**Fraud Detection System**

**Project Description**

This is a fraud-detection prediction system for predicting fraud in transactions based on the many features involved. It uses functionalities of exposure given by pre-trained features of machine learning models to come up with predictions. Other steps incorporated are data preprocessing, feature engineering, model training, evaluation, and deployment through a user-friendly web application.

Dataset The project contains features such as data on the transaction; it was downloaded from Kaggle, where there are different discrete attributes. So, the dataset is anonymized with those labels, some of which are "fraud" and "not fraud".

Link of the dataset: [Credit Card Fraud Detection (kaggle.com)](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

**Pipeline Steps**

**1. Data Preprocessing**

Load Data: Read the dataset from a CSV file.

Handle Missing Values: Ensure there are no missing values in the dataset.

A screenshot of a computer

Description automatically generated

**2. Feature Engineering**

Feature Selection: Select relevant features for the model.

We have selected and extracted some fraud transaction to know our dataset better and try at the end at our user-friendly interface and test the accuracy of the system   
A screenshot of a computer

Description automatically generated

**3. Model Training**

Split Data: Divide the data into training and testing sets.



Train Model: Use a classifier (e.g., Random Forest) to train the model.

A black and white image of a computer code

Description automatically generated with medium confidence

Save Model: Serialize the trained model for deployment.

A close up of a computer code

Description automatically generated

After splitting our data into training and testing, we have used a randomforest classifier to train our model and get a high accuracy and then after achieving a high accuracy, precision and f1 score. We decided to use this model to our system as we got 99.9% accuracy and then we saved our model into a pkl format to then use it into our flask application.

**4. Evaluation**

Evaluate Model: Assess the model's performance using metrics like accuracy, precision, recall, and F1-score.

Save Metrics: Store the evaluation metrics for future reference into a models folder that is in our main folder.

A screenshot of a computer screen

Description automatically generated

**5. Folder Setup**

Our main folder is named Final\_MLOPS and it contains folders inside (ex: Models, data, templates)

├── app.py this is our Main application script

├── Dockerfile & this is the Dockerfile for containerizing the application

├── final.ipynb our jupyter notebook for model training and preprocessing

├── models

│ ├── model.pkl model saved

│ ├── scaler.pkl scaler that was used in preprocessing

│ ├── metrics.pkl evaluation metrics that was saved

├── requirements.txt dependencies that was used

├── templates

│ └── index.html HTML & CSS template for the web interface

└── visualize.py data visualization script

A screenshot of a computer

Description automatically generated

**6. Deployment**

Web Application: We have then developed a Flask web application to serve our model and build a user friendly web interface as was asked from us.

from flask import Flask, request, jsonify, send\_file, render\_template

import numpy as np

import pickle

import matplotlib.pyplot as plt

import pandas as pd

app = Flask(\_\_name\_\_)

# Load the model, scaler, and metrics

with open('models/model.pkl', 'rb') as model\_file:

    model = pickle.load(model\_file)

with open('models/scaler.pkl', 'rb') as scaler\_file:

    scaler = pickle.load(scaler\_file)

with open('models/metrics.pkl', 'rb') as metrics\_file:

    metrics = pickle.load(metrics\_file)

we did our main code in the app.py and then did HTML template page and css in the index.html and then downloading every dependency needed in the requirements.txt after that we checked if our web application works by running the command python app.py

A screenshot of a computer error

Description automatically generated

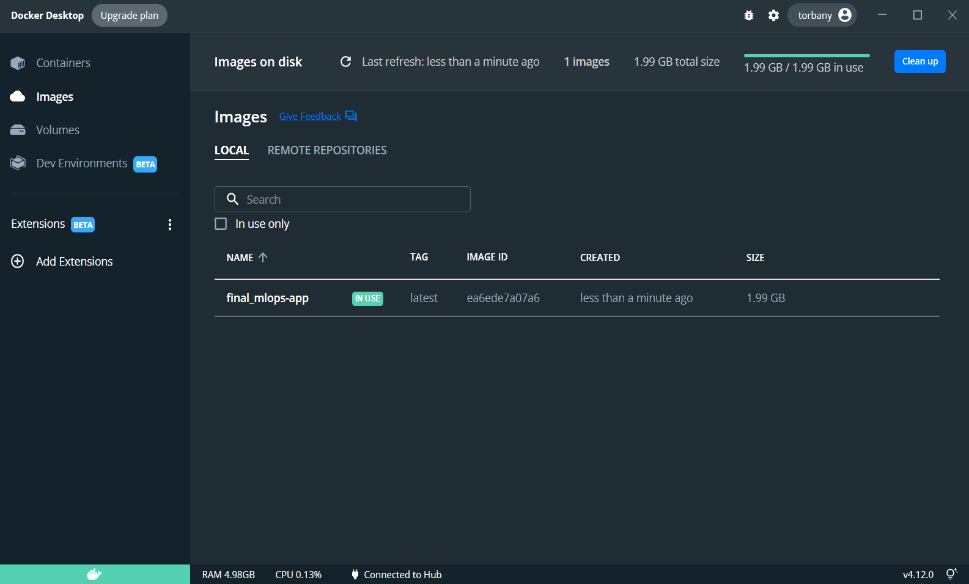
And it worked well. First the web application wasn’t user friendly but we managed to do css template to the user to ease the work for him and we made a dropdown menu also for some fraud and non fraud transactions to test if the app works and then it worked perfectly and then also added a description about our project and how to use the system.

Docker: Containerize the application using Docker for easy deployment and host it from our own desktop, after finshing all of the above we wanted now to make the docker step by building a docker image with this command.

docker build -t final\_mlops-app

and then after installing everything the image appeared in the docker desktop and now we must run the docker container after making sure everything works well by the command

docker run -d -p 5000:5000 final\_mlops-app



And now we can start our website from our localhost 5000 by making sure the container is running in the docker desktop.

<http://localhost:5000>

and these are some docker commands

Docker Commands

List Docker Images:

docker images

List Running Containers:

docker ps

Stop a Running Container:

docker stop <container\_id>

Remove a Container:

docker rm <container\_id>

Remove an Image:

docker rmi <image\_id>

**Guidelines for Contributors**

We look forward to receiving contributions from you! Here are some of the things you can help with:

Our folder will be publicly available on git hub also for people to use our system & for any bugs or errors you can email us: [saeed.eltorbany21d@eslsca.edu.eg](mailto:saeed.eltorbany21d@eslsca.edu.eg) , [Amr.Amer21d@eslsca.edu.eg](mailto:Amr.Amer21d@eslsca.edu.eg)

Report Bugs: Open a new issue and fill out the details regarding a bug.

Fix the bug, fork the repository, and submit a pull request.

Improved sighting: Recommendation of additional functionalities or Improvements. Documentation: Better Documentation and README file.

**License**:

This project is licensed under the DASAA License which stands for (Dima, Amr, Saeed, Abdelrahman & Ahmed).

Kaggle dataset: [Credit Card Fraud Detection (kaggle.com)](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

Other resources: MLOPS Lectures.