Open in Colab

Importing the libraries

In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
```

Importing the Dataset

In [2]:
```
data = pd.read_csv('Churn_Modelling.csv')

data.head(4)
```

O...

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOf
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	
3	4	15701354	Boni	699	France	Female	39	1	0.00	

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [4]:
```
data.describe()
```

O...

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfPro
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.0
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.5

O...

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Nu
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	

In [6]:
```
# Checking if our dataset contains any NULL values

data.isnull().sum()
```

Out[6]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Data Analysis

In [7]:
```
data['Gender'].value_counts()
```

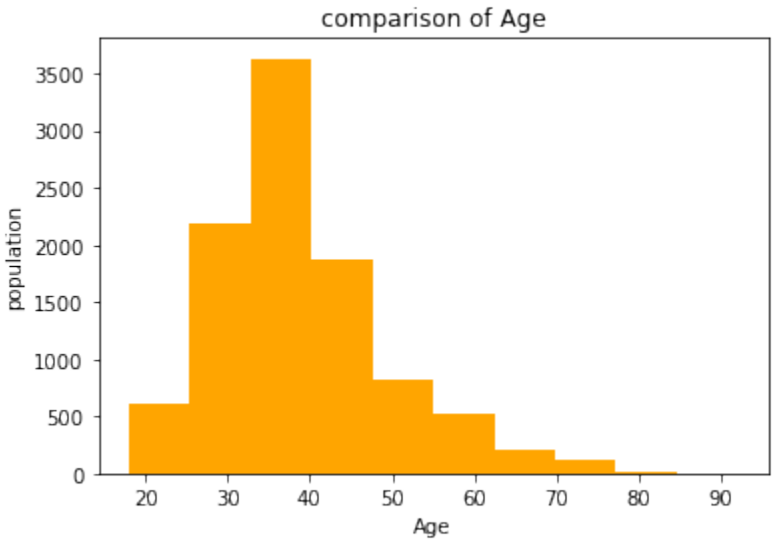
Out[7]:
```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

In [8]:
```
# Plotting the features of the dataset to see the correlation between them

plt.hist(x = data.Gender, bins = 3, color = 'pink')
plt.title('comparison of male and female')
plt.xlabel('Gender')
plt.ylabel('population')
plt.show()
```

In [9]:
```
data['Age'].value_counts()
```

Out[9]:
```
37     478
38     477
35     474
36     456
34     447
      ...
92       2
88       1
82       1
85       1
83       1
Name: Age, Length: 70, dtype: int64
```

In [10]:
```
# comparison of age in the dataset

plt.hist(x = data.Age, bins = 10, color = 'orange')
plt.title('comparison of Age')
plt.xlabel('Age')
plt.ylabel('population')
plt.show()
```



In [11]:
```
data['Geography'].value_counts()
```

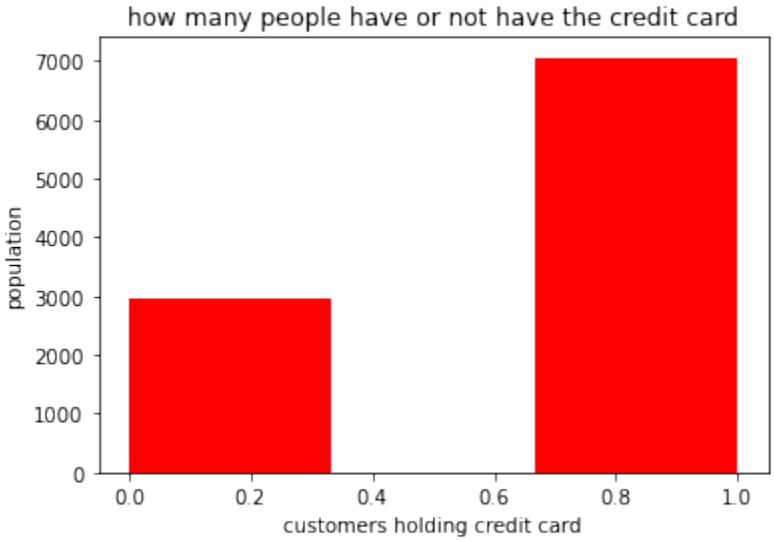
Out[11]:
```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

In [12]:
```
# comparison of geography

