Churn Rate Analysis using Artificial Neural Network

Customer churn is defined as the loss of customers because they move out to competitors. It is an expensive problem in many industries since acquiring new customers costs five to six times more than retaining existing ones. Through our project we are predicting the number of customers who might leave the bank in furture.

In our project we have analysed 10,000 bank customers and predicted the Churn rate.

We have used Churn_modelling.csv from Kaggle.com.

Key features of the data -

We are using Churn_Modelling.csv from Kaggle. The data consists of 12 independent variables and one dependent variable. The 12 independent variable give us the information about customers and the dependent variable 'Exited' suggests us whether the customer is leaving the bank or not. The independent variables which will considered to predict the churn rate will be – CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Reading the data set
dataset = pd.read_csv('Churn_Modelling.csv')
```

In [2]:
```
dataset
```

Out[2]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
0	1	15634602	Hargrave	619	France	Female	42
1	2	15647311	Hill	608	Spain	Female	41
2	3	15619304	Onio	502	France	Female	42
3	4	15701354	Boni	699	France	Female	39
4	5	15737888	Mitchell	850	Spain	Female	43
5	6	15574012	Chu	645	Spain	Male	44
6	7	15592531	Bartlett	822	France	Male	50
7	8	15656148	Obinna	376	Germany	Female	29
8	9	15792365	He	501	France	Male	44
9	10	15592389	H?	684	France	Male	27
10	11	15767821	Bearce	528	France	Male	31
11	12	15737173	Andrews	497	Spain	Male	24
12	13	15632264	Kay	476	France	Female	34
13	14	15691483	Chin	549	France	Female	25
14	15	15600882	Scott	635	Spain	Female	35
15	16	15643966	Goforth	616	Germany	Male	45
16	17	15737452	Romeo	653	Germany	Male	58
17	18	15788218	Henderson	549	Spain	Female	24
18	19	15661507	Muldrow	587	Spain	Male	45
19	20	15568982	Hao	726	France	Female	24
20	21	15577657	McDonald	732	France	Male	41
21	22	15597945	Dellucci	636	Spain	Female	32
22	23	15699309	Gerasimov	510	Spain	Female	38
23	24	15725737	Mosman	669	France	Male	46
24	25	15625047	Yen	846	France	Female	38
25	26	15738191	Maclean	577	France	Male	25
26	27	15736816	Young	756	Germany	Male	36
27	28	15700772	Nebechi	571	France	Male	44
28	29	15728693	McWilliams	574	Germany	Female	43
29	30	15656300	Lucciano	411	France	Male	29
...	...	...	...	...	...	...	...
9970	9971	15587133	Thompson	518	France	Male	42

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
9971	9972	15721377	Chou	833	France	Female	34
9972	9973	15747927	Ch'in	758	France	Male	26
9973	9974	15806455	Miller	611	France	Male	27
9974	9975	15695474	Barker	583	France	Male	33
9975	9976	15666295	Smith	610	Germany	Male	50
9976	9977	15656062	Azikiwe	637	France	Female	33
9977	9978	15579969	Mancini	683	France	Female	32
9978	9979	15703563	P'eng	774	France	Male	40
9979	9980	15692664	Diribe	677	France	Female	58
9980	9981	15719276	T'ao	741	Spain	Male	35
9981	9982	15672754	Burbidge	498	Germany	Male	42
9982	9983	15768163	Griffin	655	Germany	Female	46
9983	9984	15656710	Cocci	613	France	Male	40
9984	9985	15696175	Echezonachukwu	602	Germany	Male	35
9985	9986	15586914	Nepean	659	France	Male	36
9986	9987	15581736	Bartlett	673	Germany	Male	47
9987	9988	15588839	Mancini	606	Spain	Male	30
9988	9989	15589329	Pirozzi	775	France	Male	30
9989	9990	15605622	McMillan	841	Spain	Male	28
9990	9991	15798964	Nkemakonam	714	Germany	Male	33
9991	9992	15769959	Ajuluchukwu	597	France	Female	53
9992	9993	15657105	Chukwualuka	726	Spain	Male	36
9993	9994	15569266	Rahman	644	France	Male	28
9994	9995	15719294	Wood	800	France	Female	29
9995	9996	15606229	Obijiaku	771	France	Male	39
9996	9997	15569892	Johnstone	516	France	Male	35
9997	9998	15584532	Liu	709	France	Female	36
9998	9999	15682355	Sabbatini	772	Germany	Male	42
9999	10000	15628319	Walker	792	France	Female	28

10000 rows × 14 columns

In [3]:
```
dataset.head()
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
0	1	15634602	Hargrave	619	France	Female	42	2	0.(
1	2	15647311	Hill	608	Spain	Female	41	1	83
2	3	15619304	Onio	502	France	Female	42	8	15
3	4	15701354	Boni	699	France	Female	39	1	0.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	12

In [4]:
```
# count of Rows and Columns in our Dataset

dataset.shape
```

Out[4]: (10000, 14)

In [5]:
```
# datatypes of the variable in Dataset

dataset.dtypes
```

Out[5]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [6]:
```
#Checking for null values in the data set

dataset.isnull().any()
```

Out[6]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [7]:
```
#Creating dummy matrix for Geography column

dummies = pd.get_dummies(dataset['Geography'])
dummies = dummies.add_prefix("{}#".format('Geography'))
```

In [8]:
```
#Creating Dummy columns for gender column

dummies_1 = pd.get_dummies(dataset['Gender'])
```

In [9]:
```
#Dropping the original Geography and Gender columns

dataset.drop('Geography', axis=1, inplace=True)
dataset.drop('Gender', axis=1, inplace=True)
#dataset = dataset.join(dummies)
#dataset

# adding updated columns in Dataset
dataset=pd.concat([dummies, dataset], axis=1)
dataset=pd.concat([dummies_1, dataset],axis=1)
```

In [10]:
```
# Updated data set after dropping 'Geography', 'Gender' Column and concatinating the dummy columns
dataset.head()
```

Out[10]:

	Female	Male	Geography#France	Geography#Germany	Geography#Spain	RowNumb
0	1	0	1	0	0	1
1	1	0	0	0	1	2
2	1	0	1	0	0	3
3	1	0	1	0	0	4
4	1	0	0	0	1	5

In [11]:
```
dataset.shape
```

Out[11]: (10000, 17)

In [12]:
```
#Dropping the extra columns
```

In [13]:
```
dataset.drop('Geography#France', axis=1, inplace=True)
```

In [14]:
```
dataset.drop('Male', axis=1, inplace=True)
```

In [15]:
```
dataset.drop('CustomerId', axis=1, inplace=True)
```

In [16]:
```
dataset.drop('RowNumber', axis=1, inplace=True)
dataset.drop('Surname', axis=1, inplace=True)
```

In [17]:
```
#Final Updated Dataset

dataset.head()
```

Out[17]:

	Female	Geography#Germany	Geography#Spain	CreditScore	Age	Tenure	Balance
0	1	0	0	619	42	2	0.00
1	1	0	1	608	41	1	83807.86
2	1	0	0	502	42	8	159660.80
3	1	0	0	699	39	1	0.00
4	1	0	1	850	43	2	125510.82

In [18]:
```
#Fetching values of relevant sttributes for which values in Exited column is 1
```

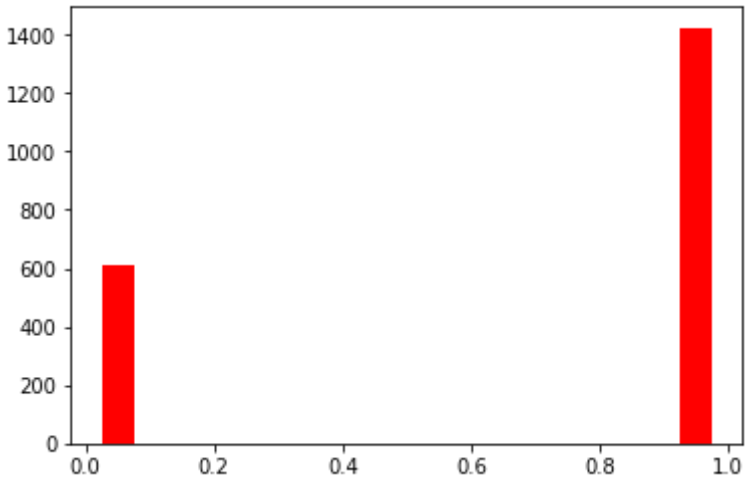
In [19]:
```
avgSalaryForLeavingCustomers=0.0
avgSalaryForRetainedCustomers=0.0
count=0
count1=0
EstSalary=[]
EstSalary1=[]
AgeGroup1=[]
AgeGroup2=[]
HasCrCardGr1=[]
HasCrCardGr2=[]
creditScores1=[]
creditScores2=[]
balance1=[]
balance2=[]
noOfProductsGroup1=[]
noOfProductsGroup2=[]
tenureGroup1=[]
tenureGroup2=[]
for i in range(len(dataset)):
    if dataset['Exited'].loc[i]==1:
        count=count+1;
        EstSalary.append(dataset['EstimatedSalary'].loc[i])
        creditScores1.append(dataset['CreditScore'].loc[i])
        AgeGroup1.append(dataset['Age'].loc[i])
        HasCrCardGr1.append(dataset['HasCrCard'].loc[i])
        noOfProductsGroup1.append(dataset['NumOfProducts'].loc[i])
        balance1.append(dataset['Balance'].loc[i])
        avgSalaryForLeavingCustomers= dataset['EstimatedSalary'].loc[i]+
avgSalaryForLeavingCustomers
    elif dataset['Exited'].loc[i]==0:
        count1=count1+1;
        EstSalary1.append(dataset['EstimatedSalary'].loc[i])
        creditScores2.append(dataset['CreditScore'].loc[i])
        HasCrCardGr2.append(dataset['HasCrCard'].loc[i])
        AgeGroup2.append(dataset['Age'].loc[i])
        balance2.append(dataset['Balance'].loc[i])
        noOfProductsGroup2.append(dataset['NumOfProducts'].loc[i])
        avgSalaryForLeavingCustomers= dataset['EstimatedSalary'].loc[i]+
avgSalaryForLeavingCustomers

print(count)
print(count1)
```

```
2037
7963
```

In [20]:
```
#Plotting graph between relevant attributes and exited column to analyse
 which group is exiting the most in each attributes.
```

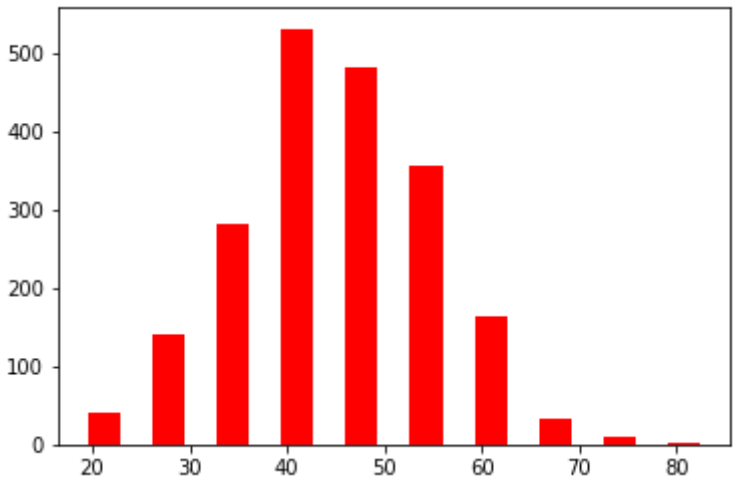
In [21]:
```
#Histogram between 'has credit card' and Exited = 1
```

In [23]:
```
import matplotlib.pyplot as plt
n_bins = 10
plt.hist(HasCrCardGr1, n_bins, histtype='bar', color='red',stacked='tru
e',rwidth=0.5)
plt.show()
```



In [24]:
```
#Histogram between AgeGroup and Exited
```

In [25]:
```
import matplotlib.pyplot as plt
n_bins = 10
plt.hist(AgeGroup1, n_bins,histtype='bar', color='red',stacked='true',rw
idth=0.5)
##plt.hist(EstSalary,num_binsalpha=0.5)
plt.show()
```



In [31]:
```
print('average salary for leaving Customers=', avgSalaryForLeavingCustom
ers/2037)
```

```
average salary for leaving Customers= 491361.020525
```

In [27]:
```
avgSalaryForRetainedCustomers=0.0
count=0
for i in range(len(dataset)):
    if dataset['Exited'].loc[i]==0:
        count=count+1;
        avgSalaryForRetainedCustomers= dataset['EstimatedSalary'].loc[i]
+avgSalaryForRetainedCustomers
print(count)
```

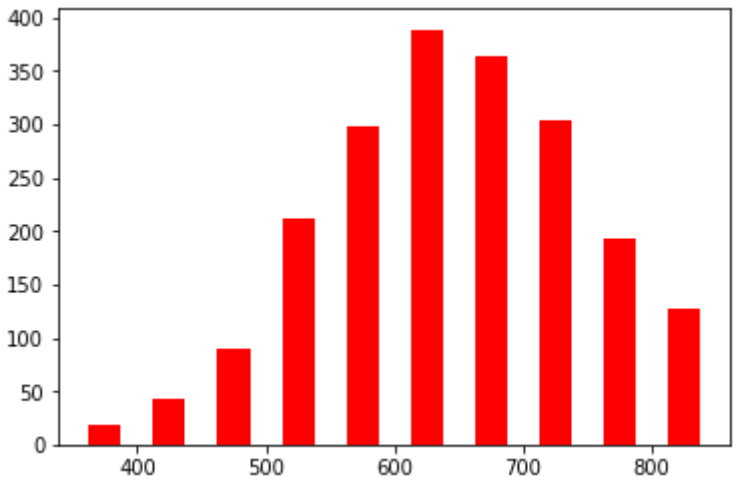
```
7963
```

In [32]:
```
print('average salary for retained Customers=', avgSalaryForRetainedCust
omers/7963)
```

```
average salary for retained Customers= 99738.3917719
```

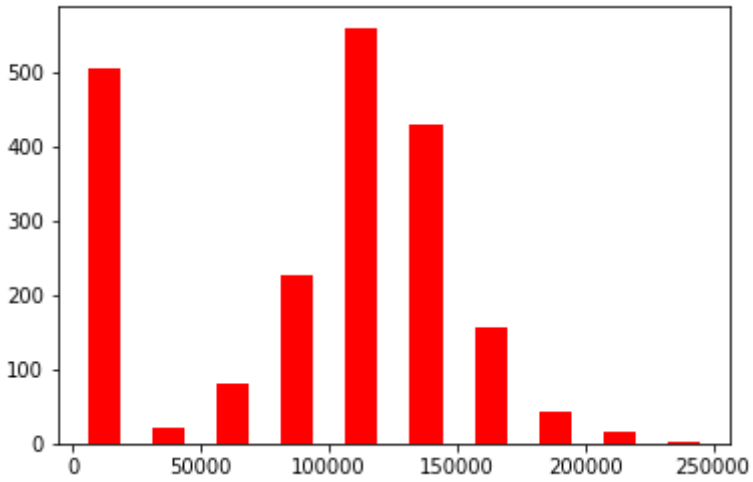
In [33]:
```
#Histogram between Credit Score and Exited
```

In [34]:
```
import matplotlib.pyplot as plt
n_bins = 10
plt.hist(creditScores1, n_bins, histtype='bar', color='red',stacked='tru
e',rwidth=0.5)
plt.show()
```



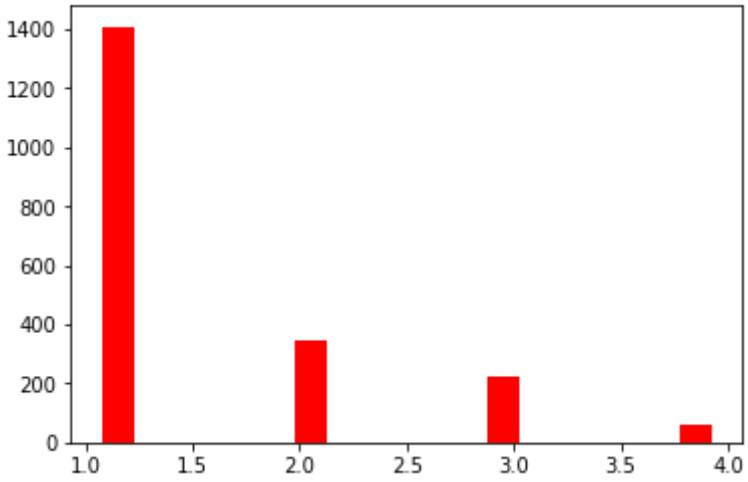
In [35]:
```
#Histogram between balance and Exited
```

In [36]:
```
import matplotlib.pyplot as plt
n_bins = 10
plt.hist(balance1, n_bins, histtype='bar', color='red',stacked='true',rwidth=0.5)
plt.show()
```



In [37]:
```
#Histogram between Number of products owned by the customer and Exited
```

In [38]:
```
import matplotlib.pyplot as plt
n_bins = 10
plt.hist(noOfProductsGroup1, n_bins, histtype='bar', color='red',stacked='true',rwidth=0.5)
plt.show()
```



In [39]:
```
dataset.shape
```

Out[39]: (10000, 12)

In [40]:
```
dataset.head()
```

Out[40]:

	Female	Geography#Germany	Geography#Spain	CreditScore	Age	Tenure	Balance
0	1	0	0	619	42	2	0.00
1	1	0	1	608	41	1	83807.86
2	1	0	0	502	42	8	159660.80
3	1	0	0	699	39	1	0.00
4	1	0	1	850	43	2	125510.82

In [41]:
```
#Assigning independent and dependent values from data set in variables
```

In [42]:
```
X = dataset.iloc[:, :11].values
y = dataset.iloc[:, 11].values
```

In [43]:
```
X.shape
```

Out[43]:
```
(10000, 11)
```

In [44]:
```
y.shape
```

Out[44]:
```
(10000,)
```

In [45]:
```
X,y
```

Out[45]:
```
(array([[  1.00000000e+00,   0.00000000e+00,   0.00000000e+00, ...,
           1.00000000e+00,   1.00000000e+00,   1.01348880e+05],
        [  1.00000000e+00,   0.00000000e+00,   1.00000000e+00, ...,
           0.00000000e+00,   1.00000000e+00,   1.12542580e+05],
        [  1.00000000e+00,   0.00000000e+00,   0.00000000e+00, ...,
           1.00000000e+00,   0.00000000e+00,   1.13931570e+05],
        ...,
        [  1.00000000e+00,   0.00000000e+00,   0.00000000e+00, ...,
           0.00000000e+00,   1.00000000e+00,   4.20855800e+04],
        [  0.00000000e+00,   1.00000000e+00,   0.00000000e+00, ...,
           1.00000000e+00,   0.00000000e+00,   9.28885200e+04],
        [  1.00000000e+00,   0.00000000e+00,   0.00000000e+00, ...,
           1.00000000e+00,   0.00000000e+00,   3.81907800e+04]]),
 array([1, 0, 1, ..., 1, 1, 0]))
```

In [46]:
```
#Creating test and training set. Splittig data into 80:20 ratio for trai
ning and test respectively.
```

In [47]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
2, random_state = 0)
```

In [50]:
```
X_test.shape,X_train.shape,y_train.shape,y_test.shape
```

Out[50]:
```
((2000, 11), (8000, 11), (8000,), (2000,))
```

In [51]:
```
#Standardizing the variables to bring them all to the same scale
```

In [52]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [53]:
```
#Importing Deep learning libraries
```

In [54]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
Using TensorFlow backend.
```

In [55]:
```
# Initialising the ANN
classifier = Sequential()
```

In [56]:
```
# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 6, kernel_initializer = 'uniform', act
ivation = 'relu', input_dim = 11))
```

```
/Users/varunpandey/anaconda3/lib/python3.6/site-packages/ipykernel_laun
cher.py:2: UserWarning: Update your `Dense` call to the Keras 2 API: `D
ense(kernel_initializer="uniform", activation="relu", input_dim=11, uni
ts=6)`
```

In [57]:
```
# Adding the second hidden layer
classifier.add(Dense(output_dim = 6, kernel_initializer = 'uniform', act
ivation = 'relu'))
```

```
/Users/varunpandey/anaconda3/lib/python3.6/site-packages/ipykernel_laun
cher.py:2: UserWarning: Update your `Dense` call to the Keras 2 API: `D
ense(kernel_initializer="uniform", activation="relu", units=6)`
```

In [58]:
```
# Adding the output layer
classifier.add(Dense(output_dim = 1, kernel_initializer = 'uniform', act
ivation = 'sigmoid'))
```

```
/Users/varunpandey/anaconda3/lib/python3.6/site-packages/ipykernel_laun
cher.py:2: UserWarning: Update your `Dense` call to the Keras 2 API: `D
ense(kernel_initializer="uniform", activation="sigmoid", units=1)`
```

In [59]:
```
# Compiling the ANN

classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', met
rics = ['accuracy'])
```

In [60]:
```
# Fitting the ANN to the Training set
classifier.fit(X_train, y_train, batch_size = 10, epochs = 100)
```

```
Epoch 1/100
8000/8000 [==============================] - 4s 481us/step - loss: 0.48
60 - acc: 0.7959
Epoch 2/100
8000/8000 [==============================] - 3s 429us/step - loss: 0.43
01 - acc: 0.7960
Epoch 3/100
8000/8000 [==============================] - 3s 401us/step - loss: 0.42
43 - acc: 0.7960
Epoch 4/100
8000/8000 [==============================] - 3s 391us/step - loss: 0.41
88 - acc: 0.8191
Epoch 5/100
8000/8000 [==============================] - 3s 424us/step - loss: 0.41
59 - acc: 0.8250
Epoch 6/100
8000/8000 [==============================] - 3s 435us/step - loss: 0.41
44 - acc: 0.8287
Epoch 7/100
8000/8000 [==============================] - 3s 393us/step - loss: 0.41
28 - acc: 0.8297
Epoch 8/100
8000/8000 [==============================] - 4s 442us/step - loss: 0.41
17 - acc: 0.8305
Epoch 9/100
8000/8000 [==============================] - 4s 464us/step - loss: 0.41
07 - acc: 0.8324
Epoch 10/100
8000/8000 [==============================] - 3s 392us/step - loss: 0.40
99 - acc: 0.8326
Epoch 11/100
8000/8000 [==============================] - 3s 396us/step - loss: 0.40
87 - acc: 0.8345
Epoch 12/100
8000/8000 [==============================] - 3s 394us/step - loss: 0.40
81 - acc: 0.8329
Epoch 13/100
8000/8000 [==============================] - 4s 484us/step - loss: 0.40
75 - acc: 0.8344
Epoch 14/100
8000/8000 [==============================] - 3s 427us/step - loss: 0.40
65 - acc: 0.8341
Epoch 15/100
8000/8000 [==============================] - 3s 389us/step - loss: 0.40
65 - acc: 0.8342
Epoch 16/100
8000/8000 [==============================] - 3s 385us/step - loss: 0.40
63 - acc: 0.8344
Epoch 17/100
8000/8000 [==============================] - 3s 430us/step - loss: 0.40
60 - acc: 0.8352
Epoch 18/100
8000/8000 [==============================] - 3s 408us/step - loss: 0.40
48 - acc: 0.8347
Epoch 19/100
8000/8000 [==============================] - 3s 435us/step - loss: 0.40
48 - acc: 0.8349
```

```
Epoch 20/100
8000/8000 [==============================] - 3s 425us/step - loss: 0.40
42 - acc: 0.8345
Epoch 21/100
8000/8000 [==============================] - 3s 401us/step - loss: 0.40
44 - acc: 0.8349
Epoch 22/100
8000/8000 [==============================] - 3s 434us/step - loss: 0.40
38 - acc: 0.8332
Epoch 23/100
8000/8000 [==============================] - 3s 394us/step - loss: 0.40
38 - acc: 0.8354
Epoch 24/100
8000/8000 [==============================] - 3s 395us/step - loss: 0.40
33 - acc: 0.8339
Epoch 25/100
8000/8000 [==============================] - 3s 399us/step - loss: 0.40
31 - acc: 0.8345
Epoch 26/100
8000/8000 [==============================] - 3s 389us/step - loss: 0.40
30 - acc: 0.8342
Epoch 27/100
8000/8000 [==============================] - 3s 361us/step - loss: 0.40
27 - acc: 0.8337
Epoch 28/100
8000/8000 [==============================] - 3s 383us/step - loss: 0.40
21 - acc: 0.8347
Epoch 29/100
8000/8000 [==============================] - 3s 395us/step - loss: 0.40
26 - acc: 0.8357
Epoch 30/100
8000/8000 [==============================] - 3s 377us/step - loss: 0.40
22 - acc: 0.8342
Epoch 31/100
8000/8000 [==============================] - 3s 400us/step - loss: 0.40
21 - acc: 0.8352
Epoch 32/100
8000/8000 [==============================] - 4s 441us/step - loss: 0.40
20 - acc: 0.8351
Epoch 33/100
8000/8000 [==============================] - 3s 408us/step - loss: 0.40
18 - acc: 0.8347
Epoch 34/100
8000/8000 [==============================] - 3s 429us/step - loss: 0.40
20 - acc: 0.8357
Epoch 35/100
8000/8000 [==============================] - 3s 388us/step - loss: 0.40
20 - acc: 0.8356
Epoch 36/100
8000/8000 [==============================] - 3s 376us/step - loss: 0.40
16 - acc: 0.8359
Epoch 37/100
8000/8000 [==============================] - 3s 396us/step - loss: 0.40
12 - acc: 0.8349
Epoch 38/100
8000/8000 [==============================] - 3s 389us/step - loss: 0.40
14 - acc: 0.8346
```

```
Epoch 39/100
8000/8000 [==============================] - 3s 399us/step - loss: 0.40
13 - acc: 0.8346
Epoch 40/100
8000/8000 [==============================] - 3s 427us/step - loss: 0.40
14 - acc: 0.8331
Epoch 41/100
8000/8000 [==============================] - 3s 385us/step - loss: 0.40
12 - acc: 0.8345
Epoch 42/100
8000/8000 [==============================] - 3s 413us/step - loss: 0.40
10 - acc: 0.8362
Epoch 43/100
8000/8000 [==============================] - 3s 381us/step - loss: 0.40
14 - acc: 0.8340
Epoch 44/100
8000/8000 [==============================] - 3s 388us/step - loss: 0.40
08 - acc: 0.8345
Epoch 45/100
8000/8000 [==============================] - 3s 402us/step - loss: 0.40
10 - acc: 0.8340
Epoch 46/100
8000/8000 [==============================] - 3s 371us/step - loss: 0.40
13 - acc: 0.8350
Epoch 47/100
8000/8000 [==============================] - 3s 355us/step - loss: 0.40
13 - acc: 0.8342
Epoch 48/100
8000/8000 [==============================] - 3s 423us/step - loss: 0.40
12 - acc: 0.8344
Epoch 49/100
8000/8000 [==============================] - 3s 417us/step - loss: 0.40
09 - acc: 0.8339
Epoch 50/100
8000/8000 [==============================] - 4s 439us/step - loss: 0.40
