

Sports Person Classifier



LIONEL
MESSI



MARIA
SHARAPOVA



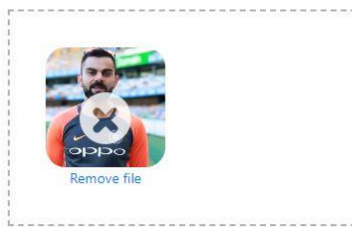
ROGER
FEDERER



SERENA
WILLIAMS



VIRAT KOHLI



Classify



VIRAT KOHLI

Player	Probability Score
Leonel Messi	0.4
Maria Sharapova	0.2
Rodger Federer	0.11
Serena Williams	0.07
Virat Kohli	99.22

Sports Celebrity Image Classification

01.11.2021

Arroud Yassir

Objectifs

Sommaire

- 1) Introduction
- 2) Collection et préparation de données
- 3) Entraînement des modèles de classification
- 4) Data engineering & déploiement du projet

Grandes étapes

I. Introduction

Notre projet vise à classer des personnalités sportives selon la détection faciale, on se restreint seulement à 5 sportifs fameux(...) en raison de ne pas ralentir l'entraînement et le déploiement de notre modèle.

Les technologies utilisées dans ce projet sont:

- Python Numpy et OpenCV pour le nettoyage des données
- Matplotlib & Seaborn pour la visualisation des données
- Sklearn pour la construction de modèles
- Bloc-notes Jupyter, code de studio visuel et pycharm en tant qu'IDE Flacon Python pour serveur http
- HTML/CSS/Javascript pour l'interface utilisateur

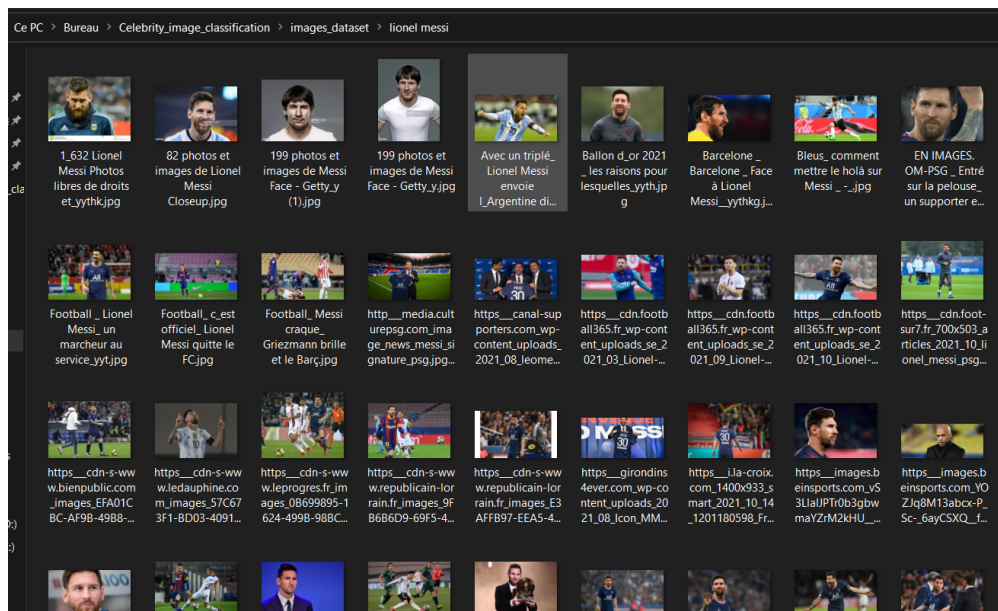
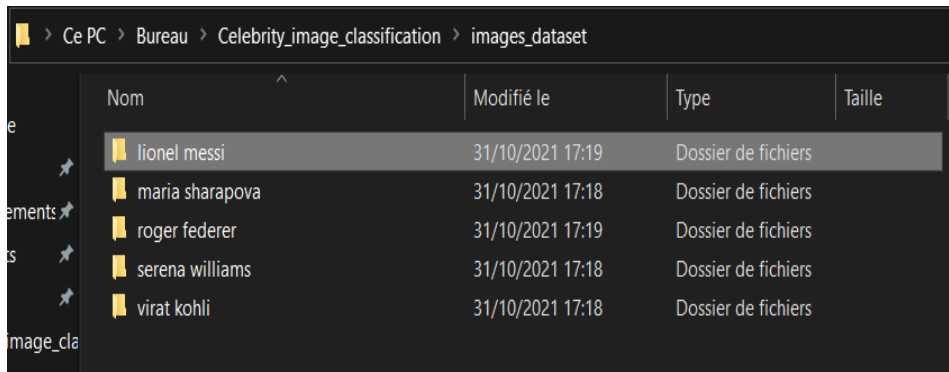
II. Collection et préparation de données

a) Collection de données

Notre jeu de données dans ce projet est l'ensemble des images de nos sportifs, afin de construire cette dataset, on doit charger les images de l'internet.

Une panoplie de manières est proposée pour la réalisation de cette tâche. Or, on opte pour l'extension "fatkun" de google chrome pour télécharger nos images.

Nos résultats seront sous cette forme là :



b) Préparation de données

Détecter le visage et les yeux

Lorsque nous regardons une image, la plupart du temps, nous identifions une personne à l'aide d'un visage. Une image peut contenir plusieurs visages, le visage peut également être obstrué et pas clair. La première étape de notre pipeline de prétraitement consiste à détecter les visages à partir d'une image. Une fois le visage détecté, nous détecterons les yeux, si deux yeux sont détectés, nous gardons uniquement cette image, sinon nous la rejetons. Maintenant, **comment on peut détecter le visage et les yeux ?**

A partir d'OpenCV, on va utiliser les "haar cascades"

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html?highlight=haar

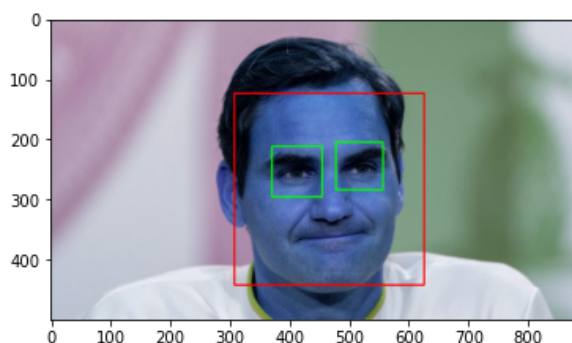
```
[ ] face_cascade = cv2.CascadeClassifier('./opencv/data/haarcascades/haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier('./opencv/data/haarcascades/haarcascade_eye.xml')

    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    faces

array([[308, 124, 318, 318]])
```

```
[ ] cv2.destroyAllWindows()
    for (x,y,w,h) in faces:
        face_img = cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = face_img[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray)
        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color, (ex,ey), (ex+ew,ey+eh), (0,255,0),2)

    plt.figure()
    plt.imshow(face_img, cmap='gray')
    plt.show()
```

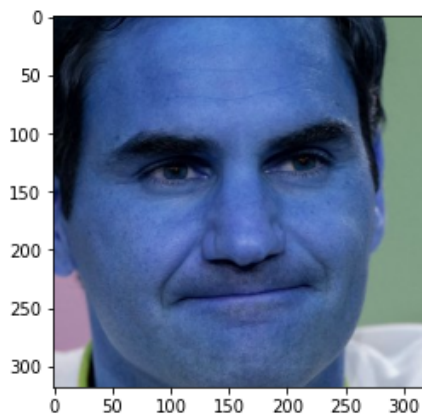


On définit ensuite une fonction `get_cropped_image_if_2_eyes()`, qui charge l'image, détecte le visage. Si yeux ≥ 2 , enregistre et recadre le visage region renvoie l'image recadrée (si le visage et les yeux ≥ 2 sont détectés).

```
def get_cropped_image_if_2_eyes(image_path):
    img = cv2.imread(image_path)
    if(img is not None):
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (x,y,w,h) in faces:
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = img[y:y+h, x:x+w]
            eyes = eye_cascade.detectMultiScale(roi_gray)
            if len(eyes) >= 2:
                return roi_color
        else:
            exit(1)

[ ] cropped_image = get_cropped_image_if_2_eyes('./test_images/https__images.ladepeche.fr')
plt.imshow(cropped_image)
```

<matplotlib.image.AxesImage at 0x242b4863fd0>



N'oublions pas que notre but est de préparer nos données (X et y) afin de les présenter comme entrée pour nos algorithmes de classification:

- X : les images des célébrités recadrées et redimensionnées.
- y : labels ou variables de réponse indiquant l'identité du sportif.

En premier lieu, on recadre tous nos images de façon à ne laisser que le visage clair et net de la célébrité, le code suivant réalise cette tâche en créant un nouveau dossier "cropped" contenant un dossier par célébrité ou toutes les images dedant sont bien redimensionnées

```

▶ cropped_image_dirs = []
celebrity_file_names_dict = {}
for img_dir in img_dirs:
    count = 1
    celebrity_name = img_dir.split('/')[-1]
    celebrity_file_names_dict[celebrity_name] = []

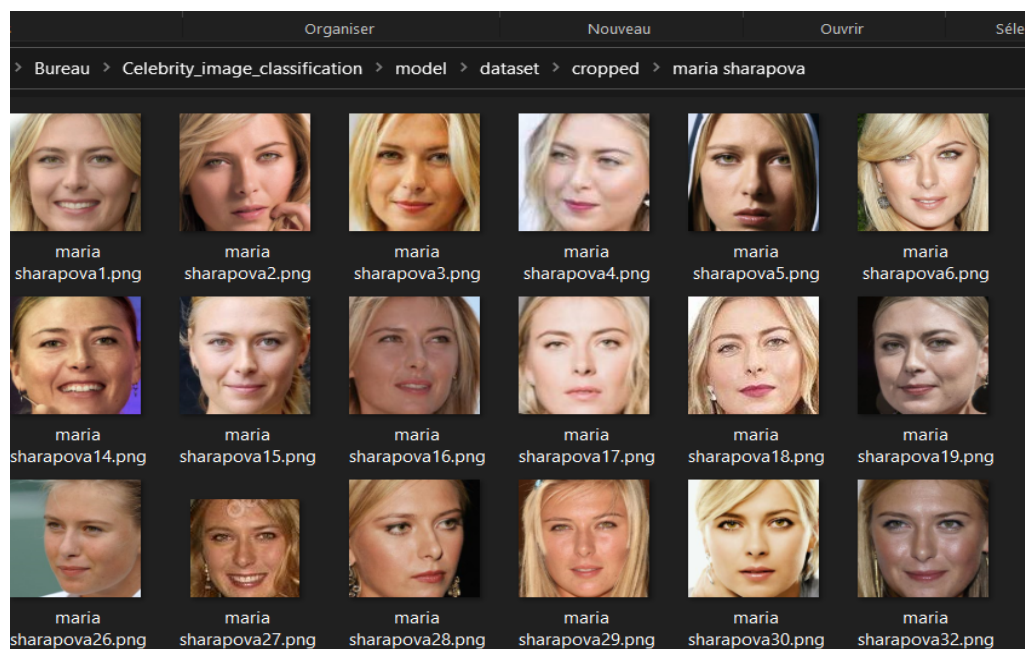
    for entry in os.scandir(img_dir):
        new_entry = entry.path.replace("\\", "/")
        roi_color = get_cropped_image_if_2_eyes(new_entry)
        if roi_color is not None:
            cropped_folder = path_to_cr_data + celebrity_name
            if not os.path.exists(cropped_folder):
                os.makedirs(cropped_folder)
            cropped_image_dirs.append(cropped_folder)
            print("Generating cropped images in folder: ",cropped_folder)

            cropped_file_name = celebrity_name + str(count) + ".png"
            cropped_file_path = cropped_folder + "/" + cropped_file_name

            cv2.imwrite(cropped_file_path, roi_color)
            celebrity_file_names_dict[celebrity_name].append(cropped_file_path)
            count += 1

Generating cropped images in folder: ./dataset/cropped/lionel messi
Generating cropped images in folder: ./dataset/cropped/maria sharapova
Generating cropped images in folder: ./dataset/cropped/roger federer
Generating cropped images in folder: ./dataset/cropped/serena williams
Generating cropped images in folder: ./dataset/cropped/virat kohli

```



On arrive à voir que maintenant nos images sont bien recadrés en ne gardant que le visage!

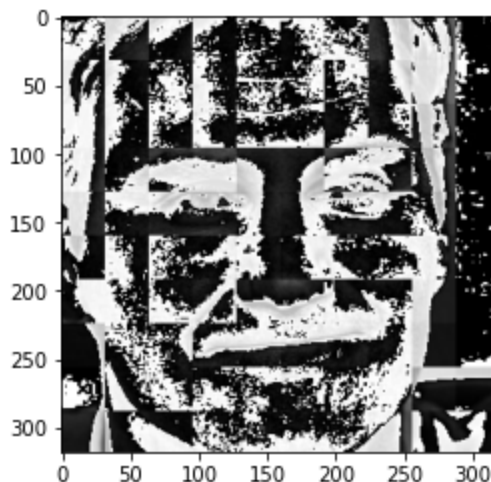
Wavelet transformation :

Maintenant nous allons faire la transformation en ondelettes (Wavelet transformation) qui nous permet d'extraire les caractéristiques importantes de nos images. On utilisera la transformation en ondelettes comme caractéristique pour former notre modèle Dans l'image transformée en ondelettes, vous pouvez voir clairement les bords et cela peut nous donner des indices sur diverses caractéristiques du visage telles que les yeux, le nez, les lèvres, etc.

On définit une fonction `w2d()` qui réalise la transformation en ondelettes comme ceci pour une simple image de Roger Federer

```
[ ] im_har = w2d(cropped_image, 'db1', 5)
plt.imshow(im_har, cmap='gray')
```

<matplotlib.image.AxesImage at 0x242b397c850>



en créant cette image effrayante en noir et blanc, vous extrayez les caractéristiques importantes (faciale) ça marche comme ça : pour détecter le visage de federer, il a une certaine taille d'yeux, a-t-il une barbe ou pas ? quelle est la taille de son nez...

Félicitations! Nos données sont presque pretes, il ne suffit que d'exécute ce code ci dessous

```

] # Images in cropped folder can be used for model training. We will use these raw images along with
# wavelet transformed images to train our classifier. Let's prepare X and y now
X, y = [], []
for celebrity_name, training_files in celebrity_file_names_dict.items():
    for training_image in training_files:
        img = cv2.imread(training_image)
        scaled_raw_img = cv2.resize(img, (32, 32))
        img_har = w2d(img, 'db1', 5)
        scaled_img_har = cv2.resize(img_har, (32, 32))
        combined_img = np.vstack((scaled_raw_img.reshape(32*32*3,1), scaled_img_har.reshape(32*32,1)))
        X.append(combined_img)
        y.append(class_dict[celebrity_name])

```

```

[ ] X = np.array(X).reshape(len(X), 4096).astype(float)
    X.shape

```

```

(255, 4096)

```

X est de taille (255, 4096) alors que notre variable de réponse y est sans surprise de (255,1)

III. Entrainement des modèles de classification

a. Première tentative

On commence par utiliser un SVM classeur avec " rbf " kernel juste pour tenter

```

# we randomly choose the param for svc, we'll fine tune em later
pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC(kernel = 'rbf', C = 10))])
pipe.fit(X_train, y_train)
pipe.score(X_test, y_test)

```

```

0.734375

```

```

[ ] print(classification_report(y_test, pipe.predict(X_test)))

```

	precision	recall	f1-score	support
0	0.71	0.92	0.80	13
1	0.67	0.62	0.64	13
2	1.00	0.33	0.50	6
3	0.67	0.50	0.57	12
4	0.79	0.95	0.86	20
accuracy			0.73	64
macro avg	0.77	0.66	0.68	64
weighted avg	0.74	0.73	0.72	64

C'est déjà pas mal ! mais on cherche quelque chose de mieux :)

b. GridSearch CV

Utilisons Grid Search pour essayer différents modèles avec différents paramètres. L'objectif est de proposer le meilleur modèle avec les meilleurs paramètres affinés

```
model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto', probability=True),
        'params': {
            'svc__C': [1, 10, 100, 1000],
            'svc__kernel': ['rbf', 'linear']
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params': {
            'randomforestclassifier__n_estimators': [1, 5, 10]
        }
    },
    'logistic_regression': {
        'model': LogisticRegression(solver='liblinear', multi_class='auto'),
        'params': {
            'logisticregression__C': [1, 5, 10]
        }
    }
}
```

```
scores = []
best_estimators = {}
import pandas as pd
for algo, mp in model_params.items():
    pipe = make_pipeline(StandardScaler(), mp['model'])
    clf = GridSearchCV(pipe, mp['params'], cv=5, return_train_score=False)
    clf.fit(X_train, y_train)
    scores.append({
        'model': algo,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
    best_estimators[algo] = clf.best_estimator_

df = pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])
df
```

On obtient comme résultat :



	model	best_score	best_params
0	svm	0.806073	{'svc__C': 1, 'svc__kernel': 'linear'}
1	random_forest	0.665182	{'randomforestclassifier__n_estimators': 10}
2	logistic_regression	0.801350	{'logisticregression__C': 5}

```
[ ] best_estimators
```

```
{'svm': Pipeline(steps=[('standardscaler', StandardScaler()),
                        ('svc',
                         SVC(C=1, gamma='auto', kernel='linear', probability=True))]),
 'random_forest': Pipeline(steps=[('standardscaler', StandardScaler()),
                                   ('randomforestclassifier',
                                    RandomForestClassifier(n_estimators=10))]),
 'logistic_regression': Pipeline(steps=[('standardscaler', StandardScaler()),
                                         ('logisticregression',
                                          LogisticRegression(C=5, solver='liblinear'))])}
```

```
[ ] best_estimators['svm'].score(X_test,y_test)
```

```
0.8125
```

```
[ ] best_estimators['random_forest'].score(X_test,y_test)
```

```
0.578125
```

```
[ ] best_estimators['logistic_regression'].score(X_test,y_test)
```

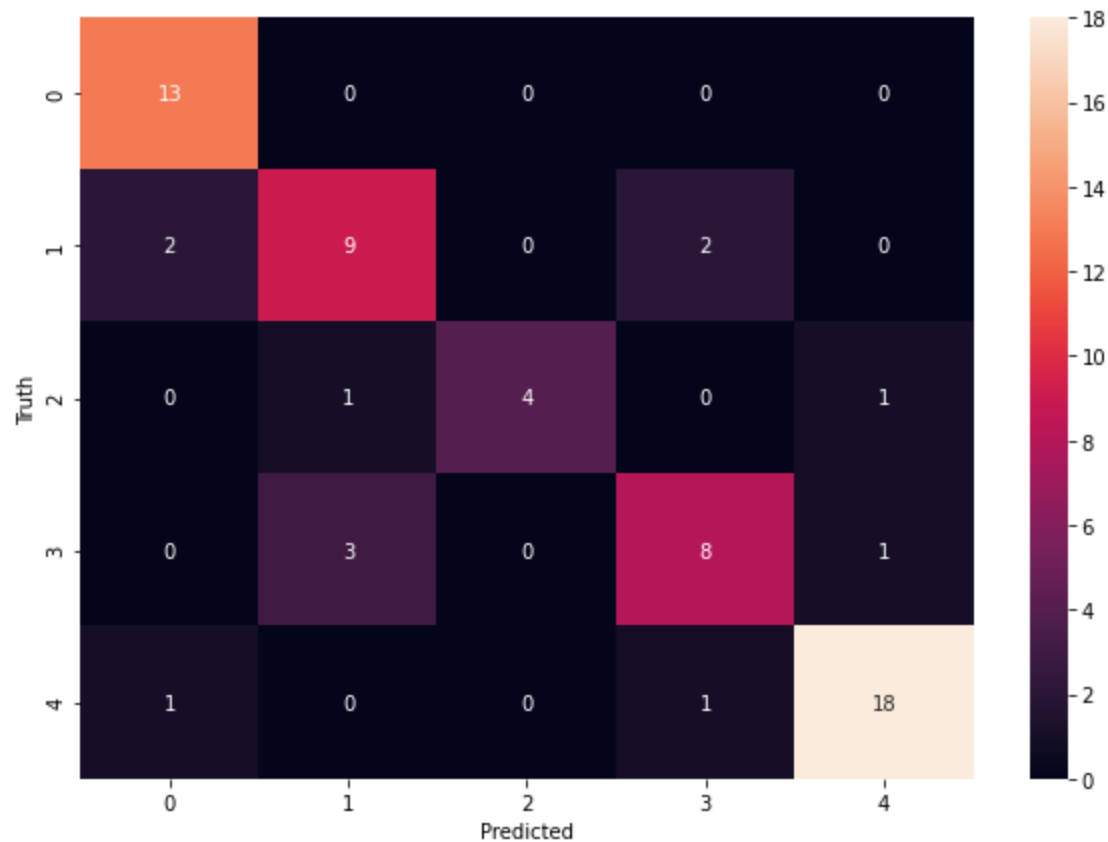
```
0.828125
```

```
[ ] best_clf = best_estimators['svm']
```

c. Evaluation et choix du modèle

On évalue notre modèle avec une matrice de confusion, enfin on l'enregistre dans un fichier "saved_model.pkl", ce fichier sera fourni pour l'équipe de data engineering afin d'être exploité dans toute application web / mobile.

```
{'lionel messi': 0,
 'maria sharapova': 1,
 'roger federer': 2,
 'serena williams': 3,
 'virat kohli': 4}
```



```
[ ] #Save the trained model¶
!pip install joblib
import joblib
# Save the model as a pickle in a file
joblib.dump(best_clf, 'saved_model.pkl')
```


```
Requirement already satisfied: joblib in c:\users\asus\anaconda3\lib\site-packages (0.17.0)
['saved_model.pkl']
```


IV. Data engineering et déploiement du modèle


Ce travail est vraiment long et ne nous intéresse pas vraiment car l'objet de ce mini projet est d'apprendre les notions de la reconnaissance faciale et la classification avec OpenCV et Sklearn. En tout cas, voilà comment nos résultats finaux apparaissent :


On import une image de notre ordinateur dans le site, on clique sur " Classify " et ça donne, sans surprise, l'identité de la célébrité ! Génial


Sports Person Classifier

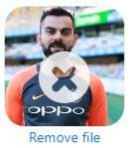

**LIONEL
MESSI**


**MARIA
SHARAPOVA**


**ROGER
FEDERER**



**SERENA
WILLIAMS**


VIRAT KOHLI

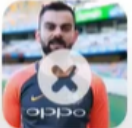


Remove file

Classify



VIRAT KOHLI

Player	Probability Score
Leonel Messi	0.4
Maria Sharapova	0.2
Rodger Federer	0.11
Serena Williams	0.07
Virat Kohli	99.22



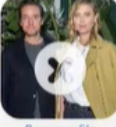
Remove file

Classify




VIRAT KOHLI

Player	Probability Score
Leonel Messi	0.4
Maria Sharapova	0.2
Rodger Federer	0.11
Serena Williams	0.07
Virat Kohli	99.22



Remove file

Classify



MARIA SHARAPOVA

Player	Probability Score
Leonel Messi	3.3
Maria Sharapova	80.31
Rodger Federer	3.92
Serena Williams	11.97
Virat Kohli	0.49

Arroud Yassir