plt.hist(x = data.Geography, bins = 5, color = 'green')
plt.title('comparison of Geography')
plt.xlabel('Geography')
plt.ylabel('population')
plt.show()
```

In [13]:
```
data['HasCrCard'].value_counts()
```

Out[13]:
```
1    7055
0    2945
Name: HasCrCard, dtype: int64
```

In [14]:
```
# comparision of how many customers hold the credit card

plt.hist(x = data.HasCrCard, bins = 3, color = 'red')
plt.title('how many people have or not have the credit card')
plt.xlabel('customers holding credit card')
plt.ylabel('population')
plt.show()
```



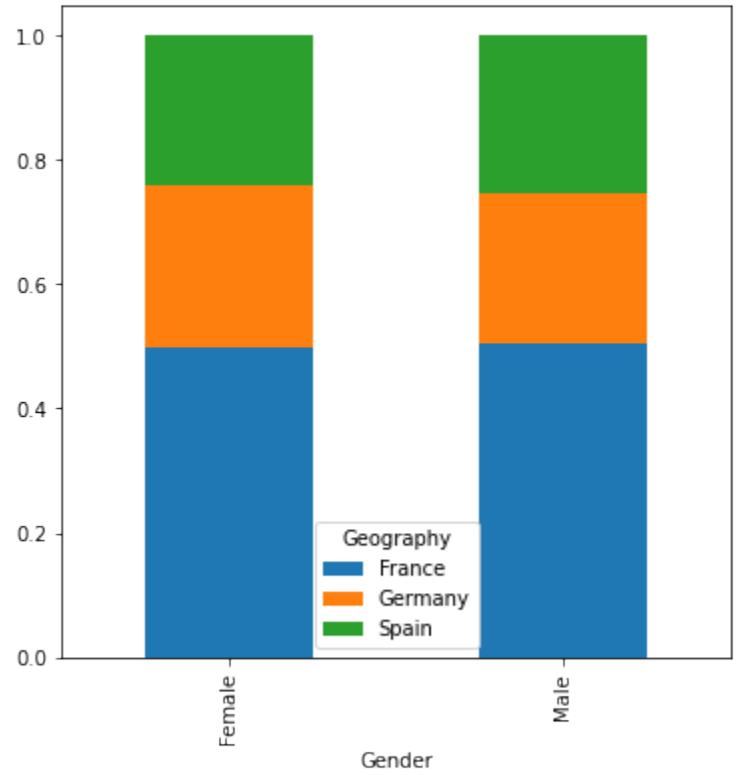
In [15]:
```
data['IsActiveMember'].value_counts()
```

Out[15]:
```
1    5151
0    4849
Name: IsActiveMember, dtype: int64
```

In [16]:
```
# How many active member does the bank have ?

plt.hist(x = data.IsActiveMember, bins = 3, color = 'brown')
plt.title('Active Members')
plt.xlabel('Customers')
plt.ylabel('population')
plt.show()
```

In [...

Out[17]:<AxesSubplot:xlabel='Gender'>

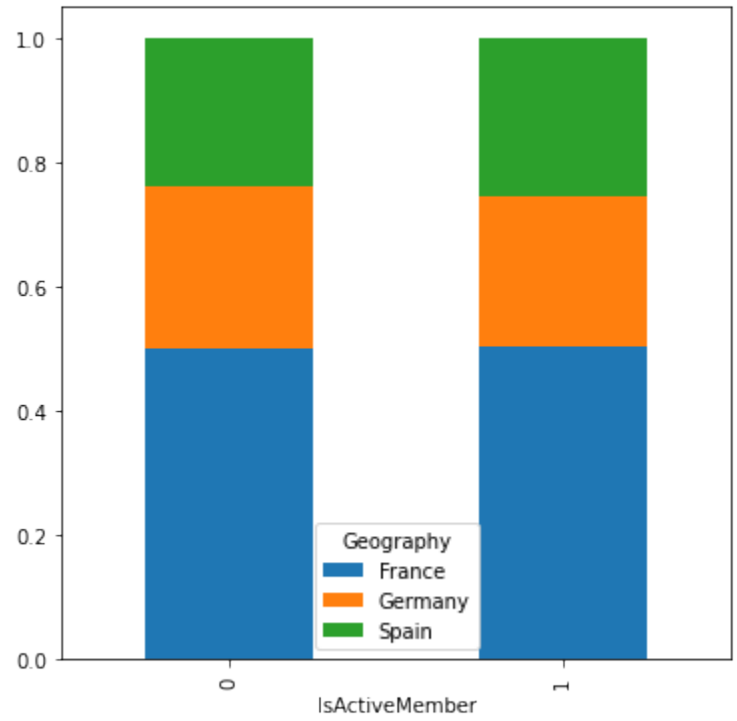


In [18]:
```
# comparison between geography and card holders

