In [84]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Importing Data

In [85]:
```
bank_data = pd.read_csv('bank-full (1).csv')
bank_data
```

Out[85]:

nce	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	Target
143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
1	no	no	unknown	5	may	198	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...
825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
715	no	no	cellular	17	nov	1127	5	184	3	success	yes
668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
971	no	no	cellular	17	nov	361	2	188	11	other	no

In [86]:
```
bank_data.head()
```

Out[86]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	d
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	

In [87]:
```
bank_data.shape
```

Out[87]:
```
(45211, 17)
```

In [88]:
```
bank_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  Target     45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

In [89]:
```
bank_data.isna().sum()
```

Out[89]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
Target       0
dtype: int64
```

In [90]:

```
bank_data.describe(include='all')
```

Out[90]:

	age	job	marital	education	default	balance	housing	loan	contact
count	45211.000000	45211	45211	45211	45211	45211.000000	45211	45211	45211
unique	NaN	12	3	4	2	NaN	2	2	3
top	NaN	blue-collar	married	secondary	no	NaN	yes	no	cellular
freq	NaN	9732	27214	23202	44396	NaN	25130	37967	29285
mean	40.936210	NaN	NaN	NaN	NaN	1362.272058	NaN	NaN	NaN
std	10.618762	NaN	NaN	NaN	NaN	3044.765829	NaN	NaN	NaN
min	18.000000	NaN	NaN	NaN	NaN	-8019.000000	NaN	NaN	NaN
25%	33.000000	NaN	NaN	NaN	NaN	72.000000	NaN	NaN	NaN
50%	39.000000	NaN	NaN	NaN	NaN	448.000000	NaN	NaN	NaN
75%	48.000000	NaN	NaN	NaN	NaN	1428.000000	NaN	NaN	NaN
max	95.000000	NaN	NaN	NaN	NaN	102127.000000	NaN	NaN	NaN

In [91]:

```
bank_data.dtypes
```

Out[91]:

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

In [92]:

```
bank_data.corr()
```

Out[92]:

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.097783	-0.009120	-0.004648	0.004760	-0.023758	0.001288
balance	0.097783	1.000000	0.004503	0.021560	-0.014578	0.003435	0.016674
day	-0.009120	0.004503	1.000000	-0.030206	0.162490	-0.093044	-0.051710
duration	-0.004648	0.021560	-0.030206	1.000000	-0.084570	-0.001565	0.001203
campaign	0.004760	-0.014578	0.162490	-0.084570	1.000000	-0.088628	-0.032855
pdays	-0.023758	0.003435	-0.093044	-0.001565	-0.088628	1.000000	0.454820
previous	0.001288	0.016674	-0.051710	0.001203	-0.032855	0.454820	1.000000

In [93]:

```
bank_data['job'].unique()
```

Out[93]:

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

In [94]:

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

Out[94]:

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

In [95]:

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

Out[95]:

```
array(['tertiary', 'secondary', 'unknown', 'primary'], dtype=object)
```

In [96]:

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

Out[96]:

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

In [97]:

```
bank_data['month'].unique()
```

Out[97]:

```
array(['may', 'jun', 'jul', 'aug', 'oct', 'nov', 'dec', 'jan', 'feb',
       'mar', 'apr', 'sep'], dtype=object)
```

In [98]:

```
df_user = pd.DataFrame(np.arange(0,len(bank_data)), columns=['user'])
bank_data =pd.concat([df_user,bank_data],axis=1)
```

In [99]:

```
bank_data.columns.values
```

Out[99]:

```
array(['user', 'age', 'job', 'marital', 'education', 'default', 'balance',
       'housing', 'loan', 'contact', 'day', 'month', 'duration',
       'campaign', 'pdays', 'previous', 'poutcome', 'Target'],
      dtype=object)
```

In [100]:

```
bank_data.groupby('Target').mean()
```

Out[100]:

	user	age	balance	day	duration	campaign	pdays	previou
Target								
no	21197.503081	40.838986	1303.714969	15.892290	221.182806	2.846350	36.421372	0.50215
yes	33228.953867	41.670070	1804.267915	15.158253	537.294574	2.141047	68.702968	1.17035

