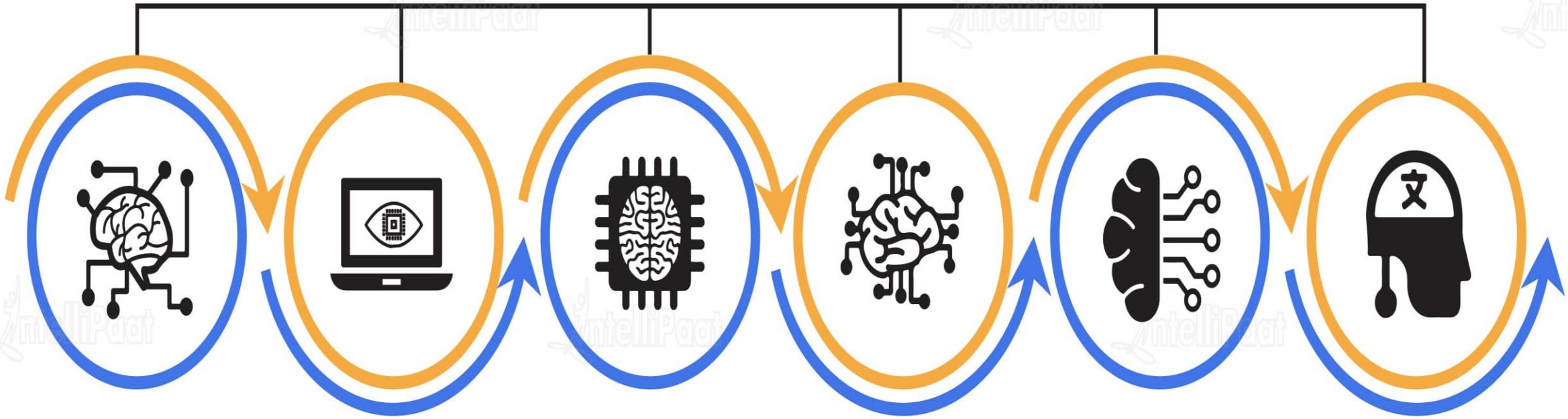
Deploying Machine Learning Models in Production as APIs

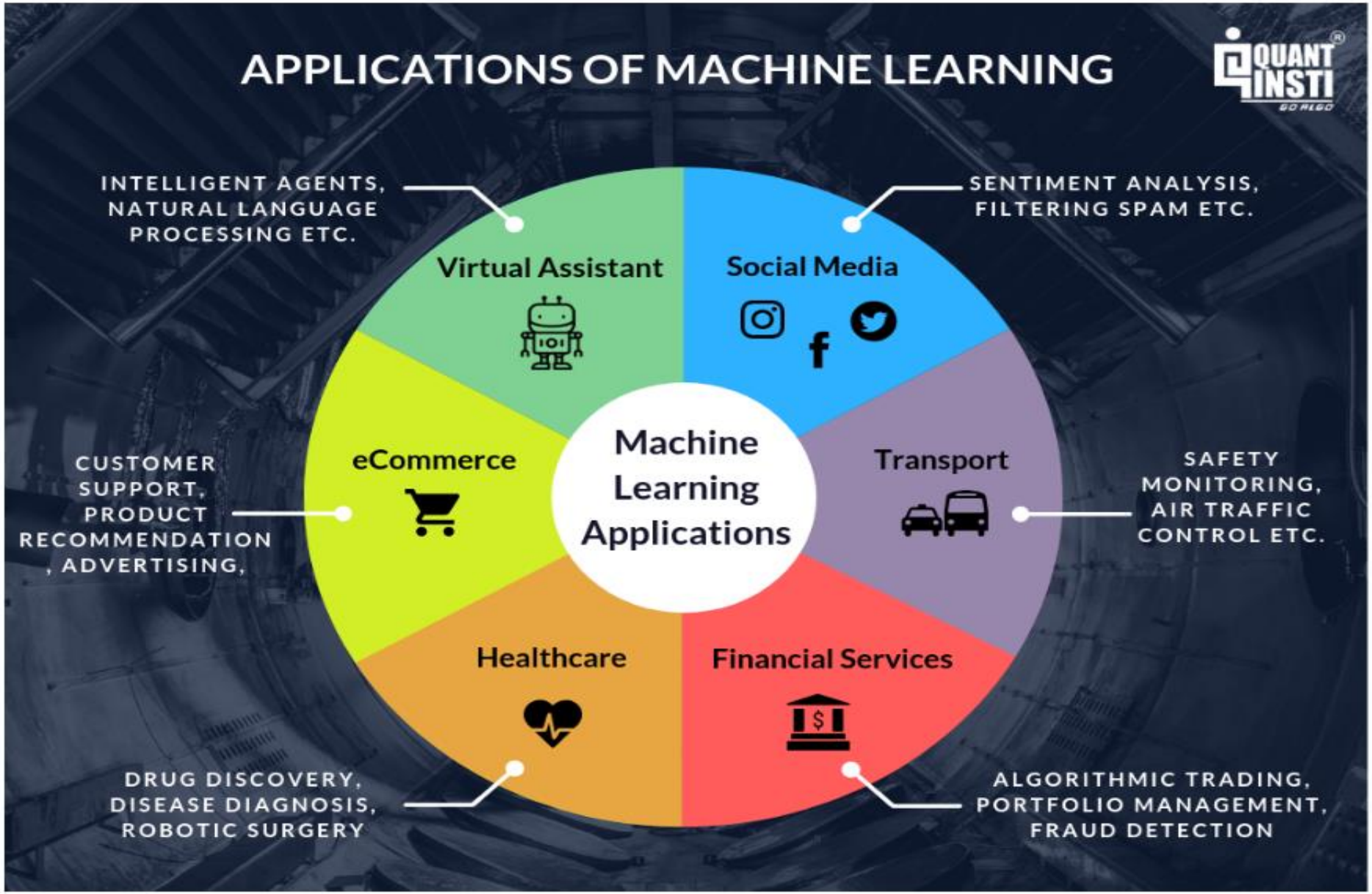
WAI'23 – PRESENTED BY DR AID AICHA

https://sites.google.com/view/wai23/accueil

Why this Presentation ?

Artificial Intelligence




APPLICATIONS OF MACHINE LEARNING
QUANT INSTI
INTELLIGENT AGENTS, NATURAL LANGUAGE PROCESSING ETC.
SENTIMENT ANALYSIS, FILTERING SPAM ETC.
Virtual Assistant
Social Media
CUSTOMER SUPPORT, PRODUCT RECOMMENDATION , ADVERTISING,
eCommerce
Machine Learning Applications
Transport
SAFETY MONITORING, AIR TRAFFIC CONTROL ETC.
Healthcare
Financial Services
DRUG DISCOVERY, DISEASE DIAGNOSIS, ROBOTIC SURGERY
ALGORITHMIC TRADING, PORTFOLIO MANAGEMENT, FRAUD DETECTION

memeBot: Towards Automatic Image Meme Generation [Sadasivam et al., 2020]