HasCrCard = pd.crosstab(data['HasCrCard'], data['Geography'])
HasCrCard.div(HasCrCard.sum(1).astype(float), axis = 0).plot(kind = 'bar',
                                                stacked = True,figsize = (6, 6))
```

Out[18]:<AxesSubplot:xlabel='HasCrCard'>

In [1...
```
# comparison of active member in differnt geographies

IsActiveMember = pd.crosstab(data['IsActiveMember'], data['Geography'])
```

Out[19]:<AxesSubplot:xlabel='IsActiveMember'>



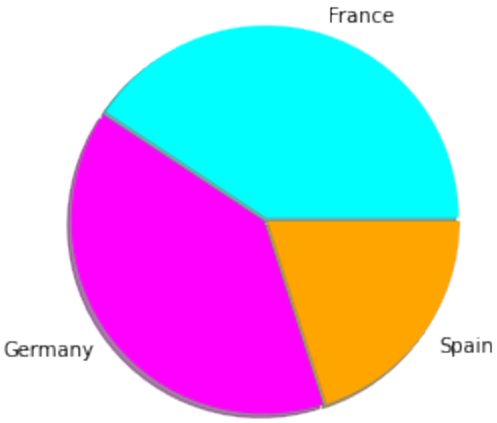
In [20]:
```
# comparing ages in different geographies

Age = pd.crosstab(data['Age'], data['Geography'])
Age.div(Age.sum(1).astype(float), axis = 0).plot(kind = 'bar',
                                                 stacked = True, figsize = (15,15))
```

Out[20]:<AxesSubplot:xlabel='Age'>

In [21]:
```
# calculating total balance in france, germany and spain

total_france = data.Balance[data.Geography == 'France'].sum()
total_germany = data.Balance[data.Geography == 'Germany'].sum()
total_spain = data.Balance[data.Geography == 'Spain'].sum()
```

```
Total Balance in France : 311332479.49
Total Balance in Germany : 300402861.38
Total Balance in Spain : 153123552.01
```

In [2...
```
# plotting a pie chart

labels = 'France', 'Germany', 'Spain'
colors = ['cyan', 'magenta', 'orange']
sizes =  [311, 300, 153]
explode = [ 0.01, 0.01, 0.01]

plt.pie(sizes, colors = colors, labels = labels, explode = explode, shadow = True

plt.axis('equal')
plt.show()
```



Data Preprocessing

In [23]:
```
# Removing the unnecassary features from the dataset

data = data.drop(['CustomerId', 'Surname', 'RowNumber'], axis = 1)



print(data.columns)
```

```
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',
       'Exited'],
      dtype='object')
```

In [24]:

Out[24]:
```
(10000, 11)
```

In [25]:
```
# splitting the dataset into x(independent variables) and y(dependent variables)

x = data.iloc[:,0:10]
y = data.iloc[:,10]

print(x.shape)
print(y.shape)
```

```
(10000, 10)
(10000,)
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary'],
      dtype='object')
```

In [26]:
```
# Encoding Categorical variables into numerical variables
# One Hot Encoding

x = pd.get_dummies(x)

x.head()
```

Out...

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	G
0	619	42	2	0.00	1	1	1	101348.88	
1	608	41	1	83807.86	1	0	1	112542.58	
2	502	42	8	159660.80	3	1	0	113931.57	
3	699	39	1	0.00	2	0	0	93826.63	
4	850	43	2	125510.82	1	1	1	79084.10	

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

Out[27]:
```
(10000, 13)
```

In [...
```
# splitting the data into training and testing set

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(7500, 13)
(7500,)
(2500, 13)
(2500,)
```

Out...

	0	1	2	3	4	5	6	7	8	
0	-0.735507	0.015266	0.008860	0.673160	2.535034	-1.553624	-1.034460	-1.640810	-1.015588	1.760
1	1.024427	-0.652609	0.008860	-1.207724	0.804242	0.643657	-1.034460	-0.079272	0.984651	-0.568
2	0.808295	-0.461788	1.393293	-0.356937	0.804242	0.643657	0.966688	-0.996840	-1.015588	-0.568
3	0.396614	-0.080145	0.008860	-0.009356	-0.926551	0.643657	0.966688	-1.591746	-1.015588	-0.568
4	-0.467915	1.255605	0.701077	-1.207724	0.804242	0.643657	0.966688	1.283302	0.984651	-0.568

Decision Tree

In [30]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix

model = DecisionTreeClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuaracy :", model.score(x_test, y_test))

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
Training Accuracy : 1.0
Testing Accuaracy : 0.8008
[[1707  284]
 [ 214  295]]
```

