Project 2 Data Programming with Python (Keras and Bokeh)

Aniket Shah

Student Id: 18200042

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System).

Bokeh is an interactive visualization library that targets modern web browsers for presentation. Its goal is to provide elegant, concise construction of versatile graphics, and to extend this capability with high-performance interactivity over very large or streaming datasets. Bokeh can help anyone who would like to quickly and easily create interactive plots, dashboards, and data applications.

Churn prediction is one of the most well known applications of machine and deep learning and data science in the Customer Relationship Management (CRM) and Marketing fields. Simply put, a **churner** is a user or customer that stops using a company’s products or services.

PROBLEM STATEMENT (CLIENT REQUIREMENTS):

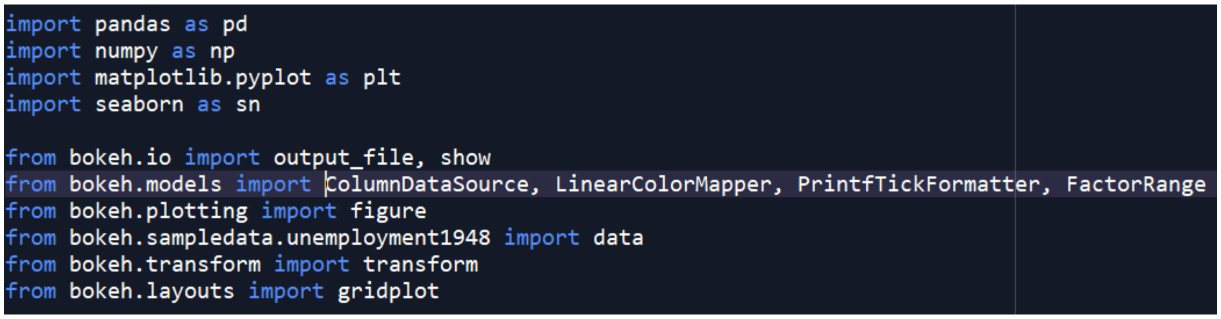
Using the Keras for deep learning and Bokeh for visualization on the Churn Dataset which contains the customer data, provided by the client for customer retention and expanding business. We have to suggest the client, on which customers of particular criteria it should target so that they get enrolled to the services provided.

Why am I using Churning dataset:

1. This is a good project because it is so well understood.
2. Attributes are numeric and categorical so you have to figure out how to load and handle data.
3. It is a Classification problem, allowing you to practice with perhaps an easier type of supervised learning algorithm using deep learning library Keras
4. Using Bokeh for visualization .
5. Creative feature engineering .

PREREQUISITES:

We will require Python libraries like Keras, Pandas, Bokeh, Numpy, Seaborn to be preinstalled. Churn dataset which is present in the given folder structure.



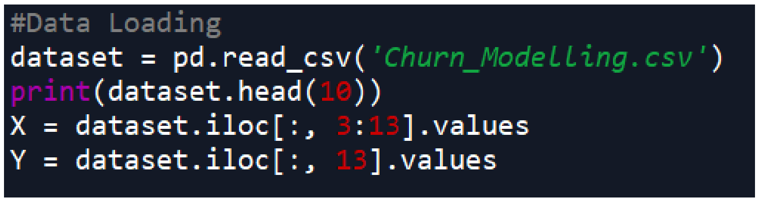
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn

from bokeh.io import output_file, show
from bokeh.models import ColumnDataSource, LinearColorMapper, PrintfTickFormatter, FactorRange
from bokeh.plotting import figure
from bokeh.sampledata.unemployment1948 import data
from bokeh.transform import transform
from bokeh.layouts import gridplot
```

DATA PREPROCESSING:

Data preprocessing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set

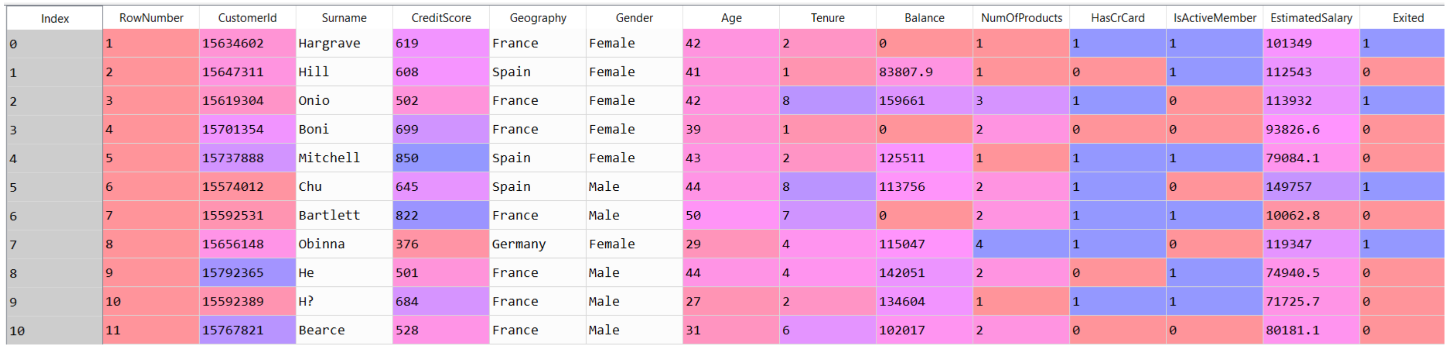
Each row represents a customer, each column contains that customer's attributes:
CustomerId: Unique customer id for each customer.
CreditScore: Number that reflects the likelihood of you payingcredit back.
Geography: Country in which customer resides.
Age: Age of customer.
Tenure: Years from which customer holding of an account.
Balance: Available balance in the account.
NumOfProducts: Count of products enrolled by customer.
HasCrCard: If customer holds the credit card or not**.**
IsActiveMember: If the customer is actively using the credit card**.**
EstimatedSalary: Estimation of salary of the customer.
Exited: If customer still holds the account with the organization.

Loading the csv file in the dataframe and assigning the input features with the desired columns and output feature with last "Exited" column. X is the list of predictors whereas Y is the output variable.



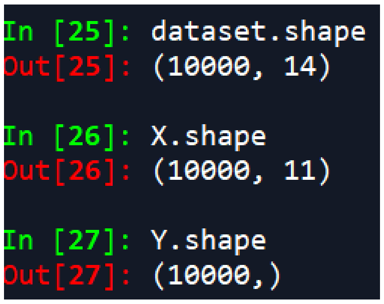
```
#Data Loading
dataset = pd.read_csv('Churn_Modelling.csv')
print(dataset.head(10))
X = dataset.iloc[:, 3:13].values
Y = dataset.iloc[:, 13].values
```

Looking at the dataset:



Index	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101349	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.9	1	0	1	112543	0
2	3	15619304	Onio	502	France	Female	42	8	159661	3	1	0	113932	1
3	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.6	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125511	1	1	1	79084.1	0
5	6	15574012	Chu	645	Spain	Male	44	8	113756	2	1	0	149757	1
6	7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
7	8	15656148	Obinna	376	Germany	Female	29	4	115047	4	1	0	119347	1
8	9	15792365	He	501	France	Male	44	4	142051	2	0	1	74940.5	0
9	10	15592389	H?	684	France	Male	27	2	134604	1	1	1	71725.7	0
10	11	15767821	Bearce	528	France	Male	31	6	102017	2	0	0	80181.1	0

Checking the dimensions of the input dataset, predictors and output variable dataframes.



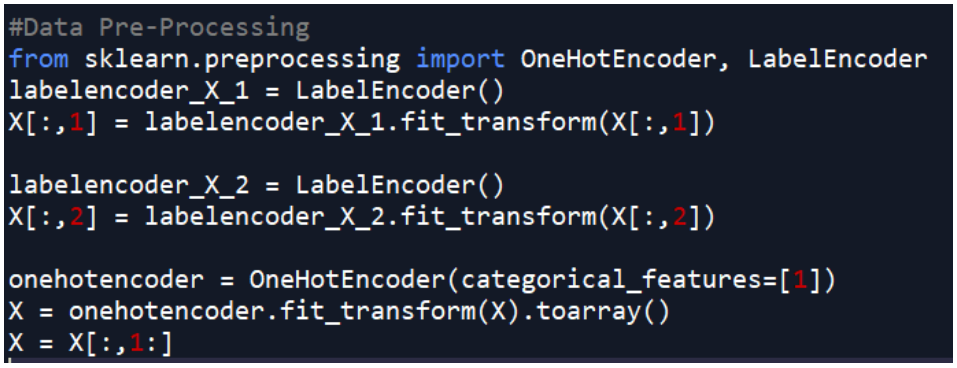
```
In [25]: dataset.shape
Out[25]: (10000, 14)

In [26]: X.shape
Out[26]: (10000, 11)

In [27]: Y.shape
Out[27]: (10000,)
```

Cleaning the data by removing the corrupt data that is removing the null values records and handling missing data by removing those records from the dataset using **dataset.dropna()** command.

Handling Categorical data from the predictors. Look at the variables, we can see that we have some wrangling to do. We will handle such data using **OneHotEncoder library from sklearn.preprocessing** package.

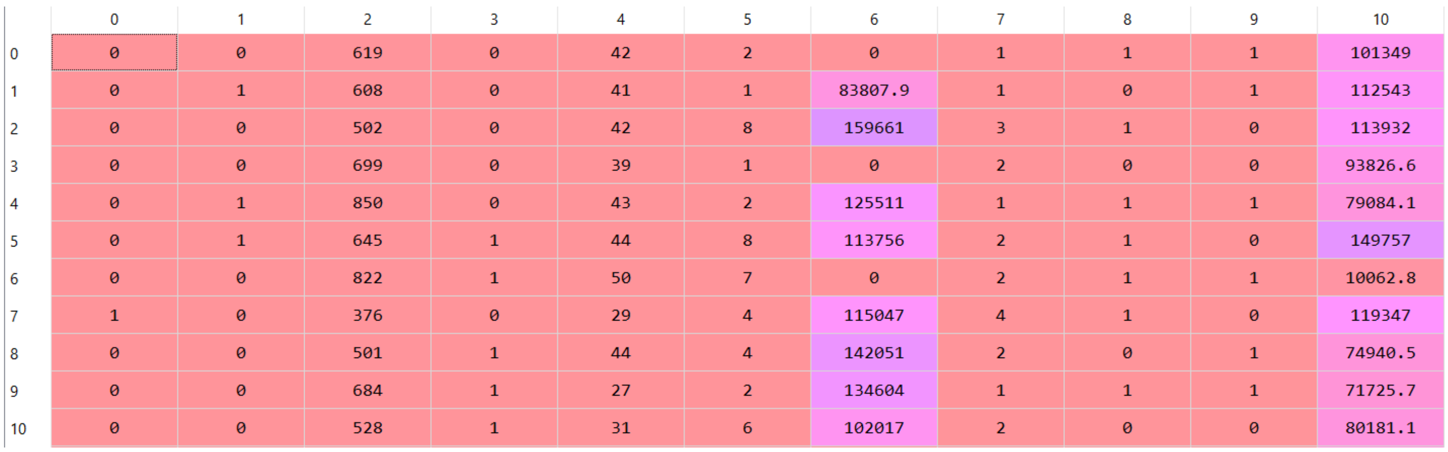
One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

Converting "Geography" and "Gender" columns in "binarization" of the category and include it as a feature to train the model.



