In [1]:

```
import numpy as np
import nltk
import json
import random
```

In [2]:

```
with open("intents.json") as file:
    data = json.load(file)
data
```

Out[2]:

```
{'intents': [{'tag': 'greeting',
   'patterns': ['Hi', 'Hello', 'Is anyone there?', 'Hey', 'Good day'],
   'responses': ['Hi, Welcome to Investment Advisor. How can I help?',
    'Hello, nice to see you. Welcome to Investment Advisor. How can I hel
p?']},
  {'tag': 'information',
   'patterns': ['I am looking for investment options',
    'Tell me investment options',
    'Investment plans',
    'What are the plans',
    'Options for money'],
   'responses': ['There are 2 plans available right now: Long Term ABC pla
n and Short term XYZ plan',
    'We have two options available: Long Term ABC plan and Short term XYZ
plan']},
  {'tag': 'long',
   'patterns': ['tell me about long term plan',
    'Long term plan',
    'What is the long term plan?',
    'I want to invest long term',
    'minimum risk investment'],
   'responses': ['Long term plan is essentially a 20% boost in your invest
ed money per year with minimum risk. This is valid till 5 years.',
    'Minimum risk plan which gives 20% boost to your money with minimum ri
sk.']},
  {'tag': 'short',
   'patterns': ['tell me about short term plan',
    'Short term plan',
    'What is the short term plan?',
    'I want to invest short term',
    'quick investment'],
   'responses': ['Short term plan is essentially a 10% boost in your inves
ted money per month. This is valid for 6 months. Market risks are involve
d']},
  {'tag': 'goodbye',
   'patterns': ['Bye', 'See you later', 'Goodbye'],
   'responses': ['See you later, thanks for visiting',
    'Have a nice day',
    'Bye! Come back again soon.']},
  {'tag': 'opentoday',
   'patterns': ['Are you open today?',
    'When do you open today?',
    'What are your hours today?'],
   'responses': ['Our hours are 9am-9pm every day',
    'We are open from 9 am to 9pm every day of the week!']}]}
```

In [3]:

```
#Defining stemmer
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

#Defining List
words = [] #store all the words
labels = [] #store all the tags
doc_x = [] #for storing x of training data
doc_y = [] #for storing y of training data (class i.e tag to which it belongs)



for intent in data["intents"]:
    for pattern in intent["patterns"]:

        wrds = nltk.word_tokenize(pattern)

        for index, w in enumerate(wrds):
            if w != "?":
                w = stemmer.stem(w.lower())
                wrds[index] = w

        words.extend(wrds)
        doc_x.append(wrds)
        doc_y.append(intent["tag"])

    if intent["tag"] not in labels:
        labels.append(intent["tag"])

print("Words:- "+str(words)+"\n")
print("labels:- "+str(labels)+"\n")
print("doc_x:- "+str(doc_x)+"\n")
print("doc_y:- "+str(doc_y)+"\n")
```

```
Words:- ['hi', 'hello', 'is', 'anyon', 'ther', '?', 'hey', 'good', 'day',
'i', 'am', 'look', 'for', 'invest', 'opt', 'tel', 'me', 'invest', 'opt',
'invest', 'plan', 'what', 'ar', 'the', 'plan', 'opt', 'for', 'money', 'te
l', 'me', 'about', 'long', 'term', 'plan', 'long', 'term', 'plan', 'what',
'is', 'the', 'long', 'term', 'plan', '?', 'i', 'want', 'to', 'invest', 'lo
ng', 'term', 'minim', 'risk', 'invest', 'tel', 'me', 'about', 'short', 'te
rm', 'plan', 'short', 'term', 'plan', 'what', 'is', 'the', 'short', 'ter
m', 'plan', '?', 'i', 'want', 'to', 'invest', 'short', 'term', 'quick', 'i
nvest', 'bye', 'see', 'you', 'lat', 'goodby', 'ar', 'you', 'op', 'today',
'?', 'when', 'do', 'you', 'op', 'today', '?', 'what', 'ar', 'yo', 'hour',
'today', '?']

labels:- ['greeting', 'information', 'long', 'short', 'goodbye', 'opentoda
y']

doc_x:- [['hi'], ['hello'], ['is', 'anyon', 'ther', '?'], ['hey'], ['goo
d', 'day'], ['i', 'am', 'look', 'for', 'invest', 'opt'], ['tel', 'me', 'in
vest', 'opt'], ['invest', 'plan'], ['what', 'ar', 'the', 'plan'], ['opt',
'for', 'money'], ['tel', 'me', 'about', 'long', 'term', 'plan'], ['long',
'term', 'plan'], ['what', 'is', 'the', 'long', 'term', 'plan', '?'], ['i',
'want', 'to', 'invest', 'long', 'term'], ['minim', 'risk', 'invest'], ['te
l', 'me', 'about', 'short', 'term', 'plan'], ['short', 'term', 'plan'],
['what', 'is', 'the', 'short', 'term', 'plan', '?'], ['i', 'want', 'to',
'invest', 'short', 'term'], ['quick', 'invest'], ['bye'], ['see', 'you',
'lat'], ['goodby'], ['ar', 'you', 'op', 'today', '?'], ['when', 'do', 'yo
u', 'op', 'today', '?'], ['what', 'ar', 'yo', 'hour', 'today', '?']]

doc_y:- ['greeting', 'greeting', 'greeting', 'greeting', 'greeting', 'info
rmation', 'information', 'information', 'information', 'information', 'lon
g', 'long', 'long', 'long', 'long', 'short', 'short', 'short', 'short', 's
hort', 'goodbye', 'goodbye', 'goodbye', 'opentoday', 'opentoday', 'opentod
ay']
```

