Transpiler un mod?le en C : r?gression lin?aire

December 15, 2021

1 Imports

[1]:
```
import pandas as pd
import joblib
import os
from sklearn.linear_model import LinearRegression
```

2 Chargement du dataset et du modèle

[2]:
```
# Chargement du dataset
df = pd.read_csv("tumors.csv")
print(df.head())

X = df[["size", "p53_concentration"]]
y = df["is_cancerous"]


# Séparation du dataset en training et testing sets
X_train = X[:-10]
X_test = X[-10:]

y_train = y[:-10]
y_test = y[-10:]

# Chargement du model
model = LinearRegression()

model.fit(X_train, y_train)

# Sauvegarde du model
joblib.dump(model, f"model.joblib")
```

```
       size  p53_concentration  is_cancerous
0 -0.004165           0.001785             1
1  0.012898           0.001899             1
```

```
2  0.013674          0.001193            1
3  0.008774          0.003673            0
4  0.009751          0.005571            0
```

[2]:
```
['model.joblib']
```

[3]:
```python
def produce_linear_regression_c_code():

    model = joblib.load('model.joblib')

    # Thetas
    n_thetas = len(model.coef_) + 1
    thetas = f"{model.intercept_}f,"
    for coef in model.coef_:
        thetas += str(coef) + "f,"
    thetas = thetas.strip(",")

    prediction_code = f"float thetas[{n_thetas}] = {{{thetas}}};"

    # Features
    features=""
    for i in range(X_test.shape[0]):
      to_predict = X_test.iloc[i].tolist()
      feature = "{"
      for value in to_predict:
          feature += str(value) + "f,"
      features += feature[:-2]
      features += "},\n"

    n_sample = X_test.shape[0]
    n_feature = X_test.shape[1]

    # Code
    code = f"""
    #include <stdio.h>

    {prediction_code}
    float prediction(float *features, int n_feature)
    {{
        float res = thetas[0];

        for (int i = 0; i < n_feature; ++i)
            res += features[i] * thetas[i+1];

        return res;
    }}
    int main()
    {{
        float features[{n_sample}][{n_feature}] = {{{features}}};

        for (int i = 0; i < {n_sample}; ++i) {{
            printf("%f\\n", prediction(features[i], 2));
        }}

        return 0;
    }}
    """



    with open("transpiler.c", "w") as f:
        f.write(code)
```

[4]:
```
produce_linear_regression_c_code()

!gcc transpiler.c -O3 -o transpiler
```

[5]:
```
print("Modèle transpilé:")
!./transpiler
```

```
Modèle transpilé:
0.485140
-1.236517
0.417540
0.787183
0.768821
0.747259
-0.069009
0.386354
0.679163
0.804862
```

[6]:
```
print('Modèle non-transpilé:')
for i in model.predict(X_test):
  print(i)
```

```
Modèle non-transpilé:
0.48513968527496
-1.2365165909081623
0.4175400010067567
0.7871834652335001
0.7688207284138135
0.7472595589533835
-0.06900933399957943
0.38635404526424
0.6791629943084592
0.8048620370086736
```

Les prédictions produites par le modèle transpilé sont bien conformes.