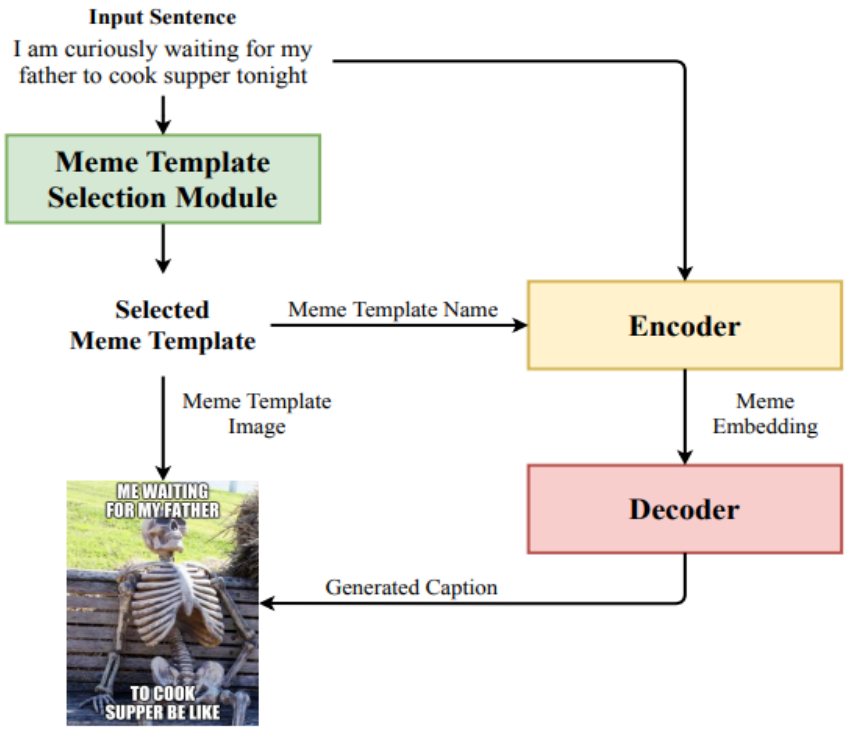
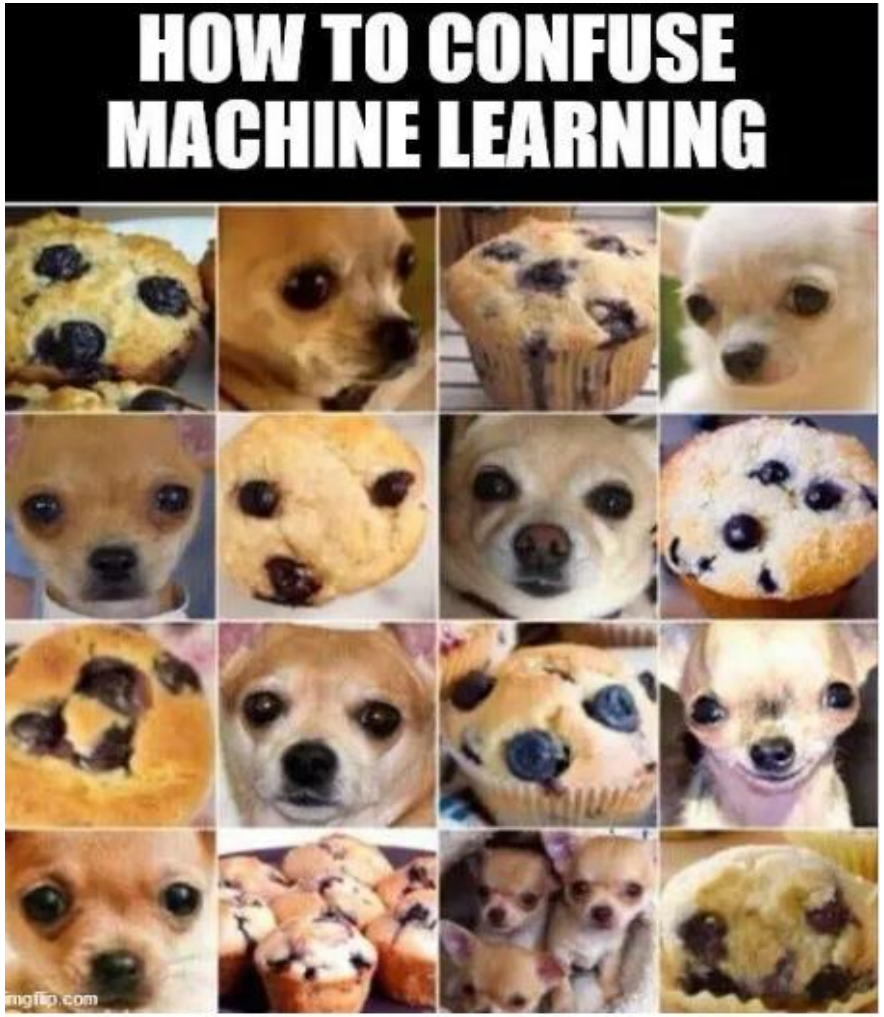


Figure 1: An illustrative figure of memeBot. It generates an image meme for a given input sentence by combining the selected meme template image and the generated meme caption.



MusicLM: Generating Music From Text

| paper | dataset |

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, Christian Frank
Google Research

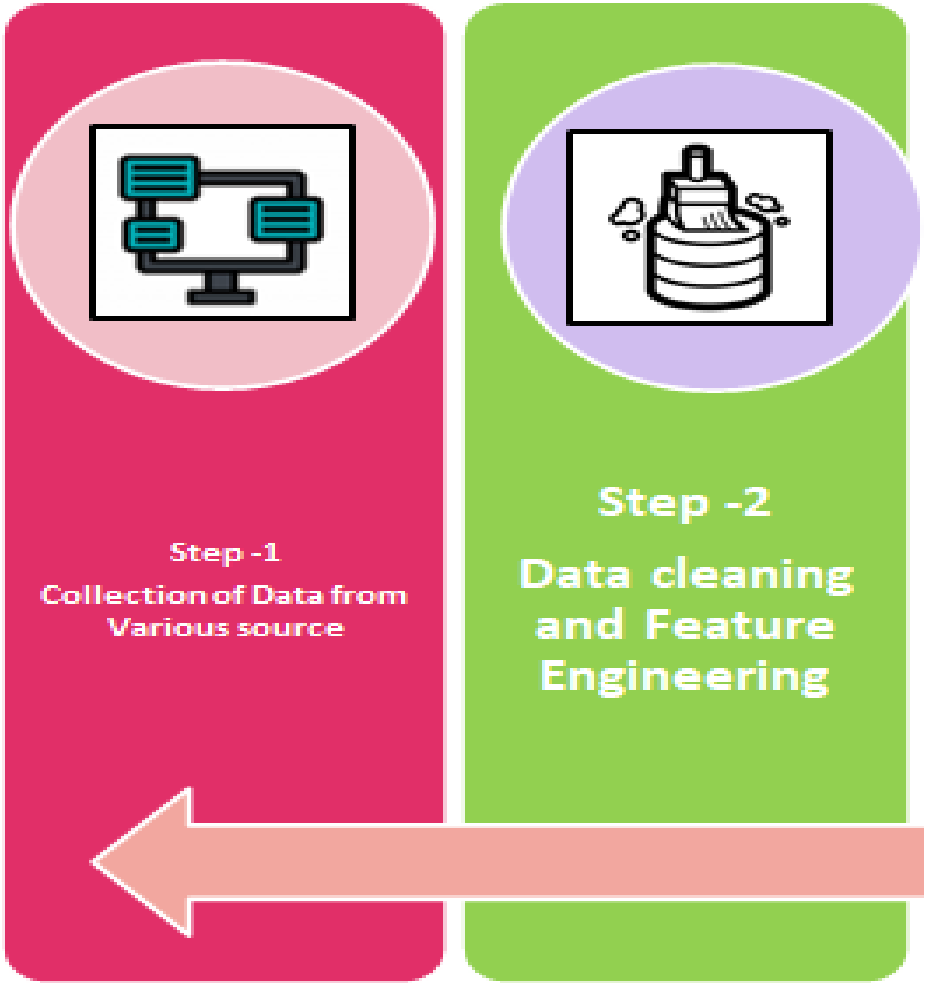
Abstract We introduce MusicLM, a model generating high-fidelity music from text descriptions such as "*a calming violin melody backed by a distorted guitar riff*". MusicLM casts the process of conditional music generation as a hierarchical sequence-to-sequence modeling task, and it generates music at 24 kHz that remains consistent over several minutes. Our experiments show that MusicLM outperforms previous systems both in audio quality and adherence to the text description. Moreover, we demonstrate that MusicLM can be conditioned on both text and a melody in that it can transform whistled and hummed melodies according to the style described in a text caption. To support future research, we publicly release MusicCaps, a dataset composed of 5.5k music-text pairs, with rich text descriptions provided by human experts.

Caption	Generated audio
The main soundtrack of an arcade game. It is fast-paced and upbeat, with a catchy electric guitar riff. The music is repetitive and easy to remember, but with unexpected sounds, like cymbal crashes or drum rolls.	0:00 / 0:30

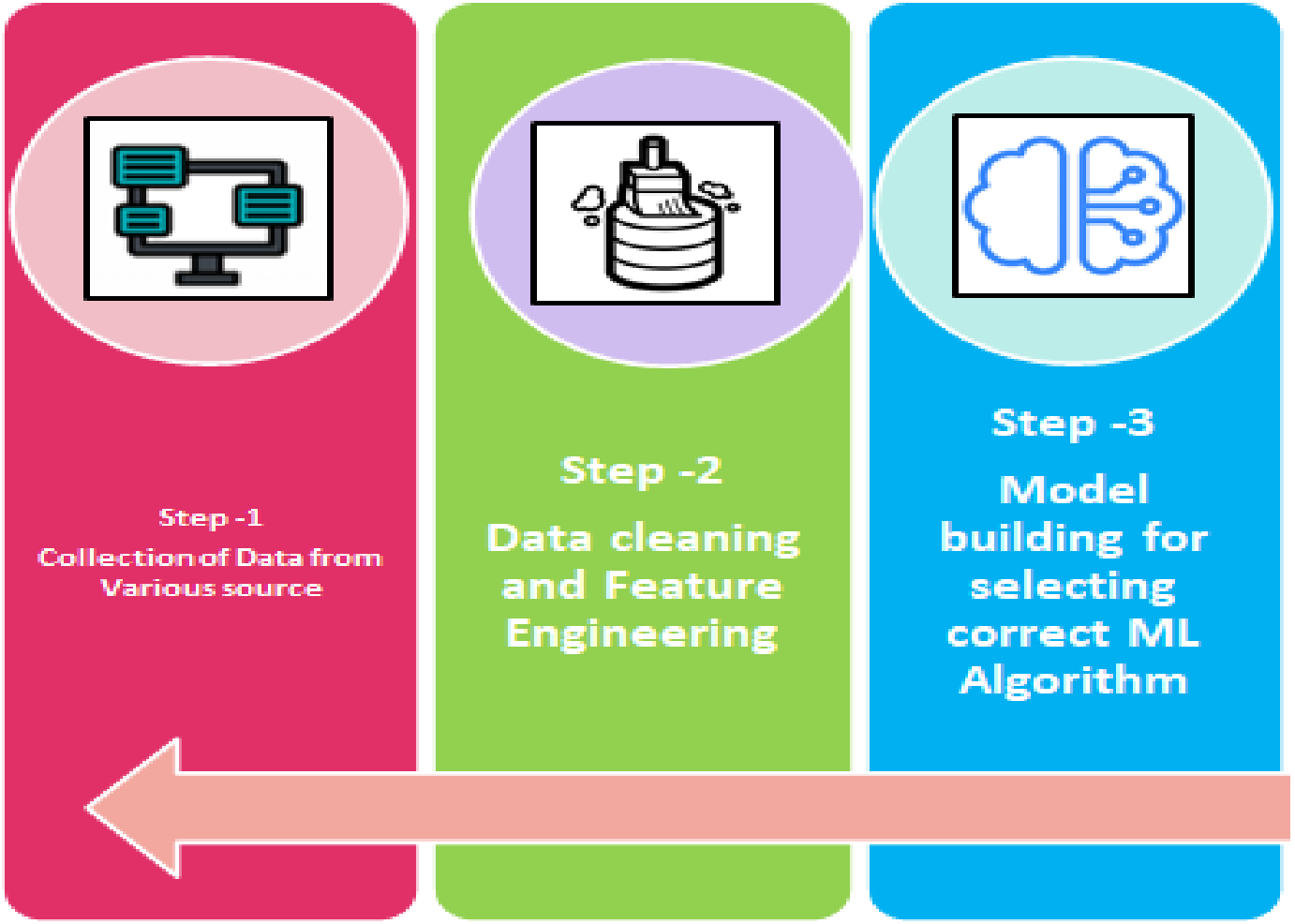
https://google-research.github.io/seanet/musiclm/examples/