In [101]:

```
bank_data['Target'].value_counts()
```

Out[101]:

```
no      39922
yes      5289
Name: Target, dtype: int64
```

In [102]:

```
 bank_data.isna().sum()
```

Out[102]:

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

In [103]:

```
x=bank_data.drop(['Target','user','job','marital','education','contact',
                'housing','loan','day','month','poutcome'],axis=1)
y=bank_data['Target']
```

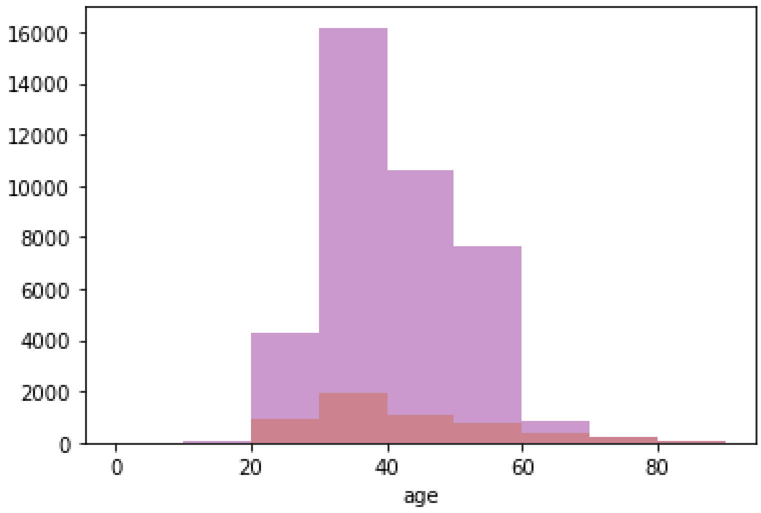
In [104]:

```
x=pd.get_dummies(x)
y=pd.get_dummies(y)
```

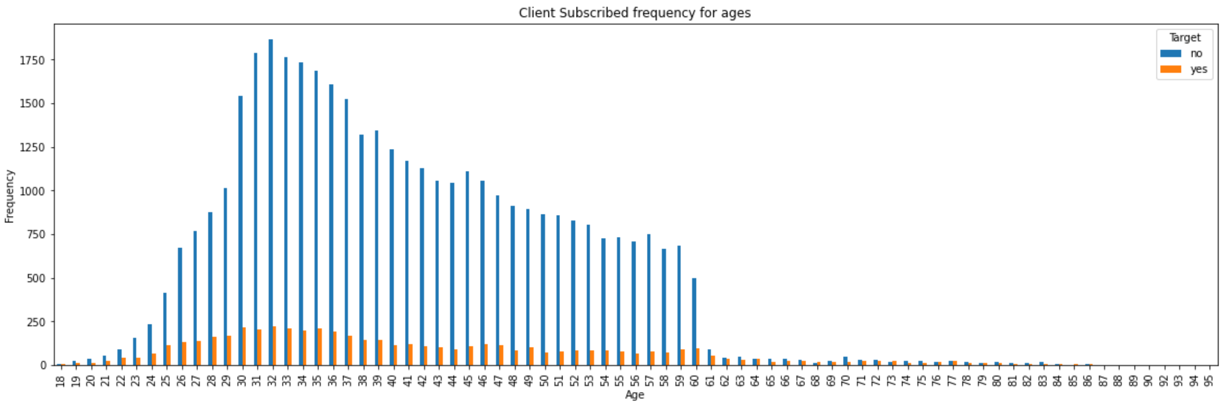
In [105]:
```
x.columns
x=x.drop(['default_no'], axis=1)
x=x.rename(columns={'default_yes':'default'})
y.columns
y=y.drop(['yes'],axis=1)
y=y.rename(columns={'no':'yes'})
```

Visualizing data

In [106]:
```
bins=range(0,100,10)
ax=sns.distplot(bank_data.age[bank_data.Target=='yes'],color='orange',kde=False,b
sns.distplot(bank_data.age[bank_data.Target=='no'],ax=ax,color='purple',kde=False
plt.legend
plt.show()
```

