Task 1: Solution overview of ML service

The overview of my solution:

Things to note about the model:

1. In real time there will be more non-spam messages than spam messages, so the data set will be imbalanced. Better handling of imbalanced data set is needed.
2. We have to predict the message, spam or not in real-time. The delay time should be less.
3. Continuous learning should be there, because data drift can happen very frequently. So, the system should have functionality to effective monitoring and re-training.
4. Scalability of model: Is important both at training and prediction time.
5. Make sure to handle the imbalance data set in a better way.

The model training workflow will be like following:

Step 1: Data collection:

1. As mentioned we don't have data from the user before we launch the tool, so we start model training from the publicly available data.
2. Once we start getting data from the users, we should continuously store those data and label it and re-train the model.

Step 2: Data pre-processing:

1. ***Tokenization***: We need to customize our tokenization technique for our need. For example, tokenize the whole link into one token, phone number into one token, the money in one token without removing the rupee symbol(later will be helpful to differentiate money with mobile number).
2. ***Remove unwanted tokens***: Remove the tokens like stop words, phone numbers, links, emojis, special characters.
3. C***ase folding***: Basic case folding to keep everything in same case.

Step 3: Convert text to numbers:

1. ***Text Vectorization:*** Use BOW or TF-IDF vectorization technique to convert the input text into numbers. Here problems are:
   1) word meaning is not captured.
   2) Order of the words are lost.
   3) Very huge dimension of dataset with lots of zeros.
2. ***Text Embedding***: Go for embedding when we want to capture the relationship between the words in the sentences. This makes the prediction better and dimensions very high. We will go for this, to get a better understanding of the input text.

Step 3: Target analysis:

1. ***Imbalance target check***: In real-world case, there will be a lot of non-spam SMSs than spam SMSs. we have to check for the class imbalance. If we have imbalance we have to try over-sampling and under-sampling or weighted ML models for better results.

Step 4: Feature Analysis:

1. ***Dimensional check***: Most of the NLP tasks will have high dimensions. We have to check the dimension of the input data after the text data is converted to numeric numbers, if that is too high we should do dimensionality reduction.
   1.1- PCA => we can use this when the input data is dense numeric values.
   1.2- SVD => we should use this when the input data is sparsh data(having lots of zeros).

Step 5: Model selection:

1. ***Basic models***: Try basic models like logistic Regression and SVM as a base model.
2. ***Moderate complex***: Try Ensemble or XGBoost models.
3. ***Complex models***: As when you have more data set available you can go for complex models like LSTM. we can try transformer models like BART to make the prediction.
4. Keep in mind, the inference time should not be too high, prediction should be real-time.

Step 6: Model Evaluation:

1. ***If imbalanced data set?*** : In this case, big no to accuracy. Go for recall and precision and f1-score.
2. ***If balanced data set?*** : Including accuracy check recall, precision, f1-score and confusion matrix to understand the prediction well.
3. ***Why f1_score?:*** Here we care about the False Negative(FN) and False Positive(FP).
   3.1) FP: Non-spam messages are predicted as spam. There are chances to miss legitimate SMS. This could lead to miss important SMS messages (like Bank withdraw message).
   3.2) FN: Spam messages are predicted as non-spam. it’s a dangerous issue, people can trust this and may make a money transaction, etc.