Machine Learning Life Cycle

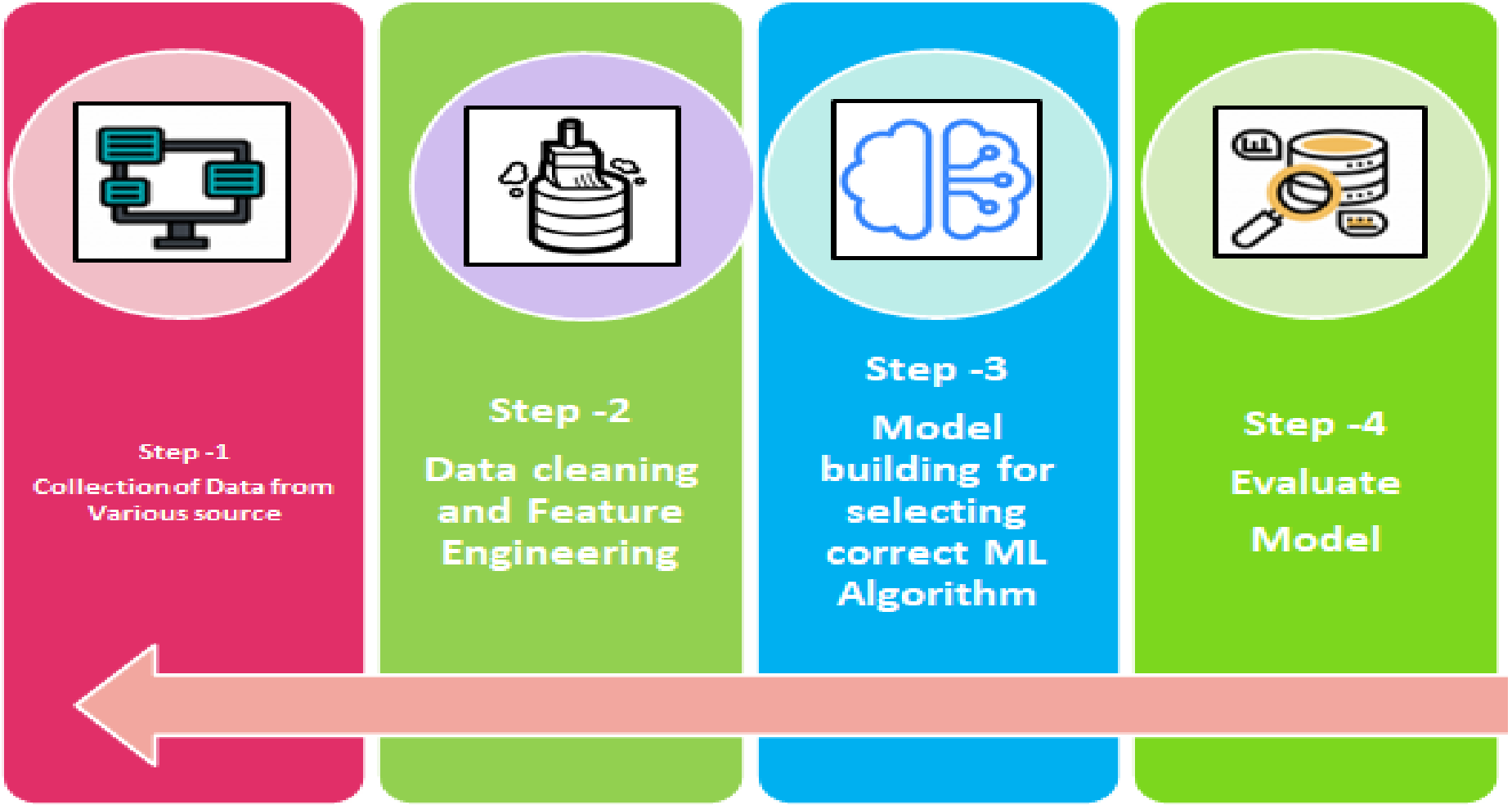
Machine Learning Life Cycle



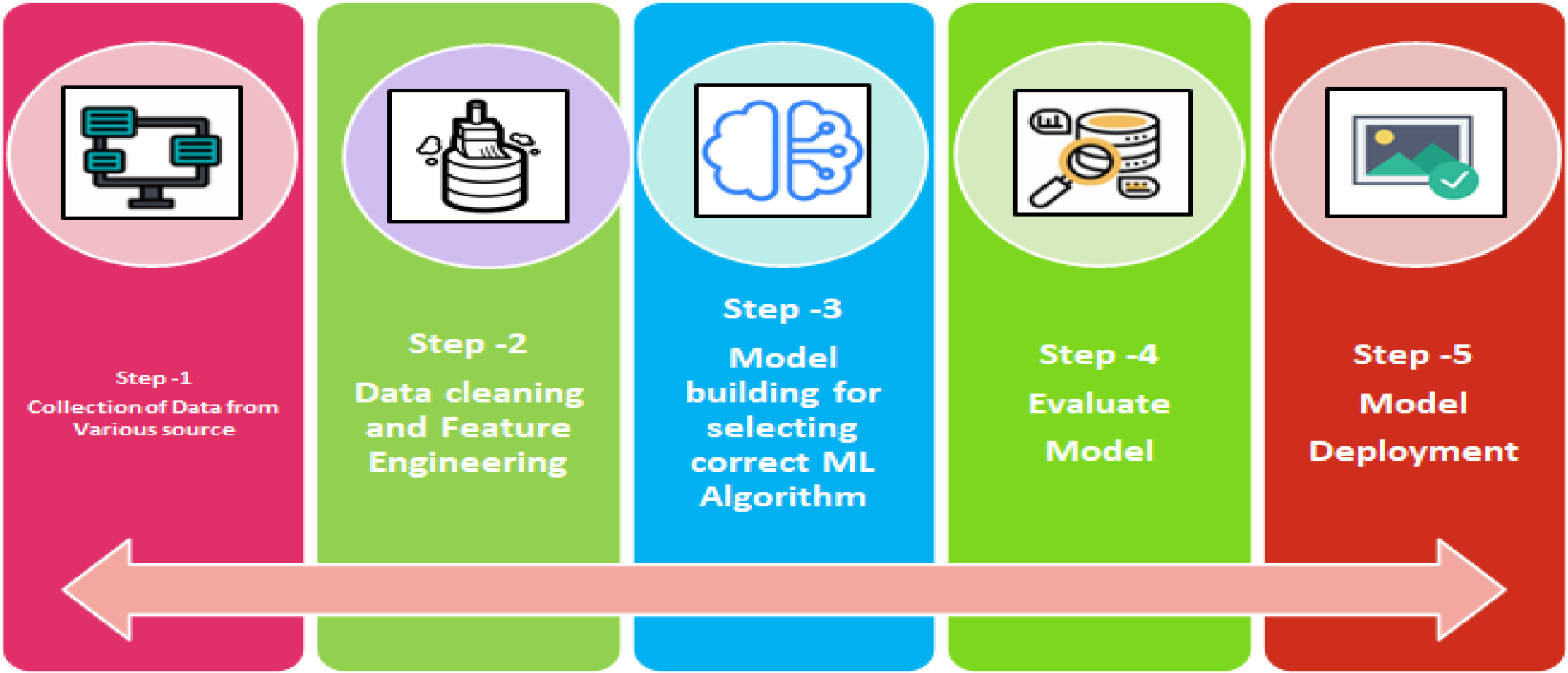
Machine Learning Life Cycle



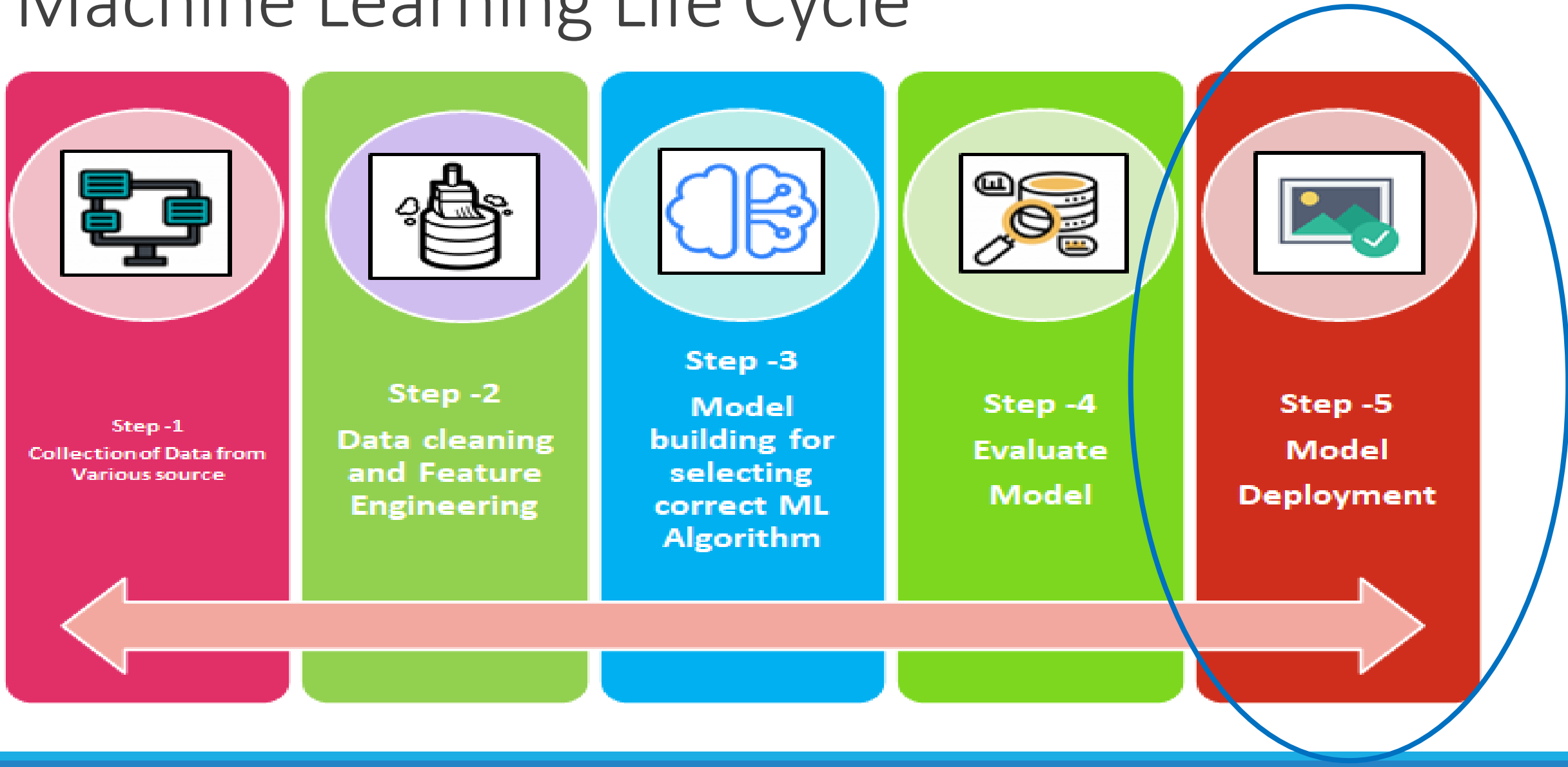
Machine Learning Life Cycle



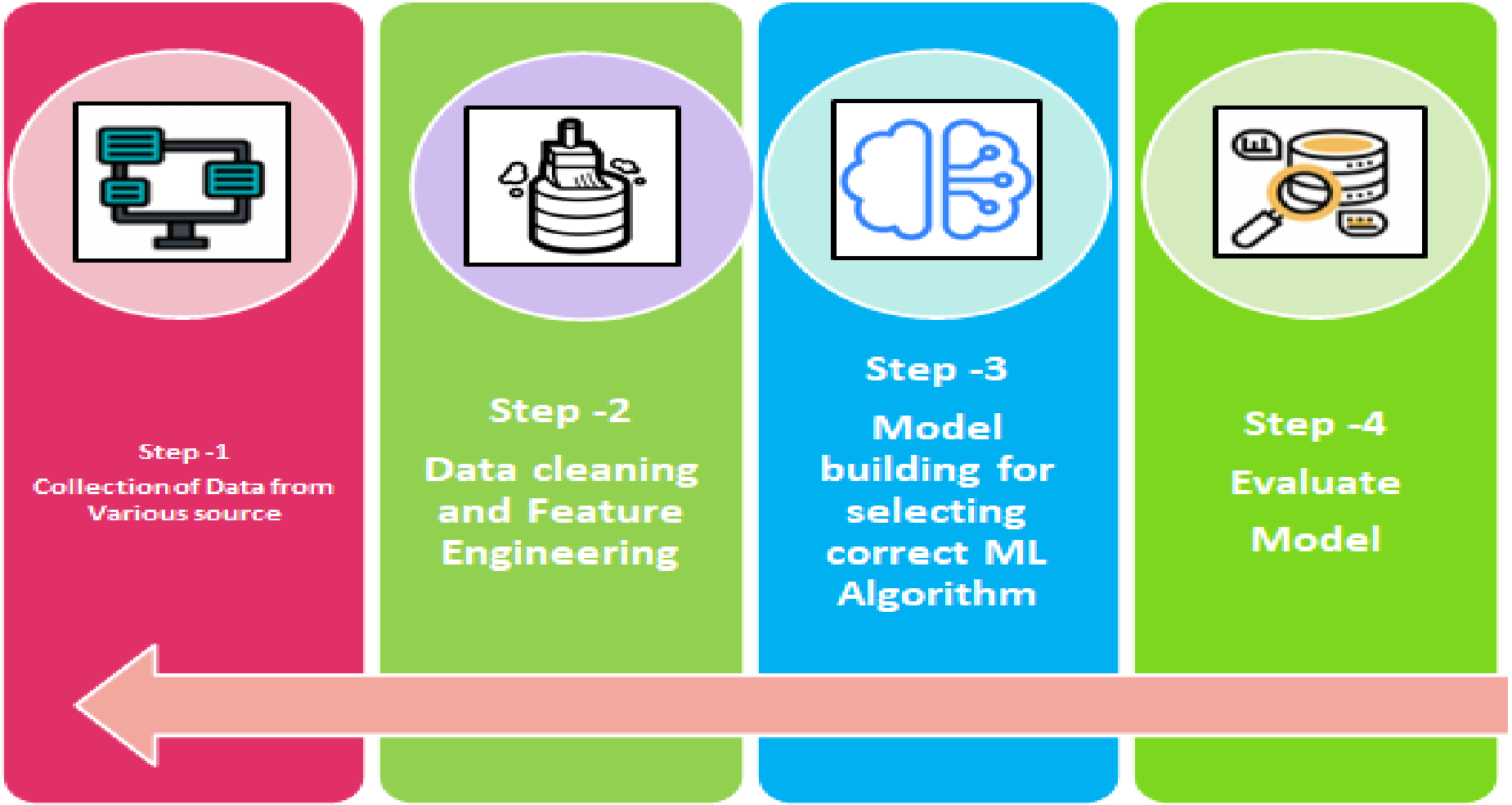
Machine Learning Life Cycle

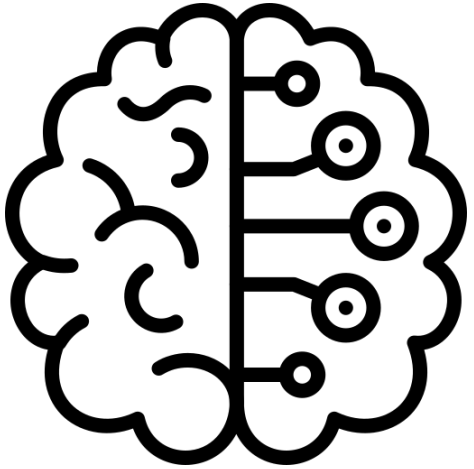


Machine Learning Life Cycle



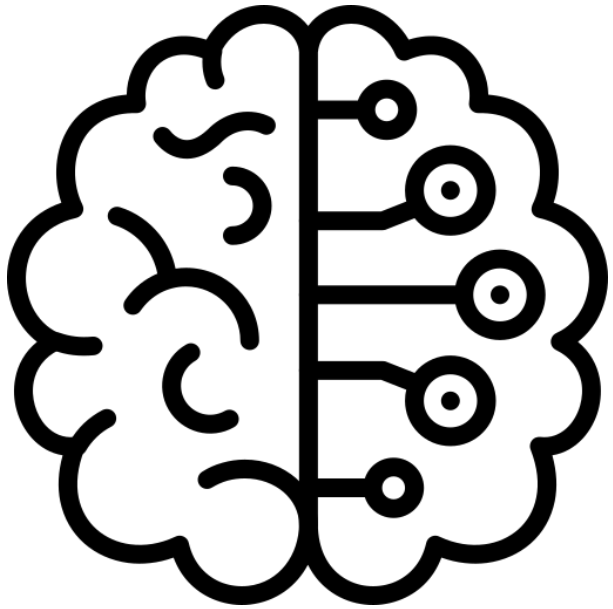
ML Models Deployment – Why ?





ML & DL Model

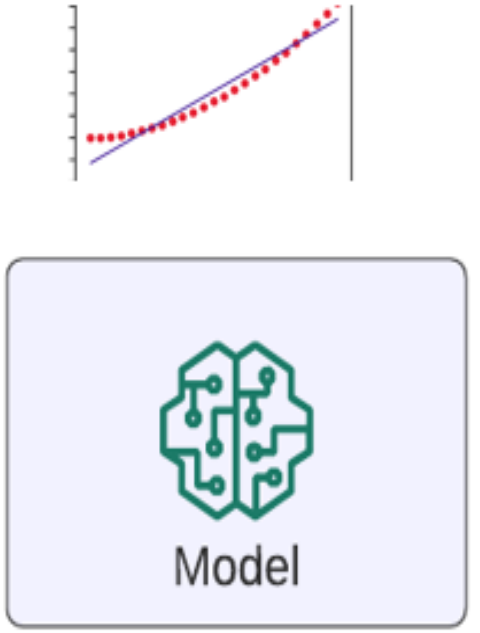
ML Models Deployment – Why ?



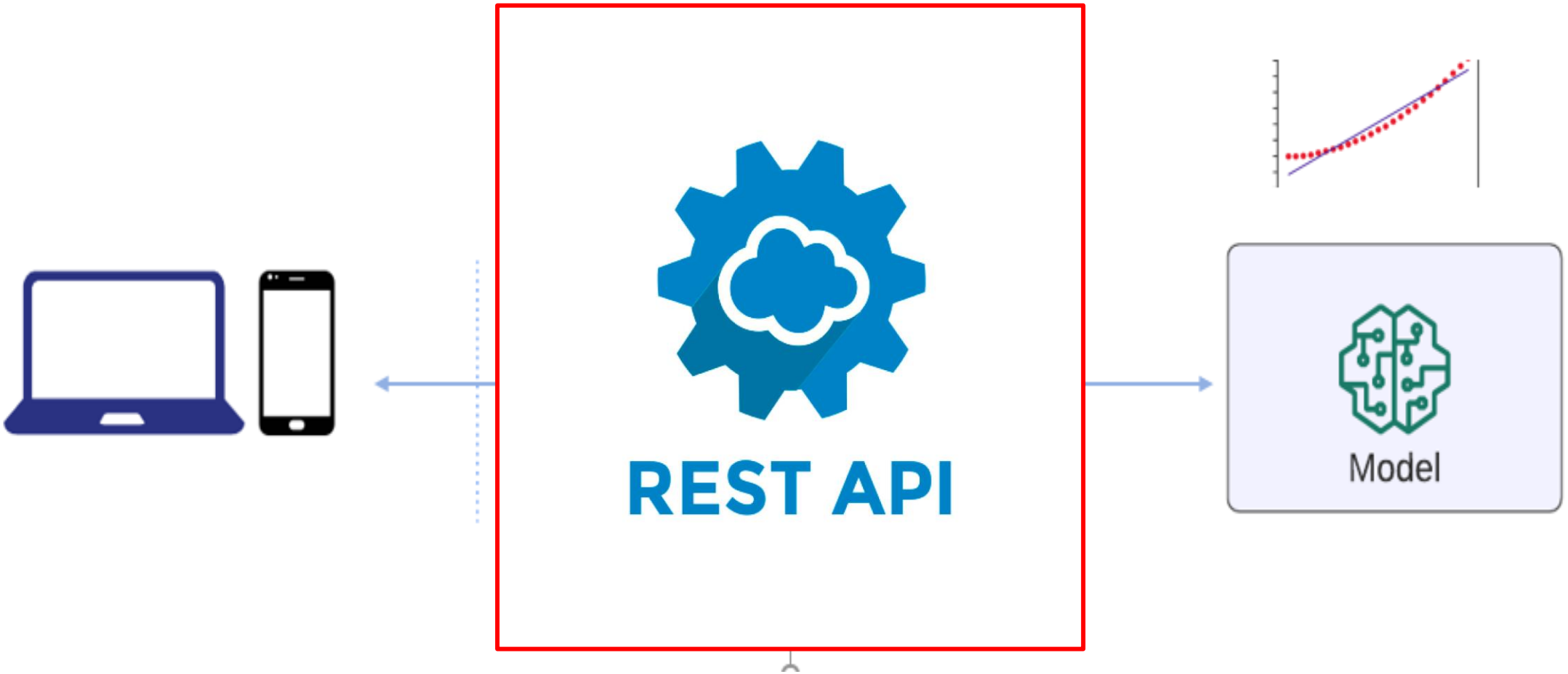
Production

STARTUP

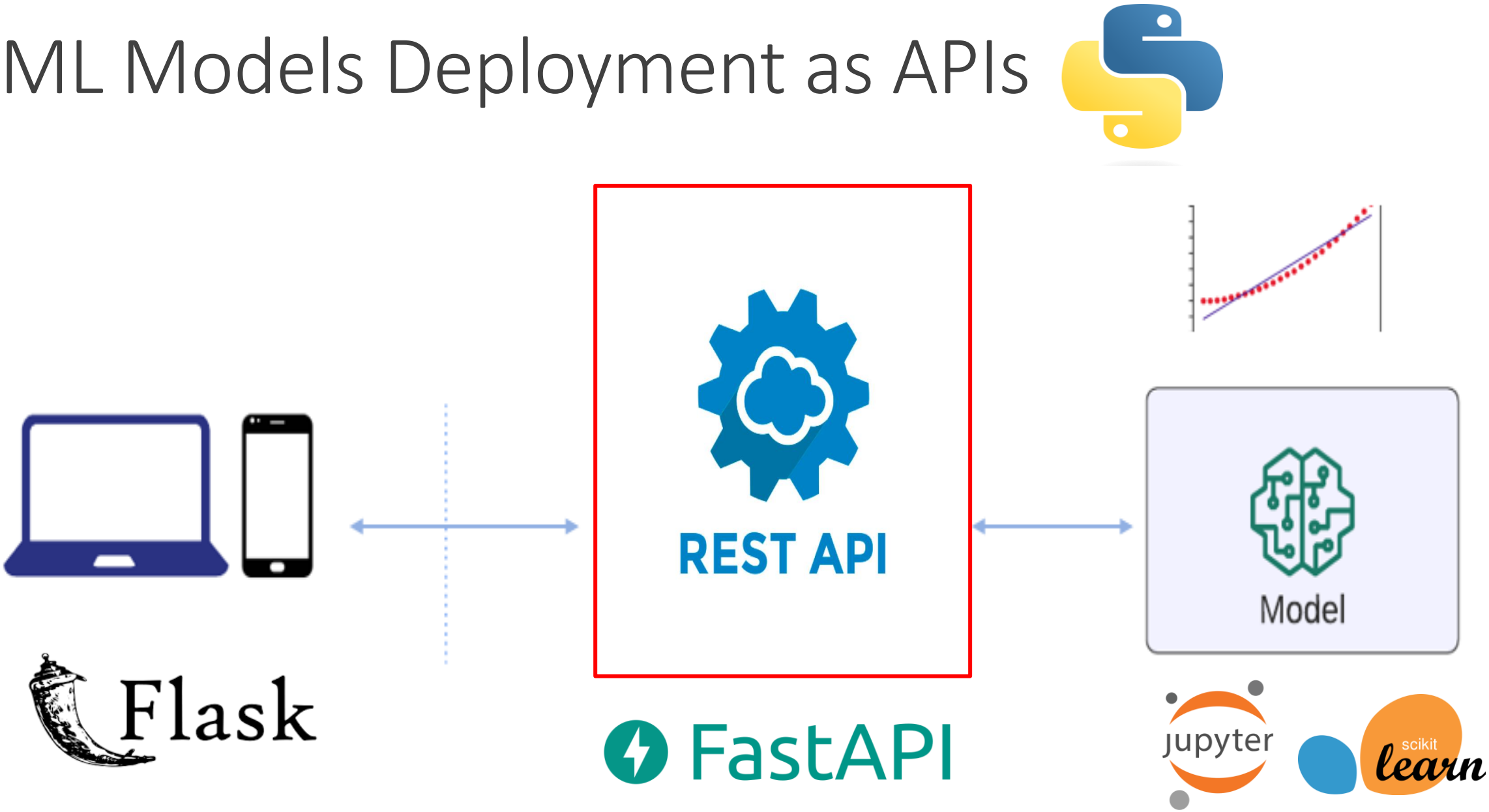
ML Models Deployment as APIs



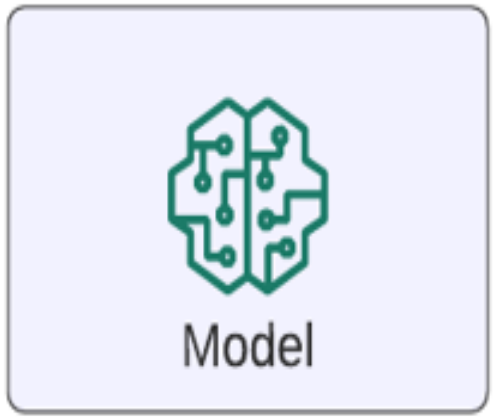
ML Models Deployment as APIs



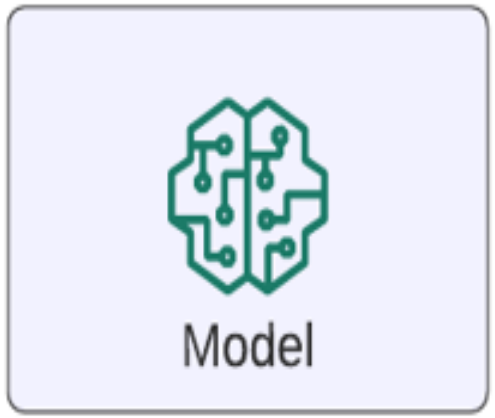
ML Models Deployment as APIs

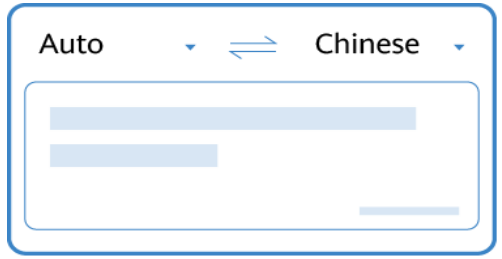


ML Models Deployment as APIs



ML Models Deployment as APIs





Text input

Develop app

Language detection

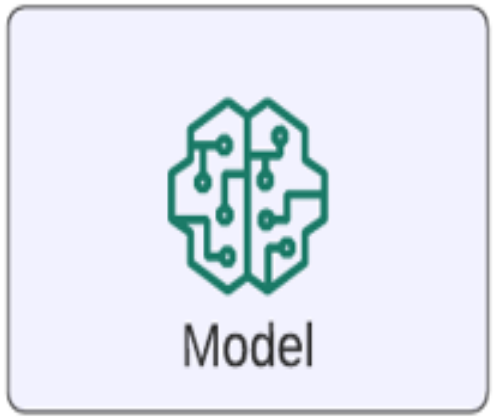
ML Kit

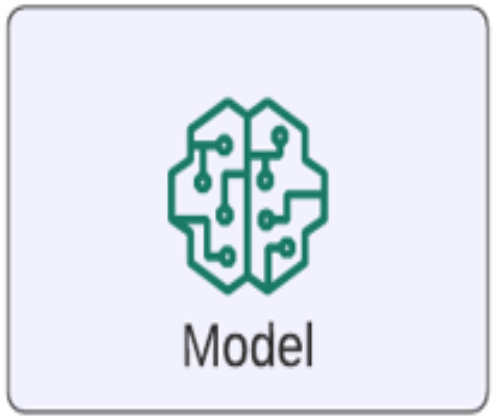
Language Detection

Result 1	Value
type	th

Result 2	Value
score	1.0

ML Models Deployment as APIs





jupyter LanguageDetection (auto-sauvegardé)

File Edit View Insert Cell Kernel Widgets Help

Exécuter Markdown

Entrée [3]:
```
# Loading the dataset
data = pd.read_csv("Language Detection.csv")
```

Entrée [4]:
```
data.head()
```

Out[4]:

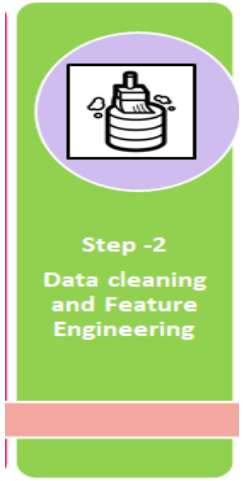
	Text	Language
0	Nature, in the broadest sense, is the natural...	English
1	"Nature" can refer to the phenomena of the phy...	English
2	The study of nature is a large, if not the onl...	English
3	Although humans are part of nature, human acti...	English
4	[1] The word nature is borrowed from the Old F...	English

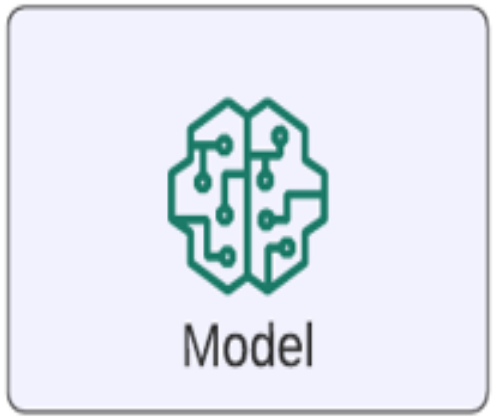
Entrée [5]:
```
data.shape
```

Out[5]:
```
(10337, 2)
```

Entrée [6]:
```
data["Language"].unique()
