# Tarea2_Machine_Learning_GABRIEL_GARCIA_ZAMBRANO

March 13, 2023

## 1 Clasificación con máquina de vectores de soporte y redes de neuronas

Alumno: Garcia Zambrano Gabriel
Materia: Machine Learning

PARTE 1: CLASIFICACIÓN CON MÁQUINA DE VECTORES DE SOPORTE

[1]:
```
##Importamos librerias:
import pandas as pd
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
##importamos librerías
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.regularizers import l2
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, BatchNormalization
from sklearn.metrics import accuracy_score
```

```
2023-03-13 17:06:52.113140: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2023-03-13 17:06:52.214511: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot
open shared object file: No such file or directory
```

```
2023-03-13 17:06:52.214527: I
tensorflow/compiler/xla/stream_executor/cuda/cudart_stub.cc:29] Ignore above
cudart dlerror if you do not have a GPU set up on your machine.
2023-03-13 17:06:52.612826: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot
open shared object file: No such file or directory
2023-03-13 17:06:52.612873: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer_plugin.so.7'; dlerror:
libnvinfer_plugin.so.7: cannot open shared object file: No such file or
directory
2023-03-13 17:06:52.612878: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with
TensorRT, please make sure the missing libraries mentioned above are installed
properly.
```

[2]:
```
##Importamos dataset
train=pd.read_csv("train.csv") ##datos de entrenamiento
test=pd.read_csv("test.csv")
print(test.keys())
print(train.keys())
```

```
Index(['id', 'battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc',
       'four_g', 'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc',
       'px_height', 'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
       'touch_screen', 'wifi'],
      dtype='object')
Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g',
       'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
       'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
       'touch_screen', 'wifi', 'price_range'],
      dtype='object')
```

[3]:
```
print(train.dtypes)
```

```
battery_power      int64
blue               int64
clock_speed      float64
dual_sim           int64
fc                 int64
four_g             int64
int_memory         int64
m_dep            float64
mobile_wt          int64
n_cores            int64
pc                 int64
```

```
px_height          int64
px_width           int64
ram                int64
sc_h               int64
sc_w               int64
talk_time          int64
three_g            int64
touch_screen       int64
wifi               int64
price_range        int64
dtype: object
```

[4]:
```
##Cargar el conjunto de datos en un DataFrame de Pandas
data = pd.read_csv('train.csv')
# Seleccionar solo las variables numéricas
numeric_vars = data.select_dtypes(include=['int64', 'float64'])
# Describir las variables numéricas
print(numeric_vars.describe())
```

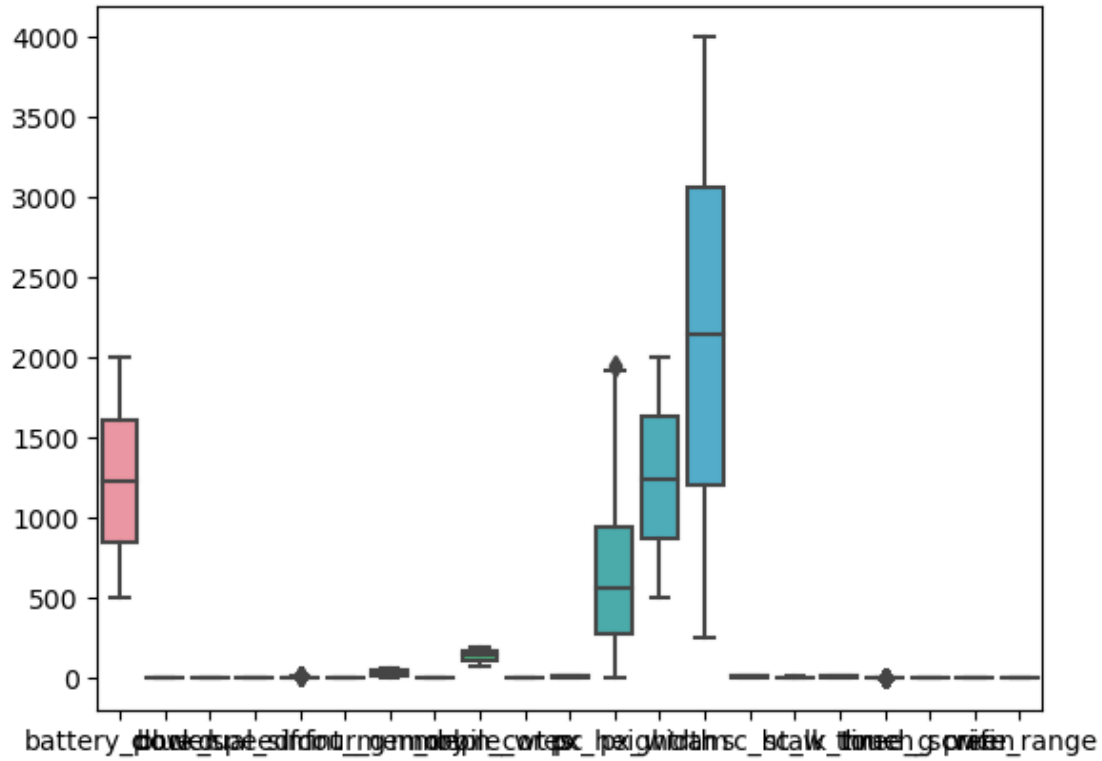
```
       battery_power       blue  clock_speed     dual_sim           fc  \
count    2000.000000  2000.0000  2000.000000  2000.000000  2000.000000
mean     1238.518500     0.4950     1.522250     0.509500     4.309500
std       439.418206     0.5001     0.816004     0.500035     4.341444
min       501.000000     0.0000     0.500000     0.000000     0.000000
25%       851.750000     0.0000     0.700000     0.000000     1.000000
50%      1226.000000     0.0000     1.500000     1.000000     3.000000
75%      1615.250000     1.0000     2.200000     1.000000     7.000000
max      1998.000000     1.0000     3.000000     1.000000    19.000000

            four_g   int_memory        m_dep    mobile_wt      n_cores  …  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  …
mean      0.521500    32.046500     0.501750   140.249000     4.520500  …
std       0.499662    18.145715     0.288416    35.399655     2.287837  …
min       0.000000     2.000000     0.100000    80.000000     1.000000  …
25%       0.000000    16.000000     0.200000   109.000000     3.000000  …
50%       1.000000    32.000000     0.500000   141.000000     4.000000  …
75%       1.000000    48.000000     0.800000   170.000000     7.000000  …
max       1.000000    64.000000     1.000000   200.000000     8.000000  …

         px_height     px_width          ram         sc_h         sc_w  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean    645.108000  1251.515500  2124.213000    12.306500     5.767000
std     443.780811   432.199447  1084.732044     4.213245     4.356398
min       0.000000   500.000000   256.000000     5.000000     0.000000
25%     282.750000   874.750000  1207.500000     9.000000     2.000000
50%     564.000000  1247.000000  2146.500000    12.000000     5.000000
75%     947.250000  1633.000000  3064.500000    16.000000     9.000000
max    1960.000000  1998.000000  3998.000000    19.000000    18.000000
```

| | talk_time | three_g | touch_screen | wifi | price_range |
|---|---|---|---|---|---|
| count | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 |
| mean | 11.011000 | 0.761500 | 0.503000 | 0.507000 | 1.500000 |
| std | 5.463955 | 0.426273 | 0.500116 | 0.500076 | 1.118314 |
| min | 2.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 6.000000 | 1.000000 | 0.000000 | 0.000000 | 0.750000 |
| 50% | 11.000000 | 1.000000 | 1.000000 | 1.000000 | 1.500000 |
| 75% | 16.000000 | 1.000000 | 1.000000 | 1.000000 | 2.250000 |
| max | 20.000000 | 1.000000 | 1.000000 | 1.000000 | 3.000000 |

[8 rows x 21 columns]

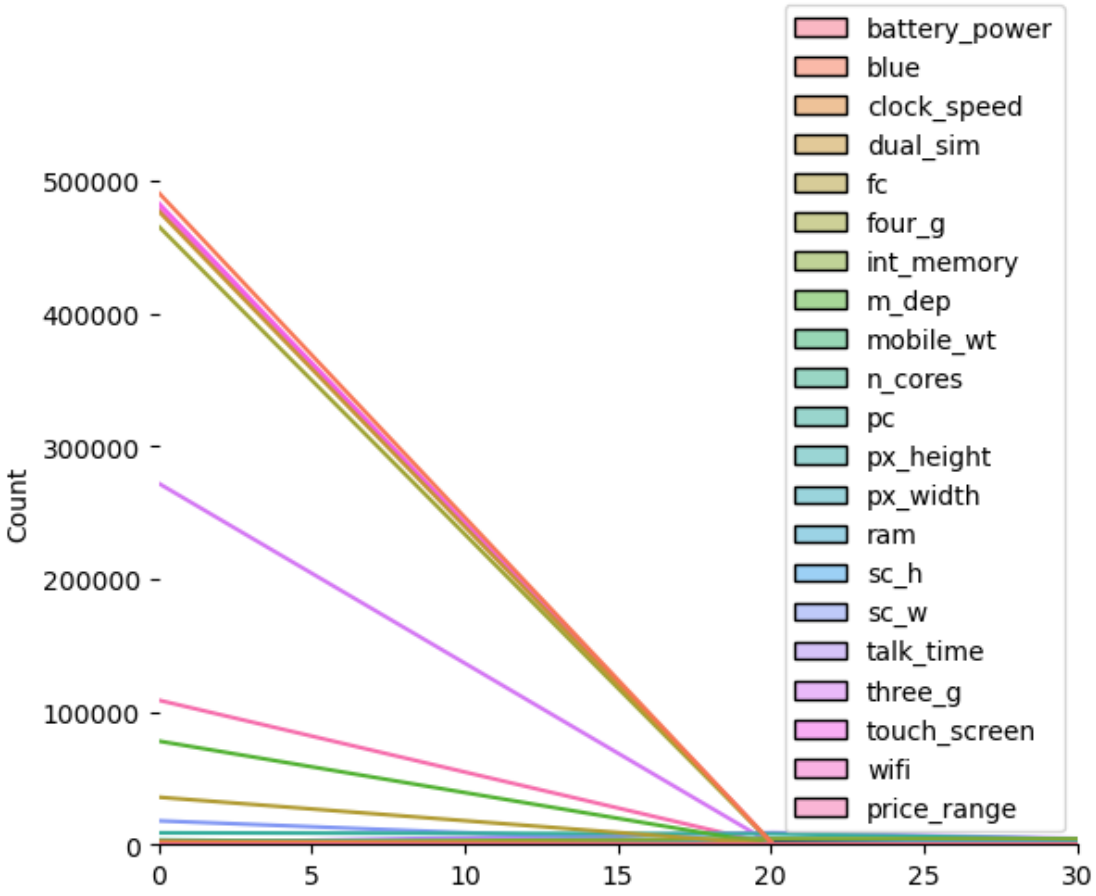
[5]:
```
# Seleccionar solo las variables numéricas
numeric_vars = train.select_dtypes(include=['int64', 'float64'])
# Crear un diagrama de caja y bigotes
sns.boxplot(data=numeric_vars)
```

[5]: <AxesSubplot:>




[6]:
```
# Leer el conjunto de datos
train = pd.read_csv("train.csv")
```

```
# Seleccionar solo las variables numéricas
numeric_vars = train.select_dtypes(include=['int64', 'float64'])
# Generar histogramas para cada variable numérica
sns.histplot(data=numeric_vars, kde=True, bins=30)
# Mostrar la figura
sns.set_style("whitegrid")
sns.despine(left=True)
plt.xlim(0, 30)
# Mostrar la figura
plt.show()
```



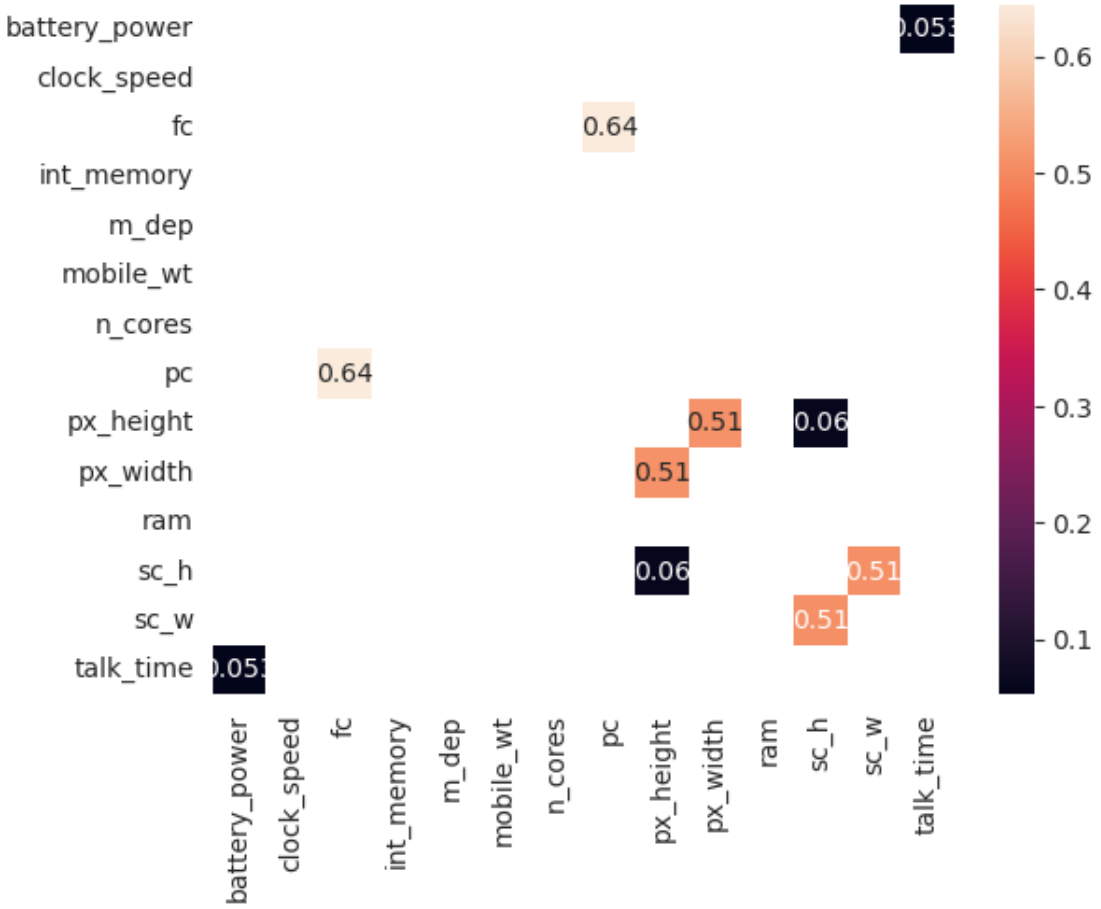

[ ]:

Realizamos matriz de correlación de los datos:

[7]:
```
# Seleccionar las columnas con variables continuas y numéricas
continuous_columns = ['battery_power', 'clock_speed', 'fc', 'int_memory',␣
 ↪'m_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height', 'px_width', 'ram',␣
 ↪'sc_h', 'sc_w', 'talk_time']
```

```
numeric_columns = ['battery_power', 'clock_speed', 'fc', 'int_memory', 'm_dep',␣
 ↪'mobile_wt', 'n_cores', 'pc', 'px_height', 'px_width', 'ram', 'sc_h',␣
 ↪'sc_w', 'talk_time']
# Crear un nuevo dataframe solo con las columnas continuas y numéricas
continuous_df = train[continuous_columns]
# Calcular la matriz de correlación de Pearson
corr_matrix = continuous_df.corr()
# Máscara para seleccionar los valores mayores a 0.05
mask = (abs(corr_matrix) > 0.05) & (corr_matrix < 1)
# Aplicar la máscara a la matriz de correlación
corr_matrix_masked = corr_matrix[mask]
# Visualizar la matriz de correlación utilizando un mapa de calor
sns.heatmap(corr_matrix_masked, annot=True)
```

[7]: <AxesSubplot:>




[8]:
```
# Leer los conjuntos de entrenamiento y prueba desde archivos CSV
train_data = pd.read_csv("train.csv")
```

```
test_data = pd.read_csv("test.csv")
# Dividir los conjuntos de datos en características y etiquetas
X_train = train_data.drop("price_range", axis=1)
y_train = train_data["price_range"]
X_test = test_data.drop("id", axis=1)
# Dividir los datos de entrenamiento en conjuntos de entrenamiento y validación
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.
 ↪2, random_state=42)
# Crear un clasificador SVM y entrenarlo con el conjunto de entrenamiento
clf = SVC(kernel="linear")
clf.fit(X_train, y_train)
y_pred = clf.predict(X_val)
# Calcular la precisión de la predicción
accuracy = accuracy_score(y_val, y_pred)
print("Precisión de la predicción: {:.2f}%".format(accuracy * 100))
# Predecir las etiquetas para el conjunto de prueba
y_test_pred = clf.predict(X_test)
# Guardar las predicciones en un archivo CSV
test_data["price_range"] = y_test_pred
test_data[["id", "price_range"]].to_csv("predicted_prices.csv", index=False)
```

```
Precisión de la predicción: 97.00%
```

[9]:
```
from sklearn.metrics import confusion_matrix

# Creación del modelo
model = SVC()

# Definición de los parámetros a ajustar
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kernel':␣
 ↪['rbf', 'linear', 'poly', 'sigmoid']}

# Ajuste de los parámetros utilizando la técnica de validación cruzada
grid = GridSearchCV(model, param_grid, refit=True, verbose=3)
grid.fit(X_train, y_train)

# Predicción utilizando el modelo ajustado
y_pred = grid.predict(X_val)

# Calcular la precisión de la predicción
accuracy = accuracy_score(y_val, y_pred)
print("Precisión de la predicción: {:.2f}%".format(accuracy * 100))

# Predicción para el conjunto de prueba
y_test = grid.predict(X_test)

# Calcular la matriz de confusión
```

```
conf_mat = confusion_matrix(y_val, y_pred)
print("Matriz de confusión:")
print(conf_mat)
```

```
Fitting 5 folds for each of 64 candidates, totalling 320 fits
[CV 1/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.256 total time=   0.1s
[CV 2/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.256 total time=   0.1s
[CV 3/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.253 total time=   0.1s
[CV 4/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.253 total time=   0.1s
[CV 5/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.253 total time=   0.1s
[CV 1/5] END …C=0.1, gamma=1, kernel=linear;, score=0.978 total time=   0.4s
[CV 2/5] END …C=0.1, gamma=1, kernel=linear;, score=0.978 total time=   0.8s
[CV 3/5] END …C=0.1, gamma=1, kernel=linear;, score=0.972 total time=   1.3s
[CV 4/5] END …C=0.1, gamma=1, kernel=linear;, score=0.959 total time=   0.5s
[CV 5/5] END …C=0.1, gamma=1, kernel=linear;, score=0.978 total time=   1.6s
[CV 1/5] END …C=0.1, gamma=1, kernel=poly;, score=0.959 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=1, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=0.1, gamma=1, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=0.1, gamma=1, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=1, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END …C=0.1, gamma=1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END …C=0.1, gamma=1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END …C=0.1, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END …C=0.1, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END …C=0.1, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.256 total time=   0.1s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.256 total time=   0.1s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.259 total time=   0.1s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.263 total time=   0.1s
[CV 5/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.253 total time=   0.1s
[CV 1/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.978 total time=   0.4s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.978 total time=   0.8s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.972 total time=   1.3s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.959 total time=   0.5s
[CV 5/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.978 total time=   1.6s
[CV 1/5] END …C=0.1, gamma=0.1, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=0.1, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END ..C=0.1, gamma=0.1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END ..C=0.1, gamma=0.1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END ..C=0.1, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END ..C=0.1, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END ..C=0.1, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
[CV 2/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
```

```
[CV 3/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
[CV 4/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.447 total time=   0.1s
[CV 5/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.253 total time=   0.1s
[CV 1/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.978 total time=   0.4s
[CV 2/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.978 total time=   0.8s
[CV 3/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.972 total time=   1.3s
[CV 4/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.959 total time=   0.5s
[CV 5/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.978 total time=   1.6s
[CV 1/5] END …C=0.1, gamma=0.01, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=0.1, gamma=0.01, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=0.1, gamma=0.01, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=0.1, gamma=0.01, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=0.01, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END .C=0.1, gamma=0.01, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END .C=0.1, gamma=0.01, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END .C=0.1, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END .C=0.1, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END .C=0.1, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.256 total time=   0.2s
[CV 2/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.256 total time=   0.2s
[CV 3/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.256 total time=   0.2s
[CV 4/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.259 total time=   0.2s
[CV 5/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.253 total time=   0.2s
[CV 1/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.978 total time=   0.5s
[CV 2/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.978 total time=   0.8s
[CV 3/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.972 total time=   1.3s
[CV 4/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.959 total time=   0.5s
[CV 5/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.978 total time=   1.6s
[CV 1/5] END …C=0.1, gamma=0.001, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=0.1, gamma=0.001, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=0.1, gamma=0.001, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=0.1, gamma=0.001, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=0.001, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END C=0.1, gamma=0.001, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END C=0.1, gamma=0.001, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END C=0.1, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END C=0.1, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END C=0.1, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=1, gamma=1, kernel=rbf;, score=0.256 total time=   0.1s
[CV 2/5] END …C=1, gamma=1, kernel=rbf;, score=0.256 total time=   0.1s
[CV 3/5] END …C=1, gamma=1, kernel=rbf;, score=0.253 total time=   0.1s
[CV 4/5] END …C=1, gamma=1, kernel=rbf;, score=0.253 total time=   0.1s
[CV 5/5] END …C=1, gamma=1, kernel=rbf;, score=0.253 total time=   0.1s
[CV 1/5] END …C=1, gamma=1, kernel=linear;, score=0.975 total time=   6.4s
[CV 2/5] END …C=1, gamma=1, kernel=linear;, score=0.978 total time=   7.7s
[CV 3/5] END …C=1, gamma=1, kernel=linear;, score=0.969 total time=   6.2s
[CV 4/5] END …C=1, gamma=1, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=1, gamma=1, kernel=linear;, score=0.981 total time=   9.5s
```

```
[CV 1/5] END …C=1, gamma=1, kernel=poly;, score=0.959 total time=   0.0s
[CV 2/5] END …C=1, gamma=1, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=1, gamma=1, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=1, gamma=1, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=1, gamma=1, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END …C=1, gamma=1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END …C=1, gamma=1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END …C=1, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END …C=1, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END …C=1, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.256 total time=   0.1s
[CV 2/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.256 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.259 total time=   0.1s
[CV 4/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.263 total time=   0.1s
[CV 5/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.253 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.1, kernel=linear;, score=0.975 total time=   6.3s
[CV 2/5] END …C=1, gamma=0.1, kernel=linear;, score=0.978 total time=   7.7s
[CV 3/5] END …C=1, gamma=0.1, kernel=linear;, score=0.969 total time=   6.2s
[CV 4/5] END …C=1, gamma=0.1, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=1, gamma=0.1, kernel=linear;, score=0.981 total time=   9.5s
[CV 1/5] END …C=1, gamma=0.1, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=1, gamma=0.1, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.1, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=1, gamma=0.1, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.1, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END …C=1, gamma=0.1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END …C=1, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END …C=1, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
[CV 2/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
[CV 4/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.259 total time=   0.1s
[CV 5/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.253 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.01, kernel=linear;, score=0.975 total time=   6.3s
[CV 2/5] END …C=1, gamma=0.01, kernel=linear;, score=0.978 total time=   7.7s
[CV 3/5] END …C=1, gamma=0.01, kernel=linear;, score=0.969 total time=   6.2s
[CV 4/5] END …C=1, gamma=0.01, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=1, gamma=0.01, kernel=linear;, score=0.981 total time=   9.5s
[CV 1/5] END …C=1, gamma=0.01, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=1, gamma=0.01, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.01, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=1, gamma=0.01, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.01, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.01, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END …C=1, gamma=0.01, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
```

```
[CV 4/5] END …C=1, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END …C=1, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.259 total time=   0.2s
[CV 2/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.263 total time=   0.2s
[CV 3/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.266 total time=   0.2s
[CV 4/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.259 total time=   0.2s
[CV 5/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.259 total time=   0.2s
[CV 1/5] END …C=1, gamma=0.001, kernel=linear;, score=0.975 total time=   6.4s
[CV 2/5] END …C=1, gamma=0.001, kernel=linear;, score=0.978 total time=   7.6s
[CV 3/5] END …C=1, gamma=0.001, kernel=linear;, score=0.969 total time=   6.2s
[CV 4/5] END …C=1, gamma=0.001, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=1, gamma=0.001, kernel=linear;, score=0.981 total time=   9.5s
[CV 1/5] END …C=1, gamma=0.001, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=1, gamma=0.001, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.001, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=1, gamma=0.001, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.001, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END ..C=1, gamma=0.001, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END ..C=1, gamma=0.001, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END ..C=1, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END ..C=1, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END ..C=1, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=10, gamma=1, kernel=rbf;, score=0.256 total time=   0.2s
[CV 2/5] END …C=10, gamma=1, kernel=rbf;, score=0.256 total time=   0.2s
[CV 3/5] END …C=10, gamma=1, kernel=rbf;, score=0.253 total time=   0.2s
[CV 4/5] END …C=10, gamma=1, kernel=rbf;, score=0.253 total time=   0.2s
[CV 5/5] END …C=10, gamma=1, kernel=rbf;, score=0.253 total time=   0.2s
[CV 1/5] END …C=10, gamma=1, kernel=linear;, score=0.972 total time=  13.0s
[CV 2/5] END …C=10, gamma=1, kernel=linear;, score=0.978 total time=   8.2s
[CV 3/5] END …C=10, gamma=1, kernel=linear;, score=0.978 total time=   9.0s
[CV 4/5] END …C=10, gamma=1, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=10, gamma=1, kernel=linear;, score=0.978 total time=   9.8s
[CV 1/5] END …C=10, gamma=1, kernel=poly;, score=0.959 total time=   0.0s
[CV 2/5] END …C=10, gamma=1, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=10, gamma=1, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=10, gamma=1, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=10, gamma=1, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END …C=10, gamma=1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END …C=10, gamma=1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END …C=10, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END …C=10, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END …C=10, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.256 total time=   0.2s
[CV 2/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.256 total time=   0.2s
[CV 3/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.259 total time=   0.2s
[CV 4/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.263 total time=   0.2s
[CV 5/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.253 total time=   0.2s
[CV 1/5] END …C=10, gamma=0.1, kernel=linear;, score=0.972 total time=  13.1s
```

