Universidad de Córdoba

Escuela Politécnica Superior de Córdoba

Grado de Ingeniería Informática - Mención en computación

Tercer curso - Segundo cuatrimestre - 2020/2021

Introducción al Aprendizaje Automático

Práctica 3: Clasificación y evaluación de los modelos

Profesor: *Nicolás Emilio García Pedrajas*

Autor: *Ventura Lucena Martínez*

Córdoba, 1 de julio de 2021

Índice

Listings

Índice de figuras

1 Ejercicio 1

Obtenga al menos 10 conjuntos de datos en formato CSV, ARFF u otro cualquiera de:

- **Weka datasets.**
- **UCI MLR.**

Los conjuntos de datos seleccionados son los siguientes[1]:

- contact-lenses.arff.
- cpu.arff.
- cpu.with.vendor.arff.
- diabetes.arff.
- glass.arff.
- ionosphere.arff.
- iris.arff.
- labor.arff.
- soybean.arff.
- weather.arff.

2 Ejercicio 2

Seleccione al menos 3 clasificadores dentro de los disponibles en Scikit. Se recomienda elegir tres de entre los siguientes: árboles de decisión, *k* vecinos más cercanos, máquinas de vectores soporte, regresión logística, redes neuronales y clasificador Naïve de Bayes.

Los clasificadores seleccionados son los siguientes:

- Árboles de decisión.
- K vecinos más cercanos.
- Máquinas de vectores soporte.

[1] Los ficheros se encuentran adjuntos en la entrega

3 Ejercicio 3

Para cada uno de los problemas seleccionados realice las siguientes tareas:

1. **Seleccione como método para obtener el error la validación cruzada de 10 particiones o el método *hold out*.**
2. **Ejecute para cada clasificador seleccionado el entrenamiento y anote el error.**
3. **Represente gráficamente el error obtenido con cada uno de los métodos de clasificación.**
4. **Para los clasificadores que admiten parámetros modifique los más relevantes y estudie cómo afectan al error de clasificación.**
5. **Pruebe alguna de las técnicas de preprocesado de datos estudiadas y estudie cómo afectan al error de clasificación.**

Se van a realizar 3 ejecuciones sobre cada conjunto de datos, mostrando como ejemplo concreto el conjunto de datos *cpu.arff*. Para ello, se ha implementado la siguiente funcionalidad:

Listing 1: **Funcionalidades de los clasificadores seleccionados.**

```
def decisionTrees(x_train, x_test, y_train, y_test, file_name):
    clf = DecisionTreeClassifier(
        class_weight='balanced').fit(x_train, y_train)
    y_predict = clf.predict(x_test)

    print("\nDecision trees", file_name, " - Mean absolute error:",
          metrics.mean_absolute_error(y_test, y_predict))

    pd.DataFrame(y_test, y_predict).boxplot()
    plt.show()


def SVM(x_train, x_test, y_train, y_test, file_name):
    clf = make_pipeline(StandardScaler(), SVC(
        gamma='auto', kernel="linear")).fit(x_train, y_train)
    y_predict = clf.predict(x_test)

    print("SVM", file_name, " - Mean absolute error:",
          metrics.mean_absolute_error(y_test, y_predict))

    pd.DataFrame(y_test, y_predict).boxplot()
    plt.show()


def k_nearest_neighbor(x_train, x_test, y_train, y_test, file_name):
    clf = KNeighborsClassifier(
        n_neighbors=6, weights="distance").fit(x_train, y_train)
    y_predict = clf.predict(x_test)
```

```
    print("K nearest neighbor", file_name, " - Mean absolute error:",
          metrics.mean_absolute_error(y_test, y_predict))

    pd.DataFrame(y_test, y_predict).boxplot()
    plt.show()
```

Los datos obtenidos para cada una de las tres ejecuciones son los siguientes:

Listing 2: **Ejercicio 3 - Ejecución 1.**

```
Decision trees .\datasets\cpu.arff  - Mean absolute error: 29.317460317460316
SVM .\datasets\cpu.arff  - Mean absolute error: 38.142857142857146
                                                              _5.py
K nearest neighbor .\datasets\cpu.arff  - Mean absolute error: 41.57142857142857
```