Random Forest

In [31]:
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
Training Accuracy : 1.0
Testing Accuracy : 0.868
[[1914   77]
 [ 253  256]]
```

In [32]:
```
# k fold cross validatio
```

```
[0.86133333 0.856      0.86266667 0.86666667 0.85466667 0.85466667
 0.864      0.85733333 0.86133333 0.86133333]
```

In [33]:
```
print("Mean Accuracy :", cvs.mean())
print("Variance :", cvs.std())
```

```
Mean Accuracy : 0.86
Variance : 0.003910100879630716
```

Logistic Regression

In [34]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
```

```
Training Accuracy : 0.8096
Testing Accuracy : 0.8092
[[1916   75]
 [ 402  107]]
```

Support Vector Machine

In [35]:
```
from sklearn.svm import SVC

model = SVC()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
Training Accuracy : 0.8625333333333334
Testing Accuracy : 0.8616
[[1951   40]
 [ 306  203]]
```

In [36]:
```
# k fold cross validatio

from sklearn.model_selection import cross_val_score

cvs = cross_val_score(estimator = model, X = x_train, y = y_train, cv = 10)
print(cvs)
```

```
[0.864      0.852      0.864      0.85733333 0.84266667 0.844
 0.852      0.85333333 0.84533333 0.85066667]

Mean Accuracy : 0.8525333333333333
Variance : 0.007160384843785353
```

Multi Layer Perceptron

In [38]:
```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(hidden_layer_sizes = (100, 100), activation ='relu',
                      solver = 'adam', max_iter = 50)
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
Training Accuracy : 0.8868
Testing Accuracy : 0.8632
```

Aritificial Neural Networks

In [39]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
~\anaconda3\lib\site-packages\tensorflow\python\pywrap_tensorflow.py in <module>
     63   try:
---> 64     from tensorflow.python._pywrap_tensorflow_internal import *
     65   # This try catch logic is because there is no bazel equivalent for py_exten
on.

ImportError: DLL load failed while importing _pywrap_tensorflow_internal: The specifi
module could not be found.

During handling of the above exception, another exception occurred:

ImportError                               Traceback (most recent call last)
<ipython-input-39-6724c5b7f2b7> in <module>
----> 1 import keras
      2 from keras.models import Sequential
      3 from keras.layers import Dense

~\anaconda3\lib\site-packages\keras\__init__.py in <module>
     19 """
     20 # pylint: disable=unused-import
```

In [ ]:
```
pip install tensorflow
```

In [ ]:

In ...
```
# creating the model
model =Sequential()

# first hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu', input_dim =

# second hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu'))

# third hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu'))

# fourth hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu'))

# fifth hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu'))

# output layer
model.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))

# Compiling the NN
```

In ...
```
# creating the model
model = Sequential()

from keras.layers import Dropout

# first hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu', input_dim =
model.add(Dropout(0.5))

# second hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu'))
model.add(Dropout(0.5))

# output layer
model.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))

# Compiling the NN
# binary_crossentropy loss function used when a binary output is expected
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accurac

model.fit(x_train, y_train, batch_size = 10, nb_epoch = 50)
```

In ...
```
# creating the model
model = Sequential()

# first hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu', input_dim =

# second hidden layer
model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu'))

# output layer
model.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))

# Compiling the NN
# binary_crossentropy loss function used when a binary output is expected
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accurac

model.fit(x_train, y_train, batch_size = 10, nb_epoch = 49)
```

In [ ]:
```
data.columns
```

In...
```
'''
predicting if the costumer having following information will leave the bank or not :

Geography : france
Age = 50
Credit score = 850
Tenure = 4
Balance = 150000
Number of Products = 5
Gender = Female
```

In ...
```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score

from keras.layers import Dense
from keras.models import Sequential

def build_classifier():
  # creating the model
  model = Sequential()

  # first hidden layer
  model.add(Dense(output_dim = 8, init = 'uniform', activation = 'relu', input_dim
```

In [ ]:
```
print("Accuracies :", accuracies)

print("Mean Accuracy :", accuracies.mean())
print("Variance :", accuracies.std())
```

In [ ]: