ML Deployement

Using FASTAPI

NAME-DIPANWITA SEN
COLLEGE-KIIT
ROLL-22052204

FAST API

What is FastAPI?

- FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.6+.
- Designed to create RESTful APIs quickly and easily.
- Uses Python type hints to provide data validation, serialization, and automatic interactive API documentation.

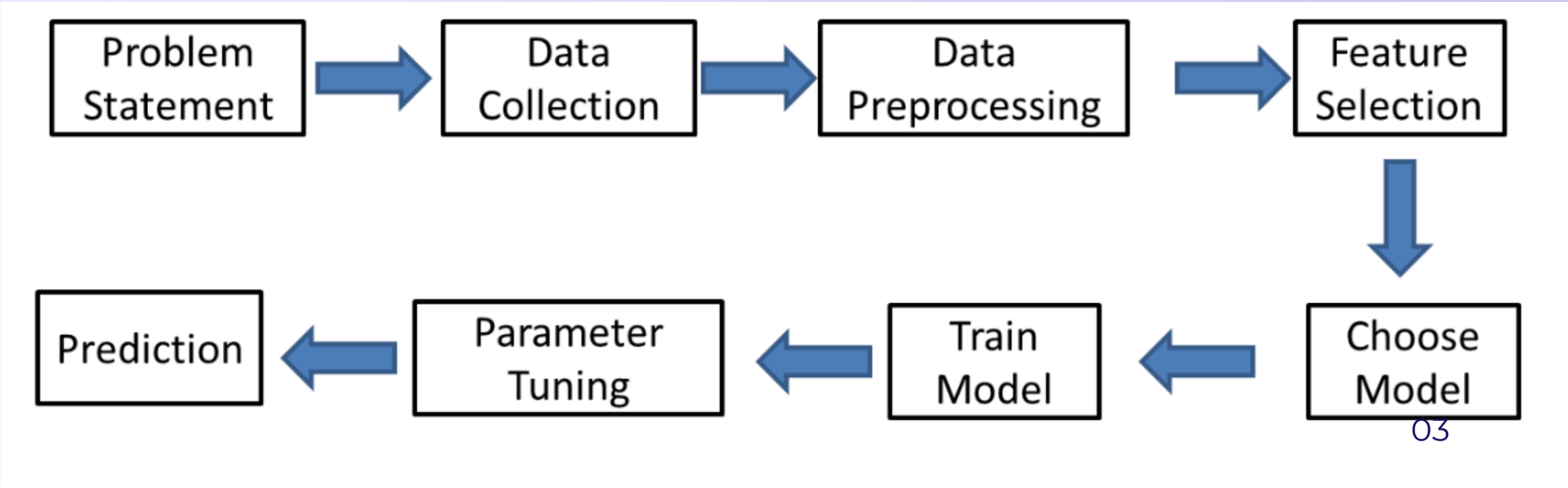
Why FastAPI is Preferred for Model Deployment

- Speed: One of the fastest Python web frameworks, comparable to Node.js and Go in performance.
- Asynchronous: Supports async code natively, allowing non-blocking requests, ideal for real-time ML model serving.
- Automatic Docs: Generates interactive API documentation (Swagger UI and ReDoc) automatically based on code and types.
- Data Validation: Uses Python type hints with Pydantic for automatic request validation and parsing.

Comparison between Django & Flask

	django	Flask
Type of Framework	Full Stack Web Framework	WSGI framework
Flexibility	Feature-packed	Full flexibility
ORM Usage	Built-in ORM	SQLAlchemy is used
Design	Batteries-included	Minimalistic design
Working Style	Monolithic	Diversified

ML MODEL TRAINING AND DIFFERENCE



Sentinal Analysis

Name: Sentiment Analysis Dataset
Source: Kaggle
🔗 Dataset: https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset
Details:

- **Contains over 30,000 labeled text samples (positive/negative sentiment).**
- **Used to train a Logistic Regression model for binary sentiment classification.**

```
df = pd.read_csv("/kaggle/input/sentiment-analysis-dataset/train.csv", encoding='latin1')
test = pd.read_csv("/kaggle/input/sentiment-analysis-dataset/test.csv", encoding='latin1')
```

+ Code + Markdown

```
df.head()
```

	textID	text	selected_text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km^2)	Density (P/Km^2)
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral	morning	0-20	Afghanistan	38928346	652860.0	60
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	noon	21-30	Albania	2877797	27400.0	105
2	088c60f138	my boss is bullying me...	bullying me	negative	night	31-45	Algeria	43851044	2381740.0	18
3	9642c003ef	what interview! leave me alone	leave me alone	negative	morning	46-60	Andorra	77265	470.0	164
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative	noon	60-70	Angola	32866272	1246700.0	26

Loading Dataset

1. Load your dataset using pandas (e.g., pd.read_csv() for CSV files).
2. Preprocess the data (cleaning, removing stopwords, etc.).
3. Convert text to numeric features using TfidfVectorizer or CountVectorizer.
4. Train your model and save it along with the vectorizer using pickle or joblib.

CODE walk through

Code Walkthrough

- Model Loading: Load the pre-trained model using pickle or joblib. Example: model = pickle.load(open('model.pkl', 'rb')).
- API Endpoints: Define routes using Flask, e.g., @app.route('/') for home and @app.route('/predict', methods=['POST']) for predictions.
- Request/Response Structure: Accept JSON input via POST, extract text, preprocess it, and return prediction as JSON.
- Error Handling: Use try-except blocks to catch and handle input format errors or model issues. Return proper error messages with HTTP status codes.
- Testing: Use tools like Postman or Python's requests module to send requests to the API and validate responses.

1.What is Logistic Regression?
Logistic Regression is a supervised learning algorithm used for binary classification problems (e.g., spam vs not spam, yes/no, 0/1). It predicts the probability that a given input belongs to a particular class.

How It Works:
It calculates a linear combination of input features:
z=w1x1+w2x2+...+wnxn+bz
Applies the sigmoid function to squash the result between 0 and 1:
The sigmoid function ensures outputs are between 0 and 1, making them interpretable as probabilities.

2.Loss Function:
Logistic Regression uses Binary Cross-Entropy Loss (Log Loss):
Loss=−[ylog(p)+(1−y)log(1−p)] Where p is the predicted probability, and y is the actual label.

Logistic Regression

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

y = encoder.fit_transform(y)
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=1)
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X_train_bow = cv.fit_transform(X_train).toarray()
X_test_bow = cv.transform(X_test).toarray()
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(n_jobs=-1)
lr.fit(X_train_bow,y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceW
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression

LogisticRegression(n_jobs=-1)

Thank You

Dear Reviewer,

I am pleased to present my project titled "Product Classification API using Logistic Regression and Flask", which is available at the following GitHub repository:

GitHub: https://github.com/DipanwitSen/Product_works

This project demonstrates my ability to develop, test, and deploy a machine learning model using a real-world dataset. I implemented a Logistic Regression model to perform binary or multiclass classification, and wrapped it in a RESTful API using Flask. The API accepts JSON input and returns prediction results, making it suitable for integration into larger systems.

The project includes the following key components:

- Preprocessing and training a logistic regression model
- Saving and loading the model using joblib
- Developing Flask endpoints for prediction and health-check
- Implementing robust error handling and request validation
- Planning for deployment, including containerization and scalability

This work showcases my skills in both machine learning and backend API development, with an emphasis on production readiness and clean code structure. I welcome any feedback and look forward to building more such integrated systems.

Sincerely,

Dipanwita Sen

GitHub: https://github.com/DipanwitSen/Product_works