Listing 3: **Ejercicio 3 - Ejecución 2.**

```
Decision trees .\datasets\cpu.arff  - Mean absolute error: 27.634920634920636
SVM .\datasets\cpu.arff  - Mean absolute error: 38.142857142857146
K nearest neighbor .\datasets\cpu.arff  - Mean absolute error: 41.57142857142857
```

Listing 4: **Ejercicio 3 - Ejecución 3.**

```
Decision trees .\datasets\cpu.arff  - Mean absolute error: 28.841269841269842
SVM .\datasets\cpu.arff  - Mean absolute error: 38.142857142857146
K nearest neighbor .\datasets\cpu.arff  - Mean absolute error: 41.57142857142857
```

En cuanto al cambio de parámetros se refiere, se han realizado las siguientes modificaciones, obteniendo los siguientes resultados:

- **Árboles de decisión:** el parámetro *class_weight* se ha modificado a *balanced*. Su valor por defecto era *none*.
- **SVM:** el parámetro *kernel* se ha modificado a *linear*. Su valor anterior era *rbf*.
- **K nearest neighbor:** el parámetro *weights* se ha modificado a *distance*. Su valor por defecto era *uniform*.

Listing 5: **Ejercicio 3 - Ejecución 1 (Parámetros modificados).**

```
Decision trees .\datasets\cpu.arff  - Mean absolute error: 29.761904761904763
SVM .\datasets\cpu.arff  - Mean absolute error: 36.23809523809524
K nearest neighbor .\datasets\cpu.arff  - Mean absolute error: 30.38095238095238
```

Listing 6: **Ejercicio 3 - Ejecución 2 (Parámetros modificados).**

```
Decision trees .\datasets\cpu.arff  - Mean absolute error: 29.746031746031747
SVM .\datasets\cpu.arff  - Mean absolute error: 36.23809523809524
K nearest neighbor .\datasets\cpu.arff  - Mean absolute error: 30.38095238095238
```

Listing 7: **Ejercicio 3 - Ejecución 3 (Parámetros modificados).**

```
Decision trees .\datasets\cpu.arff  - Mean absolute error: 28.6984126984127
SVM .\datasets\cpu.arff  - Mean absolute error: 36.23809523809524
K nearest neighbor .\datasets\cpu.arff  - Mean absolute error: 30.38095238095238
```

Por último, la representación gráfica de las medias de error absoluto quedarían de la siguiente forma:

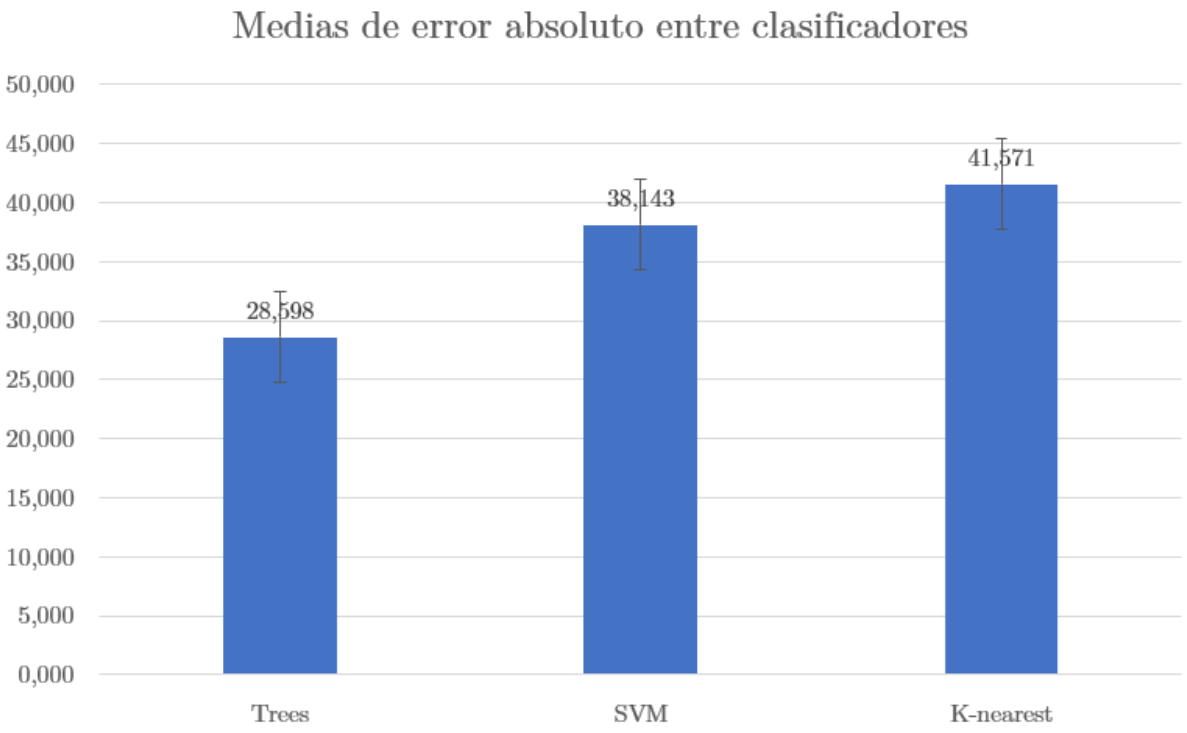

Figura 1: Gráfico con parámetros por defecto.

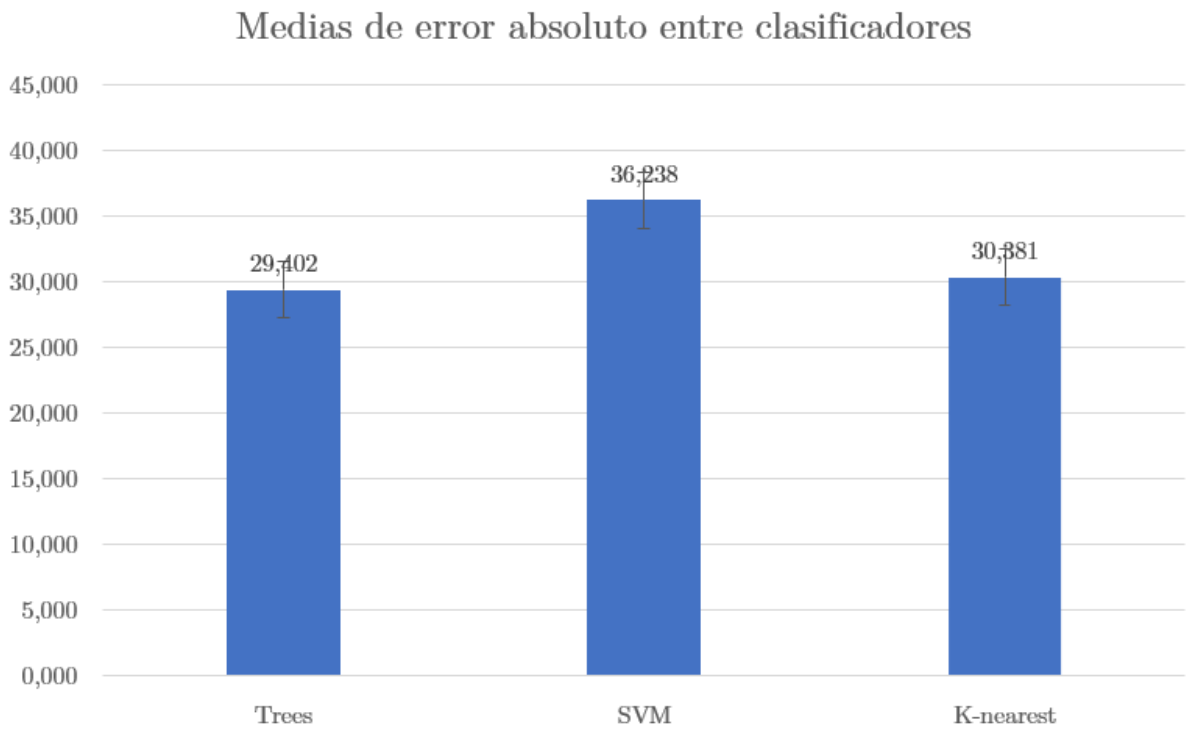

Figura 2: Gráfico con parámetros modificados.

4 Ejercicio 4

Use el test de Wilcoxon de comparación de dos algoritmos sobre *N* problemas y aplíquelo a dos de los algoritmos anteriores. Obtenga el rango de Friedman para cada clasificador y configuración y represente gráficamente los resultados. Aplique el test de Iman-Davenport sobre los tres clasificadores.

Los resultados obtenidos son los siguientes:

Listing 8: **Resultados obtenidos en los tests.**

```
Decision Trees/SVM: WilcoxonResult(statistic=746.0, pvalue=0.8405760802247746)
Decision Trees/KNN: WilcoxonResult(statistic=255.0, pvalue=2.6890504623513096e
    -05)
SVM/KNN: WilcoxonResult(statistic=226.5, pvalue=2.514192987692583e-05)

FriedmanchisquareResult(statistic=29.33031674208139, pvalue=4.275654884091536e
    -07)
```

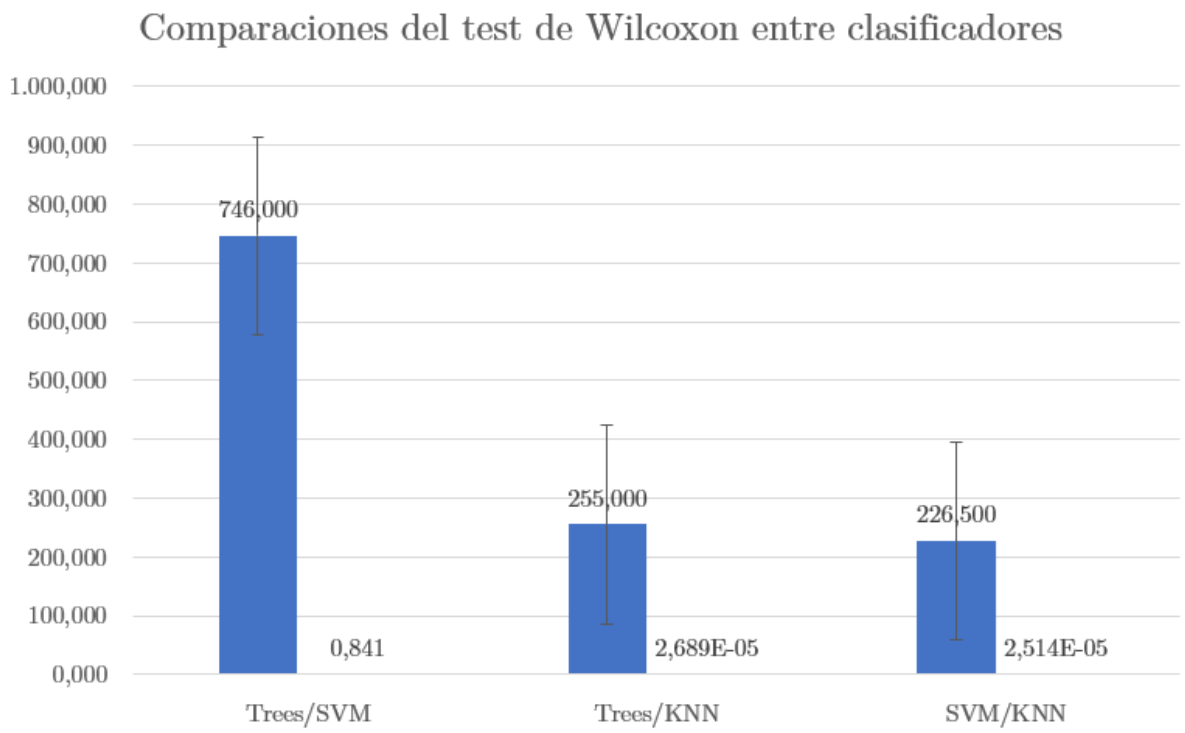

Figura 3: Comparaciones del test de Wilcoxon entre clasificadores.

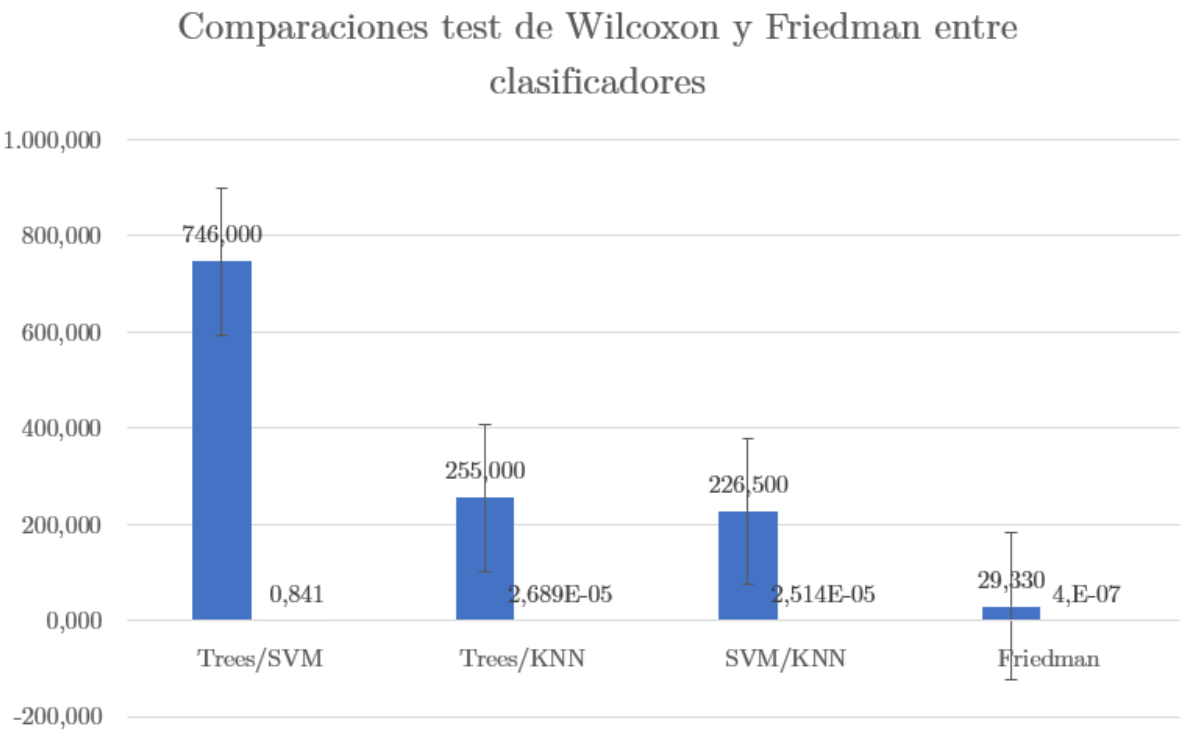

Figura 4: Comparaciones entre test de Wilcoxon y test de Friedman.

5 Ejercicio 5

Realice una gráfica *box plot* de los errores de cada método para comparar gráficamente su rendimiento.

Las gráficas obtenidas para los resultados del ejercicio 3 son las siguientes:

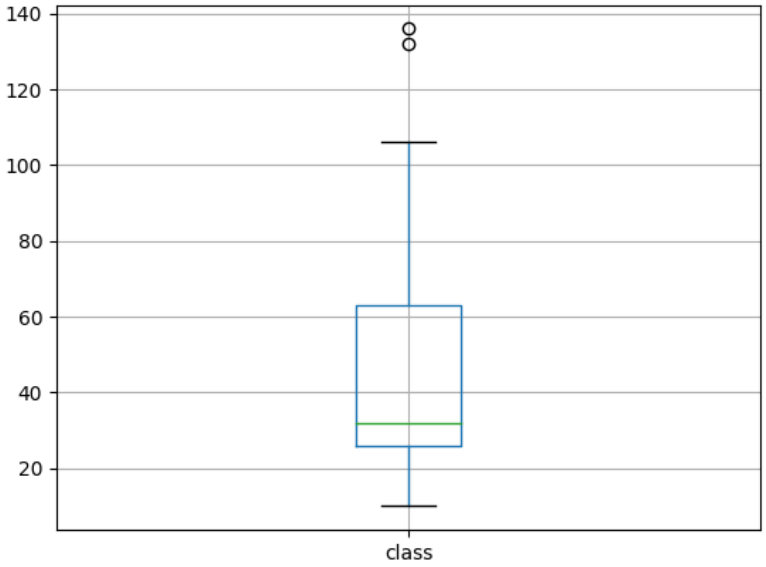

Figura 5: Boxplot - Árboles de decisión.

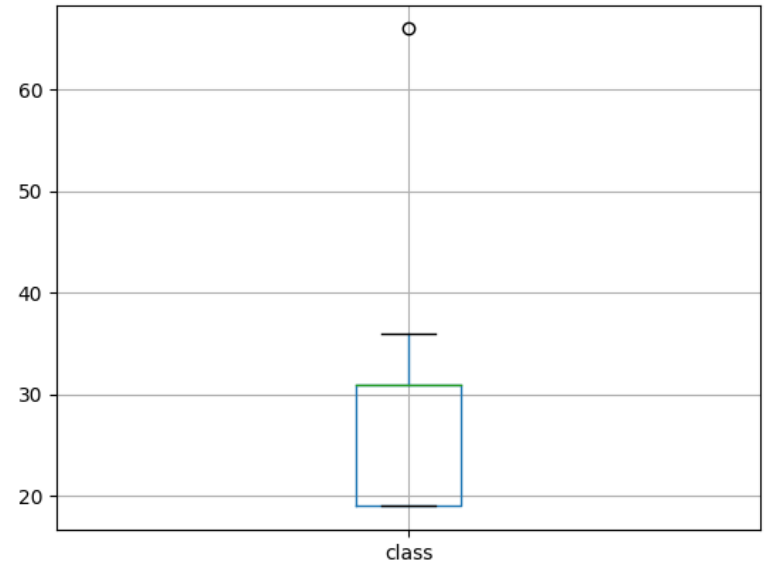

Figura 6: Boxplot - SVM.

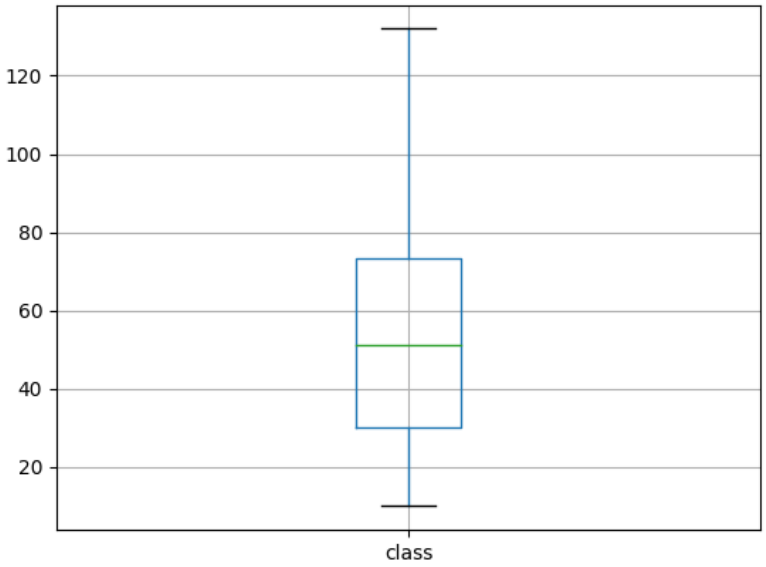

Figura 7: Boxplot - K vecinos más cercanos.

6 Ejercicio 6

Para uno de los clasificadores elegidos utilice una validación de los hiperparámetros con *grid search* y compare su rendimiento con el método con hiperparámetros fijados a priori.

