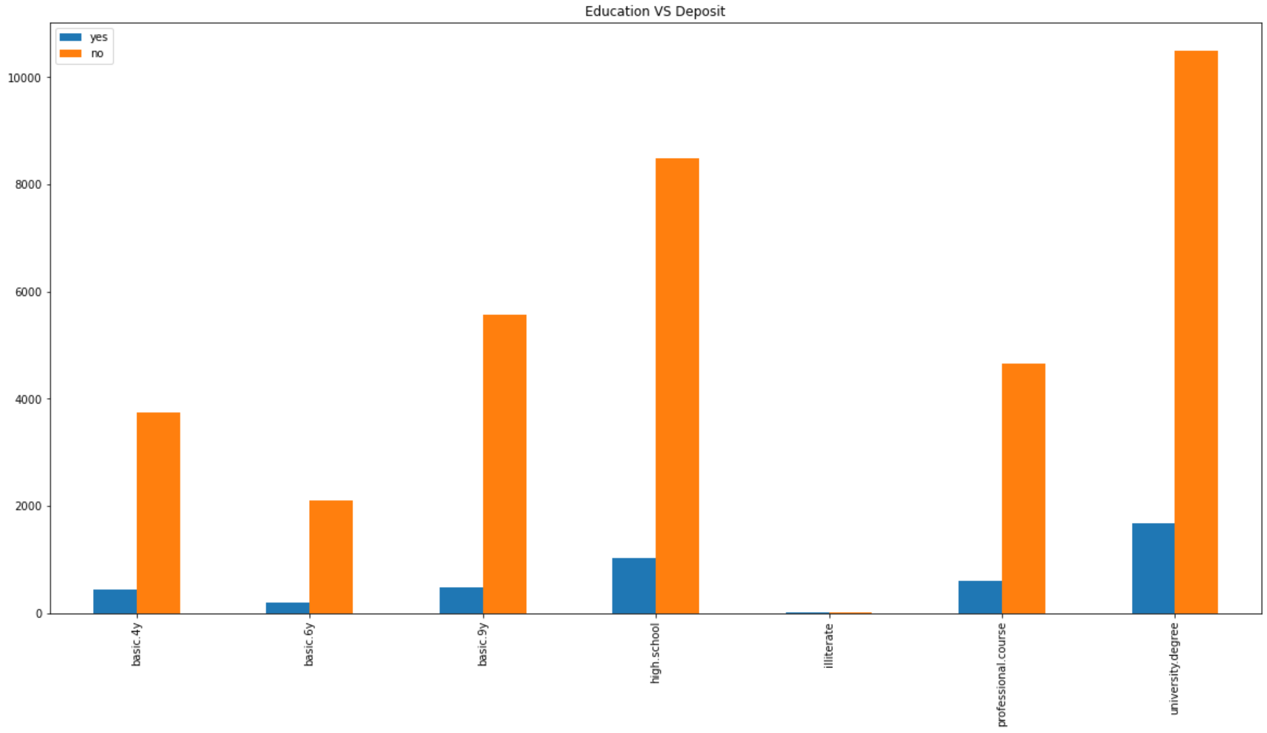


In [51]:

```
education_vs_deposit = pd.DataFrame()

education_vs_deposit['yes'] = new_bank_df[new_bank_df['y'] == 'yes']['education'].value
education_vs_deposit['no'] = new_bank_df[new_bank_df['y'] == 'no']['education'].value_c

education_vs_deposit.sort_index().plot.bar(title = "Education VS Deposit", figsize=(20,
```