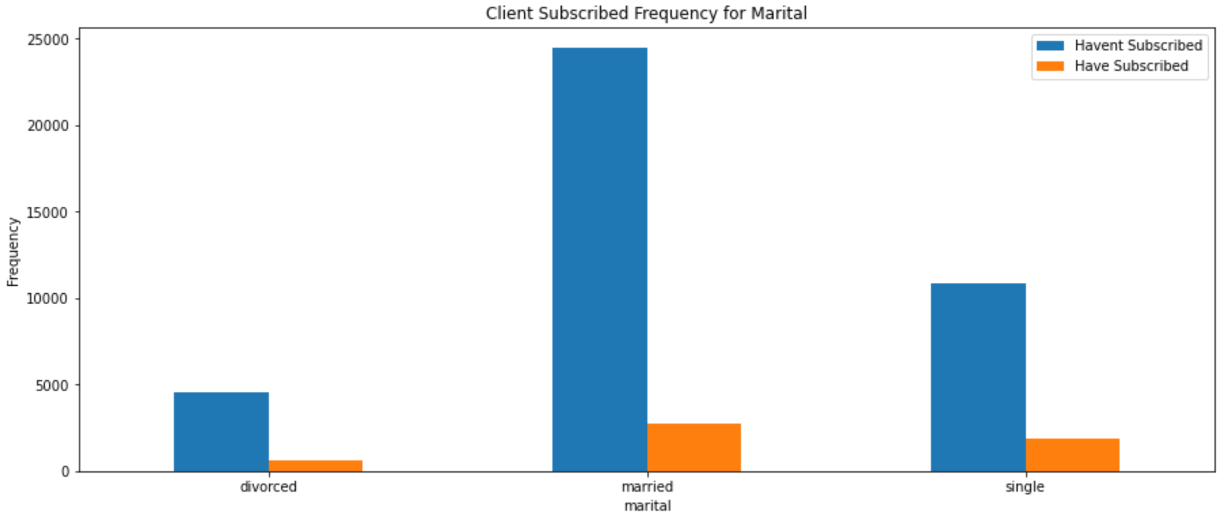


In [107]:
```
pd.crosstab(bank_data.age,bank_data.Target).plot(kind='bar',figsize=(20,6))
plt.title('Client Subscribed frequency for ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



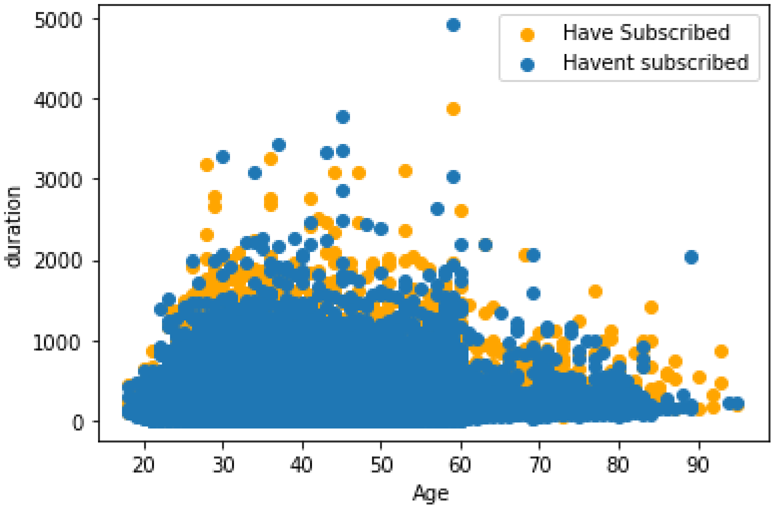
In [108]:

```
pd.crosstab(bank_data.marital,bank_data.Target).plot(kind='bar',figsize=(15,6))
plt.title('Client Subscribed Frequency for Marital')
plt.xlabel('marital')
plt.xticks(rotation=0)
plt.legend(['Havent Subscribed', 'Have Subscribed'])
plt.ylabel('Frequency')
plt.show()
```



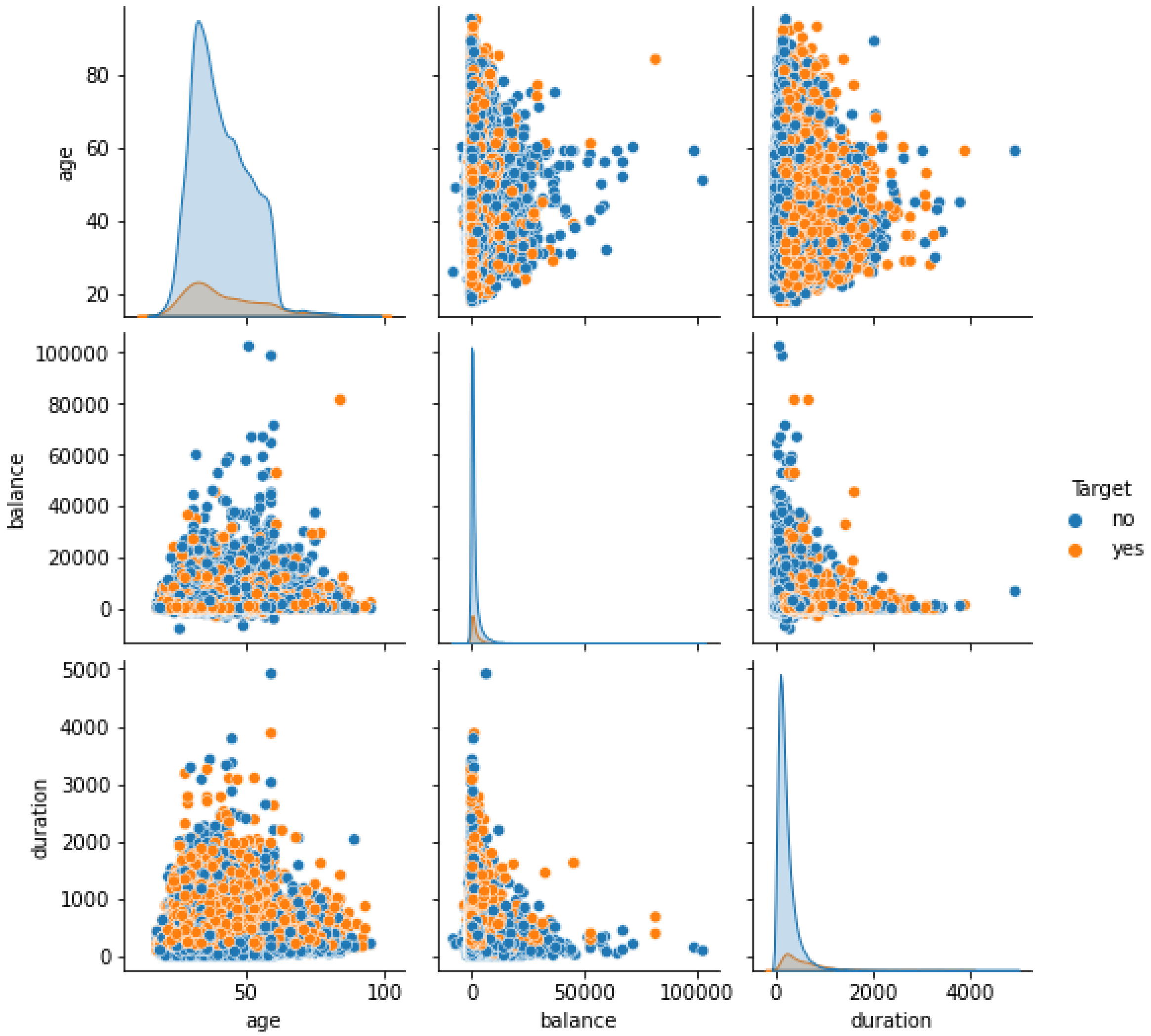
In [109]:

```
plt.scatter(x=bank_data.age[bank_data.Target=='yes'],y=bank_data.duration[(bank_c
plt.scatter(x=bank_data.age[bank_data.Target=='no'],y=bank_data.duration[(bank_da
plt.legend(['Have Subscribed','Havent subscribed'])
plt.xlabel('Age')
plt.ylabel('duration')
plt.show()
```



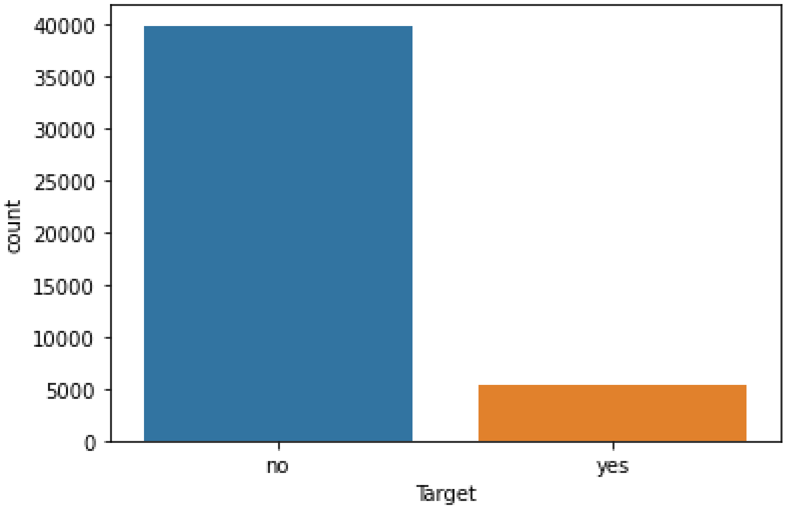
```
In [110]: sns.pairplot(data=bank_data, hue='Target', vars= ['age', 'balance', 'duration'])
```

```
Out[110]: <seaborn.axisgrid.PairGrid at 0x20eaab14a00>
```



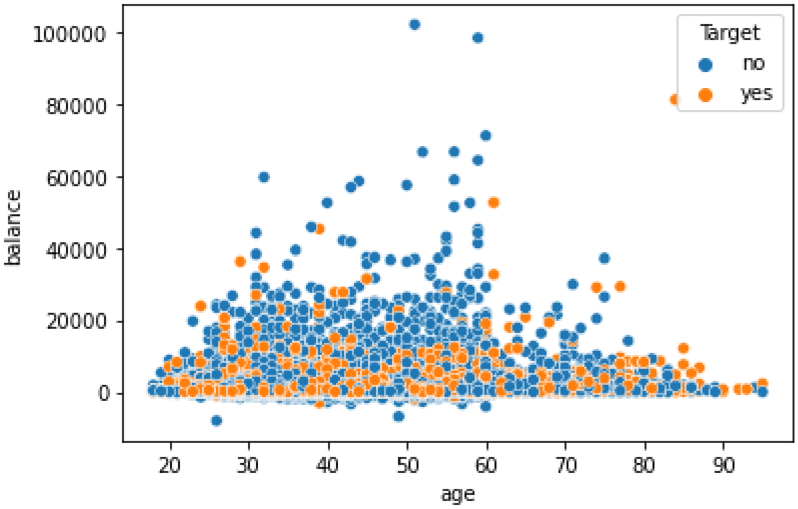
```
In [215]: sns.countplot(x='Target',data=bank_data,label='count')

Out[215]: <AxesSubplot:xlabel='Target', ylabel='count'>
```

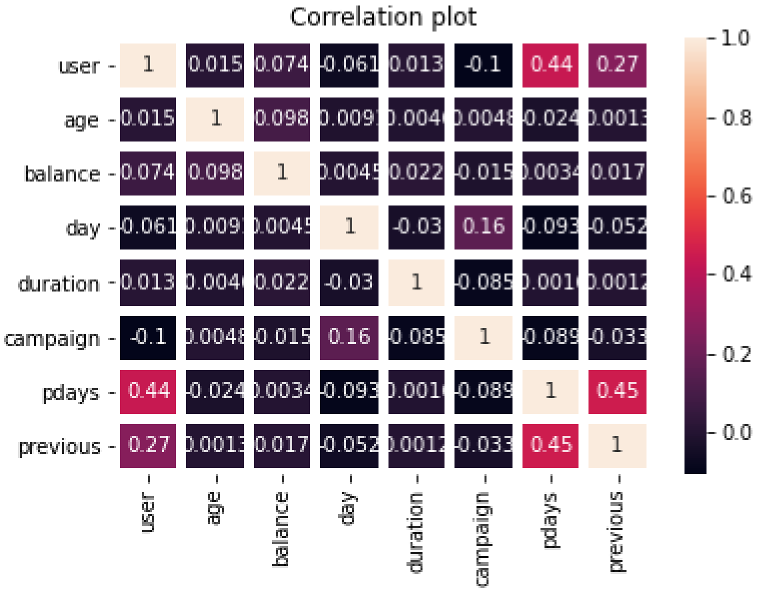