Para la realización del ejercicio se ha tomado como ejemplo una muestra de internet en la que se utiliza el dataset *load_digits* de la librería *Scikit-learn*. Los datos obtenidos han sido los siguientes:

Listing 9: **Resultados ejercicio 6.**

```
# Tuning hyper-parameters for precision

Best parameters set found on development set:

{'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}

Grid scores on development set:

0.986 (+/-0.016) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.959 (+/-0.028) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
```

```
0.988 (+/-0.017) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.982 (+/-0.026) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.988 (+/-0.017) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.983 (+/-0.026) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.988 (+/-0.017) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.983 (+/-0.026) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
0.974 (+/-0.012) for {'C': 1, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 10, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 100, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 1000, 'kernel': 'linear'}

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        89
           1       0.97      1.00      0.98        90
           2       0.99      0.98      0.98        92
           3       1.00      0.99      0.99        93
           4       1.00      1.00      1.00        76
           5       0.99      0.98      0.99       108
           6       0.99      1.00      0.99        89
           7       0.99      1.00      0.99        78
           8       1.00      0.98      0.99        92
           9       0.99      0.99      0.99        92

    accuracy                           0.99       899
   macro avg       0.99      0.99      0.99       899
weighted avg       0.99      0.99      0.99       899


Accuracy Score : 0.9911012235817576
Precision Score : 0.991212690463989
Recall Score : 0.9911012235817576
F1 Score : 0.9911031179023475

[[ 89   0   0   0   0   0   0   0   0   0]
 [  0  90   0   0   0   0   0   0   0   0]
 [  0   1  90   0   0   0   0   1   0   0]
 [  0   0   1  92   0   0   0   0   0   0]
 [  0   0   0   0  76   0   0   0   0   0]
 [  0   0   0   0   0 106   1   0   0   1]
 [  0   0   0   0   0   0  89   0   0   0]
 [  0   0   0   0   0   0   0  78   0   0]
 [  0   2   0   0   0   0   0   0  90   0]
 [  0   0   0   0   0   1   0   0   0  91]]


# Tuning hyper-parameters for recall
```

```
Best parameters set found on development set:

{'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}

Grid scores on development set:

0.986 (+/-0.019) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.957 (+/-0.028) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.987 (+/-0.019) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.981 (+/-0.028) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.987 (+/-0.019) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.982 (+/-0.026) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.987 (+/-0.019) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.982 (+/-0.026) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
0.971 (+/-0.010) for {'C': 1, 'kernel': 'linear'}
0.971 (+/-0.010) for {'C': 10, 'kernel': 'linear'}
0.971 (+/-0.010) for {'C': 100, 'kernel': 'linear'}
0.971 (+/-0.010) for {'C': 1000, 'kernel': 'linear'}

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        89
           1       0.97      1.00      0.98        90
           2       0.99      0.98      0.98        92
           3       1.00      0.99      0.99        93
           4       1.00      1.00      1.00        76
           5       0.99      0.98      0.99       108
           6       0.99      1.00      0.99        89
           7       0.99      1.00      0.99        78
           8       1.00      0.98      0.99        92
           9       0.99      0.99      0.99        92

    accuracy                           0.99       899
   macro avg       0.99      0.99      0.99       899
weighted avg       0.99      0.99      0.99       899


Accuracy Score : 0.9911012235817576
Precision Score : 0.991212690463989
Recall Score : 0.9911012235817576
F1 Score : 0.9911031179023475

[[ 89   0   0   0   0   0   0   0   0   0]
 [  0  90   0   0   0   0   0   0   0   0]
 [  0   1  90   0   0   0   0   1   0   0]
 [  0   0   1  92   0   0   0   0   0   0]
 [  0   0   0   0  76   0   0   0   0   0]
 [  0   0   0   0   0 106   1   0   0   1]
```

```
[  0   0   0   0   0   0  89   0   0   0]
[  0   0   0   0   0   0   0  78   0   0]
[  0   2   0   0   0   0   0   0  90   0]
[  0   0   0   0   0   1   0   0   0  91]]
```

Referencias

[1] Moodle Universidad de Córdoba - Enunciado práctica 3.

[2] Moodle Universidad de Córdoba - Introducción a Scikit-learn.