```
#Data Pre-Processing
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
labelencoder_X_1 = LabelEncoder()
X[:,1] = labelencoder_X_1.fit_transform(X[:,1])

labelencoder_X_2 = LabelEncoder()
X[:,2] = labelencoder_X_2.fit_transform(X[:,2])

onehotencoder = OneHotEncoder(categorical_features=[1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:,1:]
```

Looking at the input data X:

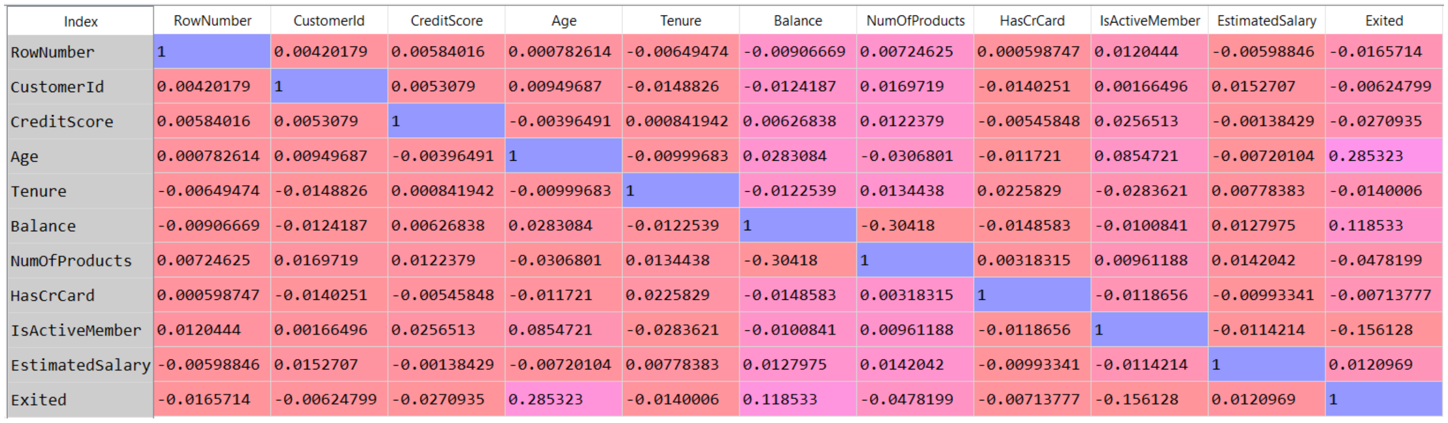


	0	1	2	3	4	5	6	7	8	9	10
0	0	0	619	0	42	2	0	1	1	1	101349
1	0	1	608	0	41	1	83807.9	1	0	1	112543
2	0	0	502	0	42	8	159661	3	1	0	113932
3	0	0	699	0	39	1	0	2	0	0	93826.6
4	0	1	850	0	43	2	125511	1	1	1	79084.1
5	0	1	645	1	44	8	113756	2	1	0	149757
6	0	0	822	1	50	7	0	2	1	1	10062.8
7	1	0	376	0	29	4	115047	4	1	0	119347
8	0	0	501	1	44	4	142051	2	0	1	74940.5
9	0	0	684	1	27	2	134604	1	1	1	71725.7
10	0	0	528	1	31	6	102017	2	0	0	80181.1

Exploratory data analysis and feature selection:

EDA helps us in the better understanding of data and only using that we derive out trends and relationship among variables. That ultimately results in generation and selection of useful features that directly impact the model performance.

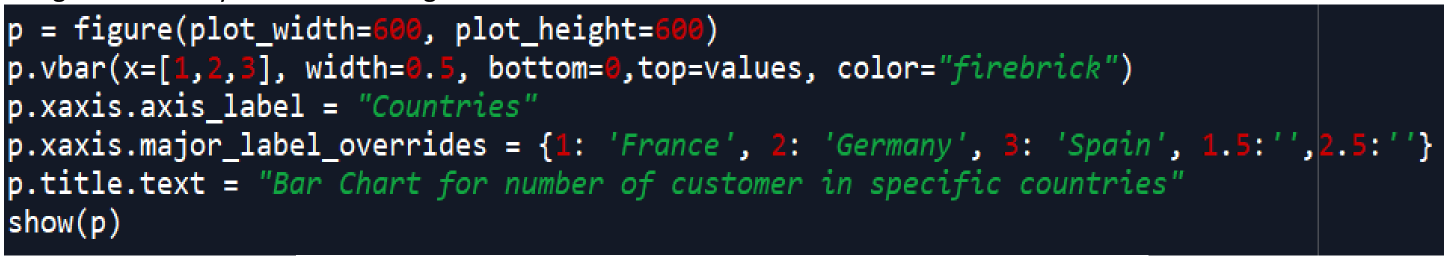
Correlation between the predictors variables:



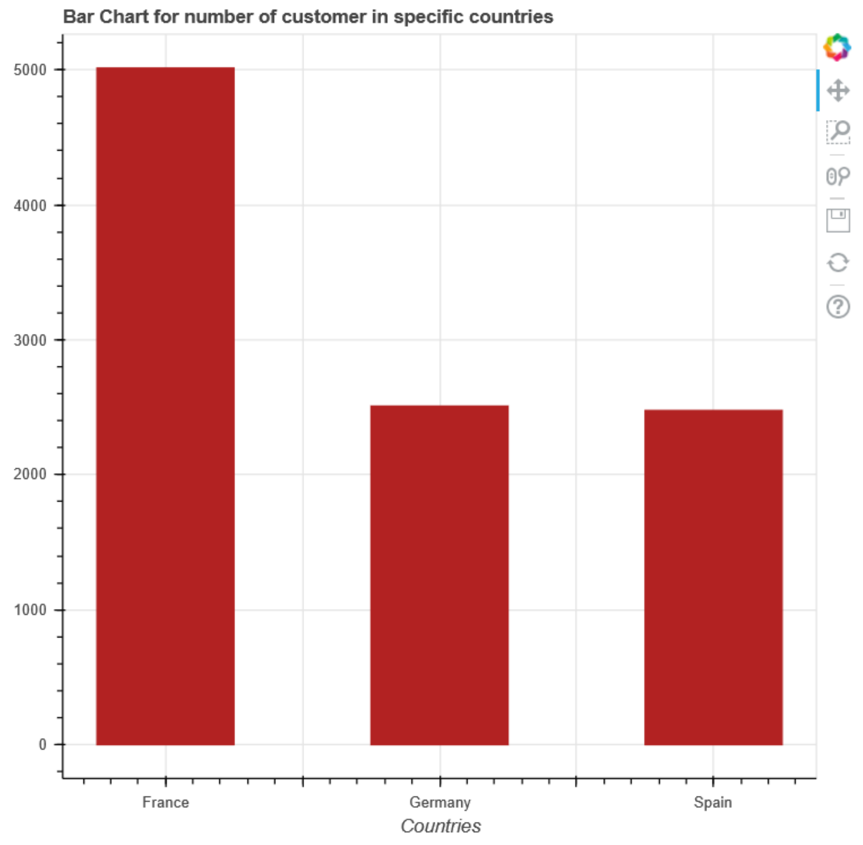
Index	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
RowNumber	1	0.00420179	0.00584016	0.000782614	-0.00649474	-0.00906669	0.00724625	0.000598747	0.0120444	-0.00598846	-0.0165714
CustomerId	0.00420179	1	0.0053079	0.00949687	-0.0148826	-0.0124187	0.0169719	-0.0140251	0.00166496	0.0152707	-0.00624799
CreditScore	0.00584016	0.0053079	1	-0.00396491	0.000841942	0.00626838	0.0122379	-0.00545848	0.0256513	-0.00138429	-0.0270935
Age	0.000782614	0.00949687	-0.00396491	1	-0.00999683	0.0283084	-0.0306801	-0.011721	0.0854721	-0.00720104	0.285323
Tenure	-0.00649474	-0.0148826	0.000841942	-0.00999683	1	-0.0122539	0.0134438	0.0225829	-0.0283621	0.00778383	-0.0140006
Balance	-0.00906669	-0.0124187	0.00626838	0.0283084	-0.0122539	1	-0.30418	-0.0148583	-0.0100841	0.0127975	0.118533
NumOfProducts	0.00724625	0.0169719	0.0122379	-0.0306801	0.0134438	-0.30418	1	0.00318315	0.00961188	0.0142042	-0.0478199
HasCrCard	0.000598747	-0.0140251	-0.00545848	-0.011721	0.0225829	-0.0148583	0.00318315	1	-0.0118656	-0.00993341	-0.00713777
IsActiveMember	0.0120444	0.00166496	0.0256513	0.0854721	-0.0283621	-0.0100841	0.00961188	-0.0118656	1	-0.0114214	-0.156128
EstimatedSalary	-0.00598846	0.0152707	-0.00138429	-0.00720104	0.00778383	0.0127975	0.0142042	-0.00993341	-0.0114214	1	0.0120969
Exited	-0.0165714	-0.00624799	-0.0270935	0.285323	-0.0140006	0.118533	-0.0478199	-0.00713777	-0.156128	0.0120969	1

Data visualization is the presentation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns.

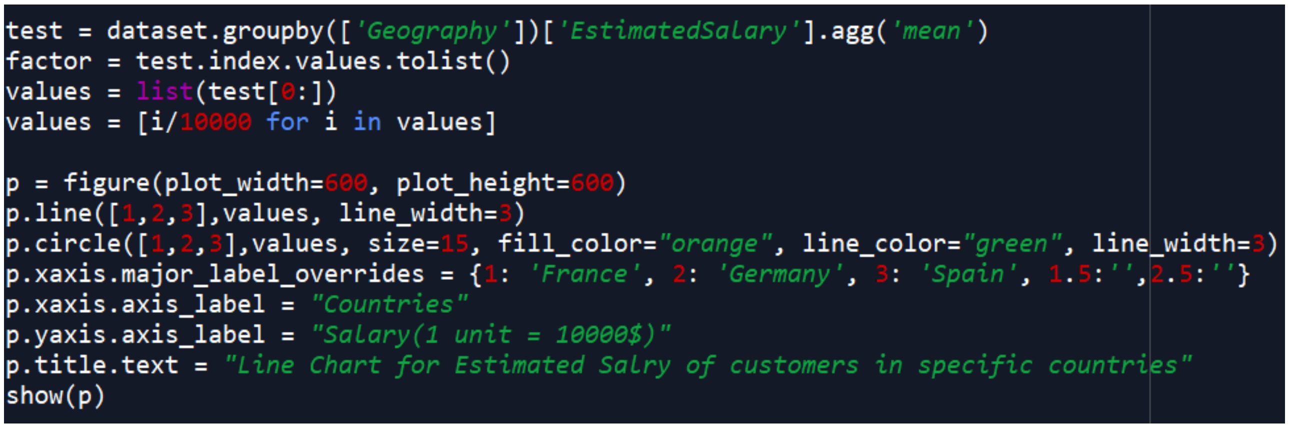
Bar plots of categorical variable of “Geography” to find out the count of customers in each country:
Using bokeh library to visualize the given feature:



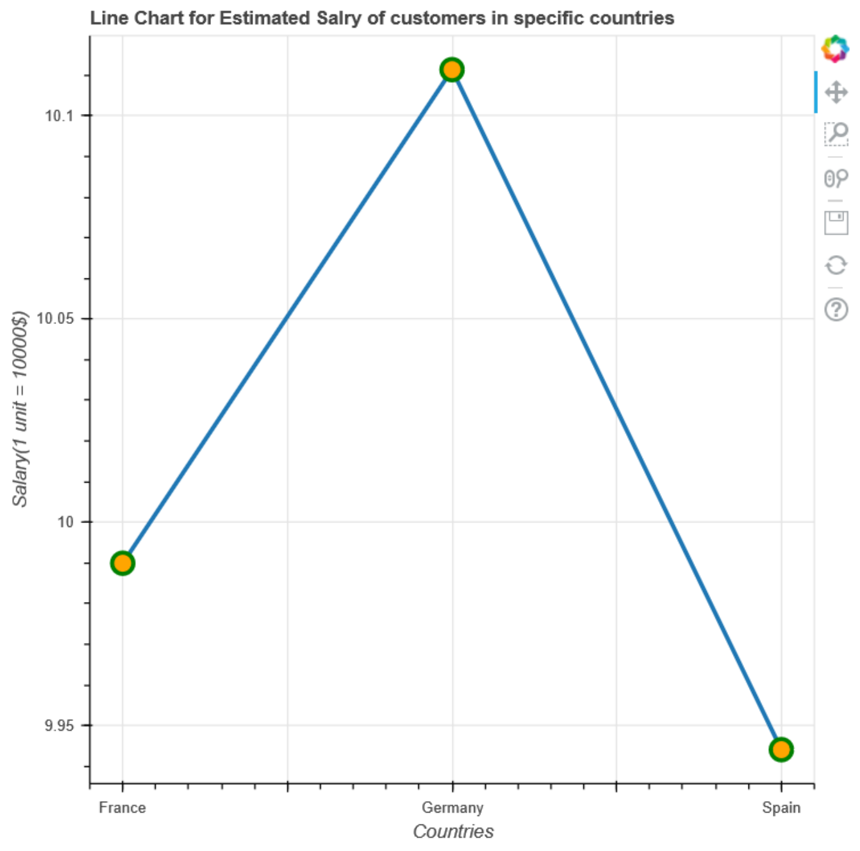
```
p = figure(plot_width=600, plot_height=600)
p.vbar(x=[1,2,3], width=0.5, bottom=0,top=values, color="firebrick")
p.xaxis.axis_label = "Countries"
p.xaxis.major_label_overrides = {1: 'France', 2: 'Germany', 3: 'Spain', 1.5:'',2.5:''}
p.title.text = "Bar Chart for number of customer in specific countries"
show(p)
```



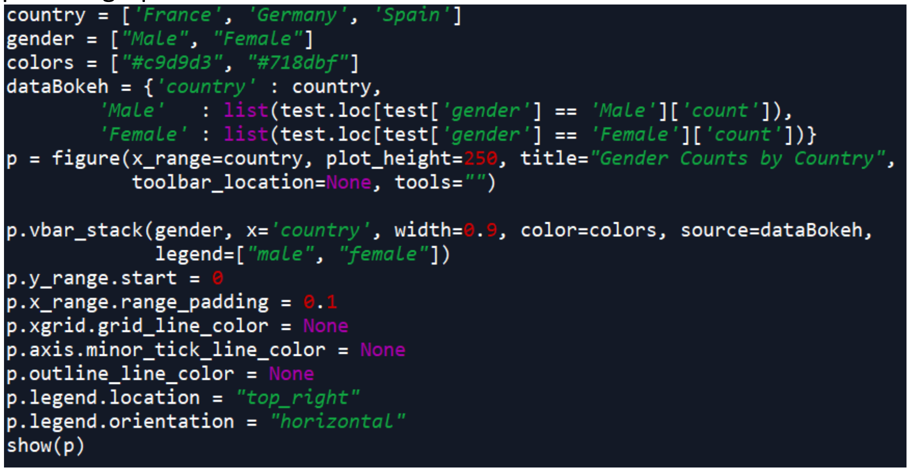
Combination of line chart and scatter plot of categorical variable of “Geography” to find out the estimated salary of customers each country:
Using bokeh library to visualize the given feature:



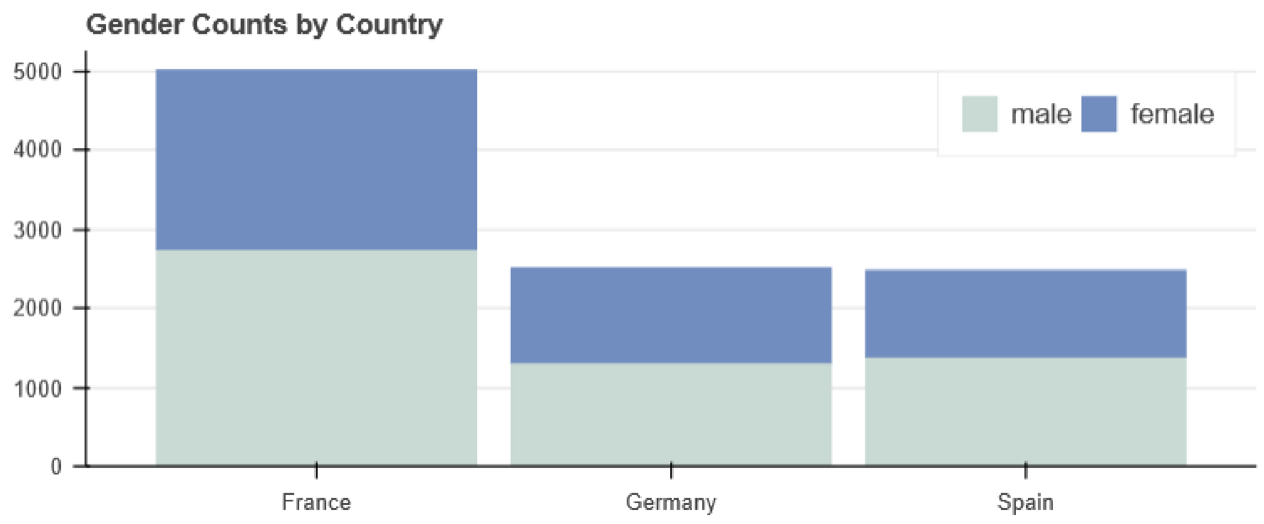
```
test = dataset.groupby(['Geography'])['EstimatedSalary'].agg('mean')
factor = test.index.values.tolist()
values = list(test[0:])
values = [i/10000 for i in values]