```
In [112]: sns.scatterplot(x='age',y='balance',hue='Target',data=bank_data)

Out[112]: <AxesSubplot:xlabel='age', ylabel='balance'>
```



In [114]:
```
sns.heatmap(data=bank_data.corr(),annot=True,linewidths=4)
plt.title('Correlation plot')
plt.show()
```



Model Building || Model Training

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=12)
```

In [30]:
```
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(36168, 7) (9043, 7) (36168, 1) (9043, 1)
```

Feature Scaling

In [115]:
```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
x_train=std.fit_transform(x_train)
x_test=std.fit_transform(x_test)
```

In [189]:
```
#imbalnced data
y_train.value_counts(normalize=True)
```

Out[189]:
```
yes
1      0.882797
0      0.117203
dtype: float64
```

In [233]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(class_weight={0:3,1:1})
model.fit(x_train,y_train)
```

Out[233]:
```
LogisticRegression(class_weight={0: 3, 1: 1})
```

Model testing

In [234]:
```
y_pred =model .predict(x_test)
y_pred
```

Out[234]:
```
array([1, 1, 0, ..., 1, 1, 1], dtype=uint8)
```

Model Evaluation

In [235]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score,recall_score
```

In [236]:
```
confusion_matrix(y_test,y_pred)
```

Out[236]:
```
array([[ 465,  585],
       [ 543, 7450]], dtype=int64)
```

In [237]:
```
accuracy_score(y_test,y_pred)
```

Out[237]:
```
0.8752626340816101
```

In [238]:
```
f1_score(y_test,y_pred)
```

Out[238]:
```
0.9296231594709259
```

In [239]:
```
precision_score(y_test,y_pred)
```

Out[239]:
```
0.9271935283136279
```

In [240]:
```
recall_score(y_test,y_pred)
```

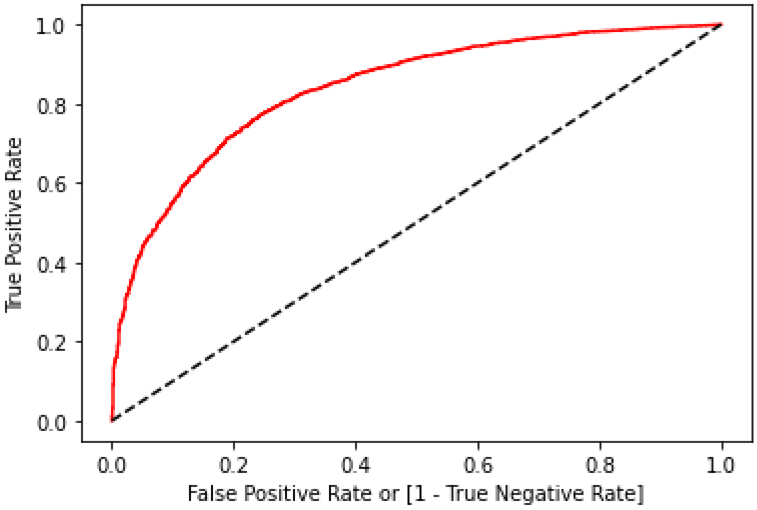
Out[240]:
```
0.9320655573626924
```

In [241]:

```
# Roc curve
fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba (x_test)[:,1])

auc = roc_auc_score(y_test, y_pred)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red', label='logit model ( area  = %0.2f)'%auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
plt.show()
```



In [ ]: