**Important files**

* **for train nlu ---**  nlu.yml, stories.yml, domain.yml
* **for UI ----** index.html, credentials.yml

**Commands**

conda deactivate

conda activate rasaenv

pip install rasa

rasa init

**Rasa commands**

rasa train, rasa shell, rasa test, rasa visualize, rasa data split nlu

rasa run -m models --enable-api --cors "\*" \_\_open index.html start conversation with bot

**Intent**

1. Greet
2. Happy
3. Unhappy
4. Bot
5. Default
6. creator
7. instructor
8. Programming
9. Popular programming
10. Variable
11. Example of variable
12. Data types
13. example of data types
14. Control structure
15. List different type of control structure
16. Function
17. OOP
18. Object
19. Class
20. Inheritance
21. Polymorphism
22. Abstraction
23. Encapsulation

* **Data Split?**
* **Training data 70% / Testing 30%?**
* **Algorithms ?**
* **Rasa NLU, Rasa Core?**
* **Confidence %?**
* **Confusion Matrix draw?**

**Data Split:** When training a machine learning model, it is common to split the available data into training and testing sets. The standard split is to use 80% of the data for training and 20% for testing. The training set is used to train the model, and the testing set is used to evaluate how well the model generalizes to new, unseen data. As 0.8, 0.2

**Algorithms:** Rasa NLU uses a combination of algorithms for intent classification and entity extraction, including Support Vector Machines (SVM), Conditional Random Fields (CRF), and TensorFlow. Rasa Core uses a form of Reinforcement Learning algorithm called Policy Gradient algorithm.

**TensorFlow:** is an open-source machine learning framework developed by Google that provides tools and libraries for building, training, and deploying machine learning and deep learning models. It offers a wide range of features, including a flexible core library, high-level APIs like Keras, tools for model visualization (TensorBoard), and deployment capabilities. TensorFlow is widely used in both research and production settings and supports various machine learning techniques.

**Confidence %:** In Rasa NLU, each intent and entity prediction comes with a confidence score, which indicates how confident the model is in its prediction. This score is represented as a percentage and ranges from 0% to 100%. A higher confidence score indicates a higher level of confidence in the model's prediction.

**Confusion Matrix:** A confusion matrix is a table that summarizes the performance of a classification model by comparing the predicted labels with the actual labels. The confusion matrix displays the number of true positives, true negatives, false positives, and false negatives. It is a useful tool for evaluating the performance of a model and identifying areas where it can be improved.

**Drawing a confusion matrix:** In Rasa NLU or Rasa Core would depend on the specific implementation and the tools used to evaluate the model's performance. There are various tools and libraries available in Python, such as scikit-learn, which can be used to draw a confusion matrix.

s

**CS304 object-oriented programming (OOP)**

Object-oriented programming (OOP) is a programming paradigm that uses objects as the building blocks for designing and structuring software. It is based on several fundamental concepts known as the "Four Pillars of OOP," which are: Object, Class, Inheritance, Polymorphism, Abstraction, Encapsulation

1. **Class:** is a template for creating objects. It defines the attributes (data members) and methods (functions or behaviors) that objects created from the class will have. Classes act as a model for real-world entities or concepts. For example, you might have a "Car" class that defines the characteristics and behaviors of cars, such as make, model, color, and methods like "start," "stop," and "accelerate."
2. **Object:** is an instance of a class. It is a concrete, tangible entity created based on the class's blueprint. Objects represent real-world entities or instances of a concept modeled by the class. if you have a "Car" class, an object could be a specific car with its unique characteristics and behaviors.
3. **Encapsulation:** refers to the bundling of data (attributes or properties) and the methods (functions or procedures) that operate on that data into a single unit called an "object**." This concept helps in hiding the internal details of an object and only exposing the necessary functionalities,** thus providing data security and preventing unintended access or modification.
4. **Inheritance:** allows you to create a new child class based on parent class. **The child class inherits the properties and methods of the parent class**, and it can also extend or override them. Inheritance **promotes code reusability** and the creation of a hierarchical class structure.
5. **Polymorphism:** means "**many shapes**" and **allows objects of different classes to be treated as objects of a common base class**. This concept is often achieved through method overriding (where a subclass provides its own implementation of a method) and method overloading (where multiple methods with the same name but different parameters are defined in the same class or subclass). **Polymorphism enables flexibility and dynamic behavior in the program.**
6. **Abstraction:** involves **simplifying complex reality by modeling classes based on the essential properties and behaviors an object should have**. It hides the unnecessary details and focuses on what is relevant for the specific problem being solved. Abstract classes and interfaces are used to define a blueprint for classes that implement them, ensuring that specific methods are implemented in derived classes.