```

Out[6]:
```
array(['English', 'Malayalam', 'Hindi', 'Tamil', 'Portugeese', 'French',
       'Dutch', 'Spanish', 'Greek', 'Russian', 'Danish', 'Italian',
       'Turkish', 'Sweedish', 'Arabic', 'German', 'Kannada'], dtype=object)
```





jupyter LanguageDetection (auto-sauvegardé)

File Edit View Insert Cell Kernel Widgets Help

Exécuter Markdown

Name: Language, Length: 10337, dtype: object

Entrée [10]:
```
# converting categorical variables to numerical
le = LabelEncoder()
y = le.fit_transform(y)
```

Entrée [11]:
```
y
```

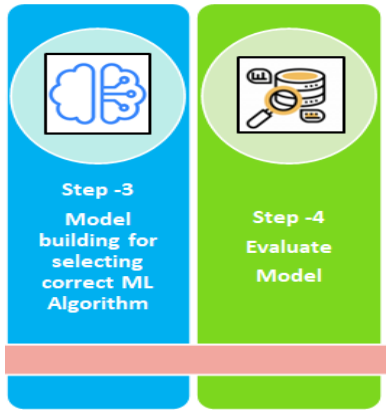
Out[11]:
```
array([3, 3, 3, ..., 9, 9, 9])
```

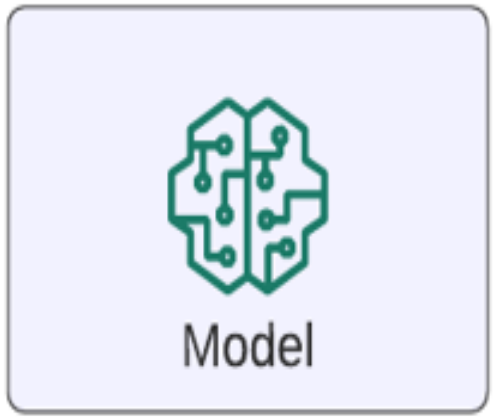
Entrée [12]:
```
le.classes_
```

Out[12]:
```
array(['Arabic', 'Danish', 'Dutch', 'English', 'French', 'German',
       'Greek', 'Hindi', 'Italian', 'Kannada', 'Malayalam', 'Portugeese',
       'Russian', 'Spanish', 'Sweedish', 'Tamil', 'Turkish'], dtype=object)
```

Entrée [13]:
```
# text preprocessing
data_list = []
for text in X:
    text = re.sub(r'[!@#$(),\n"%^*?\:;~`0-9]', ' ', text)
    text = re.sub(r'[[]]', ' ', text)
    text = text.lower()
    data_list.append(text)
```

Entrée [14]:
```
data_list
```





Entrée [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Entrée [16]:
```
# creating bag of words using countvectorizer

cv = CountVectorizer()
cv.fit(X_train)

x_train = cv.transform(X_train).toarray()
x_test  = cv.transform(X_test).toarray()
```

Entrée [17]:
```
model = MultinomialNB()
model.fit(x_train, y_train)
```

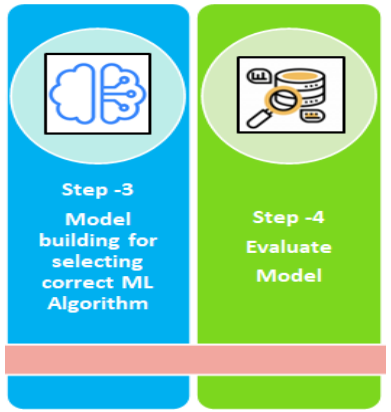
Out[17]:
```
MultinomialNB()
```

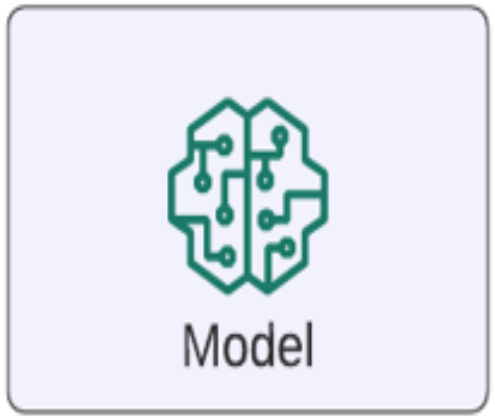
Entrée [19]:
```
y_pred = model.predict(x_test)
```

Entrée [20]:
```
ac = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)
```

Entrée [21]:
```
print("Accuracy is :", ac)
```

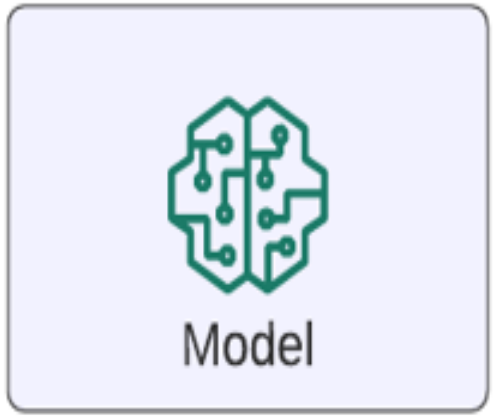
```
Accuracy is : 0.9729206963249516
```



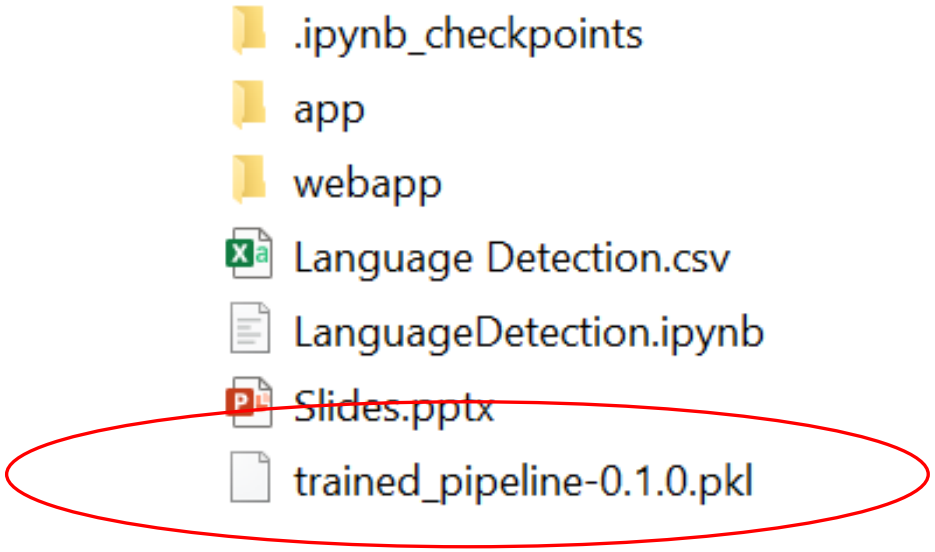


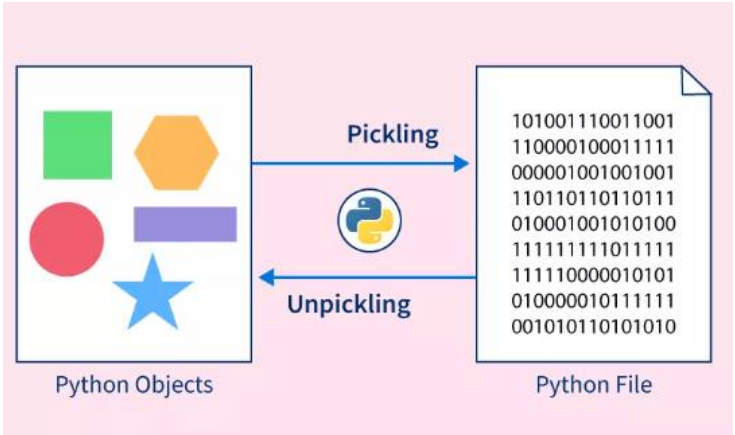
[31]:
```
text1 = "Hello, how are you?"
text2 = "Ciao, come stai?"

lang = pipe.predict([text1])
print(le.classes_[lang[0]], lang)

lang = pipe.predict([text2])
print(le.classes_[lang[0]], lang)
```

```
English [3]
Italian [8]
```




```
Entrée [26]: with open('trained_pipeline-0.1.0.pkl','wb') as f:
                 pickle.dump(pipe, f)
```





ML Models Deployment as APIs

FastAPI

ML Models Deployment as APIs

FastAPI

model.py

```python
__version__ = "0.1.0"

BASE_DIR = Path(__file__).resolve(strict=True).parent


with open(f"{BASE_DIR}/trained_pipeline-{__version__}.pkl", "rb") as f:
    model = pickle.load(f)
```

trained_pipeline-0.1.0.pkl

ML Models Deployment as APIs

FastAPI

model.py

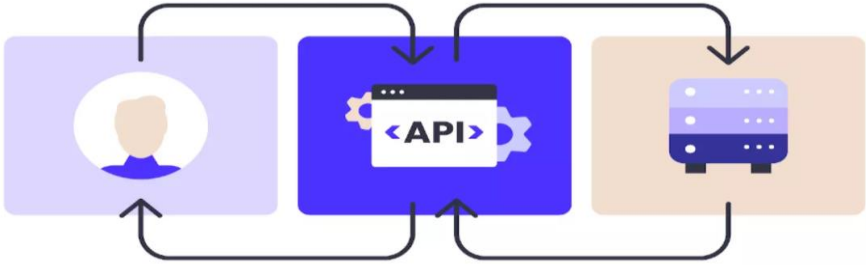
```python
def predict_pipeline(text):
    text = re.sub(r'[!@#$(),\n"%^*?\:;~`0-9]', " ", text)
    text = re.sub(r"[[]]", " ", text)
    text = text.lower()
    pred = model.predict([text])
    return classes[pred[0]]
```

ML Models Deployment as APIs

FastAPI

```python
app = FastAPI()


class TextIn(BaseModel):
    text: str


class PredictionOut(BaseModel):
    language: str

```

main.py



ML Models Deployment as APIs

FastAPI

Request URL

```
http://127.0.0.1:8000/predict
```

```python
@app.post("/predict", response_model=PredictionOut)
async def predict(payload: TextIn):
    language = predict_pipeline(payload.text)
    return {"language": language}
```

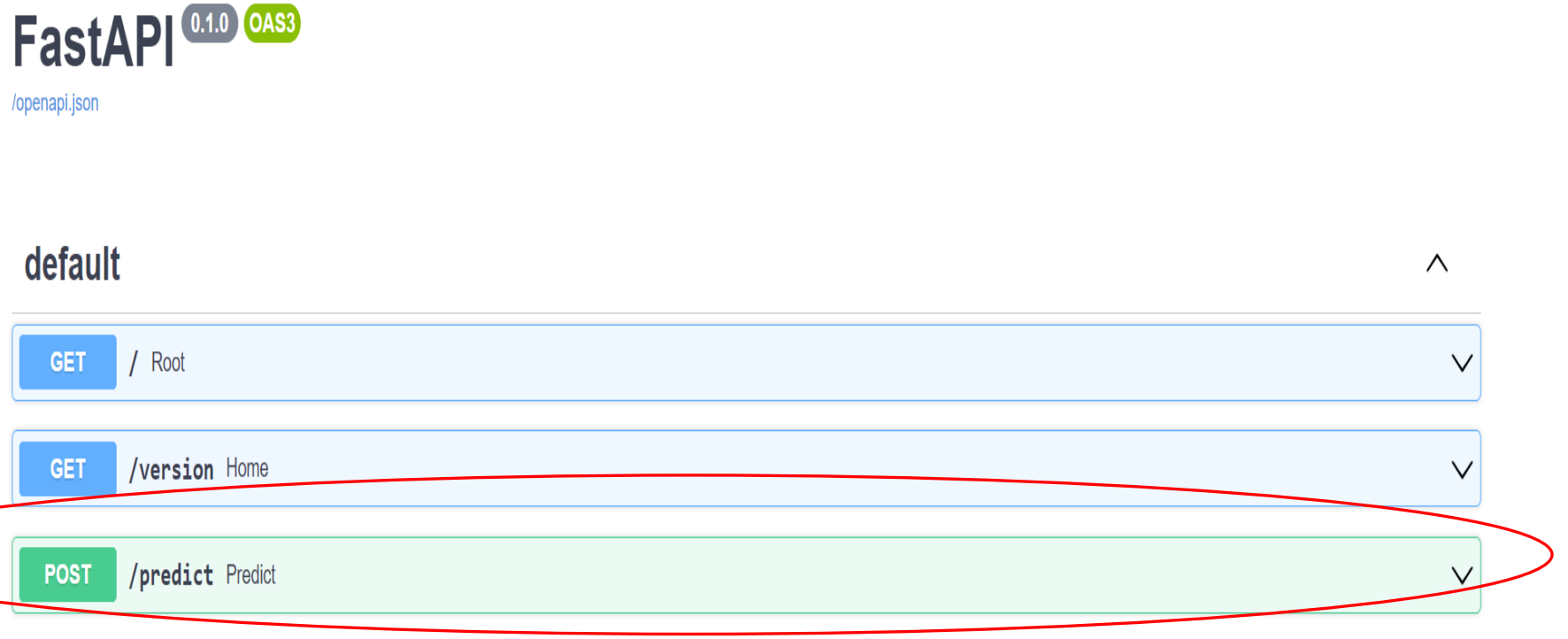
main.py

ML Models Deployment as APIs

FastAPI

```
> uvicorn main:app
```

http://127.0.0.1:8000/docs



ML Models Deployment as APIs

FastAPI

POST /predict Predict

Parameters

Cancel Reset

No parameters

Request body required application/json

```
{
  "text": "How are you ?"
}
```

Execute

ML Models Deployment as APIs

FastAPI

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "text": "How are you ?"
}'
```

Request URL

```
http://127.0.0.1:8000/predict
```

Server response

Code Details

200

Response body

```
{
  "language": "English"
}
```

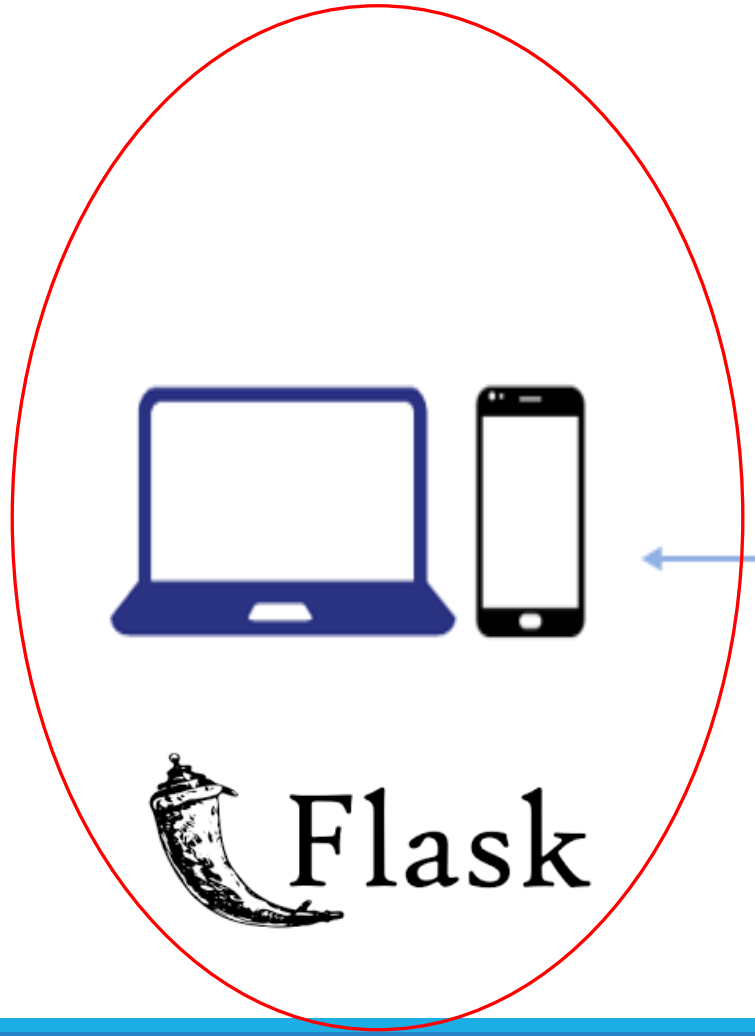
```
{
  "language": "English"
}
```

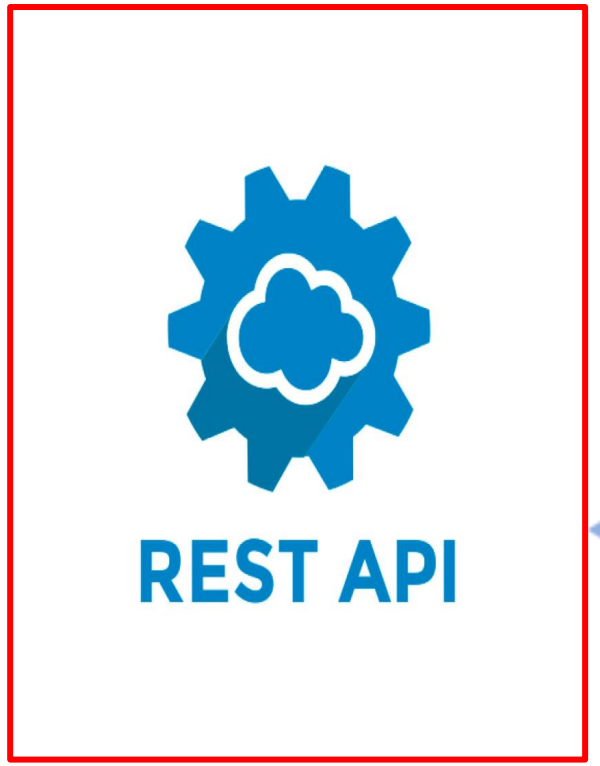
Download

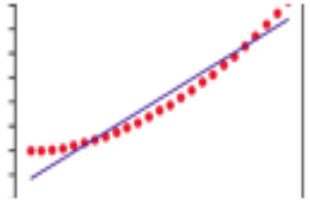
Response headers

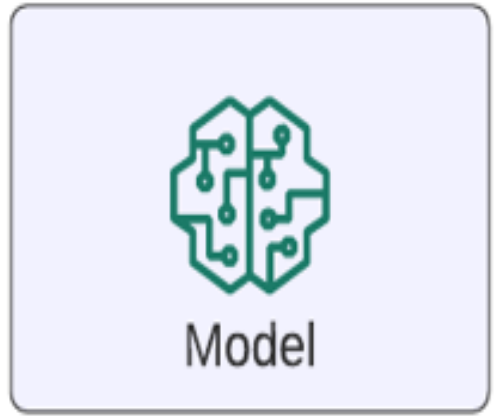
```
content-length: 22
content-type: application/json
date: Thu,13 Apr 2023 22:12:50 GMT
server: uvicorn
```

ML Models Deployment as APIs









FastAPI

jupyter

scikit learn

ML Models Deployment as APIs

WAI'23 – Language Detection

Language detection using Machine Learning algorithms. Enter your text or generate a random one from our dataset to try it.

Enter text here..

Generate Text Predict

WAI'23 - Language Detection

Language detection using Machine Learning algorithms. Enter your text or generate a random one from our dataset to try it.

عيد مبارك و كل عام و انتم بخير

عيد مبارك و كل عام و انتم بخير

Generate Text Predict

ML Models Deployment as APIs

Flask

```
> Python app.py
```

http://127.0.0.1:5000/

```python
@app.route("/", methods=['POST', 'GET'])
def home():
    form = OriginalTextForm()

    if form.generate.data:
        original_text = "Hello, how are you ? This is language detection using Ma
        form.original_text.data = str(original_text)
        return render_template('home.html', form=form, output=False)

    elif form.predict.data:
        text = {"text" : form.original_text.data}
        pred = requests.post('http://127.0.0.1:8000/predict', json=text).json()
        return render_template('home.html', form=form, output=pred)

    return render_template('home.html', form=form, output=False)
```

Request URL

http://127.0.0.1:8000/predict

Flask

WAI'23 – Language Detection

Language detection using Machine Learning algorithms. Enter your text or generate a random one from our dataset to try it.

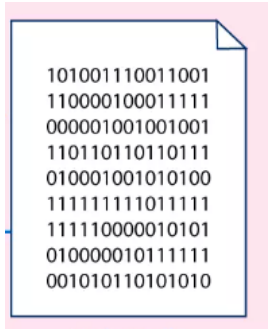
عيد مبارك و كل عام و انتم بخير

Generate Text | Predict

Arabic | See result details

ML Models Deployment as APIs

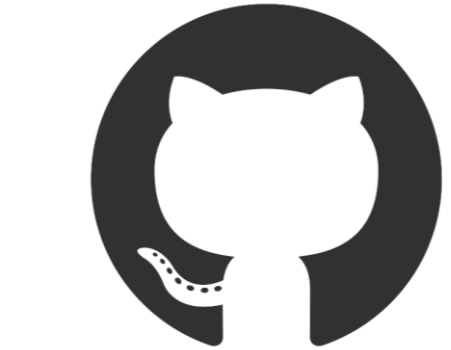
trained_pipeline-0.1.0.pkl



Thank You

HAPPY CODING

DO DIGITAL. STAY HUMAN.



SCAN ME

https://github.com/GitTeaching/Language_Detection_ML_API