p = figure(plot_width=600, plot_height=600)
p.line([1,2,3],values, line_width=3)
p.circle([1,2,3],values, size=15, fill_color="orange", line_color="green", line_width=3)
p.xaxis.major_label_overrides = {1: 'France', 2: 'Germany', 3: 'Spain', 1.5:'',2.5:''}
p.xaxis.axis_label = "Countries"
p.yaxis.axis_label = "Salary(1 unit = 10000$)"
p.title.text = "Line Chart for Estimated Salry of customers in specific countries"
show(p)
```



Using stacked bar chart to find the gender count in the countries:
Using Bokeh library to plot the graph:

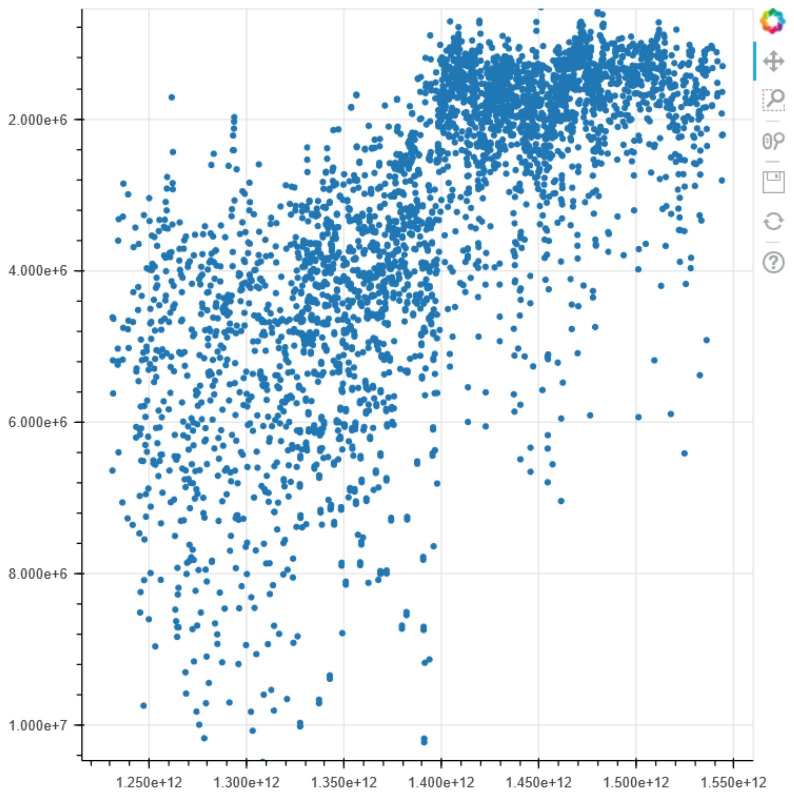


```
country = ['France', 'Germany', 'Spain']
gender = ["Male", "Female"]
colors = ["#c9d9d3", "#718dbf"]
dataBokeh = {'country' : country,
        'Male'   : list(test.loc[test['gender'] == 'Male']['count']),
        'Female' : list(test.loc[test['gender'] == 'Female']['count'])}
p = figure(x_range=country, plot_height=250, title="Gender Counts by Country",
           toolbar_location=None, tools="")

p.vbar_stack(gender, x='country', width=0.9, color=colors, source=dataBokeh,
             legend=["male", "female"])
p.y_range.start = 0
p.x_range.range_padding = 0.1
p.xgrid.grid_line_color = None
p.axis.minor_tick_line_color = None
p.outline_line_color = None
p.legend.location = "top_right"
p.legend.orientation = "horizontal"
show(p)
```



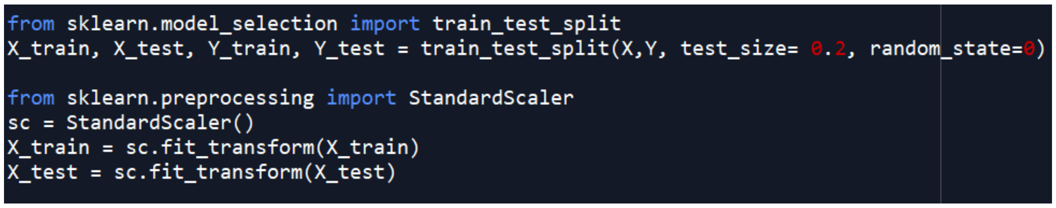
Using scatter chart to visualize age vs estimated salary of all customers in the organisation :
Using Bokeh library to plot the graph:

```
source = ColumnDataSource(dataset_train)
plot = figure()
plot.circle(x="Date",y="Volume",source = source)
show(plot)

plot = figure(x_axis_label = "x",y_axis_label = "y",tools = "pan,box_zoom")
plot.circle(x=dataset_train[0],y=dataset_train[3],size = 10,color = "black",alpha = 0.7)
plot.yaxis.axis_label = 'Age'
plot.xaxis.axis_label = 'Salary'
plot.title.text = "Chart for Age vs Salary"
show(plot)
```



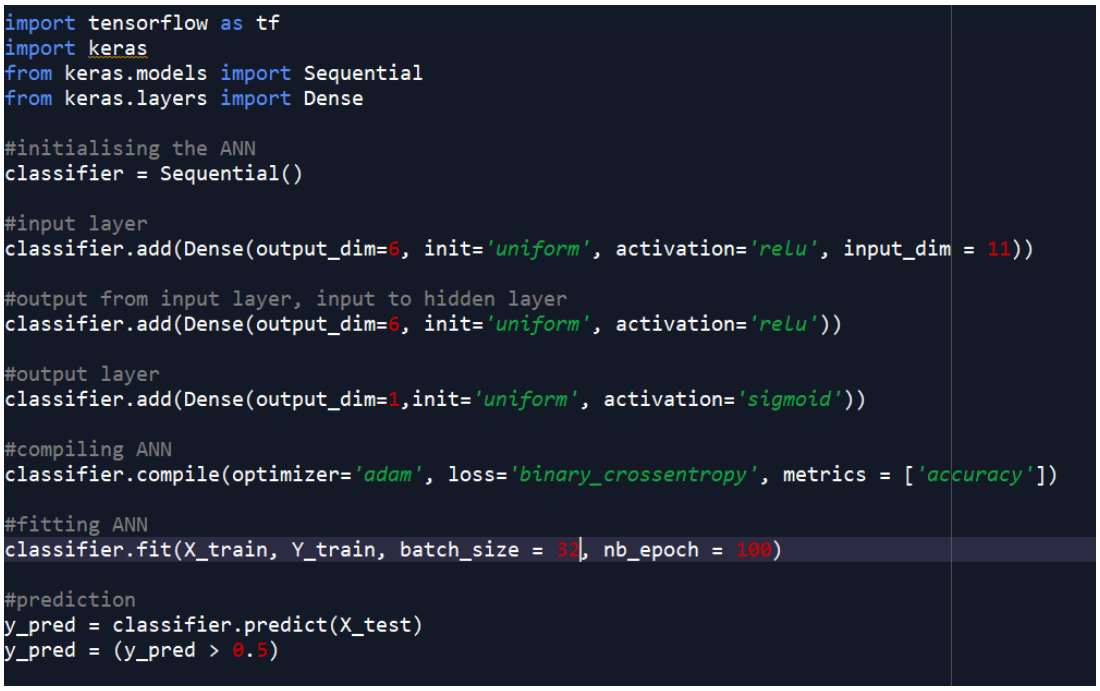
CLASSIFICATION USING KERAS:

Splitting the data in training and testing parts so that we can make the model learn on training dataset and test the accuracy on testing set. Using **sklearn.model_selection** package.



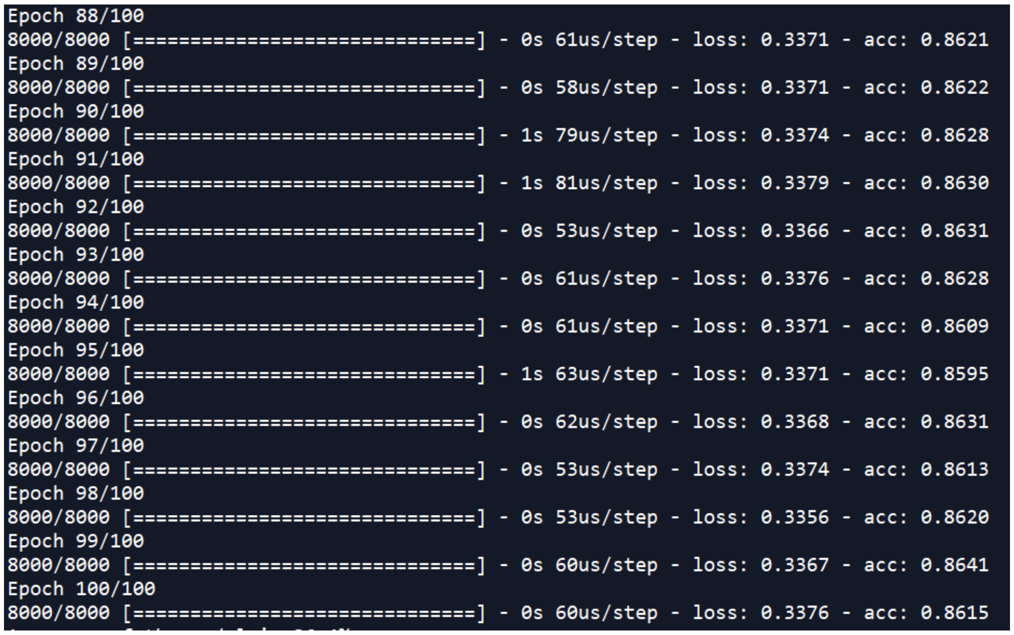
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size= 0.2, random_state=0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Using keras library with backend Tensorflow for training the model.



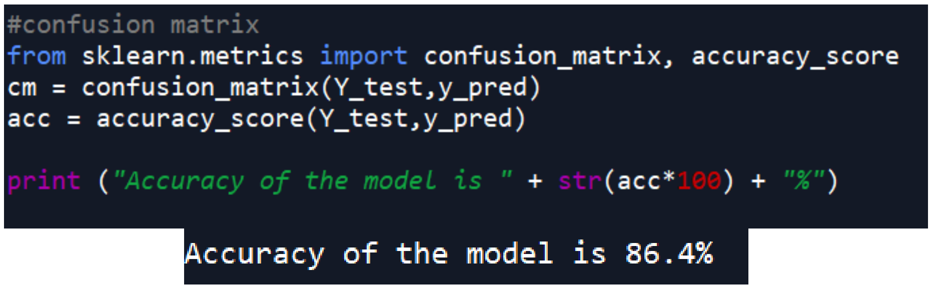
```
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense

#initialising the ANN
classifier = Sequential()

#input layer
classifier.add(Dense(output_dim=6, init='uniform', activation='relu', input_dim = 11))

#output from input layer, input to hidden layer
classifier.add(Dense(output_dim=6, init='uniform', activation='relu'))

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

#compiling ANN
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics = ['accuracy'])

#fitting ANN
classifier.fit(X_train, Y_train, batch_size = 32, nb_epoch = 100)

#prediction
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

Running for 100 Epochs:



```
Epoch 88/100
8000/8000 [==============================] - 0s 61us/step - loss: 0.3371 - acc: 0.8621
Epoch 89/100
8000/8000 [==============================] - 0s 58us/step - loss: 0.3371 - acc: 0.8622
Epoch 90/100
8000/8000 [==============================] - 1s 79us/step - loss: 0.3374 - acc: 0.8628
Epoch 91/100
8000/8000 [==============================] - 1s 81us/step - loss: 0.3379 - acc: 0.8630
Epoch 92/100
8000/8000 [==============================] - 0s 53us/step - loss: 0.3366 - acc: 0.8631
Epoch 93/100
8000/8000 [==============================] - 0s 61us/step - loss: 0.3376 - acc: 0.8628
Epoch 94/100
8000/8000 [==============================] - 0s 61us/step - loss: 0.3371 - acc: 0.8609
Epoch 95/100
8000/8000 [==============================] - 1s 63us/step - loss: 0.3371 - acc: 0.8595
Epoch 96/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.3368 - acc: 0.8631
Epoch 97/100
8000/8000 [==============================] - 0s 53us/step - loss: 0.3374 - acc: 0.8613
Epoch 98/100
8000/8000 [==============================] - 0s 53us/step - loss: 0.3356 - acc: 0.8620
Epoch 99/100
8000/8000 [==============================] - 0s 60us/step - loss: 0.3367 - acc: 0.8641
Epoch 100/100
8000/8000 [==============================] - 0s 60us/step - loss: 0.3376 - acc: 0.8615
```

Calculating accuracy of the model:



```
#confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Y_test,y_pred)
acc = accuracy_score(Y_test,y_pred)

print ("Accuracy of the model is " + str(acc*100) + "%")
```

```
Accuracy of the model is 86.4%
```

CONCLUSION:

So we can handover this model to the client. Where client can insert the new or targeted customer details and get the prediction if the customer is willing to take the product and continue with the organisation. Also, client can use this model on the specific targeted crowd to increase there overall sales of the product and maintain the good customer retaintion pattern.