1. **write the step to builing classifier in python:**
2. **Explain what is search algorithm,compinational search,minmax algorithm:**
3. **write the steps to building bots to play games:**

Steps for Building a Classifier in Python

Step 1: Import Scikit-learn: This would be very first step for building a classifier in Python. In this step, we will install a Python package called Scikit-learn which is one of the best machine learning modules in Python.
The following command will help us import the package: Import Sklearn

Step 2: Import Scikit-learn's dataset: In this step, we can begin working with the dataset for our machine learning model. Here, we are going to use the Breast Cancer Wisconsin Diagnostic Database. The dataset includes various information about breast cancer tumors, as well as classification labels of malignant or benign. The dataset has 569 instances, or data, on 569 tumors and includes information on 30 attributes, or features, such as the radius of the tumor, texture, smoothness, and area. With the help of the following command, we can import the Scikitlearn's breast cancer dataset:

```
from sklearn.datasets import load_breast_cancer
# Load the breast cancer dataset

breast_cancer = load_breast_cancer()
```

This will load the breast cancer dataset into a variable called breast_cancer. The breast_cancer variable is a dictionary-like object that contains the dataset data and metadata.
The data is stored in the data attribute, and the metadata is stored in other attributes, such as target_names and feature_names.

Step 3: Organizing data into sets: In this step, we will divide our data into two parts namely a training set and a test set. Splitting the data into these sets is very important because we have to test our model on the unseen data. To split the data into sets, sklearn has a function called the train_test_split() function. With the help of the following commands, we can split the data in these sets:

```
from sklearn.model_selection import train_test_split
```

The above command will import the train_test_split function from sklearn and the command below will split the data into training and test data. In the example given below, we are using 40 % of the data for testing and the remaining data would be used for training the model.

```
train, test, train_labels, test_labels =
train_test_split(features,labels,test_size = 0.40, random_state = 42)
```

Step 4: Building the model: In this step, we will be building our model. We are going to use Naïve Bayes algorithm for building the model. Following commands can be used to build the model:

```
from sklearn.naive_bayes import GaussianNB
```

The above command will import the GaussianNB module. Now, the following command will help you initialize the model..

```
gnb = GaussianNB()
```

We will train the model by fitting it to the data by using gnb.fit().

```
model = gnb.fit(train, train_labels)
```

Step 5: Evaluating the model and its accuracy: In this step, we are going to evaluate the model by making predictions on our test data. Then we will find out its accuracy also. For making predictions, we will use the predict() function**.**

For example, consider a classifier that is predicting whether or not a patient has cancer. The classifier makes 100 predictions, and 90 of them are correct. The accuracy score of the classifier would be 90%, because it correctly predicted the presence or absence of cancer in 90% of the patients.

```
from sklearn.metrics import accuracy_score

# Train the classifier
clf = LogisticRegression()
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print the accuracy score
 print("Accuracy:", accuracy)
```

This code will print the accuracy score of the classifier to the console.

Search Algorithm

You will have to consider computer games also with the same strategy as above. Note that Search Algorithms are the ones that figure out the strategy in computer games.

How it works:

The goal of search algorithms is to find the optimal set of moves so that they can reach at the final destination and win. These algorithms use the winning set of conditions, different for every game, to find the best moves. Visualize a computer game as the tree. We know that tree has nodes. Starting from the root, we can come to the final winning node, but with optimal moves. That is the work of search algorithms. Every node in such tree represents a future state. The search algorithms search through this tree to make decisions at each step or node of the game.

Combinational Search

The major disadvantage of using search algorithms is that they are exhaustive in nature, which is why they explore the entire search space to find the solution that leads to wastage of resources. It would be more cumbersome if these algorithms need to search the whole search space for finding the final solution. To eliminate such kind of problem, we can use combinational search which uses the heuristic to explore the search space and reduces its size by eliminating the possible wrong moves. Hence, such algorithms can save the resources. Some of the algorithms that use heuristic to search the space and save the resources are discussed here:

Minimax Algorithm

It is the strategy used by combinational search that uses heuristic to speed up the search strategy. The concept of Minimax strategy can be understood with the example of two player games, in which each player tries to predict the next move of the opponent and tries to minimize that function. Also, in order to win, the player always try to maximize its own function based on the current situation. Heuristic plays an important role in such kind of strategies like Minimax. Every node of the tree would have a heuristic function associated with it. Based on that heuristic, it will take the decision to make a move towards the node that would benefit them the most