TECNOLÓGICO DE MONTERREY®

Maestría en Inteligencia Artificial Aplicada

Curso: Navegación autónoma

Tecnológico de Monterrey

Prof Titular y Tutor: Dr. David Antonio Torres

Prof Asistente: Maricarmen Vázquez Rojí

ALUMNO: Luis Alfonso Sabanero Esquivel

MATRICULA: A01273286

ALUMNO: Jose Mtanous

MATRICULA: A00169781

ALUMNO: Guillermo Alfonso Muñiz Hermosillo

MATRICULA: A01793101

ALUMNO: Jorge Mariles Estrada

MATRICULA: A01335663

Actividad de la Semana 05

Actividad 3.1 - Detección de Peatones con SVM

Mayo 2023

La explicación de esta entrega se encuentra aquí: https://www.youtube.com/watch?v=NAQahY-G4XU

In [ ]:
```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import cv2
from skimage.feature import hog
```

In [ ]:
```
# Utilizando la libreria de glob podemos importar nuestros archivos de imagenes uti
import glob
peatones = glob.glob("data/Pedestrians/*.*")
no_peatones = glob.glob("data/NoPedestrians/*.*")
```

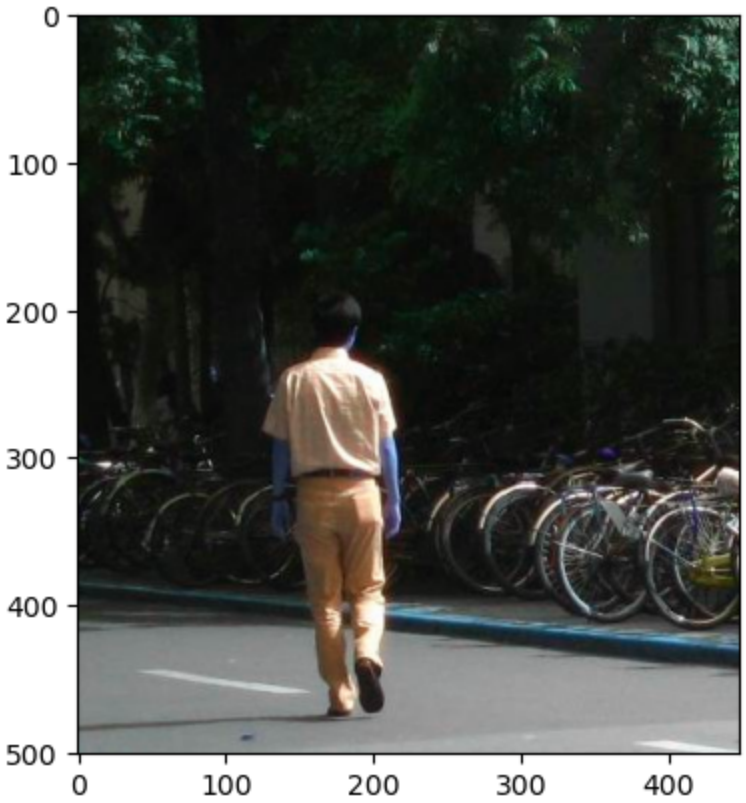
In [ ]:
```
# En esta parte imprimimos el numero de imagenes de nuestro conjunto de datos.
print(f"Imagenes con peatones: {len(peatones)}")

print(f"Imagenes sin peatones: {len(no_peatones)}")
```

```
Imagenes con peatones: 1104
Imagenes sin peatones: 1118
```

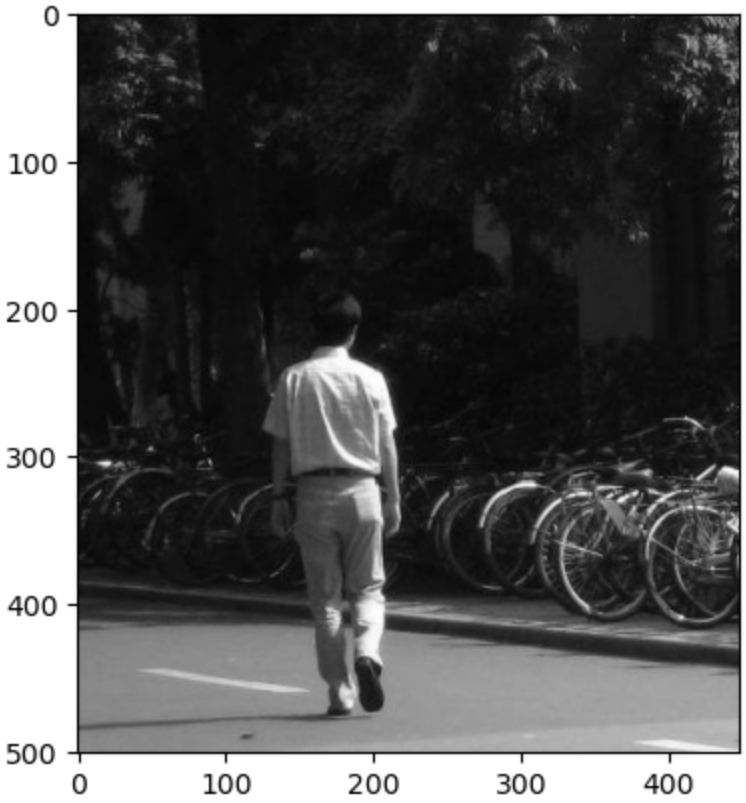
In [ ]:
```
# Enseguida validamos alguna imagen descargada mostrandola usando la libreria matpl
img_color = cv2.imread(peatones[50])
plt.imshow(img_color)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x174266910d0>
```



In [ ]:
```
# Asi mismo es necesario mencionar que debemos de convertir nuestras imagener en es
img_gray = cv2.cvtColor(img_color,cv2.COLOR_BGR2GRAY)
plt.imshow(img_gray,cmap="gray")
```

Out[ ]: <matplotlib.image.AxesImage at 0x174210b23a0>



In [ ]:
```
# Como un ejemplo, utilizamos el algoritmo HOG (Histograma de gradientes orientado)
features,hog_img = hog(img_gray,
                       orientations = 11,
                       pixels_per_cell = (16,16),
                       cells_per_block = (2,2),
                       transform_sqrt = False,
                       visualize = True,
                       feature_vector = True)
```

In [ ]:
```
# Obtenemos nuestras features para esta imagen en particular.
features.shape
```

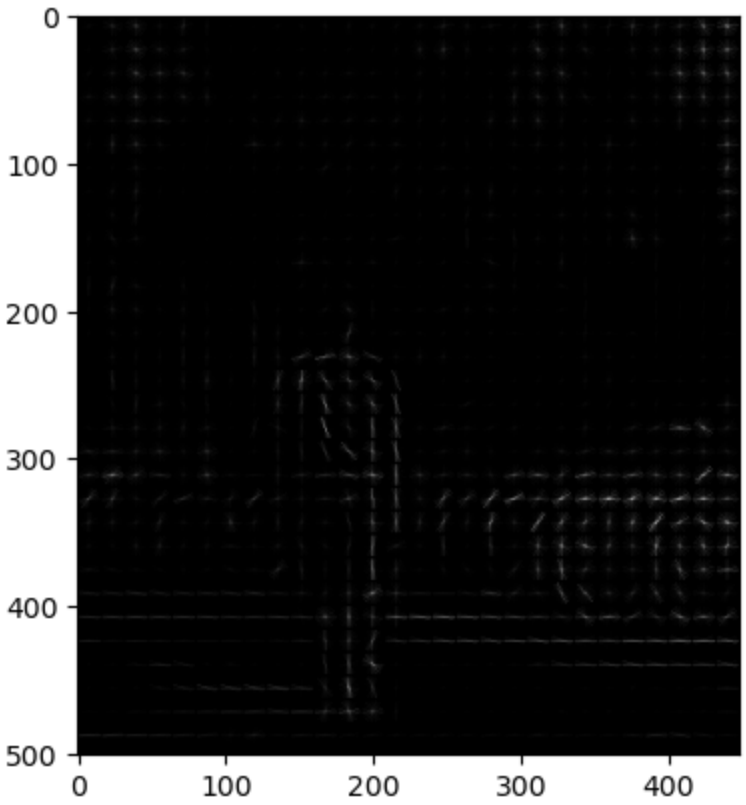
Out[ ]:
```
(35640,)
```

In [ ]:
```
# El segundo parametro contiene la imagen HOG.
hog_img.shape
```

Out[ ]:
```
(501, 448)
```

In [ ]:
```
# Mostramos dicha imagen y podemos observar los histogramas detectados por el algor
plt.imshow(hog_img,cmap = 'gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x174267764c0>
```



In [ ]:
```
# Para evitar la repeticion de codigo, creamos una funcion en la cual llevaremos a
# Utilizando el algoritmo de HOG.

def get_hog_accum(arrayOfImages):
    # Inicializamos un acumulador de caracteristicas, el cual retornaremos al final
```

```
    hog_accum = []

    # Recorremos cada una de las imagenas recibidas en el parametro de la funcion.
    for i in arrayOfImages:
        # Hacemos un resize de la imagen a color, con la finalidad de que el arregl
        img_color = cv2.resize(mpimg.imread(i), (64,128), interpolation = cv2.INTER
        # Convertimos la imagen a escala de grises.
        img_gray = cv2.cvtColor(img_color,cv2.COLOR_BGR2GRAY)

        # Obtenemos las caracteristicas de la imagen utilizando el algoritmo de HOG
        hog_feature, hog_img = hog(img_gray,
                                   orientations = 11,
                                    pixels_per_cell = (16,16),
                                    cells_per_block = (2,2),
                                    transform_sqrt = False,
                                    visualize = True,
                                    feature_vector = True)

        # Agregamos las caracteristicas obtenidas a nuestro acumulador.
        hog_accum.append(hog_feature)
    # Una vez concluido el proceso con todas las imagenes, retornamos todas las car
    return hog_accum
```

In [ ]:
```
# Asi de esta manera asignamos a esta variable el acumulador de caracteristicas par
pedestrian_hog_accum = get_hog_accum(peatones)

# Convertimos nuestro acumulador de caracteristicas en un arreglo vertical de flota
X_peaton = np.vstack(pedestrian_hog_accum).astype(np.float64)
# Creamos un arreglo de unos de la misma longitud que nuestro acumulador de caracte
# Se llena con unos al asignar esta clase como la clase positiva.
y_peaton = np.ones(len(X_peaton))
```

In [ ]:
```
# Asi pues observamos el numero de imagenes con el mismo numero de caracteristicas
X_peaton.shape
```

Out[ ]:
```
(1104, 924)
```

In [ ]:
```
# Nuestro arreglo de 1s es de la misma longitud que el numero de imagenes
y_peaton.shape
```

Out[ ]:
```
(1104,)
```

In [ ]:
```
# Repetimos el proceso anterior, pero ahora utilizando el conjunto de datos de la c
# Es decir el conjunto de imagenes que no contiene peatones.
non_pedestrian_hog_accum = get_hog_accum(no_peatones)
X_no_peaton = np.vstack(non_pedestrian_hog_accum).astype(np.float64)

# En esta ocacion es necesario crear un arreglo de 0, ya que dicho representara a l
y_no_peaton = np.zeros(len(X_no_peaton))
```

In [ ]:
```
# Asi pues observamos el numero de imagenes sin peatones con el mismo numero de car
X_no_peaton.shape
```

Out[ ]:
```
(1118, 924)
```

In [ ]:
```
# Nuestro arreglo de 0s es de la misma longitud que el numero de imagenes sin peato
y_no_peaton.shape
```

Out[ ]:
```
(1118,)
```

In [ ]:
```
# El siguiente paso es combinar ambos conjuntos de datos en un solo conjunto para u
# Utilizando el metodo vstack apilamos el conjunto de caracteristicas de ambos conj
# Obteniendo la suma total de imagenes con peatones.
X = np.vstack((X_peaton,X_no_peaton))
X.shape
```

Out[ ]:
```
(2222, 924)
```

In [ ]:
```
# Asi mismo es necesario combinar ambos conjuntos de variables dependientes
# Utilizando el metodo hstack creamos un arreglo con las etiquetas de ambos conjunt
y = np.hstack((y_peaton,y_no_peaton))
y.shape
```

Out[ ]:
```
(2222,)
```

In [ ]:
```
# Importamos el metodo train_test_split para dividir nuestro conjunto de datos en c
from sklearn.model_selection import train_test_split

# Definimos un conjunto de entrenamiento del 70% del conjunto de datos contra un co
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_s

print(f"Forma del conjunto de Entrenamiento: {X_train.shape}")

print(f"Forma del conjunto de Prueba: {X_test.shape}")
```

```
Forma del conjunto de Entrenamiento: (1555, 924)
Forma del conjunto de Prueba: (667, 924)
```

In [ ]:
```
# Es necesario importar nuestra maquina de soporte vectorial
from sklearn.svm import SVC

# Utilizando la funcion de scikitlearn entrenamos nuestro modelo utilizando nuestro
svc_model = SVC()
svc_model.fit(X_train,y_train)
```

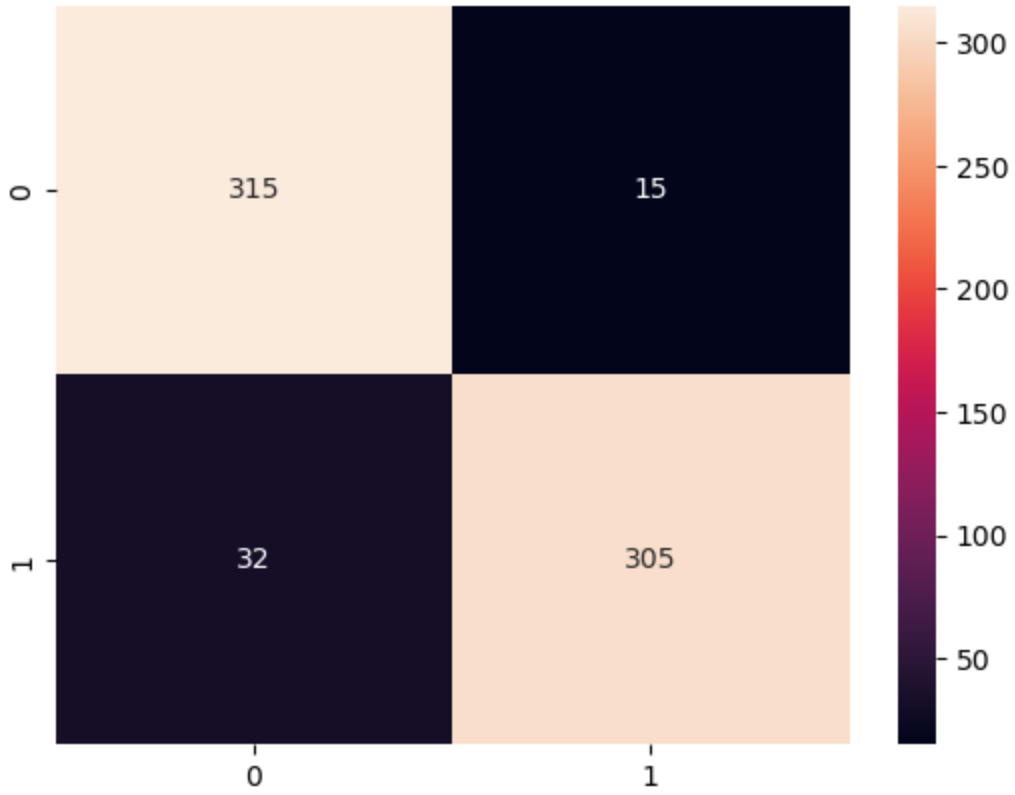
Out[ ]:
```
SVC()
```

In [ ]:
```
# Una vez entrenado el modelo, podemos predecir los resultados en nuestro conjuntos
y_predict = svc_model.predict(X_test)
```

In [ ]:
```
# Obtenidas las predicciones, es necesario crear nuestra matriz de confusion
# con el objetivo de medir el comportamiento de nuestro modelo.
from sklearn.metrics import classification_report, confusion_matrix

# Utilizando el metodo correspondiente, nuestro conjunto de prueba y las prediccion
# Obtenemos nuestra matriz. La cual podemos mostrar con un heatmat de la libreria s
cm = confusion_matrix(y_test,y_predict)
sns.heatmap(cm, annot=True, fmt = "d")
```

Out[ ]: <AxesSubplot:>



In [ ]:
```
# Tambien podemos obtener un reporte de clasificacion, el cual nos muestra las metr
# Utilizando el conjunto de pruebas y las predicciones realizadas.
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.91      0.95      0.93       330
         1.0       0.95      0.91      0.93       337

    accuracy                           0.93       667
   macro avg       0.93      0.93      0.93       667
weighted avg       0.93      0.93      0.93       667
```

In [ ]:
```
# Podemos buscar obtener un mejor resultado mediante la utilizacion de una busqueda
# utilizando la libreria GridSearch de Scikitlean.
from sklearn.model_selection import GridSearchCV

# Definimos pues las combinaciones de los parametros que queremos evaluar en nuestr
param_grid = {'C': [0.1,1, 10, 100, 1000], 'gamma': [10, 1,0.1,0.01,0.001,0.0001],

# Entrenamos nuestra malla de busqueda utilizando nuestra maquina de soporte vector
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=4)
# Y comenzamos el entrenamiento utilizando nuestros conjuntos de entrenamiento.
grid.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
[CV 1/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.508 total time=   0.4s
[CV 2/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.508 total time=   0.4s
[CV 3/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.508 total time=   0.5s
[CV 4/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.505 total time=   0.5s
[CV 5/5] END .......C=0.1, gamma=10, kernel=rbf;, score=0.505 total time=   0.4s
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.508 total time=   0.4s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.508 total time=   0.3s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.508 total time=   0.3s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.505 total time=   0.3s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.505 total time=   0.3s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.878 total time=   0.2s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.894 total time=   0.2s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.897 total time=   0.2s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.887 total time=   0.2s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.852 total time=   0.2s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.830 total time=   0.3s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.814 total time=   0.3s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.865 total time=   0.3s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.842 total time=   0.3s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.804 total time=   0.4s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.508 total time=   0.4s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.508 total time=   0.3s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.508 total time=   0.3s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.505 total time=   0.3s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.505 total time=   0.4s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.508 total time=   0.4s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.508 total time=   0.4s
[CV 3/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.508 total time=   0.4s
[CV 4/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.505 total time=   0.4s
[CV 5/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.505 total time=   0.4s
[CV 1/5] END ........C=1, gamma=10, kernel=rbf;, score=0.511 total time=   0.5s
[CV 2/5] END ........C=1, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 3/5] END ........C=1, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 4/5] END ........C=1, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 5/5] END ........C=1, gamma=10, kernel=rbf;, score=0.508 total time=   0.4s
[CV 1/5] END .........C=1, gamma=1, kernel=rbf;, score=0.746 total time=   0.4s
[CV 2/5] END .........C=1, gamma=1, kernel=rbf;, score=0.762 total time=   0.4s
[CV 3/5] END .........C=1, gamma=1, kernel=rbf;, score=0.807 total time=   0.4s
[CV 4/5] END .........C=1, gamma=1, kernel=rbf;, score=0.772 total time=   0.4s
[CV 5/5] END .........C=1, gamma=1, kernel=rbf;, score=0.743 total time=   0.4s
[CV 1/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.1s
[CV 2/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.1s
[CV 3/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.936 total time=   0.1s
[CV 4/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.926 total time=   0.1s
[CV 5/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.913 total time=   0.1s
[CV 1/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.884 total time=   0.1s
[CV 2/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.907 total time=   0.1s
[CV 3/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.900 total time=   0.1s
[CV 4/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.900 total time=   0.1s
[CV 5/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.871 total time=   0.1s
[CV 1/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.846 total time=   0.3s
[CV 2/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.823 total time=   0.3s
[CV 3/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.871 total time=   0.3s
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.842 total time=   0.3s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.807 total time=   0.3s
```

```
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.508 total time=   0.3s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.508 total time=   0.3s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.508 total time=   0.3s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.505 total time=   0.3s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.505 total time=   0.3s
[CV 1/5] END ........C=10, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 2/5] END ........C=10, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 3/5] END ........C=10, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 4/5] END ........C=10, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 5/5] END ........C=10, gamma=10, kernel=rbf;, score=0.508 total time=   0.4s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=0.756 total time=   0.4s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.772 total time=   0.4s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.817 total time=   0.4s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=0.791 total time=   0.4s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.756 total time=   0.4s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.971 total time=   0.1s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.942 total time=   0.1s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.942 total time=   0.1s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.916 total time=   0.1s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.929 total time=   0.1s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.942 total time=   0.1s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.926 total time=   0.1s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.929 total time=   0.1s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.907 total time=   0.1s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.907 total time=   0.1s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.887 total time=   0.2s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.910 total time=   0.2s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.904 total time=   0.1s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.894 total time=   0.1s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.875 total time=   0.1s
[CV 1/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.846 total time=   0.3s
[CV 2/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.826 total time=   0.3s
[CV 3/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.871 total time=   0.3s
[CV 4/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.842 total time=   0.3s
[CV 5/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.807 total time=   0.3s
[CV 1/5] END .......C=100, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 2/5] END .......C=100, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 3/5] END .......C=100, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 4/5] END .......C=100, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 5/5] END .......C=100, gamma=10, kernel=rbf;, score=0.508 total time=   0.3s
[CV 1/5] END ........C=100, gamma=1, kernel=rbf;, score=0.756 total time=   0.4s
[CV 2/5] END ........C=100, gamma=1, kernel=rbf;, score=0.772 total time=   0.4s
[CV 3/5] END ........C=100, gamma=1, kernel=rbf;, score=0.817 total time=   0.4s
[CV 4/5] END ........C=100, gamma=1, kernel=rbf;, score=0.791 total time=   0.4s
[CV 5/5] END ........C=100, gamma=1, kernel=rbf;, score=0.756 total time=   0.4s
[CV 1/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.971 total time=   0.1s
[CV 2/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.1s
[CV 3/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.1s
[CV 4/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.913 total time=   0.1s
[CV 5/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.929 total time=   0.1s
[CV 1/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.916 total time=   0.1s
[CV 2/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.913 total time=   0.1s
[CV 3/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.907 total time=   0.0s
[CV 4/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.894 total time=   0.0s
[CV 5/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.916 total time=   0.0s
[CV 1/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.936 total time=   0.1s
```

```
[CV 2/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.920 total time=   0.1s
[CV 3/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.913 total time=   0.1s
[CV 4/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.1s
[CV 5/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.904 total time=   0.1s
[CV 1/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.887 total time=   0.2s
[CV 2/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.910 total time=   0.2s
[CV 3/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.904 total time=   0.2s
[CV 4/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.891 total time=   0.2s
[CV 5/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.871 total time=   0.2s
[CV 1/5] END ......C=1000, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 2/5] END ......C=1000, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 3/5] END ......C=1000, gamma=10, kernel=rbf;, score=0.511 total time=   0.4s
[CV 4/5] END ......C=1000, gamma=10, kernel=rbf;, score=0.511 total time=   0.5s
[CV 5/5] END ......C=1000, gamma=10, kernel=rbf;, score=0.508 total time=   0.4s
[CV 1/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.756 total time=   0.4s
[CV 2/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.772 total time=   0.4s
[CV 3/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.817 total time=   0.4s
[CV 4/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.791 total time=   0.4s
[CV 5/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.756 total time=   0.3s
[CV 1/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.971 total time=   0.1s
[CV 2/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.1s
[CV 3/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.1s
[CV 4/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.913 total time=   0.1s
[CV 5/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.929 total time=   0.1s
[CV 1/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.916 total time=   0.0s
[CV 2/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.907 total time=   0.0s
[CV 3/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.913 total time=   0.0s
[CV 4/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.891 total time=   0.0s
[CV 5/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.913 total time=   0.0s
[CV 1/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.907 total time=   0.0s
[CV 2/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 3/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.913 total time=   0.0s
[CV 4/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.884 total time=   0.0s
[CV 5/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.936 total time=   0.1s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.920 total time=   0.1s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.913 total time=   0.1s
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.904 total time=   0.1s
```

Out[ ]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'gamma': [10, 1, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['rbf']},
             verbose=4)
```

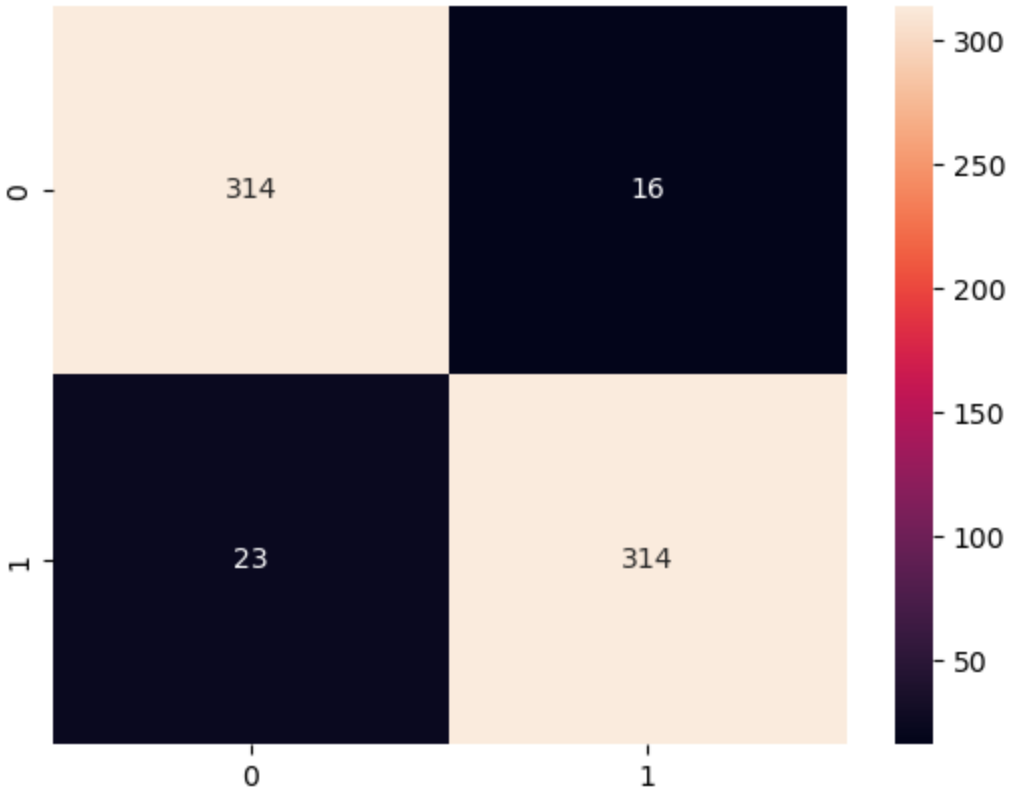
In [ ]:
```
# Una vez finalizado el entrenamiento de la malla, podemos obtener los parametros c
print(f"Mejore Parametros encontrados: {grid.best_params_}")
```

```
Mejore Parametros encontrados: {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
```

In [ ]:
```
# Para poder utilizarlo ejecutamos la linea del mejor estimador y procedemos a real
# Utilizando el conjunto de pruebas.
grid.best_estimator_
grid_predictions = grid.predict(X_test)
```

In [ ]:
```
# Una vez obtenidas las nuevas predicciones con el modelo con mejores parametros
# Repetimos el proceso de generar nuestra matriz de confusion para observar los nue
cm = confusion_matrix(y_test,grid_predictions)
sns.heatmap(cm, annot=True, fmt = "d")
```

Out[ ]: <AxesSubplot:>



In [ ]:
```
# Asi mismo re-imprimimos nuestro reporte de clasificacion para observar si hubo al
print(classification_report(y_test,grid_predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.93      0.95      0.94       330
         1.0       0.95      0.93      0.94       337

    accuracy                           0.94       667
   macro avg       0.94      0.94      0.94       667
weighted avg       0.94      0.94      0.94       667
```