```
1.data collection
2.dataprepprocessing
3.analysie
4.traing the data - ANN
5.importing the librarries
6.to initialze the model
7.add input layer
8.add hidden layers
9.add output layer
10.configure the learning process
11.train the algorithm
12.test the algorithm
13.predition
14.save the model
```

In [1]:

```python
import pandas as pd
import numpy as np
```

In [2]:

```python
#if file is in the jupyter notebook working directory then dataset=pd.read_csv("Churn_Modelling.csv"), so no need to use under c

dataset=pd.read_csv(r"D:\ML_Course\Works_on_python\ANN Algo\Churn_Modelling.csv")
```

In [3]:

```python
type(dataset)
```

Out[3]:

```
pandas.core.frame.DataFrame
```

In [4]:
```
dataset.isnull().any()
```

Out[4]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography           True
Gender             False
Age                 True
Tenure             False
Balance             True
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [5]:
```
dataset.head(10)
```

Out[5]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSal
0	1	15634602	Hargrave	619	France	Female	42.0	2	0.00	1	1	1	101348
1	2	15647311	Hill	225	Spain	Female	41.0	1	83807.86	1	0	1	112542
2	3	15619304	Onio	629	France	Female	42.0	8	159660.80	3	1	0	113931
3	4	15701354	Boni	699	France	Female	39.0	1	0.00	2	0	0	93826
4	5	15737888	Mitchell	850	Spain	Female	43.0	2	125510.82	1	1	1	79084
5	6	15574012	Chu	645	Spain	Male	44.0	8	113755.78	2	1	0	149756
6	7	15592531	Bartlett	822	NaN	Male	50.0	7	0.00	2	1	1	10062
7	8	15656148	Obinna	376	Germany	Female	29.0	4	115046.74	4	1	0	119346
8	9	15792365	He	501	France	Male	44.0	4	142051.07	2	0	1	74940
9	10	15592389	H?	684	France	Male	NaN	2	134603.88	1	1	1	71725

In [6]:
```
dataset['Geography'].fillna(dataset['Geography'].mode()[0],inplace=True)
dataset['Age'].fillna(dataset['Age'].mean(),inplace=True)
dataset['Balance'].fillna(dataset['Balance'].median(),inplace=True)
```

In [7]:
```


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

Out[7]:
```
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 [8]:
```
dataset.head(10)
```

Out[8]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimat
0	1	15634602	Hargrave	619	France	Female	42.000000	2	0.00	1	1	1	1
1	2	15647311	Hill	225	Spain	Female	41.000000	1	83807.86	1	0	1	1
2	3	15619304	Onio	629	France	Female	42.000000	8	159660.80	3	1	0	1
3	4	15701354	Boni	699	France	Female	39.000000	1	0.00	2	0	0	
4	5	15737888	Mitchell	850	Spain	Female	43.000000	2	125510.82	1	1	1	
5	6	15574012	Chu	645	Spain	Male	44.000000	8	113755.78	2	1	0	1
6	7	15592531	Bartlett	822	France	Male	50.000000	7	0.00	2	1	1	
7	8	15656148	Obinna	376	Germany	Female	29.000000	4	115046.74	4	1	0	1
8	9	15792365	He	501	France	Male	44.000000	4	142051.07	2	0	1	
9	10	15592389	H?	684	France	Male	38.922992	2	134603.88	1	1	1	

In [9]:
```
p=dataset['Geography'].mode()
```

In [10]:
```
p
```

Out[10]:
```
0    France
dtype: object
```

In [11]:
```
p[0]
```

Out[11]:
```
'France'
```

In [12]:
```

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset['Geography']=le.fit_transform(dataset['Geography'])#encoding values of Geo then storing it on dataset['Geo'] and it is p
```

In [13]:
```
dataset['Gender']=le.fit_transform(dataset['Gender'])
#male 1 , female 0
```

In [14]:
```
dataset.head(10)
```

Out[14]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimat
0	1	15634602	Hargrave	619	0	0	42.000000	2	0.00	1	1	1	1
1	2	15647311	Hill	225	2	0	41.000000	1	83807.86	1	0	1	1
2	3	15619304	Onio	629	0	0	42.000000	8	159660.80	3	1	0	1
3	4	15701354	Boni	699	0	0	39.000000	1	0.00	2	0	0	
4	5	15737888	Mitchell	850	2	0	43.000000	2	125510.82	1	1	1	
5	6	15574012	Chu	645	2	1	44.000000	8	113755.78	2	1	0	1
6	7	15592531	Bartlett	822	0	1	50.000000	7	0.00	2	1	1	
7	8	15656148	Obinna	376	1	0	29.000000	4	115046.74	4	1	0	1
8	9	15792365	He	501	0	1	44.000000	4	142051.07	2	0	1	
9	10	15592389	H?	684	0	1	38.922992	2	134603.88	1	1	1	

```
alphabetical order -
1 france ,spain and germenay
France got numerical value from 0 to 2
germany -1
spain -2
```

```
india 2
japan 3
swis 4
germany 1
australia 0
```

In [15]:
```
dataset.corr()
```

Out[15]:

	RowNumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
RowNumber	1.000000	0.004202	0.006294	-0.010358	0.018196	0.000586	-0.006495	-0.009053	0.007246	0.000599	0.012044
CustomerId	0.004202	1.000000	0.005413	0.006516	-0.002641	0.009342	-0.014883	-0.012427	0.016972	-0.014025	0.001665
CreditScore	0.006294	0.005413	1.000000	0.007163	-0.002564	-0.003962	0.001526	0.006401	0.012920	-0.004756	0.025110
Geography	-0.010358	0.006516	0.007163	1.000000	0.004719	0.022710	0.003739	0.069415	0.003972	-0.008523	0.006724
Gender	0.018196	-0.002641	-0.002564	0.004719	1.000000	-0.027442	0.014733	0.012080	-0.021859	0.005766	0.022544
Age	0.000586	0.009342	-0.003962	0.022710	-0.027442	1.000000	-0.010116	0.028422	-0.030786	-0.011648	0.085588
Tenure	-0.006495	-0.014883	0.001526	0.003739	0.014733	-0.010116	1.000000	-0.012257	0.013444	0.022583	-0.028362
Balance	-0.009053	-0.012427	0.006401	0.069415	0.012080	0.028422	-0.012257	1.000000	-0.304187	-0.014846	-0.010076
NumOfProducts	0.007246	0.016972	0.012920	0.003972	-0.021859	-0.030786	0.013444	-0.304187	1.000000	0.003183	0.009612
HasCrCard	0.000599	-0.014025	-0.004756	-0.008523	0.005766	-0.011648	0.022583	-0.014846	0.003183	1.000000	-0.011866
IsActiveMember	0.012044	0.001665	0.025110	0.006724	0.022544	0.085588	-0.028362	-0.010076	0.009612	-0.011866	1.000000
EstimatedSalary	-0.005988	0.015271	-0.001437	-0.001369	-0.008112	-0.007258	0.007784	0.012800	0.014204	-0.009933	-0.011421
Exited	-0.016571	-0.006248	-0.026611	0.035943	-0.106512	0.285284	-0.014001	0.118537	-0.047820	-0.007138	-0.156128

In [16]:
```
x=dataset.iloc[:,3:13].values # iloc is index location -> input and .value is to converted to array coz onehot only works on arr
```

```
if ouput is middle and input is on both side of output then -
x=dataset.iloc[:,[0,1,2..input cols]]
y=dataset.iloc[:,start op col:end op col]
```

In [17]:
```
y=dataset.iloc[:,13].values # output
```

In [18]:
```
x
```

Out[18]:
```
array([[6.1900000e+02, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [2.2500000e+02, 2.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [6.2900000e+02, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [7.0900000e+02, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [7.7200000e+02, 1.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [7.9200000e+02, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

In [19]:
```
y
```

Out[19]:
```
array([1, 0, 1, ..., 1, 1, 0], dtype=int64)
```

In [20]:
```
x.shape
```

Out[20]:
```
(10000, 10)
```

In [21]:
```
type(x)
```

Out[21]:
```
numpy.ndarray
```

In [22]:

```
#onehot only done to arrays
#applying on geography coz when more than 2 values in catagory of dataset it can applies but gender having 0 and 1 so if encode

from sklearn.preprocessing import OneHotEncoder
one=OneHotEncoder()
z=one.fit_transform(x[:,1:2]).toarray()
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

In [23]:

```
x=np.delete(x,1,axis=1) # coz there are more than 2 values in geography so each one will have different binary col so deleting t
x
```

Out[23]:

```
array([[6.1900000e+02, 0.0000000e+00, 4.2000000e+01, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [2.2500000e+02, 0.0000000e+00, 4.1000000e+01, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [6.2900000e+02, 0.0000000e+00, 4.2000000e+01, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [7.0900000e+02, 0.0000000e+00, 3.6000000e+01, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [7.7200000e+02, 1.0000000e+00, 4.2000000e+01, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [7.9200000e+02, 0.0000000e+00, 2.8000000e+01, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

In [24]:
```
x=np.concatenate((z,x),axis=1)
x
```

Out[24]:
```
array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

In [25]:
```
z
```

Out[25]:
```
array([[1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       ...,
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.]])
```

France 1 0 0 Spain 0 0 1 Germany 0 1 0

In [26]:
```
x.shape # don't compile again coz more columns will get add
```

Out[26]:
```
(10000, 12)
```

In [27]:
```
y.shape
```

Out[27]:
```
(10000,)
```

In [28]:
```
#step 7

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)#test_size=0.2 means for testing only 20% of the
```

In [29]:
```
x_test.shape # questions in machine xam
```

Out[29]:
```
(2000, 12)
```

In [30]:
```
y_test.shape # key paper for machine xam
```

Out[30]:
```
(2000,)
```

In [31]:
```
x_train.shape # question for training
```

Out[31]:
```
(8000, 12)
```

In [32]:
```
y_train.shape # answer for training
```

Out[32]:
```
(8000,)
```

In [33]:
```
x_train[:,5:10]
```

Out[33]:
```
array([[3.4000000e+01, 5.0000000e+00, 0.0000000e+00, 2.0000000e+00,
        1.0000000e+00],
       [4.2000000e+01, 1.0000000e+00, 7.5681520e+04, 1.0000000e+00,
        1.0000000e+00],
       [2.9000000e+01, 2.0000000e+00, 1.1236734e+05, 1.0000000e+00,
        1.0000000e+00],
       ...,
       [3.5000000e+01, 5.0000000e+00, 1.6127405e+05, 2.0000000e+00,
        1.0000000e+00],
       [3.8000000e+01, 9.0000000e+00, 0.0000000e+00, 2.0000000e+00,
        1.0000000e+00],
       [4.8000000e+01, 1.0000000e+00, 1.0807633e+05, 1.0000000e+00,
        1.0000000e+00]])
```

```
x=np.array([1,2,3,4,5,6,7,8,9])
y=np.array([0,1,0,1,0,1,0,1,0])
```

```
# random_stae=0 is keep the training process same which means with random we will be able to train with same data no matter
how many times you execute the code


from sklearn.model_selection import train_test_split
for i in range(5):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
    print("x_train without random sate for the iteration",i,x_train)
for i in range(5):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
    print("x_train with random sate for the iteration",i,x_train)
```

In [34]:

```
#in order to handel outlayer all data need in proper range

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

In [35]:

```
x_train
```

Out[35]:

```
array([[-1.01460667, -0.5698444 ,  1.74309049, ...,  0.64259497,
        -1.03227043,  1.10643166],
       [-1.01460667,  1.75486502, -0.57369368, ...,  0.64259497,
         0.9687384 , -0.74866447],
       [ 0.98560362, -0.5698444 , -0.57369368, ...,  0.64259497,
        -1.03227043,  1.48533467],
       ...,
       [ 0.98560362, -0.5698444 , -0.57369368, ...,  0.64259497,
        -1.03227043,  1.41231994],
       [-1.01460667, -0.5698444 ,  1.74309049, ...,  0.64259497,
         0.9687384 ,  0.84432121],
       [-1.01460667,  1.75486502, -0.57369368, ...,  0.64259497,
        -1.03227043,  0.32472465]])
```

In [36]:
```
x_test
```

Out[36]:
```
array([[-0.95692675,  1.62776996, -0.57427105, ...,  0.66011376,
         0.97628121,  1.62185911],
       [ 1.04501206, -0.61433742, -0.57427105, ...,  0.66011376,
        -1.02429504,  0.504204  ],
       [-0.95692675, -0.61433742,  1.74133801, ...,  0.66011376,
         0.97628121, -0.41865644],
       ...,
       [-0.95692675, -0.61433742,  1.74133801, ...,  0.66011376,
        -1.02429504,  0.72775202],
       [-0.95692675,  1.62776996, -0.57427105, ...,  0.66011376,
         0.97628121, -1.54162886],
       [-0.95692675,  1.62776996, -0.57427105, ...,  0.66011376,
        -1.02429504,  1.62356528]])
```

ANN

In [38]:
```
#step 5

import keras
from keras.layers import Dense # used to initilize input output hidden layer
```

In [40]:
```
#step 6

from keras.models import Sequential
model=Sequential() # will work on sequential format
```

In [41]:
```
#step 7

model.add(Dense(units=12,init="random_uniform",activation="relu"))#randomly initilize weight by init
#units=12 coz 12 cols
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your `Dense` call to the Keras 2 API:
`Dense(units=12, activation="relu", kernel_initializer="random_uniform")`
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [42]:
```
#step 8

model.add(Dense(units=24,init="random_uniform",activation="relu"))#hidden layer added with 24 neurons 12*2 or 12*3...
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(units=24, activation="relu", kernel_initializer="random_uniform")`
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [46]:
```
#step 9

model.add(Dense(units=1,init="random_uniform",activation="sigmoid"))#unit=1 coz eithier yes or no
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(units=1, activation="sigmoid", kernel_initializer="random_uniform")`
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [49]:
```
#  step 10

model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
```

```
WARNING:tensorflow:From C:\Users\anikp\Anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [59]:
```
#step 11

model.fit(x_train,y_train,batch_size=32,epochs=500)# batch size 8 16 32 128 256 1024
#more epochs means more accuracy in o/p
```

```
Epoch 1/500
8000/8000 [==============================] - 0s 50us/step - loss: 0.3403 - accuracy: 0.8614
Epoch 2/500
8000/8000 [==============================] - 0s 49us/step - loss: 0.3390 - accuracy: 0.8625
Epoch 3/500
8000/8000 [==============================] - 0s 51us/step - loss: 0.3384 - accuracy: 0.8636
Epoch 4/500
8000/8000 [==============================] - 0s 52us/step - loss: 0.3389 - accuracy: 0.8650
Epoch 5/500
8000/8000 [==============================] - 0s 57us/step - loss: 0.3388 - accuracy: 0.8633
Epoch 6/500
8000/8000 [==============================] - 0s 50us/step - loss: 0.3379 - accuracy: 0.8635
Epoch 7/500
8000/8000 [==============================] - 0s 53us/step - loss: 0.3387 - accuracy: 0.8631
Epoch 8/500
8000/8000 [==============================] - 0s 55us/step - loss: 0.3384 - accuracy: 0.8631
Epoch 9/500
8000/8000 [==============================] - 0s 50us/step - loss: 0.3379 - accuracy: 0.8622
Epoch 10/500
```

In [60]:
```
y_pred=model.predict(x_test)
```

In [61]:
```
y_pred
```

Out[61]:
```
array([[0.15553224],
       [0.13688385],
       [0.05968964],
       ...,
       [0.05140147],
       [0.31251594],
       [0.25613296]], dtype=float32)
```

In [62]:
```
y_pred=(y_pred>0.5)# want to see yes or no then thrs taken 0.5 that is rule
```

In [63]:
```
y_pred
```

Out[63]:
```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

In [64]:
```
y_test
```

Out[64]:
```
array([0, 1, 0, ..., 0, 0, 0], dtype=int64)
```

In [70]:
```
dataset.head(1)
```

Out[70]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalar
0	1	15634602	Hargrave	619	0	0	42.0	2	0.0	1	1	1	101348.8

In [66]:
```
y_p=model.predict(sc.transform(np.array([[1,0,0,619,0,42,8,12345,1,1,1,1234567]])))
#1st 3 col not considered.. 1st geograpy then credit socre then gender ....
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int32
was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [67]:
```
y_p
```

Out[67]:
```
array([[0.06048154]], dtype=float32)
```

In [68]:
```
y_p=y_p>0.5
```

In [69]:
```
y_p
```

Out[69]:
```
array([[False]])
```

to increase accuracy add more hidden layers add more epochs

In [76]:
```
keras.__version__
```
Out[76]: '2.3.1'

In [75]:
```
import tensorflow
tensorflow.__version__
```
Out[75]: '1.13.1'

In [ ]: