

# CREATE A CHATBOT IN PYTHON

## TEAM MEMBER

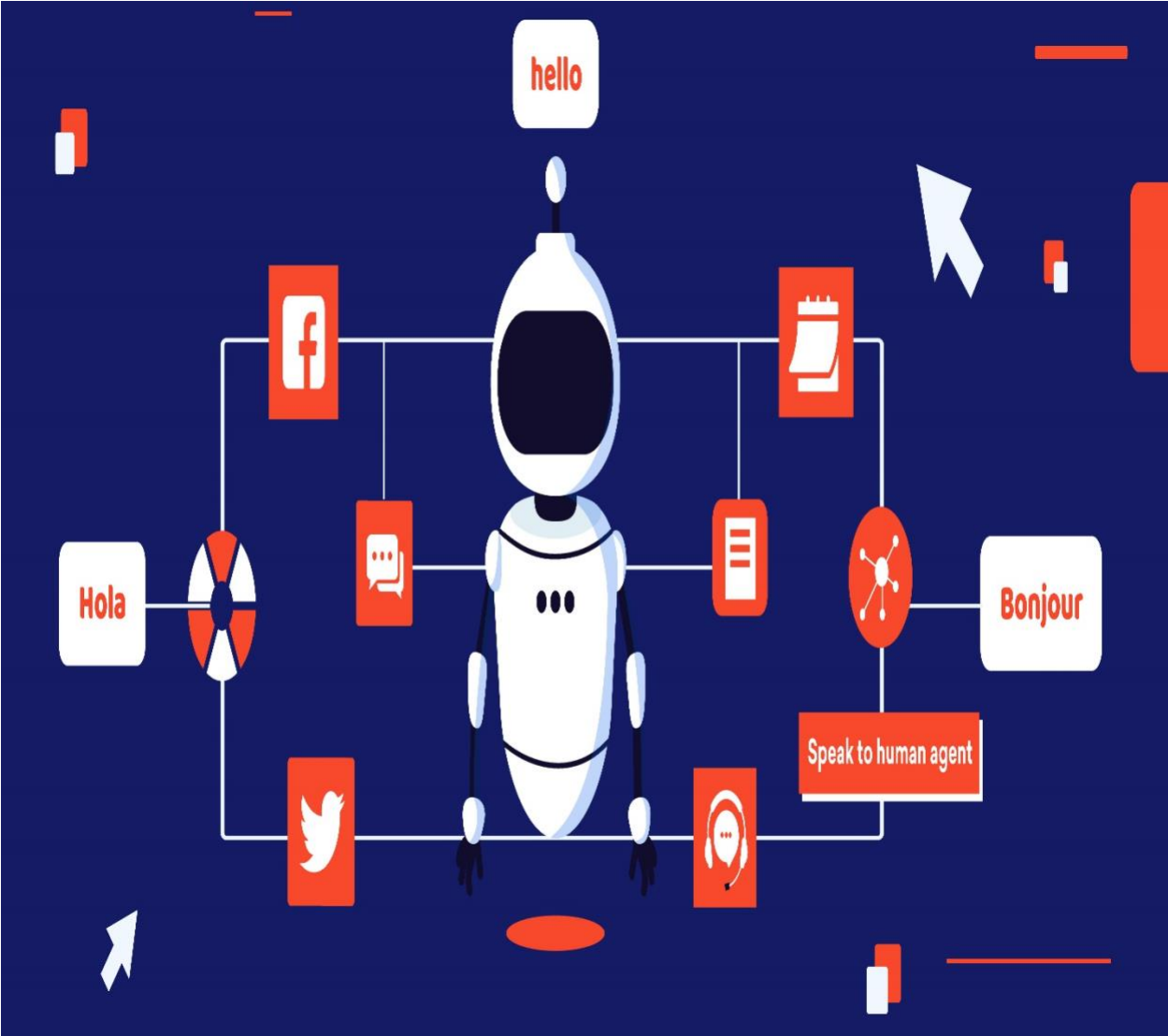
NAME : Tharuna Selvi R

PHASE-2: Innovation

### SYNOPSIS:

- Overview
- Innovation techniques
  1. NLTK
  2. Ensemble learning
  3. Deep learning
- Visualizing Accuracy
- Pre-Trained Language Models (Gpt-3)
- Conclusion





## **OVERVIEW:**

Chatbots are developed using innovative techniques like ensemble learning, deep learning, and natural language processing (NLP). Ensemble learning enhances accuracy, deep learning enables complex user queries, and NLP enhances human language understanding, making interactions more engaging.

## **INNOVATION TECHNIQUES:**

Achieving robustness and accuracy in AI systems is a complex and ongoing challenge. There are several innovative techniques exploring to enhance the robustness and accuracy of AI systems. Here are some of them: Deep Learning, Ensemble Learning, Reinforcement Learning, and NLTK (Natural Language Toolkit)

### **1. NLTK:**

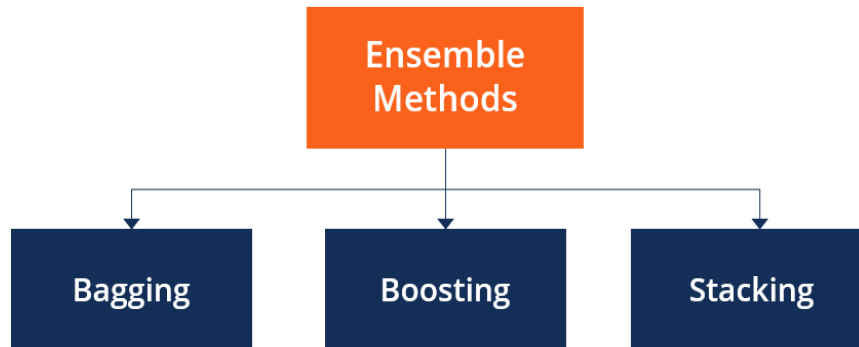
NLTK provides a wide range of tools, resources, and libraries for various NLP tasks, such as text processing, tokenization, stemming, tagging, parsing, and sentiment analysis. NLTK provides a set of tools and resources for text processing and analysis, making it for creating chatbots and improving the accuracy of prediction systems.

## **IMPLEMENTATION:**

```
import nltk
import random
from sklearn.feature_extraction.text import TfidfVectorizer
nltk.download("punkt")
def tokenizer(text):
    text = text.lower()
    tokens = word_tokenize(text)
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    tokens = [token for token in tokens if token not in stop_words]
    return token
```

## 2. ENSEMBLE LEARNING:

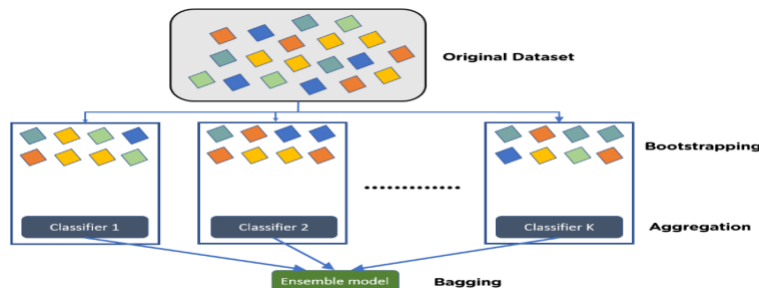
Ensemble learning is a technique in machine learning where multiple models are combined to improve predictive performance.



In this project, I used bagging method to achieve prediction system accuracy and robustness.

### **BAGGING:**

Bagging (Bootstrap Aggregating) is an ensemble machine learning technique designed to improve the performance and robustness of predictive models by creating multiple subsets of training data, training individual models, and combining their predictions.

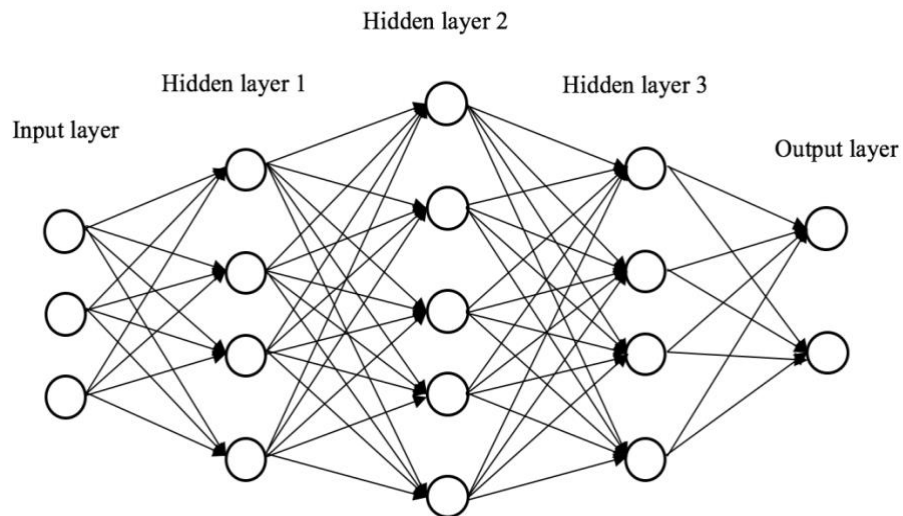


### **IMPLEMENTATION:**

```
def bow(sentence, words, show_details=True):
    sentence_words = clean_up_sentence(sentence)
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                bag[i] = 1
            if show_details:
                print ("found in bag: %s" % w)
    return(np.array(bag))
```

### **3.DEEP LEARNING:**

Deep Learning is a subfield of machine learning and artificial intelligence (AI) that focuses on the development of neural networks, particularly deep neural networks, inspired by the structure and function of the human brain.



### **IMPLEMENTATION:**

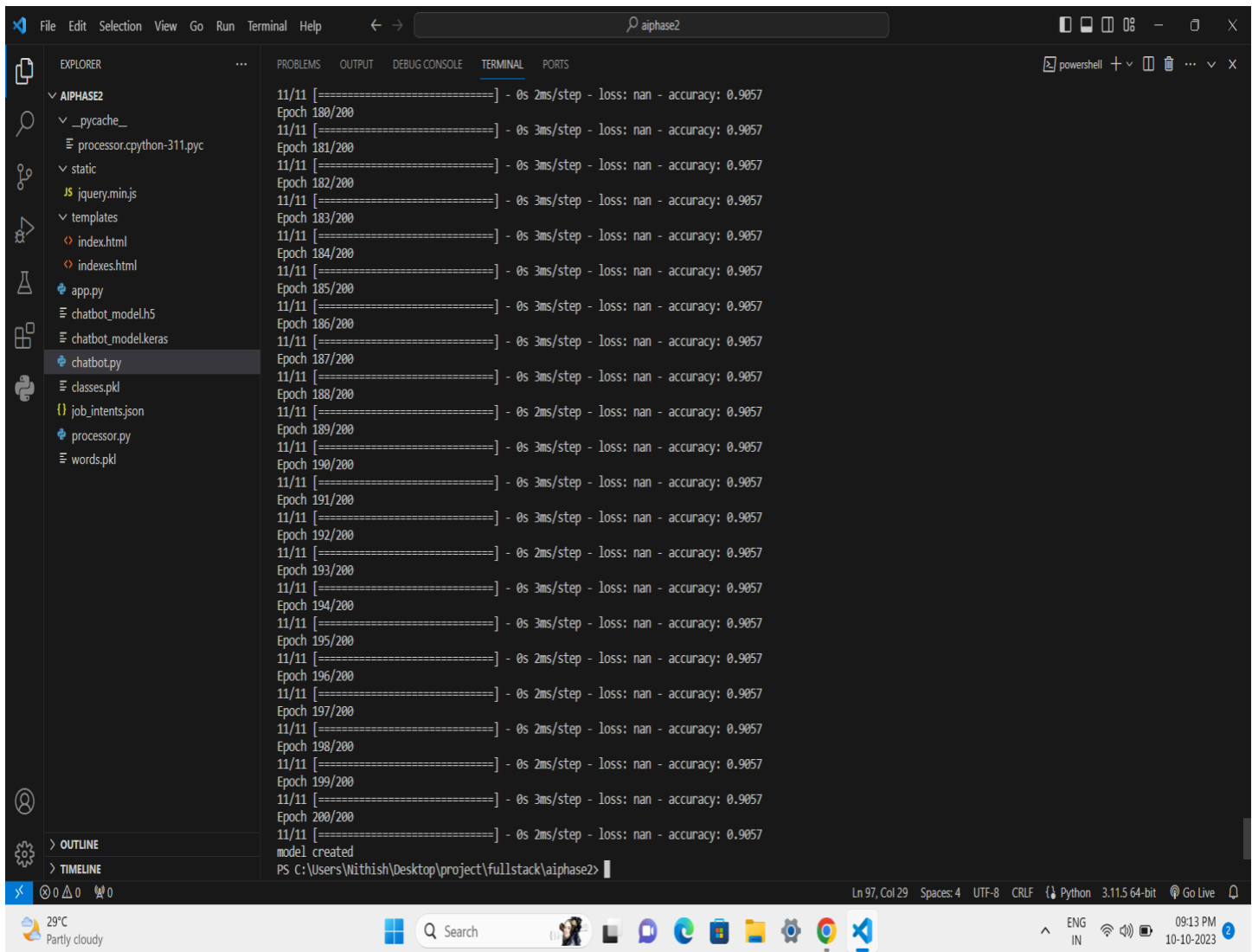
```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random
import tensorflow as tf
words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('job_intents.json', encoding='utf-8').read()
intents = json.loads(data_file)
for intent in intents['intents']:
    for pattern in intent['patterns']:
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        documents.append((w, intent['tag']))
    if intent['tag'] not in classes:
        classes.append(intent['tag'])
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
classes = sorted(list(set(classes)))
print (len(documents), "documents")
```

```

print (len(classes), "classes", classes)
print (len(words), "unique lemmatized words", words)


pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
training = []
output_empty = [0] * len(classes)
for doc in documents:
    bag = []
    pattern_words = doc[0]
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1
    training.append([bag, output_row])
    random.shuffle(training)
    max_length = max(len(item[0]) for item in training)
    training_padded = np.array([item[0] + [0] * (max_length - len(item[0])) +
item[1] for item in training])
    training = np.array(training_padded)
    train_x = list(training[:, :-1])
    train_y = list(training[:, -1:])
    print("Training data created")
    model = Sequential()
    model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(len(train_y[0]), activation='softmax'))
    sgd = tf.keras.optimizers.legacy.SGD(lr=0.01, decay=1e-6, momentum=0.9,
nesterov=True)
    model.compile(loss='categorical_crossentropy', optimizer=sgd,
metrics=['accuracy'])
    hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5,
verbose=1)
    model.save('chatbot_model.h5', hist)
print("model created")

```



```
File Edit Selection View Go Run Terminal Help
aiphase2
powerShell + - X

EXPLORER
AIPHASE2
  _pycache_
    processor.python-311.pyc
  static
  jquery.min.js
  templates
    index.html
    indexes.html
  app.py
  chatbot_model.h5
  chatbot_model.keras
  chatbot.py
  classes.pkl
  job_intents.json
  processor.py
  words.pkl

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
11/11 [=====] - 0s 2ms/step - loss: nan - accuracy: 0.9057
Epoch 180/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 181/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 182/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 183/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 184/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 185/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 186/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 187/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 188/200
11/11 [=====] - 0s 2ms/step - loss: nan - accuracy: 0.9057
Epoch 189/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 190/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 191/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 192/200
11/11 [=====] - 0s 2ms/step - loss: nan - accuracy: 0.9057
Epoch 193/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 194/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 195/200
11/11 [=====] - 0s 2ms/step - loss: nan - accuracy: 0.9057
Epoch 196/200
11/11 [=====] - 0s 2ms/step - loss: nan - accuracy: 0.9057
Epoch 197/200
11/11 [=====] - 0s 2ms/step - loss: nan - accuracy: 0.9057
Epoch 198/200
11/11 [=====] - 0s 2ms/step - loss: nan - accuracy: 0.9057
Epoch 199/200
11/11 [=====] - 0s 3ms/step - loss: nan - accuracy: 0.9057
Epoch 200/200
11/11 [=====] - 0s 2ms/step - loss: nan - accuracy: 0.9057
model created
PS C:\Users\Nithish\Desktop\project\fullstack\ai\phase2>
```

**VISUALIZATING ACCURACY:**

```
fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5))
ax[0].plot(history.history['loss'],label='loss',c='red')
ax[0].plot(history.history['val_loss'],label='val_loss',c='blue')
ax[0].set_xlabel('Epochs')
ax[1].set_xlabel('Epochs')
ax[0].set_ylabel('Loss')
```

plt.show()

The image shows a Windows 10 desktop with a VS Code editor open. The editor has a dark theme. On the left is the Explorer sidebar showing a file tree for a project named 'aiPhase2'. The files include '\_pycache\_', 'processor.cpython-311.pyc', 'static', 'jquery.min.js', 'templates', 'index.html', 'indexes.html', 'app.py', 'chatbot\_model.h5', 'chatbot\_model.keras', 'chatbot.py', 'classes.pkl', 'job\_intents.json', 'processor.py', and 'words.pkl'. The 'chatbot.py' file is selected. On the right is the Terminal window, which shows the output of a training process. The output consists of 20 lines, each representing an epoch from 11/11 to 11/11. Each line shows the progress bar, the step number (e.g., 180/200), the loss (nan), and the accuracy (0.9057). The terminal also shows the command 'model created' and the current directory 'PS C:\Users\Nithish\Desktop\project\fullstack\aiPhase2>'. The Windows taskbar at the bottom shows the Start button, a search bar, and several pinned applications including Edge, VS Code, and File Explorer. The system tray on the right shows the date and time as '09:13 PM 10-10-2023'.



## PRE-TRAINED LANGUAGE MODELS (GPT-3)

GPT-3 can be effectively utilized to enhance the quality of responses in a chatbot. By integrating GPT-3 into a chatbot system, you can take advantage of its natural language understanding and generation capabilities.

The screenshot shows a VS Code editor with a project named 'aiphase2'. The Explorer pane on the left lists files: index.html, app.py, chatbot1.py (selected), job\_intents.json, classes.pkl, processor.cpython-311.pyc, static, templates, chatbot\_model.h5, chatbot\_model.keras, chatbot.py, and chatbot1.py. The chatbot1.py file is open in the editor, showing a chatbot interface with a text input and a 'Send' button. The chat history shows a conversation where the user asks 'hi' and the chatbot responds 'i'm fine. how about yourself?'. The chatbot's internal logic is visible in the code, showing a simple stateful model using Keras and TensorFlow. The chatbot's state is saved to a file named 'chatbot\_model.h5'.

## CONCLUSION:

The integration of NLP, Deep Learning, and Ensemble Learning has significantly improved chatbot development, enhancing their comprehension, response accuracy, and performance, leading to potential applications in various fields.