Out[51]:

```
<AxesSubplot:title={'center':'Education VS Deposit'}>
```

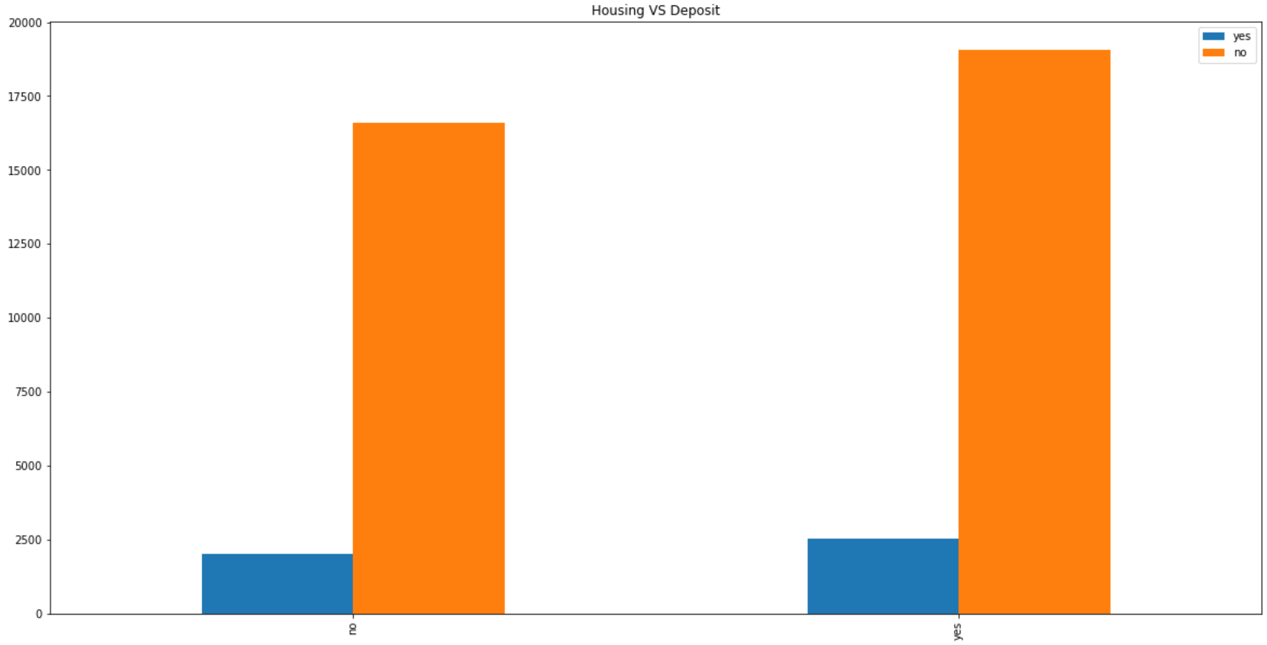


In [52]:

```
housing_vs_deposit = pd.DataFrame()

housing_vs_deposit['yes'] = new_bank_df[new_bank_df['y'] == 'yes']['housing'].value_cou
housing_vs_deposit['no'] = new_bank_df[new_bank_df['y'] == 'no']['housing'].value_count

housing_vs_deposit.sort_index().plot.bar(title = "Housing VS Deposit", figsize=(20,10))
```

Out[52]:

```
<AxesSubplot:title={'center':'Housing VS Deposit'}>
```



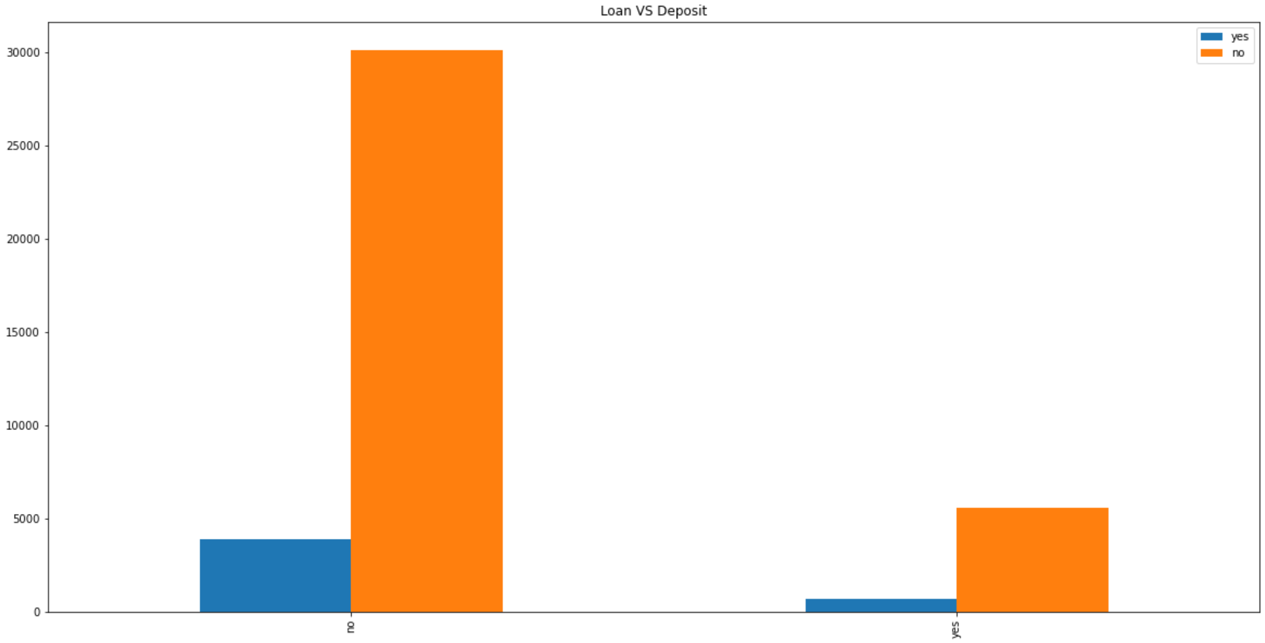
In [53]:

```
loan_vs_deposit = pd.DataFrame()

loan_vs_deposit['yes'] = new_bank_df[new_bank_df['y'] == 'yes']['loan'].value_counts()
loan_vs_deposit['no'] = new_bank_df[new_bank_df['y'] == 'no']['loan'].value_counts()
```

```
loan_vs_deposit.sort_index().plot.bar(title = "Loan VS Deposit", figsize=(20,10))
```