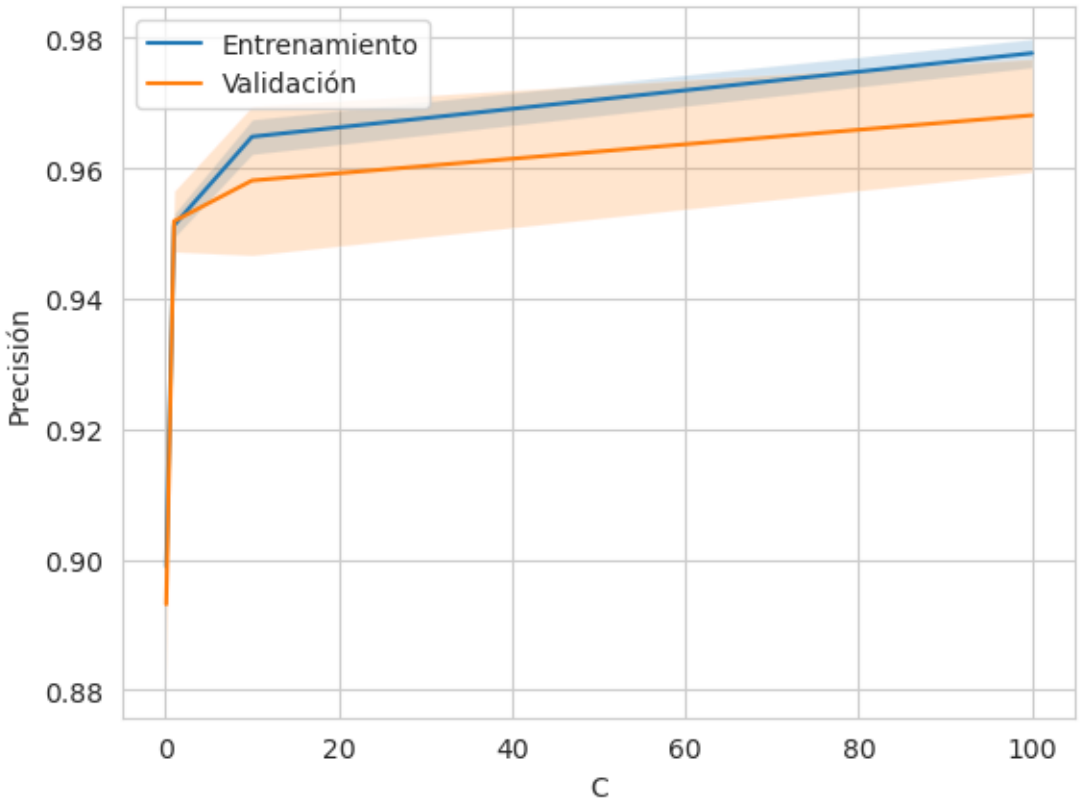
```
[CV 2/5] END …C=10, gamma=0.1, kernel=linear;, score=0.978 total time=   8.3s
[CV 3/5] END …C=10, gamma=0.1, kernel=linear;, score=0.978 total time=   9.0s
[CV 4/5] END …C=10, gamma=0.1, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=10, gamma=0.1, kernel=linear;, score=0.978 total time=   9.8s
[CV 1/5] END …C=10, gamma=0.1, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=10, gamma=0.1, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=10, gamma=0.1, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=10, gamma=0.1, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.1, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END …C=10, gamma=0.1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END …C=10, gamma=0.1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END …C=10, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END …C=10, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END …C=10, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.2s
[CV 2/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.2s
[CV 3/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.2s
[CV 4/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.259 total time=   0.2s
[CV 5/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.253 total time=   0.2s
[CV 1/5] END …C=10, gamma=0.01, kernel=linear;, score=0.972 total time=  13.1s
[CV 2/5] END …C=10, gamma=0.01, kernel=linear;, score=0.978 total time=   8.3s
[CV 3/5] END …C=10, gamma=0.01, kernel=linear;, score=0.978 total time=   9.0s
[CV 4/5] END …C=10, gamma=0.01, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=10, gamma=0.01, kernel=linear;, score=0.978 total time=   9.7s
[CV 1/5] END …C=10, gamma=0.01, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=10, gamma=0.01, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=10, gamma=0.01, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=10, gamma=0.01, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.01, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END ..C=10, gamma=0.01, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END ..C=10, gamma=0.01, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END ..C=10, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END ..C=10, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END ..C=10, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.259 total time=   0.2s
[CV 2/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.263 total time=   0.2s
[CV 3/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.266 total time=   0.2s
[CV 4/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.263 total time=   0.2s
[CV 5/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.263 total time=   0.2s
[CV 1/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.972 total time=  13.0s
[CV 2/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.978 total time=   8.3s
[CV 3/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.978 total time=   9.1s
[CV 4/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.959 total time=   2.0s
[CV 5/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.978 total time=   9.8s
[CV 1/5] END …C=10, gamma=0.001, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=10, gamma=0.001, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=10, gamma=0.001, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=10, gamma=0.001, kernel=poly;, score=0.953 total time=   0.0s
```

```
[CV 5/5] END …C=10, gamma=0.001, kernel=poly;, score=0.969 total time=   0.2s
[CV 1/5] END .C=10, gamma=0.001, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END .C=10, gamma=0.001, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END .C=10, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END .C=10, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END .C=10, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=100, gamma=1, kernel=rbf;, score=0.256 total time=   0.2s
[CV 2/5] END …C=100, gamma=1, kernel=rbf;, score=0.256 total time=   0.2s
[CV 3/5] END …C=100, gamma=1, kernel=rbf;, score=0.253 total time=   0.2s
[CV 4/5] END …C=100, gamma=1, kernel=rbf;, score=0.253 total time=   0.2s
[CV 5/5] END …C=100, gamma=1, kernel=rbf;, score=0.253 total time=   0.2s
[CV 1/5] END …C=100, gamma=1, kernel=linear;, score=0.975 total time=  17.3s
[CV 2/5] END …C=100, gamma=1, kernel=linear;, score=0.978 total time=  20.6s
[CV 3/5] END …C=100, gamma=1, kernel=linear;, score=0.984 total time=  12.3s
[CV 4/5] END …C=100, gamma=1, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=100, gamma=1, kernel=linear;, score=0.984 total time=  22.8s
[CV 1/5] END …C=100, gamma=1, kernel=poly;, score=0.959 total time=   0.0s
[CV 2/5] END …C=100, gamma=1, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=100, gamma=1, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=100, gamma=1, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=100, gamma=1, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END …C=100, gamma=1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END …C=100, gamma=1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END …C=100, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END …C=100, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END …C=100, gamma=1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.256 total time=   0.2s
[CV 2/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.256 total time=   0.2s
[CV 3/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.259 total time=   0.2s
[CV 4/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.263 total time=   0.2s
[CV 5/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.253 total time=   0.2s
[CV 1/5] END …C=100, gamma=0.1, kernel=linear;, score=0.975 total time=  17.5s
[CV 2/5] END …C=100, gamma=0.1, kernel=linear;, score=0.978 total time=  20.7s
[CV 3/5] END …C=100, gamma=0.1, kernel=linear;, score=0.984 total time=  12.2s
[CV 4/5] END …C=100, gamma=0.1, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END …C=100, gamma=0.1, kernel=linear;, score=0.984 total time=  22.5s
[CV 1/5] END …C=100, gamma=0.1, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=100, gamma=0.1, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=100, gamma=0.1, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=100, gamma=0.1, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.1, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END ..C=100, gamma=0.1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END ..C=100, gamma=0.1, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END ..C=100, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END ..C=100, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END ..C=100, gamma=0.1, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
[CV 2/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
```

```
[CV 3/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.256 total time=   0.1s
[CV 4/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.259 total time=   0.1s
[CV 5/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.253 total time=   0.2s
[CV 1/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.975 total time=  17.1s
[CV 2/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.978 total time=  20.2s
[CV 3/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.984 total time=  12.0s
[CV 4/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.984 total time=  22.1s
[CV 1/5] END …C=100, gamma=0.01, kernel=poly;, score=0.959 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.01, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=100, gamma=0.01, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=100, gamma=0.01, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.01, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END .C=100, gamma=0.01, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END .C=100, gamma=0.01, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END .C=100, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END .C=100, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END .C=100, gamma=0.01, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 1/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.259 total time=   0.2s
[CV 2/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.263 total time=   0.2s
[CV 3/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.266 total time=   0.2s
[CV 4/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.263 total time=   0.2s
[CV 5/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.263 total time=   0.2s
[CV 1/5] END .C=100, gamma=0.001, kernel=linear;, score=0.975 total time=  17.0s
[CV 2/5] END .C=100, gamma=0.001, kernel=linear;, score=0.978 total time=  20.1s
[CV 3/5] END .C=100, gamma=0.001, kernel=linear;, score=0.984 total time=  12.0s
[CV 4/5] END .C=100, gamma=0.001, kernel=linear;, score=0.959 total time=   1.9s
[CV 5/5] END .C=100, gamma=0.001, kernel=linear;, score=0.984 total time=  22.2s
[CV 1/5] END …C=100, gamma=0.001, kernel=poly;, score=0.959 total time=   0.1s
[CV 2/5] END …C=100, gamma=0.001, kernel=poly;, score=0.956 total time=   0.1s
[CV 3/5] END …C=100, gamma=0.001, kernel=poly;, score=0.944 total time=   0.1s
[CV 4/5] END …C=100, gamma=0.001, kernel=poly;, score=0.953 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.001, kernel=poly;, score=0.969 total time=   0.1s
[CV 1/5] END C=100, gamma=0.001, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 2/5] END C=100, gamma=0.001, kernel=sigmoid;, score=0.256 total time=   0.1s
[CV 3/5] END C=100, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 4/5] END C=100, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
[CV 5/5] END C=100, gamma=0.001, kernel=sigmoid;, score=0.253 total time=   0.1s
Precisión de la predicción: 97.25%
Matriz de confusión:
[[ 99   6   0   0]
 [  0  91   0   0]
 [  0   2  89   1]
 [  0   0   2 110]]
```

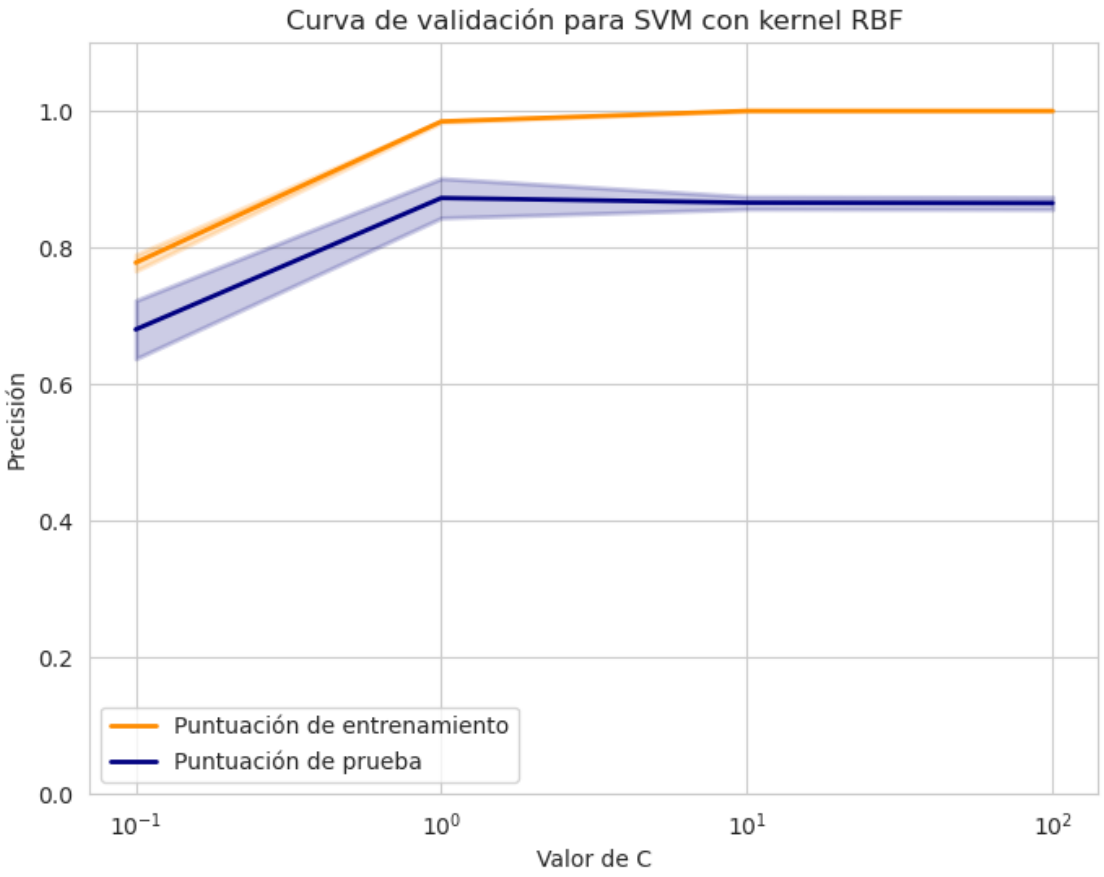
[10]:
```
from sklearn.model_selection import validation_curve
```

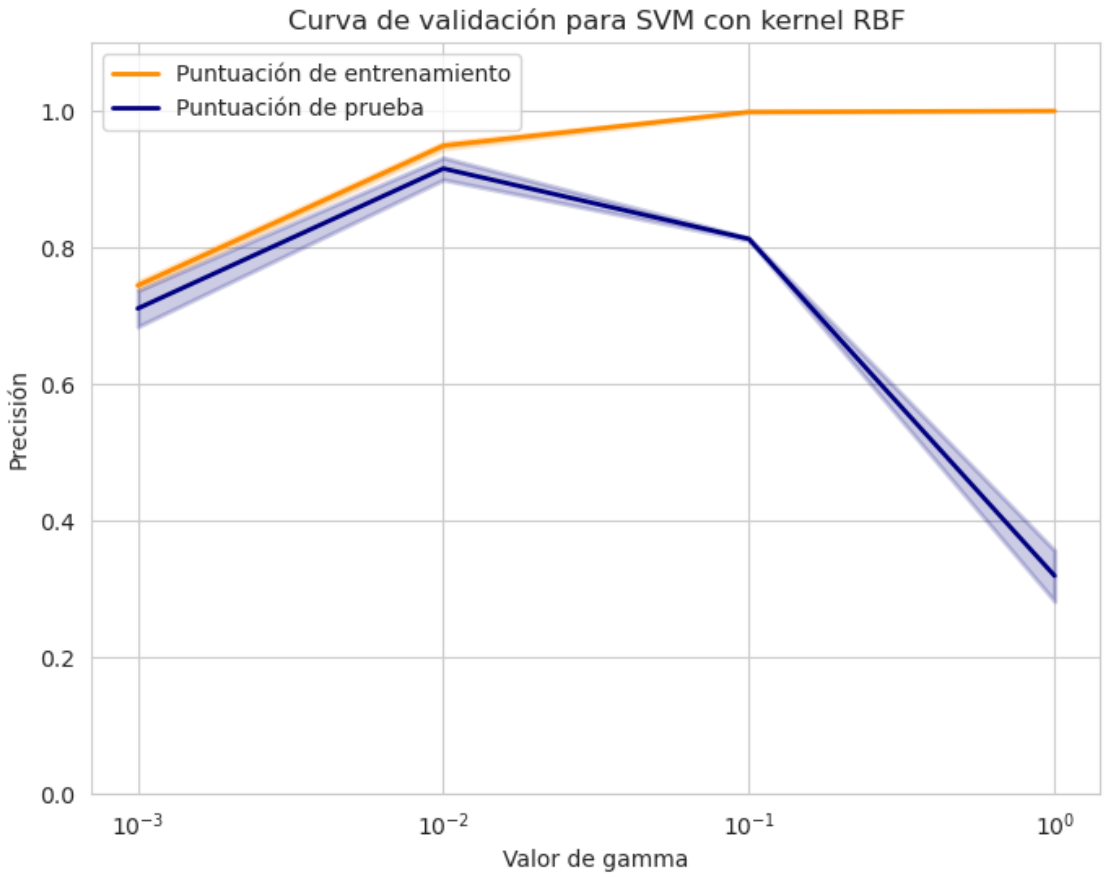
```
# Creación del modelo
model = SVC()

# Definición de los parámetros a ajustar
param_range = [0.1, 1, 10, 100]
param_name = 'C'

# Validación cruzada para obtener las curvas de validación y entrenamiento
train_scores, test_scores = validation_curve(model, X_train, y_train,
                                             param_name=param_name,
                                             param_range=param_range,
                                             cv=5)

# Cálculo de las medias y desviaciones estándar de las curvas de validación y␣
 ↪entrenamiento
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Graficar las curvas de validación y entrenamiento
plt.plot(param_range, train_mean, label="Entrenamiento")
plt.fill_between(param_range, train_mean - train_std, train_mean + train_std,␣
 ↪alpha=0.2)
plt.plot(param_range, test_mean, label="Validación")
plt.fill_between(param_range, test_mean - test_std, test_mean + test_std,␣
 ↪alpha=0.2)
plt.xlabel(param_name)
plt.ylabel("Precisión")
plt.legend(loc="best")
plt.show()
```




[21]:
```
from sklearn.model_selection import validation_curve

# Definición de los valores para los parámetros que se van a ajustar
param_range_C = [0.1, 1, 10, 100]
param_range_gamma = [1, 0.1, 0.01, 0.001]

# Calcular la curva de validación para cada hiperparámetro
train_scores, test_scores = validation_curve(
    SVC(), X_train, y_train, param_name="C", param_range=param_range_C,
    cv=5, scoring="accuracy", n_jobs=-1)

train_scores2, test_scores2 = validation_curve(
    SVC(), X_train, y_train, param_name="gamma", param_range=param_range_gamma,
    cv=5, scoring="accuracy", n_jobs=-1)

# Calcular el promedio y la desviación estándar de las puntuaciones de␣
 ↪entrenamiento y prueba
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)
```

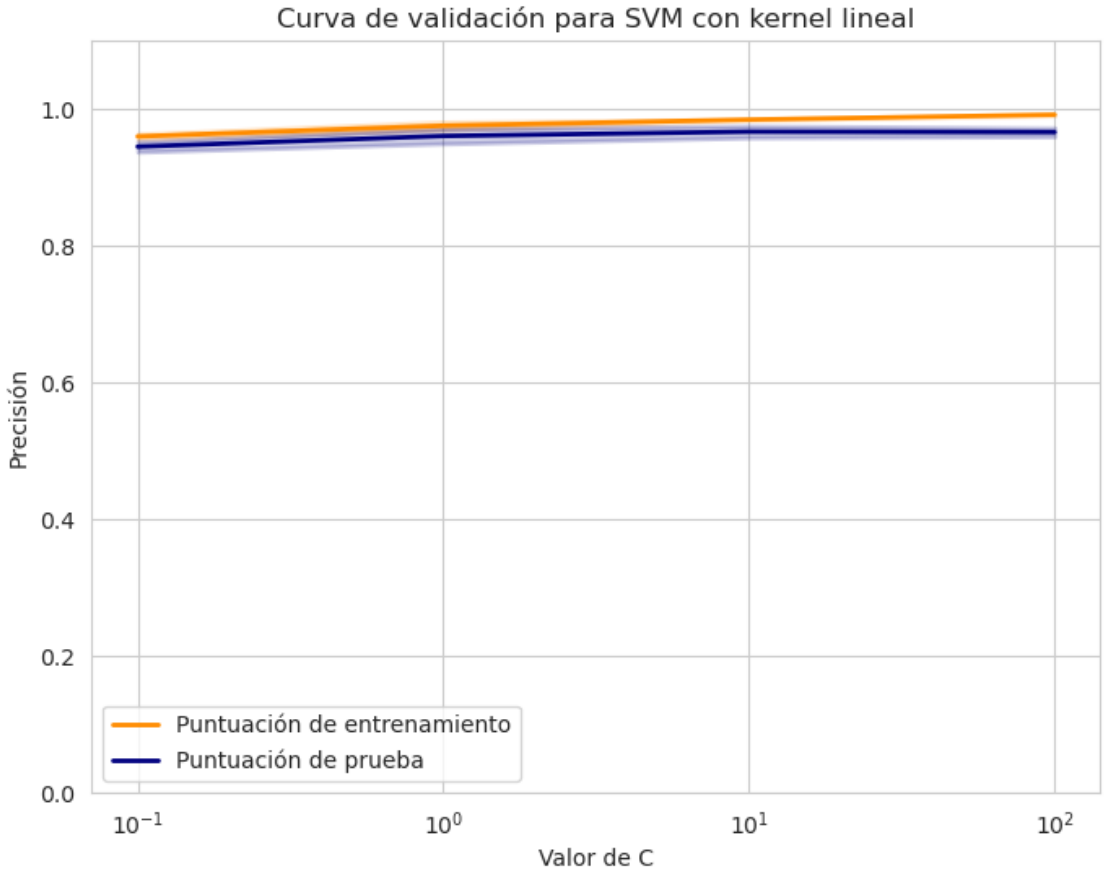
```
train_mean2 = np.mean(train_scores2, axis=1)
train_std2 = np.std(train_scores2, axis=1)
test_mean2 = np.mean(test_scores2, axis=1)
test_std2 = np.std(test_scores2, axis=1)

# Graficar la curva de validación para C
plt.figure(figsize=(8, 6))
plt.title("Curva de validación para SVM con kernel RBF")
plt.xlabel("Valor de C")
plt.ylabel("Precisión")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range_C, train_mean, label="Puntuación de entrenamiento",
             color="darkorange", lw=lw)
plt.fill_between(param_range_C, train_mean - train_std,
                 train_mean + train_std, alpha=0.2,
                 color="darkorange", lw=lw)
plt.semilogx(param_range_C, test_mean, label="Puntuación de prueba",
             color="navy", lw=lw)
plt.fill_between(param_range_C, test_mean - test_std,
                 test_mean + test_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")

# Graficar la curva de validación para gamma
plt.figure(figsize=(8, 6))
plt.title("Curva de validación para SVM con kernel RBF")
plt.xlabel("Valor de gamma")
plt.ylabel("Precisión")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range_gamma, train_mean2, label="Puntuación de␣
 ↪entrenamiento",
             color="darkorange", lw=lw)
plt.fill_between(param_range_gamma, train_mean2 - train_std2,
                 train_mean2 + train_std2, alpha=0.2,
                 color="darkorange", lw=lw)
plt.semilogx(param_range_gamma, test_mean2, label="Puntuación de prueba",
             color="navy", lw=lw)
plt.fill_between(param_range_gamma, test_mean2 - test_std2,
                 test_mean2 + test_std2, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```


Curva de validación para SVM con kernel RBF
Precisión
1.0
0.8
0.6
0.4
0.2
0.0
$10^{-1}$
$10^{0}$
$10^{1}$
$10^{2}$
Valor de C
Puntuación de entrenamiento
Puntuación de prueba

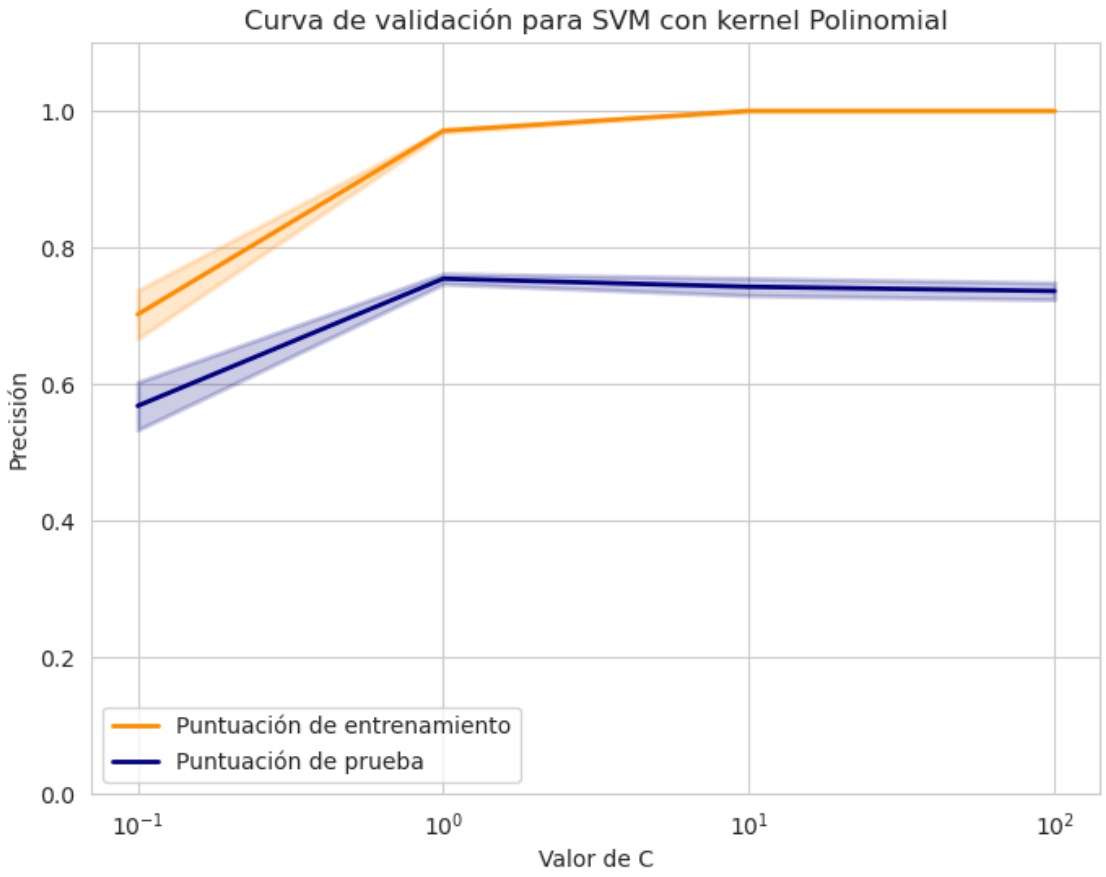



[22]:
```
from sklearn.svm import SVC
from sklearn.model_selection import validation_curve

# Definición de los valores para los parámetros que se van a ajustar
param_range_C = [0.1, 1, 10, 100]

# Calcular la curva de validación para C
train_scores, test_scores = validation_curve(
    SVC(kernel='linear'), X_train, y_train, param_name="C",␣
 ↪param_range=param_range_C,
    cv=5, scoring="accuracy", n_jobs=-1)

# Calcular el promedio y la desviación estándar de las puntuaciones de␣
 ↪entrenamiento y prueba
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)
```

```
# Graficar la curva de validación para C
plt.figure(figsize=(8, 6))
plt.title("Curva de validación para SVM con kernel lineal")
plt.xlabel("Valor de C")
plt.ylabel("Precisión")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range_C, train_mean, label="Puntuación de entrenamiento",
             color="darkorange", lw=lw)
plt.fill_between(param_range_C, train_mean - train_std,
                 train_mean + train_std, alpha=0.2,
                 color="darkorange", lw=lw)
plt.semilogx(param_range_C, test_mean, label="Puntuación de prueba",
             color="navy", lw=lw)
plt.fill_between(param_range_C, test_mean - test_std,
                 test_mean + test_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

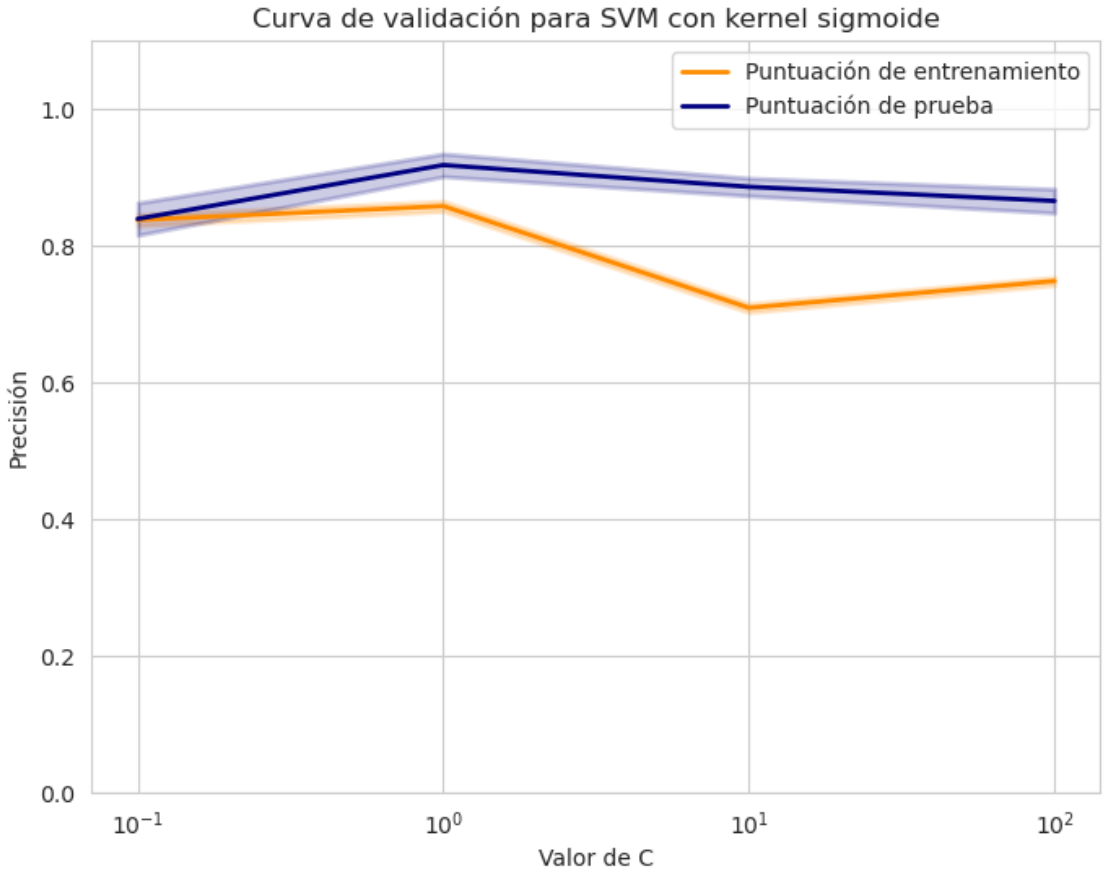


[23]:
```
from sklearn.svm import SVC
from sklearn.model_selection import validation_curve

# Definición de los valores para los parámetros que se van a ajustar
param_range_C = [0.1, 1, 10, 100]

# Calcular la curva de validación para C
train_scores, test_scores = validation_curve(
    SVC(kernel='poly'), X_train, y_train, param_name="C",␣
 ↪param_range=param_range_C,
    cv=5, scoring="accuracy", n_jobs=-1)

# Calcular el promedio y la desviación estándar de las puntuaciones de␣
 ↪entrenamiento y prueba
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Graficar la curva de validación para C
plt.figure(figsize=(8, 6))
plt.title("Curva de validación para SVM con kernel Polinomial")
plt.xlabel("Valor de C")
plt.ylabel("Precisión")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range_C, train_mean, label="Puntuación de entrenamiento",
             color="darkorange", lw=lw)
plt.fill_between(param_range_C, train_mean - train_std,
                 train_mean + train_std, alpha=0.2,
                 color="darkorange", lw=lw)
plt.semilogx(param_range_C, test_mean, label="Puntuación de prueba",
             color="navy", lw=lw)
plt.fill_between(param_range_C, test_mean - test_std,
                 test_mean + test_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```




[24]:
```
from sklearn.svm import SVC
from sklearn.model_selection import validation_curve

# Definición de los valores para los parámetros que se van a ajustar
param_range_C = [0.1, 1, 10, 100]

# Calcular la curva de validación para C
train_scores, test_scores = validation_curve(
    SVC(kernel='sigmoid'), X_train, y_train, param_name="C",␣
 ↪param_range=param_range_C,
    cv=5, scoring="accuracy", n_jobs=-1)

# Calcular el promedio y la desviación estándar de las puntuaciones de␣
 ↪entrenamiento y prueba
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)
```

```
# Graficar la curva de validación para C
plt.figure(figsize=(8, 6))
plt.title("Curva de validación para SVM con kernel sigmoide")
plt.xlabel("Valor de C")
plt.ylabel("Precisión")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range_C, train_mean, label="Puntuación de entrenamiento",
             color="darkorange", lw=lw)
plt.fill_between(param_range_C, train_mean - train_std,
                 train_mean + train_std, alpha=0.2,
                 color="darkorange", lw=lw)
plt.semilogx(param_range_C, test_mean, label="Puntuación de prueba",
             color="navy", lw=lw)
plt.fill_between(param_range_C, test_mean - test_std,
                 test_mean + test_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```



# 2 SEGUNDA PARTE: CLASIFICACIÓN CON REDES NEURONALES

[15]:
```
# Lectura de los archivos CSV
data = pd.read_csv("train.csv")
scaler = StandardScaler()
X = scaler.fit_transform(data.drop("price_range", axis=1))

y = data["price_range"]

# División del dataset en datos de entrenamiento y de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Creación del modelo de red neuronal
model = Sequential()

# Añadir capas ocultas
model.add(Dense(128, input_dim=X_train.shape[1], activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu', kernel_regularizer=l2(0.01)))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu', kernel_regularizer=l2(0.01)))
model.add(Dropout(0.5))

# Añadir capa de salida
model.add(Dense(4, activation='softmax'))

# Compilación del modelo
optimizer = Adam(learning_rate=0.0001)
model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizer,␣
 ↪metrics=['accuracy'])

# Entrenamiento del modelo
early_stop = EarlyStopping(monitor='val_loss', patience=20)
history = model.fit(X_train, y_train, epochs=200, batch_size=128,␣
 ↪validation_split=0.1, callbacks=[early_stop])

# Evaluación del modelo
loss, accuracy = model.evaluate(X_test, y_test)
print('Precisión: %.2f' % (accuracy*100))
```

```
Epoch 1/200
```

```
2023-03-13 17:18:29.403898: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open
```

```
shared object file: No such file or directory
2023-03-13 17:18:29.403917: W
tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:265] failed call to
cuInit: UNKNOWN ERROR (303)
2023-03-13 17:18:29.403930: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_diagnostics.cc:156] kernel
driver does not appear to be running on this host (gavinci-B550M-DS3H):
/proc/driver/nvidia/version does not exist
2023-03-13 17:18:29.404081: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
12/12 [==============================] - 0s 12ms/step - loss: 2.9143 - accuracy:
0.2576 - val_loss: 2.6777 - val_accuracy: 0.2438
Epoch 2/200
12/12 [==============================] - 0s 2ms/step - loss: 2.8570 - accuracy:
0.2507 - val_loss: 2.6637 - val_accuracy: 0.2500
Epoch 3/200
12/12 [==============================] - 0s 2ms/step - loss: 2.8583 - accuracy:
0.2438 - val_loss: 2.6507 - val_accuracy: 0.2438
Epoch 4/200
12/12 [==============================] - 0s 2ms/step - loss: 2.8240 - accuracy:
0.2403 - val_loss: 2.6387 - val_accuracy: 0.2375
Epoch 5/200
12/12 [==============================] - 0s 2ms/step - loss: 2.7908 - accuracy:
0.2514 - val_loss: 2.6277 - val_accuracy: 0.2313
Epoch 6/200
12/12 [==============================] - 0s 2ms/step - loss: 2.7946 - accuracy:
0.2417 - val_loss: 2.6167 - val_accuracy: 0.2375
Epoch 7/200
12/12 [==============================] - 0s 2ms/step - loss: 2.7521 - accuracy:
0.2667 - val_loss: 2.6059 - val_accuracy: 0.2438
Epoch 8/200
12/12 [==============================] - 0s 2ms/step - loss: 2.7295 - accuracy:
0.2458 - val_loss: 2.5954 - val_accuracy: 0.2562
Epoch 9/200
12/12 [==============================] - 0s 2ms/step - loss: 2.7512 - accuracy:
0.2472 - val_loss: 2.5853 - val_accuracy: 0.2625
Epoch 10/200
12/12 [==============================] - 0s 2ms/step - loss: 2.7020 - accuracy:
0.2597 - val_loss: 2.5752 - val_accuracy: 0.2625
Epoch 11/200
12/12 [==============================] - 0s 2ms/step - loss: 2.6889 - accuracy:
0.2618 - val_loss: 2.5653 - val_accuracy: 0.2500
Epoch 12/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 2.6707 - accuracy:
0.2549 - val_loss: 2.5554 - val_accuracy: 0.2500
Epoch 13/200
12/12 [==============================] - 0s 2ms/step - loss: 2.6713 - accuracy:
0.2611 - val_loss: 2.5459 - val_accuracy: 0.2500
Epoch 14/200
12/12 [==============================] - 0s 2ms/step - loss: 2.6506 - accuracy:
0.2667 - val_loss: 2.5364 - val_accuracy: 0.2688
Epoch 15/200
12/12 [==============================] - 0s 2ms/step - loss: 2.6288 - accuracy:
0.2618 - val_loss: 2.5273 - val_accuracy: 0.2688
Epoch 16/200
12/12 [==============================] - 0s 2ms/step - loss: 2.6083 - accuracy:
0.2868 - val_loss: 2.5182 - val_accuracy: 0.2750
Epoch 17/200
12/12 [==============================] - 0s 2ms/step - loss: 2.6075 - accuracy:
0.2674 - val_loss: 2.5088 - val_accuracy: 0.2812
Epoch 18/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5919 - accuracy:
0.2660 - val_loss: 2.4994 - val_accuracy: 0.2812
Epoch 19/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5926 - accuracy:
0.2639 - val_loss: 2.4908 - val_accuracy: 0.3000
Epoch 20/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5881 - accuracy:
0.2674 - val_loss: 2.4818 - val_accuracy: 0.3187
Epoch 21/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5571 - accuracy:
0.2785 - val_loss: 2.4726 - val_accuracy: 0.3187
Epoch 22/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5558 - accuracy:
0.2632 - val_loss: 2.4634 - val_accuracy: 0.3250
Epoch 23/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5192 - accuracy:
0.2965 - val_loss: 2.4545 - val_accuracy: 0.3187
Epoch 24/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5263 - accuracy:
0.2944 - val_loss: 2.4460 - val_accuracy: 0.3250
Epoch 25/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5057 - accuracy:
0.2903 - val_loss: 2.4374 - val_accuracy: 0.3250
Epoch 26/200
12/12 [==============================] - 0s 2ms/step - loss: 2.5205 - accuracy:
0.2785 - val_loss: 2.4287 - val_accuracy: 0.3250
Epoch 27/200
12/12 [==============================] - 0s 2ms/step - loss: 2.4719 - accuracy:
0.2986 - val_loss: 2.4201 - val_accuracy: 0.3313
Epoch 28/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 2.4882 - accuracy:
0.2708 - val_loss: 2.4115 - val_accuracy: 0.3313
Epoch 29/200
12/12 [==============================] - 0s 2ms/step - loss: 2.4640 - accuracy:
0.3035 - val_loss: 2.4029 - val_accuracy: 0.3375
Epoch 30/200
12/12 [==============================] - 0s 2ms/step - loss: 2.4593 - accuracy:
0.3021 - val_loss: 2.3942 - val_accuracy: 0.3438
Epoch 31/200
12/12 [==============================] - 0s 2ms/step - loss: 2.4354 - accuracy:
0.3042 - val_loss: 2.3854 - val_accuracy: 0.3562
Epoch 32/200
12/12 [==============================] - 0s 2ms/step - loss: 2.4458 - accuracy:
0.2896 - val_loss: 2.3769 - val_accuracy: 0.3562
Epoch 33/200
12/12 [==============================] - 0s 2ms/step - loss: 2.4237 - accuracy:
0.3035 - val_loss: 2.3680 - val_accuracy: 0.3625
Epoch 34/200
12/12 [==============================] - 0s 2ms/step - loss: 2.4230 - accuracy:
0.3063 - val_loss: 2.3591 - val_accuracy: 0.3812
Epoch 35/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3987 - accuracy:
0.3049 - val_loss: 2.3503 - val_accuracy: 0.3875
Epoch 36/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3959 - accuracy:
0.2937 - val_loss: 2.3412 - val_accuracy: 0.3938
Epoch 37/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3837 - accuracy:
0.3187 - val_loss: 2.3324 - val_accuracy: 0.4000
Epoch 38/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3900 - accuracy:
0.2917 - val_loss: 2.3232 - val_accuracy: 0.4000
Epoch 39/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3715 - accuracy:
0.3104 - val_loss: 2.3140 - val_accuracy: 0.4000
Epoch 40/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3544 - accuracy:
0.3076 - val_loss: 2.3051 - val_accuracy: 0.4062
Epoch 41/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3333 - accuracy:
0.3229 - val_loss: 2.2960 - val_accuracy: 0.4125
Epoch 42/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3225 - accuracy:
0.3222 - val_loss: 2.2865 - val_accuracy: 0.4250
Epoch 43/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3134 - accuracy:
0.3187 - val_loss: 2.2772 - val_accuracy: 0.4313
Epoch 44/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 2.3024 - accuracy:
0.3347 - val_loss: 2.2677 - val_accuracy: 0.4375
Epoch 45/200
12/12 [==============================] - 0s 2ms/step - loss: 2.3021 - accuracy:
0.3243 - val_loss: 2.2581 - val_accuracy: 0.4375
Epoch 46/200
12/12 [==============================] - 0s 2ms/step - loss: 2.2982 - accuracy:
0.3097 - val_loss: 2.2479 - val_accuracy: 0.4437
Epoch 47/200
12/12 [==============================] - 0s 2ms/step - loss: 2.2959 - accuracy:
0.3076 - val_loss: 2.2385 - val_accuracy: 0.4500
Epoch 48/200
12/12 [==============================] - 0s 2ms/step - loss: 2.2879 - accuracy:
0.3271 - val_loss: 2.2285 - val_accuracy: 0.4500
Epoch 49/200
12/12 [==============================] - 0s 2ms/step - loss: 2.2616 - accuracy:
0.3569 - val_loss: 2.2184 - val_accuracy: 0.4625
Epoch 50/200
12/12 [==============================] - 0s 2ms/step - loss: 2.2561 - accuracy:
0.3368 - val_loss: 2.2085 - val_accuracy: 0.4750
Epoch 51/200
12/12 [==============================] - 0s 2ms/step - loss: 2.2464 - accuracy:
0.3375 - val_loss: 2.1986 - val_accuracy: 0.4750
Epoch 52/200
12/12 [==============================] - 0s 2ms/step - loss: 2.2406 - accuracy:
0.3299 - val_loss: 2.1884 - val_accuracy: 0.4812
Epoch 53/200
12/12 [==============================] - 0s 2ms/step - loss: 2.2156 - accuracy:
0.3500 - val_loss: 2.1780 - val_accuracy: 0.4875
Epoch 54/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1945 - accuracy:
0.3757 - val_loss: 2.1678 - val_accuracy: 0.4938
Epoch 55/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1958 - accuracy:
0.3694 - val_loss: 2.1566 - val_accuracy: 0.5063
Epoch 56/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1950 - accuracy:
0.3646 - val_loss: 2.1456 - val_accuracy: 0.5000
Epoch 57/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1788 - accuracy:
0.3576 - val_loss: 2.1346 - val_accuracy: 0.5000
Epoch 58/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1783 - accuracy:
0.3715 - val_loss: 2.1232 - val_accuracy: 0.5000
Epoch 59/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1475 - accuracy:
0.3757 - val_loss: 2.1123 - val_accuracy: 0.5000
Epoch 60/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 2.1500 - accuracy:
0.3562 - val_loss: 2.1015 - val_accuracy: 0.5063
Epoch 61/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1332 - accuracy:
0.3667 - val_loss: 2.0902 - val_accuracy: 0.5125
Epoch 62/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1289 - accuracy:
0.3750 - val_loss: 2.0787 - val_accuracy: 0.5063
Epoch 63/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1224 - accuracy:
0.3847 - val_loss: 2.0673 - val_accuracy: 0.5063
Epoch 64/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1051 - accuracy:
0.3938 - val_loss: 2.0553 - val_accuracy: 0.5125
Epoch 65/200
12/12 [==============================] - 0s 2ms/step - loss: 2.1158 - accuracy:
0.3778 - val_loss: 2.0445 - val_accuracy: 0.5188
Epoch 66/200
12/12 [==============================] - 0s 2ms/step - loss: 2.0937 - accuracy:
0.3833 - val_loss: 2.0328 - val_accuracy: 0.5188
Epoch 67/200
12/12 [==============================] - 0s 2ms/step - loss: 2.0875 - accuracy:
0.3785 - val_loss: 2.0207 - val_accuracy: 0.5250
Epoch 68/200
12/12 [==============================] - 0s 2ms/step - loss: 2.0774 - accuracy:
0.4042 - val_loss: 2.0086 - val_accuracy: 0.5375
Epoch 69/200
12/12 [==============================] - 0s 2ms/step - loss: 2.0472 - accuracy:
0.4056 - val_loss: 1.9955 - val_accuracy: 0.5312
Epoch 70/200
12/12 [==============================] - 0s 2ms/step - loss: 2.0439 - accuracy:
0.4042 - val_loss: 1.9824 - val_accuracy: 0.5312
Epoch 71/200
12/12 [==============================] - 0s 2ms/step - loss: 2.0290 - accuracy:
0.4194 - val_loss: 1.9691 - val_accuracy: 0.5437
Epoch 72/200
12/12 [==============================] - 0s 2ms/step - loss: 2.0192 - accuracy:
0.4104 - val_loss: 1.9560 - val_accuracy: 0.5562
Epoch 73/200
12/12 [==============================] - 0s 2ms/step - loss: 2.0159 - accuracy:
0.4243 - val_loss: 1.9427 - val_accuracy: 0.5750
Epoch 74/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9847 - accuracy:
0.4222 - val_loss: 1.9291 - val_accuracy: 0.5688
Epoch 75/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9811 - accuracy:
0.4194 - val_loss: 1.9153 - val_accuracy: 0.5750
Epoch 76/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 1.9836 - accuracy:
0.4271 - val_loss: 1.9012 - val_accuracy: 0.5813
Epoch 77/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9546 - accuracy:
0.4299 - val_loss: 1.8875 - val_accuracy: 0.5875
Epoch 78/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9660 - accuracy:
0.4326 - val_loss: 1.8737 - val_accuracy: 0.6125
Epoch 79/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9304 - accuracy:
0.4257 - val_loss: 1.8605 - val_accuracy: 0.6250
Epoch 80/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9408 - accuracy:
0.4187 - val_loss: 1.8467 - val_accuracy: 0.6250
Epoch 81/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9350 - accuracy:
0.4326 - val_loss: 1.8338 - val_accuracy: 0.6250
Epoch 82/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9122 - accuracy:
0.4479 - val_loss: 1.8206 - val_accuracy: 0.6438
Epoch 83/200
12/12 [==============================] - 0s 2ms/step - loss: 1.9020 - accuracy:
0.4514 - val_loss: 1.8067 - val_accuracy: 0.6313
Epoch 84/200
12/12 [==============================] - 0s 2ms/step - loss: 1.8996 - accuracy:
0.4493 - val_loss: 1.7934 - val_accuracy: 0.6313
Epoch 85/200
12/12 [==============================] - 0s 2ms/step - loss: 1.8694 - accuracy:
0.4639 - val_loss: 1.7802 - val_accuracy: 0.6375
Epoch 86/200
12/12 [==============================] - 0s 2ms/step - loss: 1.8625 - accuracy:
0.4611 - val_loss: 1.7663 - val_accuracy: 0.6375
Epoch 87/200
12/12 [==============================] - 0s 2ms/step - loss: 1.8477 - accuracy:
0.4563 - val_loss: 1.7524 - val_accuracy: 0.6438
Epoch 88/200
12/12 [==============================] - 0s 2ms/step - loss: 1.8625 - accuracy:
0.4493 - val_loss: 1.7395 - val_accuracy: 0.6500
Epoch 89/200
12/12 [==============================] - 0s 2ms/step - loss: 1.8405 - accuracy:
0.4667 - val_loss: 1.7272 - val_accuracy: 0.6313
Epoch 90/200
12/12 [==============================] - 0s 2ms/step - loss: 1.8226 - accuracy:
0.4826 - val_loss: 1.7136 - val_accuracy: 0.6375
Epoch 91/200
12/12 [==============================] - 0s 2ms/step - loss: 1.8069 - accuracy:
0.4910 - val_loss: 1.7002 - val_accuracy: 0.6313
Epoch 92/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 1.8075 - accuracy:
0.4694 - val_loss: 1.6872 - val_accuracy: 0.6313
Epoch 93/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7886 - accuracy:
0.4792 - val_loss: 1.6751 - val_accuracy: 0.6125
Epoch 94/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7794 - accuracy:
0.4993 - val_loss: 1.6631 - val_accuracy: 0.6000
Epoch 95/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7845 - accuracy:
0.4681 - val_loss: 1.6509 - val_accuracy: 0.6062
Epoch 96/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7464 - accuracy:
0.4938 - val_loss: 1.6380 - val_accuracy: 0.6000
Epoch 97/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7442 - accuracy:
0.4729 - val_loss: 1.6255 - val_accuracy: 0.6000
Epoch 98/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7451 - accuracy:
0.4806 - val_loss: 1.6137 - val_accuracy: 0.6000
Epoch 99/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7202 - accuracy:
0.4826 - val_loss: 1.6017 - val_accuracy: 0.6000
Epoch 100/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7062 - accuracy:
0.5014 - val_loss: 1.5895 - val_accuracy: 0.6000
Epoch 101/200
12/12 [==============================] - 0s 2ms/step - loss: 1.7089 - accuracy:
0.4979 - val_loss: 1.5778 - val_accuracy: 0.6000
Epoch 102/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6967 - accuracy:
0.4965 - val_loss: 1.5659 - val_accuracy: 0.5938
Epoch 103/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6669 - accuracy:
0.5292 - val_loss: 1.5537 - val_accuracy: 0.6000
Epoch 104/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6777 - accuracy:
0.5056 - val_loss: 1.5416 - val_accuracy: 0.6062
Epoch 105/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6767 - accuracy:
0.5097 - val_loss: 1.5303 - val_accuracy: 0.6062
Epoch 106/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6601 - accuracy:
0.5083 - val_loss: 1.5194 - val_accuracy: 0.6000
Epoch 107/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6390 - accuracy:
0.5243 - val_loss: 1.5088 - val_accuracy: 0.6062
Epoch 108/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 1.6203 - accuracy:
0.5347 - val_loss: 1.4979 - val_accuracy: 0.6187
Epoch 109/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6325 - accuracy:
0.5118 - val_loss: 1.4879 - val_accuracy: 0.6187
Epoch 110/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6257 - accuracy:
0.5146 - val_loss: 1.4783 - val_accuracy: 0.6187
Epoch 111/200
12/12 [==============================] - 0s 2ms/step - loss: 1.6181 - accuracy:
0.5160 - val_loss: 1.4696 - val_accuracy: 0.6250
Epoch 112/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5937 - accuracy:
0.5312 - val_loss: 1.4602 - val_accuracy: 0.6250
Epoch 113/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5868 - accuracy:
0.5285 - val_loss: 1.4505 - val_accuracy: 0.6250
Epoch 114/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5824 - accuracy:
0.5472 - val_loss: 1.4407 - val_accuracy: 0.6375
Epoch 115/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5706 - accuracy:
0.5417 - val_loss: 1.4313 - val_accuracy: 0.6375
Epoch 116/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5656 - accuracy:
0.5222 - val_loss: 1.4224 - val_accuracy: 0.6250
Epoch 117/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5431 - accuracy:
0.5625 - val_loss: 1.4128 - val_accuracy: 0.6125
Epoch 118/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5664 - accuracy:
0.5347 - val_loss: 1.4042 - val_accuracy: 0.6187
Epoch 119/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5521 - accuracy:
0.5285 - val_loss: 1.3954 - val_accuracy: 0.6313
Epoch 120/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5278 - accuracy:
0.5389 - val_loss: 1.3869 - val_accuracy: 0.6250
Epoch 121/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5224 - accuracy:
0.5319 - val_loss: 1.3779 - val_accuracy: 0.6375
Epoch 122/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5084 - accuracy:
0.5500 - val_loss: 1.3687 - val_accuracy: 0.6375
Epoch 123/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4942 - accuracy:
0.5611 - val_loss: 1.3590 - val_accuracy: 0.6438
Epoch 124/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 1.4964 - accuracy:
0.5493 - val_loss: 1.3493 - val_accuracy: 0.6438
Epoch 125/200
12/12 [==============================] - 0s 2ms/step - loss: 1.5053 - accuracy:
0.5340 - val_loss: 1.3405 - val_accuracy: 0.6625
Epoch 126/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4880 - accuracy:
0.5465 - val_loss: 1.3320 - val_accuracy: 0.6625
Epoch 127/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4869 - accuracy:
0.5465 - val_loss: 1.3234 - val_accuracy: 0.6750
Epoch 128/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4770 - accuracy:
0.5535 - val_loss: 1.3147 - val_accuracy: 0.6812
Epoch 129/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4573 - accuracy:
0.5500 - val_loss: 1.3059 - val_accuracy: 0.6812
Epoch 130/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4554 - accuracy:
0.5562 - val_loss: 1.2974 - val_accuracy: 0.6875
Epoch 131/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4419 - accuracy:
0.5618 - val_loss: 1.2893 - val_accuracy: 0.6938
Epoch 132/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4462 - accuracy:
0.5611 - val_loss: 1.2817 - val_accuracy: 0.6938
Epoch 133/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4288 - accuracy:
0.5729 - val_loss: 1.2740 - val_accuracy: 0.6938
Epoch 134/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4186 - accuracy:
0.5708 - val_loss: 1.2657 - val_accuracy: 0.7063
Epoch 135/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4117 - accuracy:
0.5646 - val_loss: 1.2574 - val_accuracy: 0.7063
Epoch 136/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3943 - accuracy:
0.5826 - val_loss: 1.2498 - val_accuracy: 0.7063
Epoch 137/200
12/12 [==============================] - 0s 2ms/step - loss: 1.4016 - accuracy:
0.5806 - val_loss: 1.2418 - val_accuracy: 0.7125
Epoch 138/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3911 - accuracy:
0.5750 - val_loss: 1.2337 - val_accuracy: 0.7125
Epoch 139/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3778 - accuracy:
0.5903 - val_loss: 1.2254 - val_accuracy: 0.7125
Epoch 140/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 1.3894 - accuracy:
0.5653 - val_loss: 1.2179 - val_accuracy: 0.7188
Epoch 141/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3775 - accuracy:
0.5764 - val_loss: 1.2107 - val_accuracy: 0.7250
Epoch 142/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3726 - accuracy:
0.5882 - val_loss: 1.2028 - val_accuracy: 0.7250
Epoch 143/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3617 - accuracy:
0.5965 - val_loss: 1.1956 - val_accuracy: 0.7250
Epoch 144/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3386 - accuracy:
0.5868 - val_loss: 1.1888 - val_accuracy: 0.7188
Epoch 145/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3461 - accuracy:
0.5993 - val_loss: 1.1812 - val_accuracy: 0.7188
Epoch 146/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3490 - accuracy:
0.5840 - val_loss: 1.1746 - val_accuracy: 0.7312
Epoch 147/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3293 - accuracy:
0.6028 - val_loss: 1.1675 - val_accuracy: 0.7375
Epoch 148/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3355 - accuracy:
0.5715 - val_loss: 1.1603 - val_accuracy: 0.7437
Epoch 149/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3318 - accuracy:
0.5875 - val_loss: 1.1534 - val_accuracy: 0.7437
Epoch 150/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3087 - accuracy:
0.5951 - val_loss: 1.1469 - val_accuracy: 0.7500
Epoch 151/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3116 - accuracy:
0.5965 - val_loss: 1.1395 - val_accuracy: 0.7563
Epoch 152/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3002 - accuracy:
0.5993 - val_loss: 1.1324 - val_accuracy: 0.7500
Epoch 153/200
12/12 [==============================] - 0s 2ms/step - loss: 1.3079 - accuracy:
0.6049 - val_loss: 1.1256 - val_accuracy: 0.7750
Epoch 154/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2961 - accuracy:
0.6097 - val_loss: 1.1187 - val_accuracy: 0.7625
Epoch 155/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2760 - accuracy:
0.6139 - val_loss: 1.1119 - val_accuracy: 0.7625
Epoch 156/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 1.2853 - accuracy:
0.6056 - val_loss: 1.1053 - val_accuracy: 0.7875
Epoch 157/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2767 - accuracy:
0.6000 - val_loss: 1.0990 - val_accuracy: 0.7875
Epoch 158/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2673 - accuracy:
0.6229 - val_loss: 1.0927 - val_accuracy: 0.7875
Epoch 159/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2698 - accuracy:
0.6069 - val_loss: 1.0859 - val_accuracy: 0.7937
Epoch 160/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2498 - accuracy:
0.6396 - val_loss: 1.0787 - val_accuracy: 0.8000
Epoch 161/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2627 - accuracy:
0.6243 - val_loss: 1.0721 - val_accuracy: 0.8000
Epoch 162/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2423 - accuracy:
0.6153 - val_loss: 1.0661 - val_accuracy: 0.8062
Epoch 163/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2354 - accuracy:
0.6257 - val_loss: 1.0597 - val_accuracy: 0.8062
Epoch 164/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2387 - accuracy:
0.6132 - val_loss: 1.0536 - val_accuracy: 0.8125
Epoch 165/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2082 - accuracy:
0.6389 - val_loss: 1.0471 - val_accuracy: 0.8125
Epoch 166/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2220 - accuracy:
0.6250 - val_loss: 1.0402 - val_accuracy: 0.8125
Epoch 167/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2128 - accuracy:
0.6403 - val_loss: 1.0335 - val_accuracy: 0.8125
Epoch 168/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2266 - accuracy:
0.6160 - val_loss: 1.0271 - val_accuracy: 0.8188
Epoch 169/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2116 - accuracy:
0.6181 - val_loss: 1.0218 - val_accuracy: 0.8188
Epoch 170/200
12/12 [==============================] - 0s 2ms/step - loss: 1.2054 - accuracy:
0.6139 - val_loss: 1.0169 - val_accuracy: 0.8250
Epoch 171/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1829 - accuracy:
0.6340 - val_loss: 1.0104 - val_accuracy: 0.8250
Epoch 172/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 1.1956 - accuracy:
0.6139 - val_loss: 1.0047 - val_accuracy: 0.8250
Epoch 173/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1971 - accuracy:
0.6194 - val_loss: 0.9994 - val_accuracy: 0.8188
Epoch 174/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1629 - accuracy:
0.6701 - val_loss: 0.9937 - val_accuracy: 0.8250
Epoch 175/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1829 - accuracy:
0.6278 - val_loss: 0.9878 - val_accuracy: 0.8250
Epoch 176/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1470 - accuracy:
0.6569 - val_loss: 0.9815 - val_accuracy: 0.8188
Epoch 177/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1637 - accuracy:
0.6431 - val_loss: 0.9754 - val_accuracy: 0.8250
Epoch 178/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1502 - accuracy:
0.6382 - val_loss: 0.9700 - val_accuracy: 0.8250
Epoch 179/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1440 - accuracy:
0.6583 - val_loss: 0.9644 - val_accuracy: 0.8250
Epoch 180/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1502 - accuracy:
0.6368 - val_loss: 0.9584 - val_accuracy: 0.8375
Epoch 181/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1423 - accuracy:
0.6556 - val_loss: 0.9531 - val_accuracy: 0.8375
Epoch 182/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1267 - accuracy:
0.6646 - val_loss: 0.9473 - val_accuracy: 0.8375
Epoch 183/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1330 - accuracy:
0.6375 - val_loss: 0.9413 - val_accuracy: 0.8375
Epoch 184/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1306 - accuracy:
0.6340 - val_loss: 0.9361 - val_accuracy: 0.8375
Epoch 185/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1203 - accuracy:
0.6569 - val_loss: 0.9313 - val_accuracy: 0.8438
Epoch 186/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1188 - accuracy:
0.6479 - val_loss: 0.9260 - val_accuracy: 0.8438
Epoch 187/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1066 - accuracy:
0.6653 - val_loss: 0.9206 - val_accuracy: 0.8438
Epoch 188/200
```

```
12/12 [==============================] - 0s 2ms/step - loss: 1.1177 - accuracy:
0.6750 - val_loss: 0.9158 - val_accuracy: 0.8438
Epoch 189/200
12/12 [==============================] - 0s 2ms/step - loss: 1.1210 - accuracy:
0.6528 - val_loss: 0.9111 - val_accuracy: 0.8500
Epoch 190/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0902 - accuracy:
0.6806 - val_loss: 0.9074 - val_accuracy: 0.8500
Epoch 191/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0937 - accuracy:
0.6701 - val_loss: 0.9031 - val_accuracy: 0.8375
Epoch 192/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0810 - accuracy:
0.6736 - val_loss: 0.8987 - val_accuracy: 0.8375
Epoch 193/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0912 - accuracy:
0.6750 - val_loss: 0.8933 - val_accuracy: 0.8438
Epoch 194/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0605 - accuracy:
0.7000 - val_loss: 0.8879 - val_accuracy: 0.8438
Epoch 195/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0567 - accuracy:
0.6715 - val_loss: 0.8829 - val_accuracy: 0.8500
Epoch 196/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0785 - accuracy:
0.6583 - val_loss: 0.8766 - val_accuracy: 0.8562
Epoch 197/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0540 - accuracy:
0.6868 - val_loss: 0.8710 - val_accuracy: 0.8562
Epoch 198/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0498 - accuracy:
0.6819 - val_loss: 0.8669 - val_accuracy: 0.8500
Epoch 199/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0525 - accuracy:
0.6826 - val_loss: 0.8638 - val_accuracy: 0.8562
Epoch 200/200
12/12 [==============================] - 0s 2ms/step - loss: 1.0414 - accuracy:
0.6896 - val_loss: 0.8598 - val_accuracy: 0.8562
13/13 [==============================] - 0s 561us/step - loss: 0.8204 -
accuracy: 0.8775
Precisión: 87.75
```

[ ]:

[16]:
```python
def create_model():
    model = Sequential()
    model.add(Dense(128, input_dim=X_train.shape[1], activation='relu'))
```

```
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.5))
    model.add(Dense(32, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.5))
    model.add(Dense(4, activation='softmax'))
    optimizer = Adam(learning_rate=0.0001)
    model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizer,␣
 ↪metrics=['accuracy'])
    return model

data = pd.read_csv("train.csv")
scaler = StandardScaler()
X = scaler.fit_transform(data.drop("price_range", axis=1))
y = data["price_range"]

# Crear el modelo
model = KerasClassifier(build_fn=create_model, epochs=200, batch_size=128,␣
 ↪verbose=0)

# Validación cruzada
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
results = cross_val_score(model, X, y, cv=kfold)

print("Precisión promedio: %.2f%%" % (results.mean()*100))
```

```
/tmp/ipykernel_122845/1950703141.py:20: DeprecationWarning: KerasClassifier is
deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See
https://www.adriangb.com/scikeras/stable/migration.html for help migrating.
  model = KerasClassifier(build_fn=create_model, epochs=200, batch_size=128,
verbose=0)
```

```
Precisión promedio: 85.50%
```

[17]:
```
def create_model():
    model = Sequential()
    model.add(Dense(256, input_dim=X_train.shape[1], activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.5))
    model.add(Dense(32, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.5))
    model.add(Dense(16, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.5))
    model.add(Dense(4, activation='softmax'))
```

```
    optimizer = Adam(learning_rate=0.0005)
    model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizer,␣
 ↪metrics=['accuracy'])
    return model

data = pd.read_csv("train.csv")
scaler = StandardScaler()
X = scaler.fit_transform(data.drop("price_range", axis=1))
y = data["price_range"]

# Validación cruzada
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Crear el modelo
model = KerasClassifier(build_fn=create_model, epochs=200, batch_size=128,␣
 ↪verbose=0)

results = cross_val_score(model, X, y, cv=kfold)

print("Precisión promedio: %.2f%%" % (results.mean()*100))
```

```
/tmp/ipykernel_122845/4171429106.py:27: DeprecationWarning: KerasClassifier is
deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See
https://www.adriangb.com/scikeras/stable/migration.html for help migrating.
  model = KerasClassifier(build_fn=create_model, epochs=200, batch_size=128,
verbose=0)
```

```
Precisión promedio: 93.40%
```

[19]:
```
def create_model():
    model = Sequential()
    model.add(Dense(256, input_dim=X_train.shape[1], activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(32, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(16, activation='relu', kernel_regularizer=l2(0.01)))
```

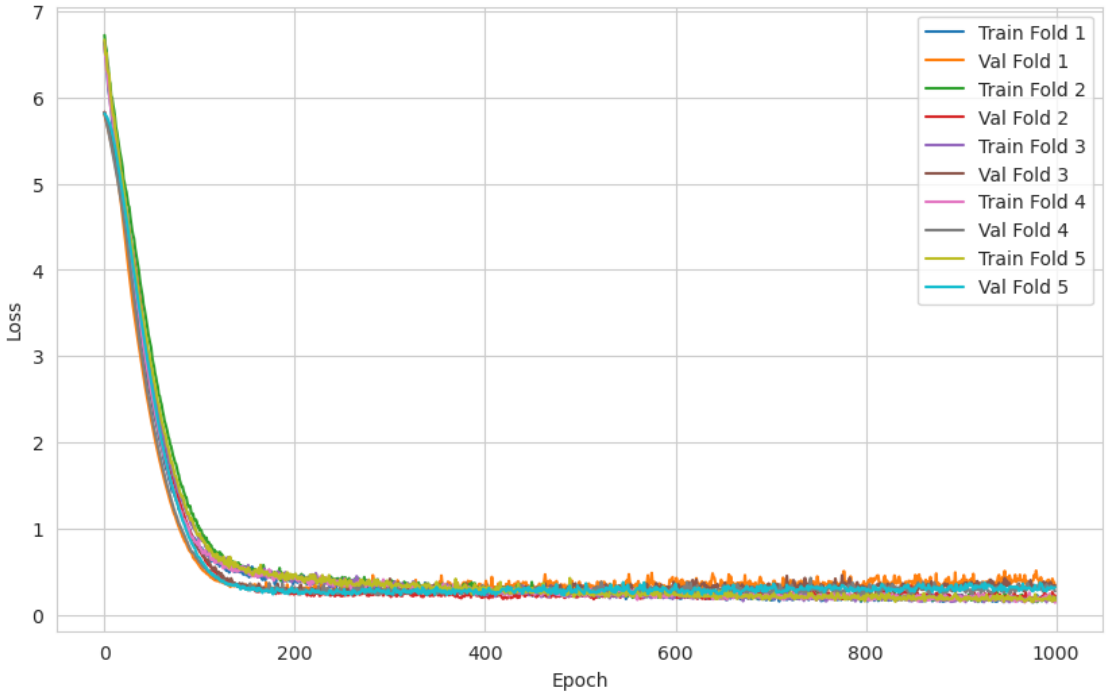
```
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(4, activation='softmax'))
    optimizer = Adam(learning_rate=0.0005)
    model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizer,␣
 ↪metrics=['accuracy'])
    return model

data = pd.read_csv("train.csv")
scaler = StandardScaler()
X = scaler.fit_transform(data.drop("price_range", axis=1))
y = data["price_range"]

# Validación cruzada
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Crear el modelo
model = KerasClassifier(build_fn=create_model, epochs=1000, batch_size=64,␣
 ↪verbose=0)

results = cross_val_score(model, X, y, cv=kfold)

print("Precisión promedio: %.2f%%" % (results.mean()*100))
```

```
/tmp/ipykernel_122845/4181710883.py:35: DeprecationWarning: KerasClassifier is
deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See
https://www.adriangb.com/scikeras/stable/migration.html for help migrating.
  model = KerasClassifier(build_fn=create_model, epochs=1000, batch_size=64,
verbose=0)
```

```
Precisión promedio: 94.45%
```

[20]:
```
def create_model():
    model = Sequential()
    model.add(Dense(256, input_dim=X_train.shape[1], activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(32, activation='relu', kernel_regularizer=l2(0.01)))
```

```
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(16, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(4, activation='softmax'))
    optimizer = Adam(learning_rate=0.0005)
    model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizer,␣
 ↪metrics=['accuracy'])

    # Return the history object for the trained model
    history = model.fit(X_train, y_train, validation_data=(X_val, y_val),␣
 ↪epochs=1000, batch_size=64, verbose=0)
    return history

histories = []
for train_idx, val_idx in kfold.split(X, y):
    X_train, y_train = X[train_idx], y[train_idx]
    X_val, y_val = X[val_idx], y[val_idx]
    history = create_model()
    histories.append(history)

# Plot the training and validation loss for each epoch across all folds
plt.figure(figsize=(10, 6))
for i, history in enumerate(histories):
    plt.plot(history.history['loss'], label='Train Fold {}'.format(i+1))
    plt.plot(history.history['val_loss'], label='Val Fold {}'.format(i+1))
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```


Train Fold 1
Val Fold 1
Train Fold 2
Val Fold 2
Train Fold 3
Val Fold 3
Train Fold 4
Val Fold 4
Train Fold 5
Val Fold 5
Loss
Epoch
0
1
2
3
4
5
6
7
0
200
400
600
800
1000