11 - acc: 0.8342
Epoch 51/100
8000/8000 [==============================] - 3s 380us/step - loss: 0.40
08 - acc: 0.8341
Epoch 52/100
8000/8000 [==============================] - 3s 417us/step - loss: 0.40
08 - acc: 0.8354
Epoch 53/100
8000/8000 [==============================] - 2s 312us/step - loss: 0.40
08 - acc: 0.8350
Epoch 54/100
8000/8000 [==============================] - 3s 347us/step - loss: 0.40
07 - acc: 0.8340
Epoch 55/100
8000/8000 [==============================] - 3s 389us/step - loss: 0.40
09 - acc: 0.8336
Epoch 56/100
8000/8000 [==============================] - 3s 422us/step - loss: 0.40
02 - acc: 0.8369
Epoch 57/100
8000/8000 [==============================] - 2s 310us/step - loss: 0.40
05 - acc: 0.8360
```

```
Epoch 58/100
8000/8000 [==============================] - 3s 407us/step - loss: 0.40
05 - acc: 0.8356
Epoch 59/100
8000/8000 [==============================] - 3s 392us/step - loss: 0.40
06 - acc: 0.8362
Epoch 60/100
8000/8000 [==============================] - 3s 345us/step - loss: 0.40
04 - acc: 0.8365
Epoch 61/100
8000/8000 [==============================] - 3s 389us/step - loss: 0.40
03 - acc: 0.8369
Epoch 62/100
8000/8000 [==============================] - 3s 352us/step - loss: 0.40
06 - acc: 0.8350
Epoch 63/100
8000/8000 [==============================] - 3s 314us/step - loss: 0.40
05 - acc: 0.8367
Epoch 64/100
8000/8000 [==============================] - 3s 385us/step - loss: 0.40
03 - acc: 0.8364
Epoch 65/100
8000/8000 [==============================] - 3s 426us/step - loss: 0.40
05 - acc: 0.8346
Epoch 66/100
8000/8000 [==============================] - 3s 421us/step - loss: 0.40
07 - acc: 0.8359
Epoch 67/100
8000/8000 [==============================] - 3s 387us/step - loss: 0.40
03 - acc: 0.8350
Epoch 68/100
8000/8000 [==============================] - 3s 363us/step - loss: 0.40
02 - acc: 0.8359
Epoch 69/100
8000/8000 [==============================] - 3s 378us/step - loss: 0.39
96 - acc: 0.8382
Epoch 70/100
8000/8000 [==============================] - 2s 281us/step - loss: 0.40
05 - acc: 0.8372
Epoch 71/100
8000/8000 [==============================] - 3s 386us/step - loss: 0.40
00 - acc: 0.8364
Epoch 72/100
8000/8000 [==============================] - 3s 314us/step - loss: 0.40
04 - acc: 0.8351
Epoch 73/100
8000/8000 [==============================] - 3s 330us/step - loss: 0.40
05 - acc: 0.8364
Epoch 74/100
8000/8000 [==============================] - 3s 322us/step - loss: 0.40
03 - acc: 0.8369
Epoch 75/100
8000/8000 [==============================] - 3s 362us/step - loss: 0.40
02 - acc: 0.8364
Epoch 76/100
8000/8000 [==============================] - 3s 334us/step - loss: 0.40
02 - acc: 0.8359
```

```
Epoch 77/100
8000/8000 [==============================] - 3s 319us/step - loss: 0.40
03 - acc: 0.8361
Epoch 78/100
8000/8000 [==============================] - 3s 411us/step - loss: 0.39
98 - acc: 0.8351
Epoch 79/100
8000/8000 [==============================] - 3s 413us/step - loss: 0.40
01 - acc: 0.8367
Epoch 80/100
8000/8000 [==============================] - 3s 426us/step - loss: 0.40
04 - acc: 0.8341
Epoch 81/100
8000/8000 [==============================] - 3s 399us/step - loss: 0.40
00 - acc: 0.8352
Epoch 82/100
8000/8000 [==============================] - 3s 397us/step - loss: 0.40
03 - acc: 0.8361
Epoch 83/100
8000/8000 [==============================] - 4s 451us/step - loss: 0.40
02 - acc: 0.8362
Epoch 84/100
8000/8000 [==============================] - 3s 411us/step - loss: 0.40
03 - acc: 0.8354
Epoch 85/100
8000/8000 [==============================] - 3s 400us/step - loss: 0.40
00 - acc: 0.8362
Epoch 86/100
8000/8000 [==============================] - 3s 375us/step - loss: 0.40
01 - acc: 0.8352
Epoch 87/100
8000/8000 [==============================] - 3s 406us/step - loss: 0.40
02 - acc: 0.8374
Epoch 88/100
8000/8000 [==============================] - 3s 435us/step - loss: 0.40
03 - acc: 0.8350
Epoch 89/100
8000/8000 [==============================] - 3s 409us/step - loss: 0.40
02 - acc: 0.8357
Epoch 90/100
8000/8000 [==============================] - 3s 413us/step - loss: 0.40
01 - acc: 0.8354
Epoch 91/100
8000/8000 [==============================] - 3s 436us/step - loss: 0.39
98 - acc: 0.8365
Epoch 92/100
8000/8000 [==============================] - 3s 411us/step - loss: 0.39
98 - acc: 0.8362
Epoch 93/100
8000/8000 [==============================] - 3s 395us/step - loss: 0.39
97 - acc: 0.8354
Epoch 94/100
8000/8000 [==============================] - ETA: 0s - loss: 0.4002 - a
cc: 0.834 - 3s 356us/step - loss: 0.3999 - acc: 0.8349
Epoch 95/100
8000/8000 [==============================] - 3s 434us/step - loss: 0.40
02 - acc: 0.8354
```

```
Epoch 96/100
8000/8000 [==============================] - 3s 376us/step - loss: 0.39
97 - acc: 0.8361
Epoch 97/100
8000/8000 [==============================] - 3s 386us/step - loss: 0.39
92 - acc: 0.8347
Epoch 98/100
8000/8000 [==============================] - 3s 378us/step - loss: 0.40
04 - acc: 0.8354
Epoch 99/100
8000/8000 [==============================] - 3s 361us/step - loss: 0.40
01 - acc: 0.8355
Epoch 100/100
8000/8000 [==============================] - 4s 447us/step - loss: 0.40
03 - acc: 0.8352
```