In [4]:

```python
#Considering only unique words and sorting the words and labels list
words = sorted(list(set(words)))
labels = sorted(labels)

print("Words:- "+str(words)+"\n")
print("labels:- "+str(labels)+"\n")
```

```
Words:- ['?', 'about', 'am', 'anyon', 'ar', 'bye', 'day', 'do', 'for', 'go
od', 'goodby', 'hello', 'hey', 'hi', 'hour', 'i', 'invest', 'is', 'lat',
'long', 'look', 'me', 'minim', 'money', 'op', 'opt', 'plan', 'quick', 'ris
k', 'see', 'short', 'tel', 'term', 'the', 'ther', 'to', 'today', 'want',
'what', 'when', 'yo', 'you']

labels:- ['goodbye', 'greeting', 'information', 'long', 'opentoday', 'shor
t']
```

In [5]:

```
#Now to provide input to our model, we will use one hot encoding

x_train = []
y_train = []

output_empty = []  #temporary row for temporary use

for i in range(0, len(labels)):
    output_empty.append(0)


for index, x in enumerate(doc_x):
    bag = []

    for word in words:
        if word in x:
            bag.append(1)
        else:
            bag.append(0)

    x_train.append(bag)

    output_row = output_empty[:]
    output_row[labels.index(doc_y[index])] = 1
    y_train.append(output_row)


for index, x in enumerate(x_train):
    print(str(x)+"\t"+str(y_train[index]))
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 0]
[1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0]   [0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 1, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 1, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0]   [0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0]   [0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 1, 0, 0]
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 1]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0]   [0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]   [1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]   [1, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1]   [0, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1]   [0, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0]   [0, 0, 0, 0, 1, 0]
```

In [6]:

```
#Converting them to numpy array
x_train = np.array(x_train)
y_train = np.array(y_train)
```

In [7]:

```
#Training our neural network
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(hidden_layer_sizes=(len(x_train[0]),len(x_train[0])),  max_iter=5
00)
model.fit(x_train, y_train)
```

Out[7]:

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=
0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(42, 42), learning_rate='constant',
              learning_rate_init=0.001, max_iter=500, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=None, shuffle=True, solver='adam', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

In [8]:

```
#Client Side

def convert_user_input(user_input, words):

    nltk_user_input = nltk.word_tokenize(user_input)

    for index, word in enumerate(nltk_user_input):
        nltk_user_input[index] = stemmer.stem(word)

    bag = []

    for w in words:
        if w in nltk_user_input:
            bag.append(1)
        else:
            bag.append(0)

    bag = np.array(bag)

    return bag

def output(result):
    flag = False

    for index, value in enumerate(result):
        if value == 1:
            flag = True
            break

    if flag == True:
        result_index = index

        result_tag = labels[result_index]

        for intent in data["intents"]:
            if intent["tag"] == result_tag:
                responses = intent["responses"]
                break

        return random.choice(responses)
    else:
        return "Sorry, didn't undestand what you were saying. Please try again"





def chat():
    print("Chat with our bot:- (Enter quit to exit) \n")
    while True:
        user_input = input("You:- ")

        user_input = user_input.lower()

        if user_input == "quit":
            break

        converted_user_input = convert_user_input(user_input, words)
        result = model.predict([converted_user_input])
        bot_response = output(result[0])
        print("\nBot:- "+bot_response+"\n")
```

In [10]:

```
#Speaking with Chatbot
chat()
```

```
Chat with our bot:- (Enter quit to exit)

You:- hi

Bot:- Hello, nice to see you. Welcome to Investment Advisor. How can I hel
p?

You:- i am looking for investment plan

Bot:- There are 2 plans available right now: Long Term ABC plan and Short
term XYZ plan

You:- Tell me about long term plab

Bot:- Minimum risk plan which gives 20% boost to your money with minimum r
isk.

You:- what about the short term plan

Bot:- Short term plan is essentially a 10% boost in your invested money pe
r month. This is valid for 6 months. Market risks are involved

You:- see you later

Bot:- See you later, thanks for visiting

You:- quit
```

In [ ]:

```

```