Out[53]:

```
<AxesSubplot:title={'center':'Loan VS Deposit'}>
```

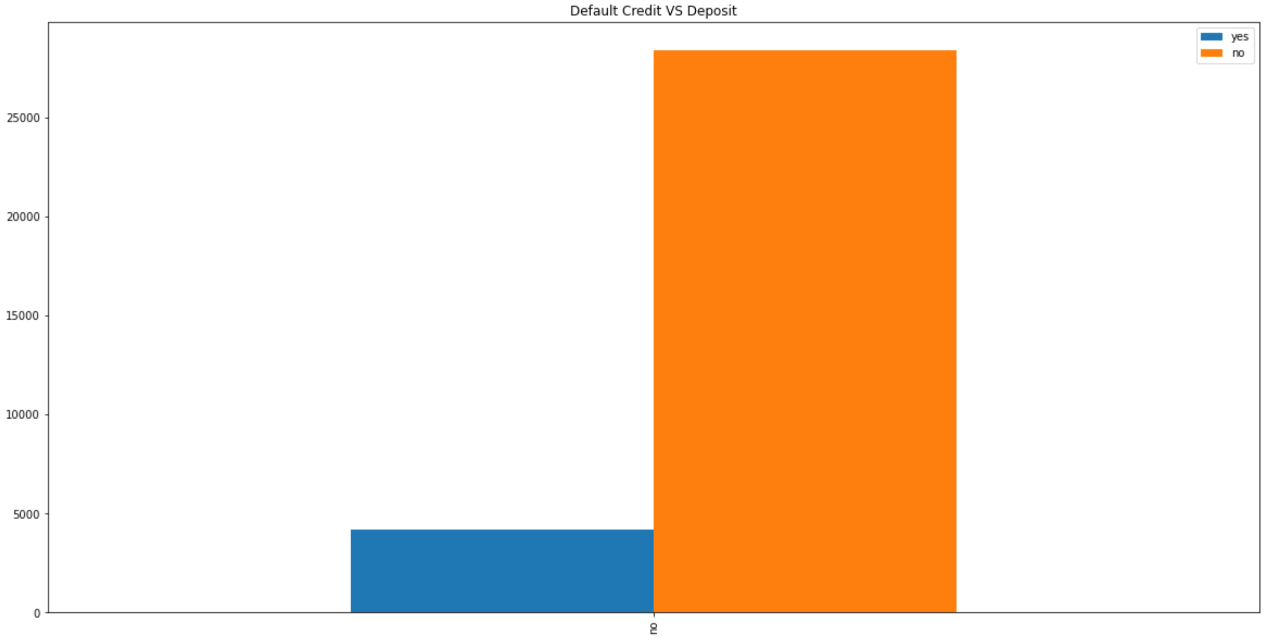


In [54]:

```
default_vs_deposit = pd.DataFrame()

default_vs_deposit['yes'] = new_bank_df[new_bank_df['y'] == 'yes']['default'].value_cou
default_vs_deposit['no'] = new_bank_df[new_bank_df['y'] == 'no']['default'].value_count

default_vs_deposit.sort_index().plot.bar(title = "Default Credit VS Deposit", figsize=(
```

Out[54]:

```
<AxesSubplot:title={'center':'Default Credit VS Deposit'}>
```



In [55]:

```
old_age = new_bank_df[new_bank_df['age'] > 61]
```

In [56]:

```
old_age
```

Out[56]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_we
30104	62	self-employed	married	university.degree	no	no	no	cellular	apr	m
39268	62	technician	married	NaN	no	yes	no	cellular	mar	m
39274	62	technician	married	NaN	no	no	no	cellular	mar	m
37678	62	unemployed	married	high.school	no	yes	no	cellular	aug	w
41178	62	retired	married	university.degree	no	no	no	cellular	nov	t
...	...	...	...	...	...	...	...	...	...	
40450	92	retired	married	NaN	no	no	yes	cellular	aug	t
38921	94	retired	married	basic.9y	no	no	no	cellular	nov	w
27826	95	retired	divorced	basic.6y	no	no	no	cellular	mar	t
38455	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	
38452	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	

837 rows × 21 columns

In [57]:

```
old_age[old_age['y'] == 'yes']
```

Out[57]:

	age	job	marital	education	default	housing	loan	contact	month	day_
39268	62	technician	married	NaN	no	yes	no	cellular	mar	
41178	62	retired	married	university.degree	no	no	no	cellular	nov	
37594	62	management	divorced	high.school	no	no	no	cellular	aug	
37803	62	retired	married	professional.course	no	yes	no	cellular	aug	
27958	62	admin.	married	high.school	no	no	no	telephone	mar	
...	...	...	...	...	...	...	...	...	...	
39734	92	retired	divorced	NaN	NaN	no	no	cellular	may	
40469	92	retired	married	NaN	no	no	yes	cellular	aug	
40450	92	retired	married	NaN	no	no	yes	cellular	aug	
38455	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	
38452	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	

382 rows × 21 columns

In [58]:

```
old_age
```

Out[58]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_we
30104	62	self-employed	married	university.degree	no	no	no	cellular	apr	m
39268	62	technician	married	NaN	no	yes	no	cellular	mar	m
39274	62	technician	married	NaN	no	no	no	cellular	mar	m
37678	62	unemployed	married	high.school	no	yes	no	cellular	aug	w
41178	62	retired	married	university.degree	no	no	no	cellular	nov	t
...	...	...	...	...	...	...	...	...	...	
40450	92	retired	married	NaN	no	no	yes	cellular	aug	t
38921	94	retired	married	basic.9y	no	no	no	cellular	nov	w
27826	95	retired	divorced	basic.6y	no	no	no	cellular	mar	t
38455	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	
38452	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	

837 rows × 21 columns

In [59]:

```
new_bank_df
```

Out[59]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	ca
38274	17	student	single	NaN	no	no	yes	cellular	oct	tue	...	
37579	17	student	single	basic.9y	no	NaN	NaN	cellular	aug	fri	...	
37539	17	student	single	basic.9y	no	yes	no	cellular	aug	fri	...	
37140	17	student	single	NaN	no	yes	no	cellular	aug	wed	...	
37558	17	student	single	basic.9y	no	yes	no	cellular	aug	fri	...	
...	...	...	...	...	...	...	...	...	...	...	...	
40450	92	retired	married	NaN	no	no	yes	cellular	aug	tue	...	
38921	94	retired	married	basic.9y	no	no	no	cellular	nov	wed	...	
27826	95	retired	divorced	basic.6y	no	no	no	cellular	mar	thu	...	
38455	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	fri	...	
38452	98	retired	married	basic.4y	NaN	yes	no	cellular	oct	fri	...	

41188 rows × 21 columns

In [60]:

```
new_bank_df.rename(columns = {'y':'deposited?'}, inplace = True)
```

In [61]:

```
new_bank_df['default'] = new_bank_df['default'].replace({'yes': 1, 'no': 0})
```

In [62]:

```
new_bank_df['deposited?'] = new_bank_df['deposited?'].replace({'yes': 1, 'no': 0})
```

In [63]:

```
new_bank_df['housing'] = new_bank_df['housing'].replace({'yes': 1, 'no': 0})
```

In [64]:

```
new_bank_df['loan'] = new_bank_df['loan'].replace({'yes': 1, 'no': 0})
```

In [65]:

```
new_bank_df
```

Out[65]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	ca
38274	17	student	single	NaN	0.0	0.0	1.0	cellular	oct	tue	...	
37579	17	student	single	basic.9y	0.0	NaN	NaN	cellular	aug	fri	...	
37539	17	student	single	basic.9y	0.0	1.0	0.0	cellular	aug	fri	...	
37140	17	student	single	NaN	0.0	1.0	0.0	cellular	aug	wed	...	
37558	17	student	single	basic.9y	0.0	1.0	0.0	cellular	aug	fri	...	
...	...	...	...	...	...	...	...	...	...	...	...	
40450	92	retired	married	NaN	0.0	0.0	1.0	cellular	aug	tue	...	
38921	94	retired	married	basic.9y	0.0	0.0	0.0	cellular	nov	wed	...	
27826	95	retired	divorced	basic.6y	0.0	0.0	0.0	cellular	mar	thu	...	
38455	98	retired	married	basic.4y	NaN	1.0	0.0	cellular	oct	fri	...	
38452	98	retired	married	basic.4y	NaN	1.0	0.0	cellular	oct	fri	...	

41188 rows × 21 columns

In [66]:

```
new_bank_df.dtypes
```

Out[66]:

```
age                 int64
job                object
marital            object
education          object
default           float64
housing           float64
loan              float64
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
deposited?          int64
dtype: object
```

In [67]:

```
new_bank_df['default'] = new_bank_df['default'].astype('Int64')
```

In [68]:

```
new_bank_df['housing'] = new_bank_df['housing'].astype('Int64')
```

In [69]:

```
new_bank_df['loan'] = new_bank_df['loan'].astype('Int64')
```

In [70]:

```
new_bank_df
```

Out[70]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...
38274	17	student	single	NaN	0	0	1	cellular	oct	tue	...
37579	17	student	single	basic.9y	0	<NA>	<NA>	cellular	aug	fri	...
37539	17	student	single	basic.9y	0	1	0	cellular	aug	fri	...
37140	17	student	single	NaN	0	1	0	cellular	aug	wed	...
37558	17	student	single	basic.9y	0	1	0	cellular	aug	fri	...
...	...	...	...	...	...	...	...	...	...	...	...
40450	92	retired	married	NaN	0	0	1	cellular	aug	tue	...
38921	94	retired	married	basic.9y	0	0	0	cellular	nov	wed	...
27826	95	retired	divorced	basic.6y	0	0	0	cellular	mar	thu	...
38455	98	retired	married	basic.4y	<NA>	1	0	cellular	oct	fri	...
38452	98	retired	married	basic.4y	<NA>	1	0	cellular	oct	fri	...

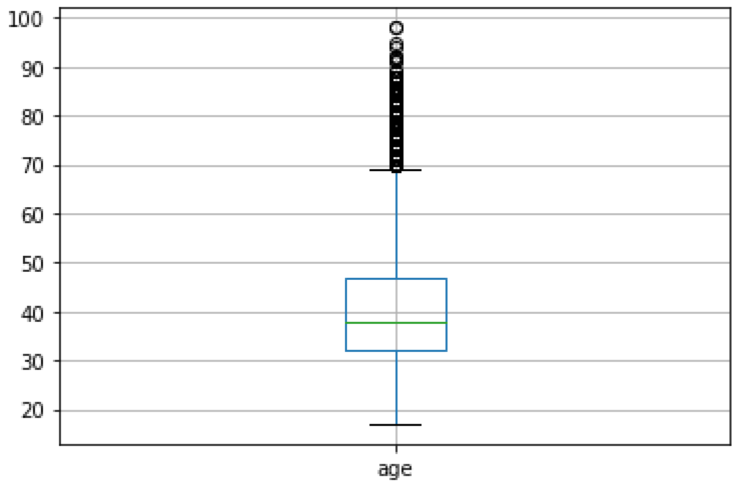
41188 rows × 21 columns

In [71]:

```
new_bank_df.boxplot(column='age')
```

Out[71]:

```
<AxesSubplot:>
```



In [72]:
```
new_bank_df.drop('duration', axis='columns', inplace = True)
```

In [73]:
```
new_bank_df
```

Out[73]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	cam
38274	17	student	single	NaN	0	0	1	cellular	oct	tue	
37579	17	student	single	basic.9y	0	<NA>	<NA>	cellular	aug	fri	
37539	17	student	single	basic.9y	0	1	0	cellular	aug	fri	
37140	17	student	single	NaN	0	1	0	cellular	aug	wed	
37558	17	student	single	basic.9y	0	1	0	cellular	aug	fri	
...	...	...	...	...	...	...	...	...	...	...	
40450	92	retired	married	NaN	0	0	1	cellular	aug	tue	
38921	94	retired	married	basic.9y	0	0	0	cellular	nov	wed	
27826	95	retired	divorced	basic.6y	0	0	0	cellular	mar	thu	
38455	98	retired	married	basic.4y	<NA>	1	0	cellular	oct	fri	
38452	98	retired	married	basic.4y	<NA>	1	0	cellular	oct	fri	

41188 rows × 20 columns

In [87]:
```
new_bank_df.isnull().sum()
```

Out[87]:
```
age               0
job             330
marital          80
education      1731
default        8597
housing         990
loan            990
contact           0
month             0
```

```
day_of_week         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
deposited?          0
dtype: int64
```

In [88]:

```
new_bank_df.shape
```

Out[88]:

```
(41188, 20)
```

In [92]:

```
new_bank_df['default'].fillna(0,inplace = True)
new_bank_df['housing'].fillna(0,inplace = True)
new_bank_df['loan'].fillna(0,inplace = True)
```

In [93]:

```
new_bank_df.isnull().sum()
```

Out[93]:

```
age                 0
job               330
marital            80
education        1731
default             0
housing             0
loan                0
contact             0
month               0
day_of_week         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
deposited?          0
dtype: int64
```

In [95]:

```
new_bank_df.dropna(inplace=True)
```

In [96]:

```
new_bank_df.isnull().sum()
```

Out[96]:

```
age              0
job              0
marital          0
education        0
default          0
```

```
housing           0
loan              0
contact           0
month             0
day_of_week       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
deposited?        0
dtype: int64
```

Models Building

In [97]:

```
X=new_bank_df.drop(['deposited?'],axis=1)
y=new_bank_df['deposited?']
```

In [98]:

```
X = pd.get_dummies(X)
X.columns=[x.lower() for x in X.columns]
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=42,test_size=0.3, strat
```

In [99]:

```
df_train = X_train.copy()
df_train['deposited?'] = y_train
df_train.head()
```

Out[99]:

	age	default	housing	loan	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf
731	48	0	0	0	3	999	0	1.1	93.994	-
15805	37	0	1	0	1	999	0	1.4	93.918	-
23451	41	0	0	0	5	999	0	1.4	93.444	-
33990	36	0	0	0	6	999	0	-1.8	92.893	-
19390	45	0	1	1	3	999	0	1.4	93.444	-

5 rows × 54 columns

In [100…

```
classes=df_train['deposited?'].value_counts()
normal_share=round(classes[0]/df_train['deposited?'].count()*100,2)
fraud_share=round(classes[1]/df_train['deposited?'].count()*100, 2)
print("Non-deposited? : {} %".format(normal_share))
print("deposited? : {} %".format(fraud_share))
```

```
Non-deposited? : 88.87 %
deposited? : 11.13 %
```

In [101…

```
fig = px.histogram(df_train, x="deposited?", color="deposited?", title="deposited class
fig.show()
```

In [102…

```
X_train=df_train.drop(['deposited?'],axis=1)
y_train=df_train['deposited?']
```

In [103…

```
fig = px.histogram(df_train, x="deposited?", color="deposited?", title="deposited class
fig.show()
```

In [106…

```
def evaluation_metrics(y_test, y_pre, target_names):
    #scores
    print("Accuracy :",accuracy_score(y_test,y_pre))
    print("Precision :",precision_score(y_test,y_pre))
    print("Recall :",recall_score(y_test,y_pre))
    print("F1 Score :",f1_score(y_test,y_pre))

    print(classification_report(y_test, y_pre, target_names=target_names))

    #AUC
    fpr, tpr, _ = roc_curve(y_test,  y_pre)
    auc = roc_auc_score(y_test, y_pre)
    print("AUC :", auc)

    #ROC
    plt.plot(fpr,tpr,label="uc={:.3f})".format(auc))
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.title('ROC curve')
    plt.legend(loc=4)
    plt.show()

    #CM matrix
    matrix = confusion_matrix(y_test, y_pre)
    cm = pd.DataFrame(matrix, index=target_names, columns=target_names)

    sns.heatmap(cm, annot=True, cbar=None, cmap="Blues", fmt = 'g')
    plt.title("Confusion Matrix"), plt.tight_layout()
    plt.ylabel("True Class"), plt.xlabel("Predicted Class")
    plt.show()
```

In [107…

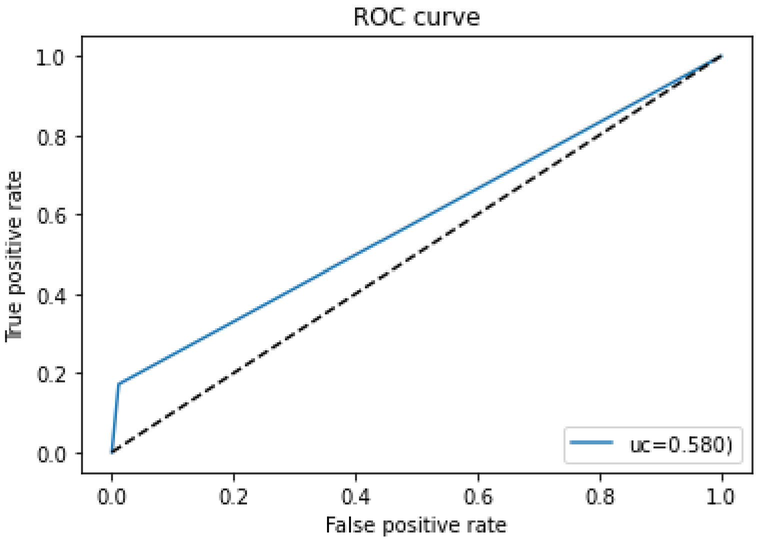
```
target_names=['No Deposited', 'Deposited']
```

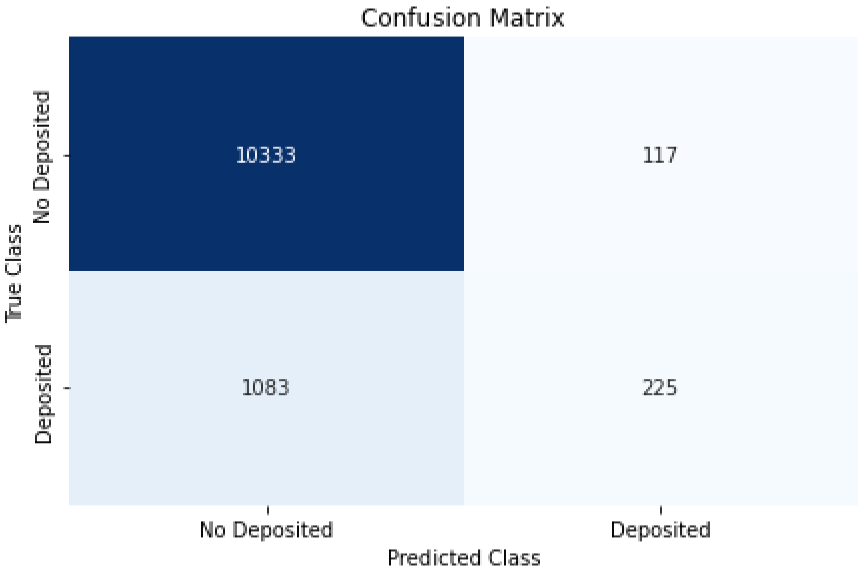
Logistic Regression

In [108…

```
def log(X_train,X_test,y_train,y_test):
    model=LogisticRegression()
    model.fit(X_train,y_train)
    y_pre=model.predict(X_test)
    evaluation_metrics(y_test, y_pre, target_names)

log(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.8979418268412995
Precision : 0.6578947368421053
Recall : 0.1720183486238532
F1 Score : 0.2727272727272727
              precision    recall  f1-score   support

No Deposited       0.91      0.99      0.95     10450
   Deposited       0.66      0.17      0.27      1308

    accuracy                           0.90     11758
   macro avg       0.78      0.58      0.61     11758
weighted avg       0.88      0.90      0.87     11758

AUC : 0.5804110881875246
```





RidgeClassifier

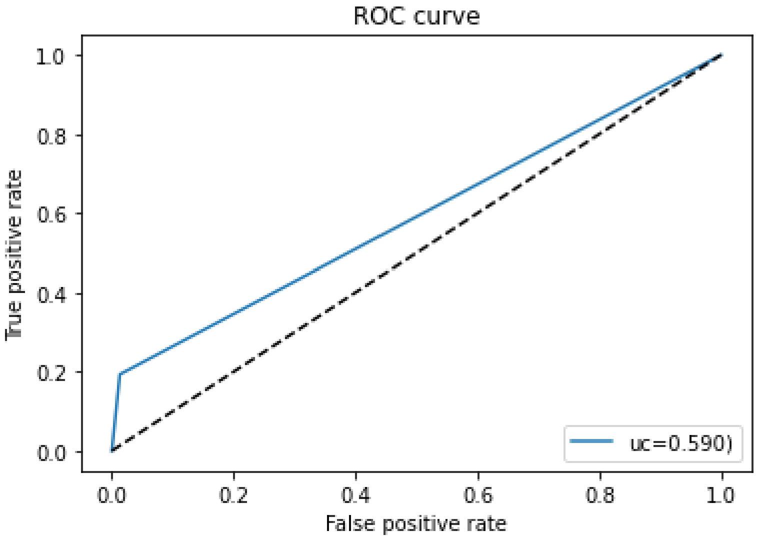
In [109…

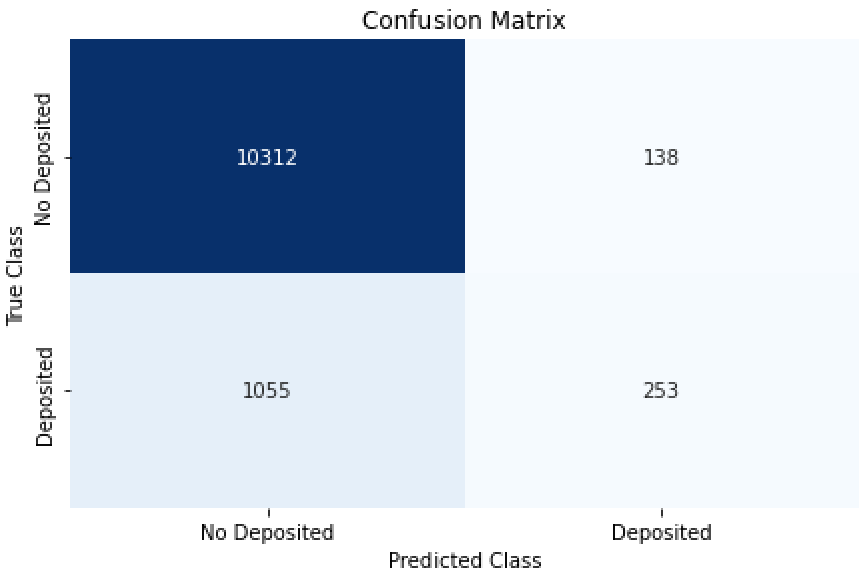
```
def Ridge(X_train,X_test,y_train,y_test):
    #train the model
    model = RidgeClassifier(random_state=2)
    model.fit(X_train, y_train)
    #predictions
    y_pre = model.predict(X_test)
    evaluation_metrics(y_test, y_pre, target_names)
```

In [110…

```
Ridge(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.8985371661847253
Precision : 0.6470588235294118
Recall : 0.19342507645259938
F1 Score : 0.2978222483814008
              precision    recall  f1-score   support

No Deposited       0.91      0.99      0.95     10450
   Deposited       0.65      0.19      0.30      1308

    accuracy                           0.90     11758
   macro avg       0.78      0.59      0.62     11758
weighted avg       0.88      0.90      0.87     11758

AUC : 0.5901096674129026
```





RandomForestClassifier

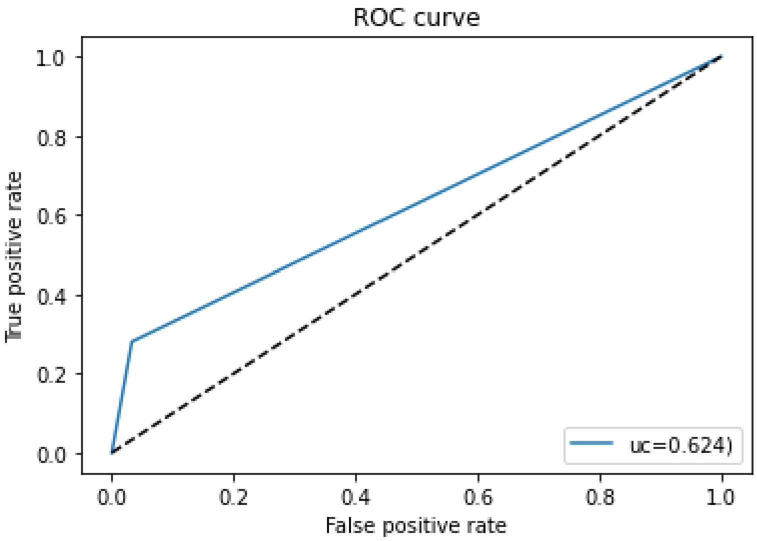
In [111…

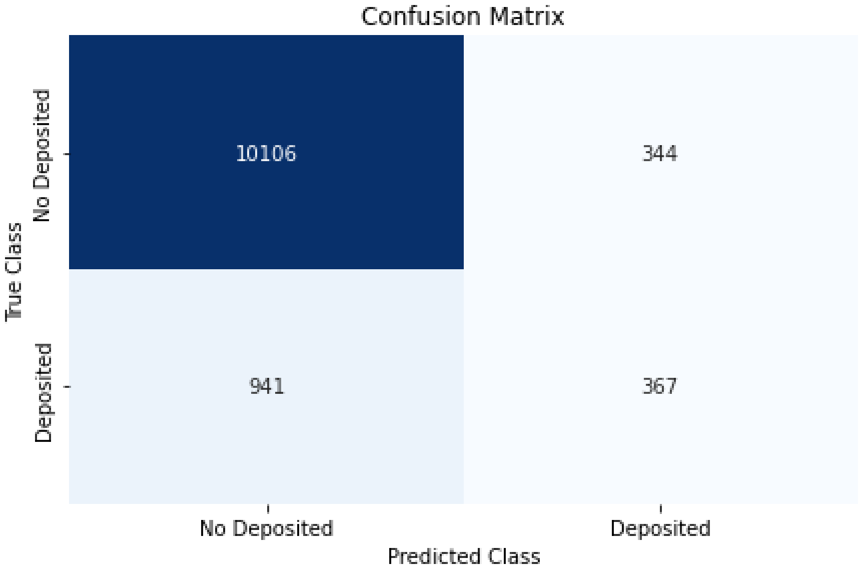
```
def RF(X_train,X_test,y_train,y_test):
  #train the model
  model = RandomForestClassifier(random_state=2)
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

In [112…

```
RF(X_train,X_test,y_train,y_test)
```

```
Accuracy : 0.8907127062425583
Precision : 0.5161744022503516
Recall : 0.2805810397553517
F1 Score : 0.36354631005448246
              precision    recall  f1-score   support

No Deposited       0.91      0.97      0.94     10450
   Deposited       0.52      0.28      0.36      1308

    accuracy                           0.89     11758
   macro avg       0.72      0.62      0.65     11758
weighted avg       0.87      0.89      0.88     11758

AUC : 0.6238311897341351
```





Conclusion

- Approximately all the classifiers have same result, but Random Forest was the best one.
- The model has around 89% Accuracy.
- Random Forest has 87% Precision, 89% Recall, & 88% F1 Score.
- We can also see the results for each classifier as well.

Model Deployment

In [113…

```
from sklearn.ensemble import StackingClassifier
```

In [114…

```
def Stacking(X_train,X_test,y_train,y_test):
  #train the model
  estimators = [('rf', RandomForestClassifier(n_estimators=10, random_state=42)), ('svr
  model = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression(
  model.fit(X_train, y_train)
  #predictions
  y_pre = model.predict(X_test)
  evaluation_metrics(y_test, y_pre, target_names)
```

In [ ]:

```
###Stacking classifier
import pickle
estimators = [('rf', RandomForestClassifier(n_estimators=10, random_state=42)), ('svr',
final_model = StackingClassifier(estimators=estimators, final_estimator=LogisticRegress
final_model.fit(X, y)
filename = 'final_model.sav'
pickle.dump(final_model, open(filename, 'wb'))
```