Out[60]: <keras.callbacks.History at 0x1a1fb21c50>

In [61]:
```
#Part 3 - Making the predictions and evaluating the model
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

In [62]:
```
y_pred
```

Out[62]:
```
array([[ 0.24377614],
       [ 0.36874709],
       [ 0.18942001],
       ...,
       [ 0.18134569],
       [ 0.1650631 ],
       [ 0.12963018]], dtype=float32)
```

In [63]:
```
y_pred = (y_pred > 0.5)
```

In [64]:
```
y_pred
```

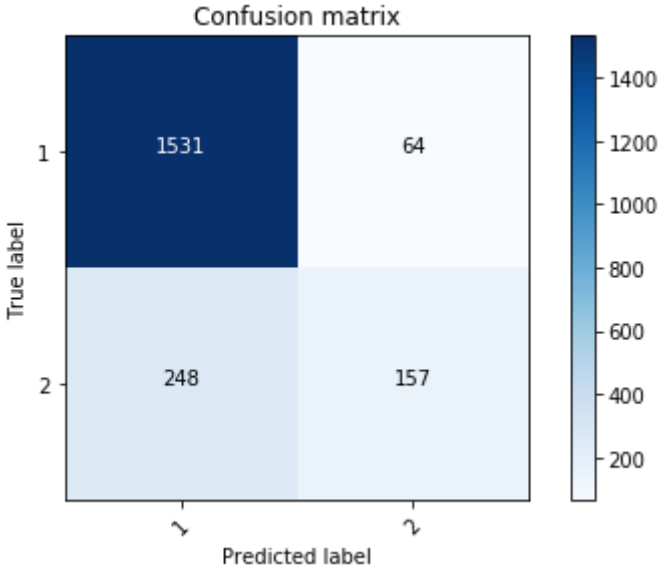
Out[64]:
```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]], dtype=bool)
```

In [65]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

In [69]:
```
#This function will plot a confusion matrix and is taken from the sklear
n documentation with just some minor tweaks
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = np.around((cm.astype('float') / cm.sum(axis=1)[:, np.newaxi
s]),decimals=2)
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1
])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [70]:
```
import itertools

class_names = ['1', '2']

plt.figure()
plot_confusion_matrix(cm, classes=class_names,normalize=False,title='Con
fusion matrix')
plt.show()
```

```
Confusion matrix, without normalization
[[1531   64]
 [ 248  157]]
```



In [71]:
```
accuracy=(1531+157)/2000
```

In [72]:
```
accuracy
```

Out[72]: 0.844