Validation de TP2&TP3
~COMPUTER VISION~

Réalisé par :
- Ichrak Ben Saad
- Aya Fekih Romdhane

Classe :3DNI01

Enseignante:Mme Olfa Besbes

Année Universitaire : 2022/2023

“FACE DETECTION APPLICATION”

1.Feature Engineering

2.Binary Classification using Scikit-Learn

3.Evaluating the Best Face Detector

4.Finding Faces in a New Image

5.Model Deployment with Python and Streamlit

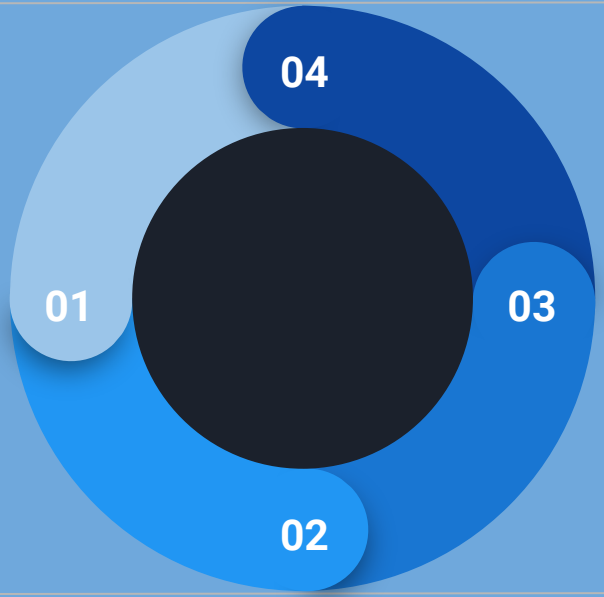
1. Feature Engineering

Dans un premier temps, nous réalisons feature engineering en extrayant l'histogramme d'orientation

Fonctions Gradients (HOG) à l'aide de la bibliothèque Scikit-Image.

sklearn

skimage



sklearn.feature_extraction.image

matplotlib.pyplot

2. Binary Classification using Scikit-Learn

Divisé les données en ensembles de formation et de test

```
# Split the data into training and test sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,  test_size=0.2, random_s
tate=0)

# View the shape of the data
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

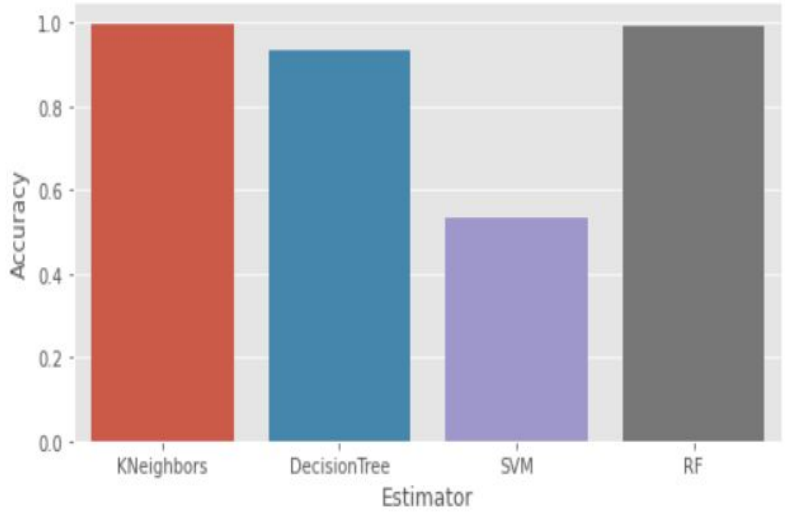
Construisons différents classificateurs binaires avec le meilleur estimateur pour chacun :

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
```

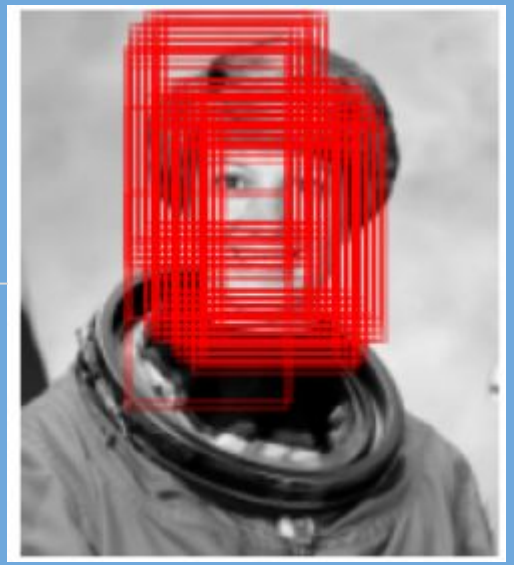
```python
# Create model
clf = model["estimater"]
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = clf, param_grid = model["params"],
                           cv = 5)

# Fit the model
grid_search.fit(X_train, y_train);

# Make predictions on the test set compute accuracy metric
predicted = grid_search.predict(X_test)
acc = accuracy_score(predicted, y_test)
entries.append(acc)

print(grid_search.best_params_)

# Get the best model with the highest accuracy
if acc > max_acc:
  max_acc = acc
  best_model = grid_search
```

```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
df = pd.DataFrame({"Estimator": column_names, "Accuracy": entries})
plt.figure(figsize=(8, 4))
sns.barplot(x='Estimator', y='Accuracy', data=df)
print(df)
```

```
      Estimator  Accuracy
0    KNeighbors  0.997922
1  DecisionTree  0.936154
2           SVM  0.533245
3            RF  0.989611
```



Finding Faces in a New Image



`skimage.data`

```
labels = best_model.predict(patches_hog)
```

Face Detection App

Ichrak Ben Saad & Aya Fekih Romdhane

Face Detection

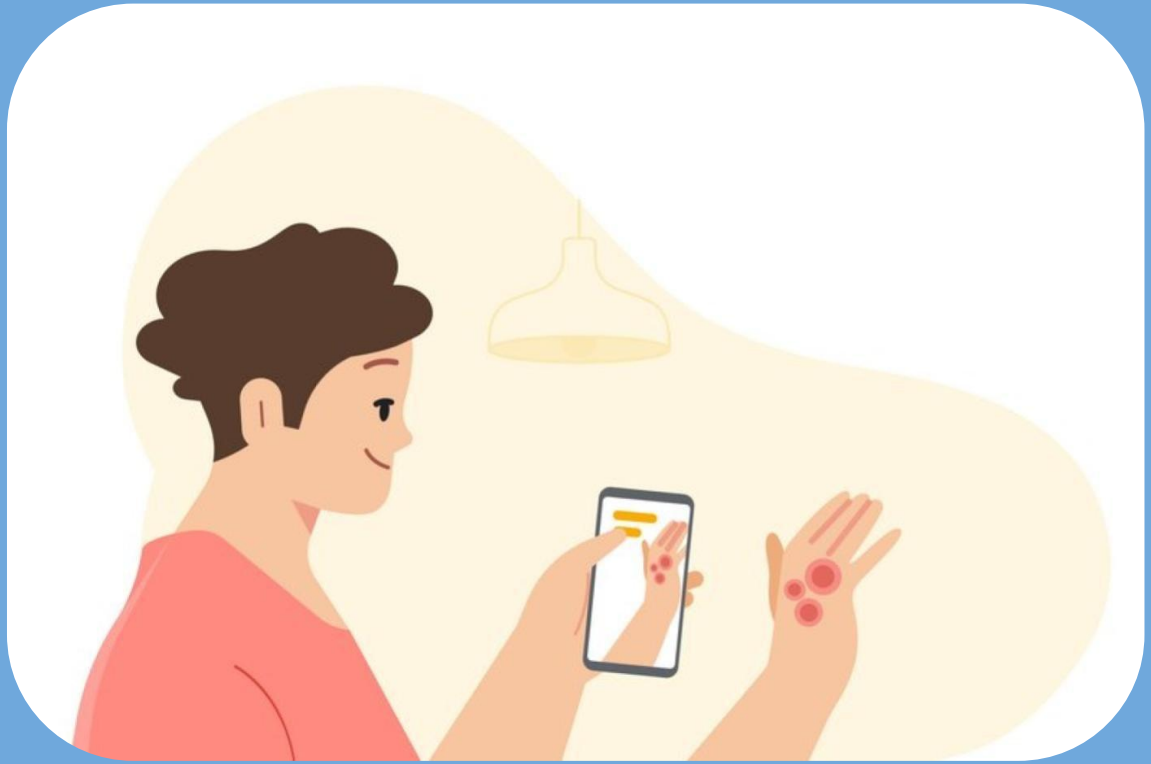
Upload File

Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files

test3.PNG 28.8KB

Process

TP3 : DÉTECTION DU CANCER DE LA PEAU AVEC L'APPLICATION ANDROID



Le projet est principalement divisé en deux parties

01 **TensorFlow** & **Keras** sont utilisés pour construire et créer un modèle de machine learning.

02 **TFLite** est utilisé pour déployer le modèle sur une application Android.

Nos tâches pour la Training model

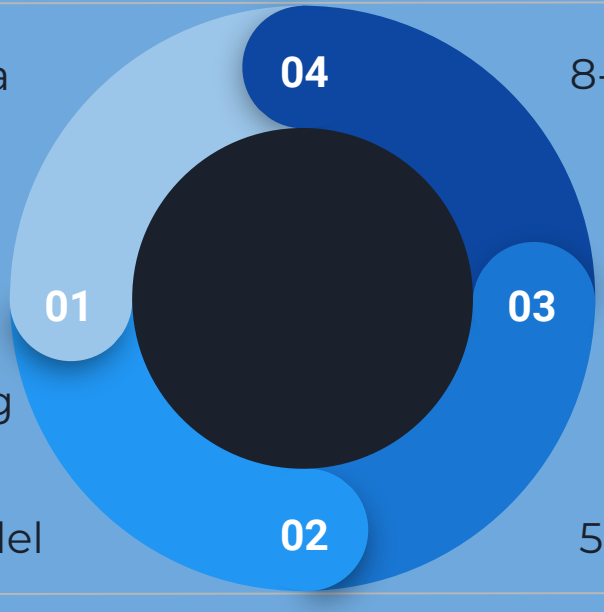
9-Export as saved model and convert to TFLite

1-Loading the data

2-Transfer Learning with TensorFlow hub

3-Data Preprocessing

4-Build the model



8-Confusion Matrix

7-Test at random

6-Checking Performance

5-Training Model

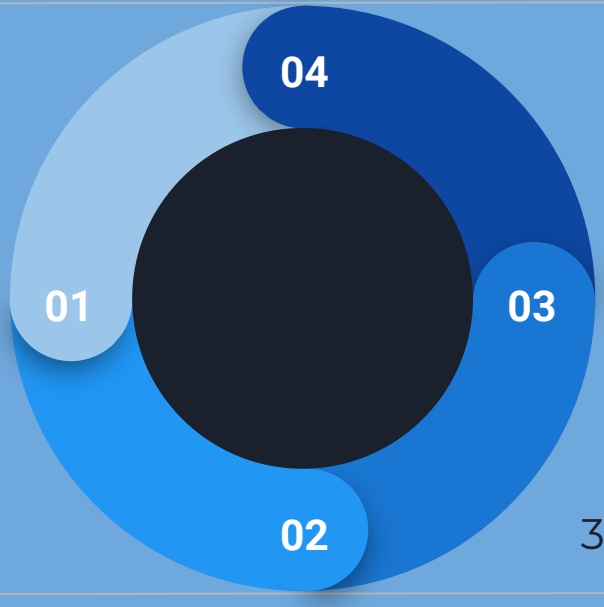
Android App Part

Nous utiliserons TFLite pour créer une application Android de bout en bout pour ce projet. Nous avons décidé de créer une application Android qui détecte le cancer de la peau.

Nos tâches pour l’application mobile

1-Load the model in our Android project

2-Add TFLite dependency to app/build.gradle file



5-convert the preprocessed bitmap into ByteBuffer

4-Recognition is our result data class

3-create a classifier


DETECT
Benign


Thanks
for your
attention