Nombre: Implementacion del modelo Iris

Código de lote: 001

Fecha de envío: 26/10/2024

```
from sklearn.datasets import load_iris
import pandas as pd

iris= load_iris()
df= pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target']= iris.target

print(df.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import joblib
```

```
X_train, X_test, y_train, y_test= train_test_split(df[iris.feature_names], df['target'], test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
joblib.dump(model, 'C:/Users/kfall/Desktop/Proyecto3/model.pkl')
```

```
['C:/Users/kfall/Desktop/Proyecto3/model.pkl']
```

```
C: > Users > kfall > Desktop > Proyecto3 > main.py > predict
from flask import Flask, request, jsonify
import joblib
import os

app = Flask(__name__)

model = joblib.load('C:/Users/kfall/Desktop/Proyecto3/model.pkl')
# we do the route
@app.route('/', methods=['GET'])
def home():
    return jsonify({"Mensaje ": "Bienvenido a la API de predicción"}), 200

# Path to make predictions using the loaded model
@app.route('/predict', methods=['POST'])
def predict():
    if model is None:
        return jsonify({"error": "El modelo no ha sido cargado."}), 400

    data = request.get_json()
    if not data or 'features' not in data:
        return jsonify({"error": "No se enviaron datos válidos"}), 400

    features = data['features']

    try:

        prediction = model.predict([features])
        return jsonify({"prediction": prediction.tolist()}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500

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

```
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
from datetime import datetime

def create_pdf():
    c = canvas.Canvas("implementacion.pdf", pagesize=letter)
    c.drawString(100, 750, "Nombre: Implementacion del modelo Iris")
    c.drawString(100, 730, "Código de lote: 001")
    c.drawString(100, 710, f"Fecha de envío: {datetime.now().strftime('%d/%m/%Y')}")
    c.drawString(100, 690, "Enviado a: Repositorio de GitHub")


    x = 100
    width = 400
    height = 200
    y_positions = [500, 280, 60]


    c.drawImage("modelo.png", x, y_positions[0], width, height)
    c.drawImage("main.png", x, y_positions[1], width, height)
    c.drawImage("pdf.png", x, y_positions[2], width, height)


    c.save()

create_pdf()
```