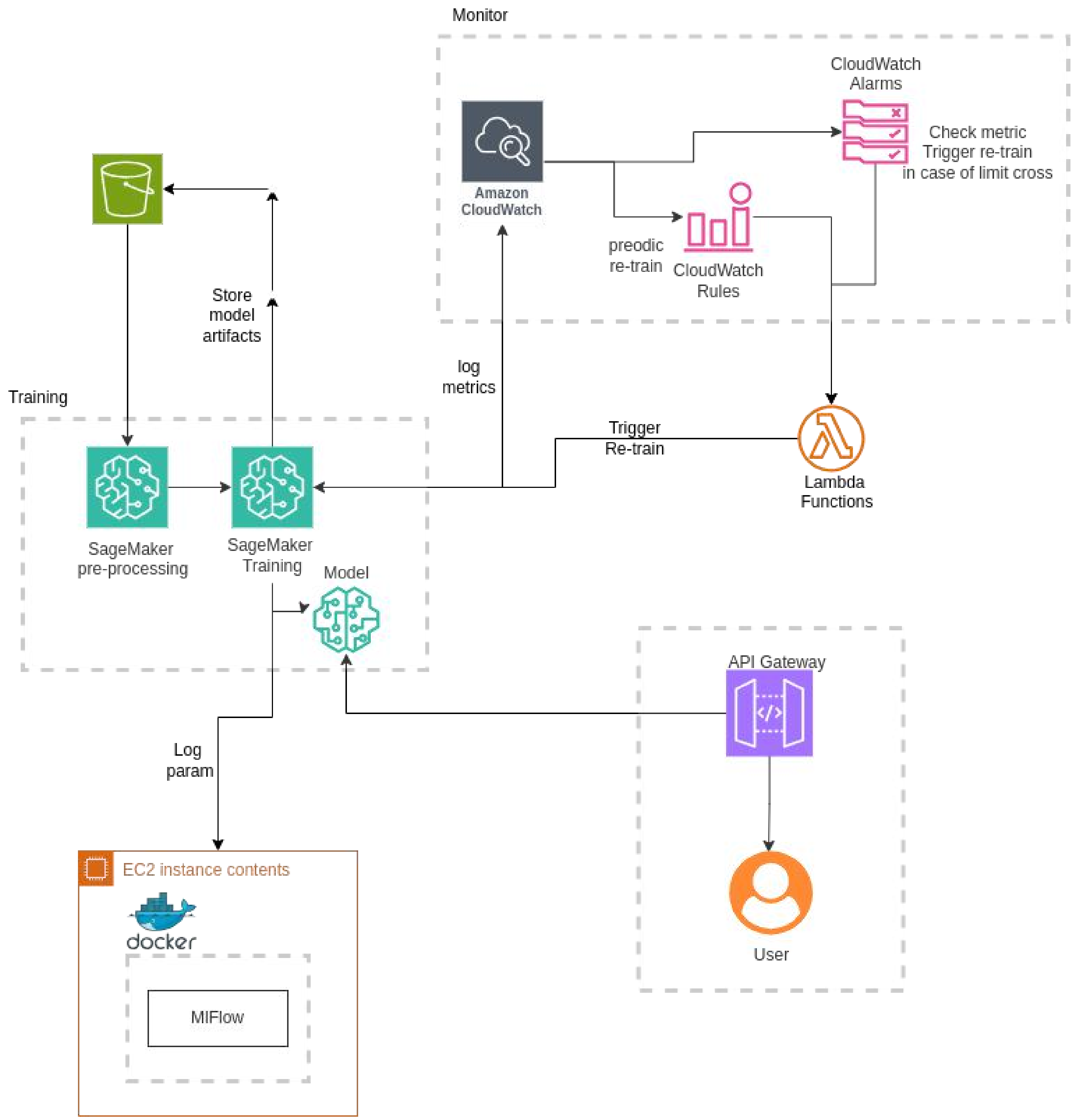
Step 7: Model Monitoring:

1. Monitor for data drift and re-train the model in that case.
2. We will get more data once we launch our model, so it’s good to re-train our model periodically.

Tech Stack:

- **S3 storage** - For storing our data and model artifacts.
- **Amazon SageMaker** - For pre-processing, training and deploying our model.
- **MlFlow** - For tracking our model performance and versioning of models and data sets.
- **EC2 instance and Docker** - To run our Mlflow inside it.
- **Lambda functions** - To trigger the re-training pipeline of SageMaker.
- **CloudWatch** - To store all the model prediction logs, metrics like precision, recall, F1-score, population Stability Index.
- **CloudWatch Alarms** - To trigger the Lambda function when a particular metric goes over the threshold limit.
- **Amazon API Gateway** - To make the API call to the deployed model.
- **Programming Language** - Python.
- **Code version control** - git.

Work Flow Diagram:


Monitor
CloudWatch Alarms
Check metric Trigger re-train in case of limit cross
Amazon CloudWatch
preodic re-train
CloudWatch Rules
Store model artifacts
log metrics
Training
Trigger Re-train
Lambda Functions
SageMaker pre-processing
SageMaker Training
Model
API Gateway
Log param
EC2 instance contents
docker
MlFlow
User