Deployment

To deploy our model, the python Flask library was used to create a local API.

Within the deployment, there are three main files: settings.py, app.py, and testing_app.py

settings.py: It has variables specific to the problem, which will be used within app.py.

```
columnas_f = ['monto',
 'hora',
 'linea_tc',
 'interes_tc',
 'is_prime',
 'dcto',
 'cashback',
 'device_score',
 'dia',
 'genero_--',
 'genero_F',
 'genero_M',
 'establecimiento_Abarrotes',
 'establecimiento_Farmacia',
 'establecimiento_MPago',
 'establecimiento_Restaurante',
 'establecimiento_Super',
 'establecimiento_na',
 'ciudad_Guadalajara',
 'ciudad_Merida',
 'ciudad_Monterrey',
 'ciudad_Toluca',
 'ciudad_na',
 'tipo_tc_Física',
 'tipo_tc_Virtual',
 'status_txn_Aceptada',
 'status_txn_Rechazada',
 'os_%%',
 'os_.',
 'os_ANDROID',
 'os_WEB']
list_categoricas = ['genero', 'establecimiento', 'ciudad', 'tipo_tc', 'status_txn', 'os']
```

app.py: Contains the deployment of the model.

```python
from flask import Flask
import redis
import time
import pandas as pd
from settings import columnas_f, list_categoricas
from joblib import dump, load
from flask_restful import Api, Resource, reqparse
import json
import numpy as np

#Inicializando Flask application

app = Flask(__name__)
API_app = Api(app)

model = load('logistic_regressor.joblib')
scaler = load('scaler.joblib')
columnas = ['genero', 'monto', 'hora', 'establecimiento', 'ciudad',
       'tipo_tc', 'linea_tc', 'interes_tc', 'status_txn', 'is_prime', 'dcto',
       'cashback', 'device_score', 'os', 'dia', 'fecha']

X_solve = pd.DataFrame(np.zeros((1, len(columnas_f))), columns = columnas_f)

@app.route("/")
def homepage():
    return "Bienvenido a la API de detección de fraude"

class Predict(Resource):

    @staticmethod
    def post():
        parser = reqparse.RequestParser()
        parser.add_argument('genero')
        parser.add_argument('monto')
        parser.add_argument('hora')
        parser.add_argument('establecimiento')
        parser.add_argument('ciudad')
        parser.add_argument('tipo_tc')
        parser.add_argument('linea_tc')
        parser.add_argument('interes_tc')
        parser.add_argument('status_txn')
        parser.add_argument('is_prime')
        parser.add_argument('dcto')
        parser.add_argument('cashback')
        parser.add_argument('device_score')
        parser.add_argument('os')
        parser.add_argument('dia')
        args = parser.parse_args()

        #X_solve = pd.DataFrame(args.values(), columns = columnas)

        X = args


        for key, value in X.items():
            if key in list_categoricas:
                key_m = key + '_' + value
                X_solve.loc[0, key_m] = 1
            else:
                X_solve.loc[0, key] = value


        df_solve = scaler.transform(X_solve.values)

        predicciones = model.predict(df_solve)
        print(predicciones)
        out = {'Predicciones': str(predicciones[0])}

        return out, 200

API_app.add_resource(Predict, '/predict')

if __name__ == '__main__':
    app.run(debug=True )
```

To deploy the model, all you must do is open a terminal with Python and run the command python app.py.

The server will run on the address http://127.0.0.1:5000/

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

testing_app.py: This file contains an example script, to send a request to the server with our data and return the prediction of the model.

To run, just type in a terminal with Python: python testing_app.py

```python
import requests
url = 'http://127.0.0.1:5000/predict'

test = {'genero': 'F',
 'monto': 608.3456335342977,
 'hora': 20,
 'establecimiento': 'Super',
 'ciudad': 'Merida',
 'tipo_tc': 'Física',
 'linea_tc': 71000,
 'interes_tc': 51,
 'status_txn': 'Aceptada',
 'is_prime': 0,
 'dcto': 60.834563353429786,
 'cashback': 5.4751107018086795,
 'device_score': 3,
 'os': 'ANDROID',
 'dia': 1}
```

Response

```
<Response [200]>
{'Prediction': '0'}
```