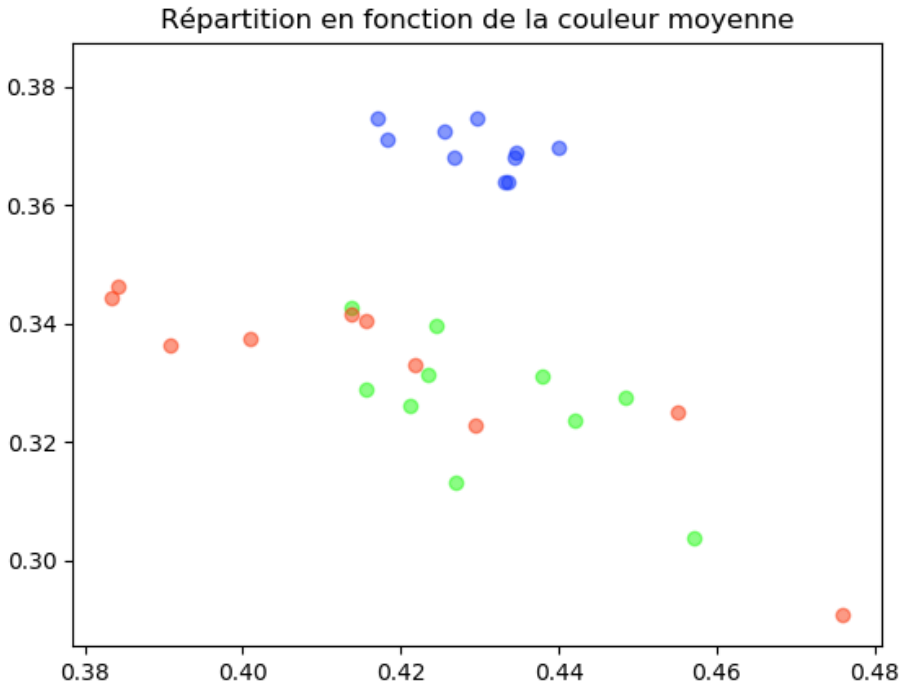
TP4 : Classification bayésienne

1 Prétraitement : calcul de la couleur moyenne d'une image

Pour calculer la couleur moyenne d'une image, il nous faut dans un premier temps obtenir un niveau de couleur normalisé pour chaque pixel. Ensuite on peut faire une moyenne. On travaille uniquement avec R et V car B est déductible.

```
#parcours de tout les pixels
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        #normalisation
        px = image[i,j]
        tot = int(px[0])+int(px[1])+int(px[2])
        x = px[0]/max(1,tot)
        y = px[1] / max(1, tot)

        sum_x += x
        sum_y += y
        np_px += 1

return(sum_x/np_px, sum_y/np_px)
```

Le résultat obtenu nous montre que la couleur moyenne n'est pas une caractéristique suffisamment discriminante



2 Estimation de la vraisemblance de chaque espèce de fleurs

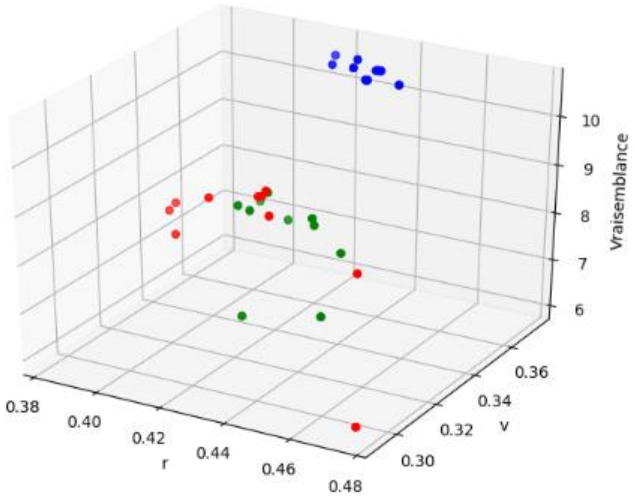
Dans un premier temps nous devons calculer le vecteur moyen ainsi que la matrice de covariance à l'aide de la fonction fit du tp3.

```
def fit(self, X:np.ndarray, y:np.ndarray):
    """Learning of parameters
    X : shape (n_data, n_features)
    y : shape (n_data)
    """
    # number of random variables and classes
    n_features = X.shape[1]
    n_classes = len(np.unique(y))
    # initialization of parameters
    self.mu = np.zeros((n_classes, n_features))
    self.sigma = np.zeros((n_classes, n_features, n_features))

    for i in range(n_classes):
        #calcul vecteur moyen
        self.mu[i] = np.average(X[y == i], 0)
        #calcul de la matrice
        self.sigma[i] = np.cov(X[y == i].T)

    return self.mu, self.sigma
```

```
# Apprentissage
mu, sig = g.fit(color, labels)
```

Ensuite j'ai modifié la fonction predict afin qu'il renvoie non plus le tableau de prédiction des classes mais le tableau avec la valeur de la vraisemblance.

Enfin on peut afficher chaque point avec une coordonnée en plus, la vraisemblance.



3 Classification d'image de fleurs

Sans les probabilité à priori on obtient un score de 0.766666

Dans la fonction fit on a :

```
#rajout de la probabilité à priori
val = val * self.priors[j]
```

```
g = GaussianBayes(priors=[0.33,0.3,0.326])
```

Avec ces probas à priori on arrive à obtenir un score de 0.8.

4 Amélioration du classifieur

Pour améliorer le classifieur j'ai créé une nouvelle fonction de prétraitement qui va ajouter en plus la moyenne de rouge dans le centre en tant que coordonnées supplémentaire. Le fichier Bayes peut fonctionner avec cette coordonnée en plus sans avoir besoin de modification.

```
if (i < (image.shape[0] / 2) - 20 or i > (image.shape[0] / 2)+20) and (j < (image.shape[1] / 2)-20 or j > 3*(image.shape[1] / 2)+20):
    #normalisation
    px = image[i,j]
    tot = int(px[0])+int(px[1])+int(px[2])
    x = px[0]/max(1,tot)
    y = px[1] / max(1, tot)

    sum_x += x
    sum_y += y
    np_px += 1
else:
    # normalisation
    px = image[i, j]
    tot = int(px[0]) + int(px[1]) + int(px[2])
    c = px[0] / max(1, tot)

    sum_c += c
    np_px_c += 1
```

Si le pixel dont on calcul la couleur se situe dans le carré de côté 20pixels au centre de l'image, on l'ajoute pour la moyenne de rouge du centre.