Reporte Weka y Jupyter Notebook | Inteligencia Artificial

Marco Ferraro | B82957

Grabiel Revillat | B86524

Steven Nuñez | B95614

**Enunciado**

1. Vea el video de SVM usando Weka que se agregó.

2. Durante el tiempo de clase ( o antes si lo desean) ustedes llevarán a cabo en grupo ( con su equipo de trabajo ) lo siguiente:

a) Seleccionarán un archivo de datos .csv creado para la Tarea de Naive Bayes por alguno de los tres integrantes del grupo.

b) Siguiendo la información del video, habilitarán LibSVM en Weka para crear un modelo con la colección de Trainning y Evaluarán sobre la de Testing que eligieron.

c) Comparen los resultados de: Naive Bayes contra SVM ( kernel lineal, radial y poli), ajusten los hiperparámetros vistos en la presentación de la clase del martes antes de evaluar sobre el data set de testing. Reporten tanto la calidad del modelo como los resultados de evaluación sobre el testing.

3. Usando Jupiter Notebook carguen los datos de training y usando el cuaderno visto el martes, modifíquenlo para evaluar el data set de testing con los mismos hiperparámetros que usaron con Weka.

4. Deben hacer una síntesis y un análisis de los resultados obtenidos en general.

**Bayes y Máquinas de Soporte Vectorial en Weka**

Para este trabajo se utiliza el set de datos de películas y su país de origen. El *target variable* es USA o Germany.

Para la primera parte de esta tarea, volvimos a probar el modelo de Bayes en los datos de prueba y entrenamiento.

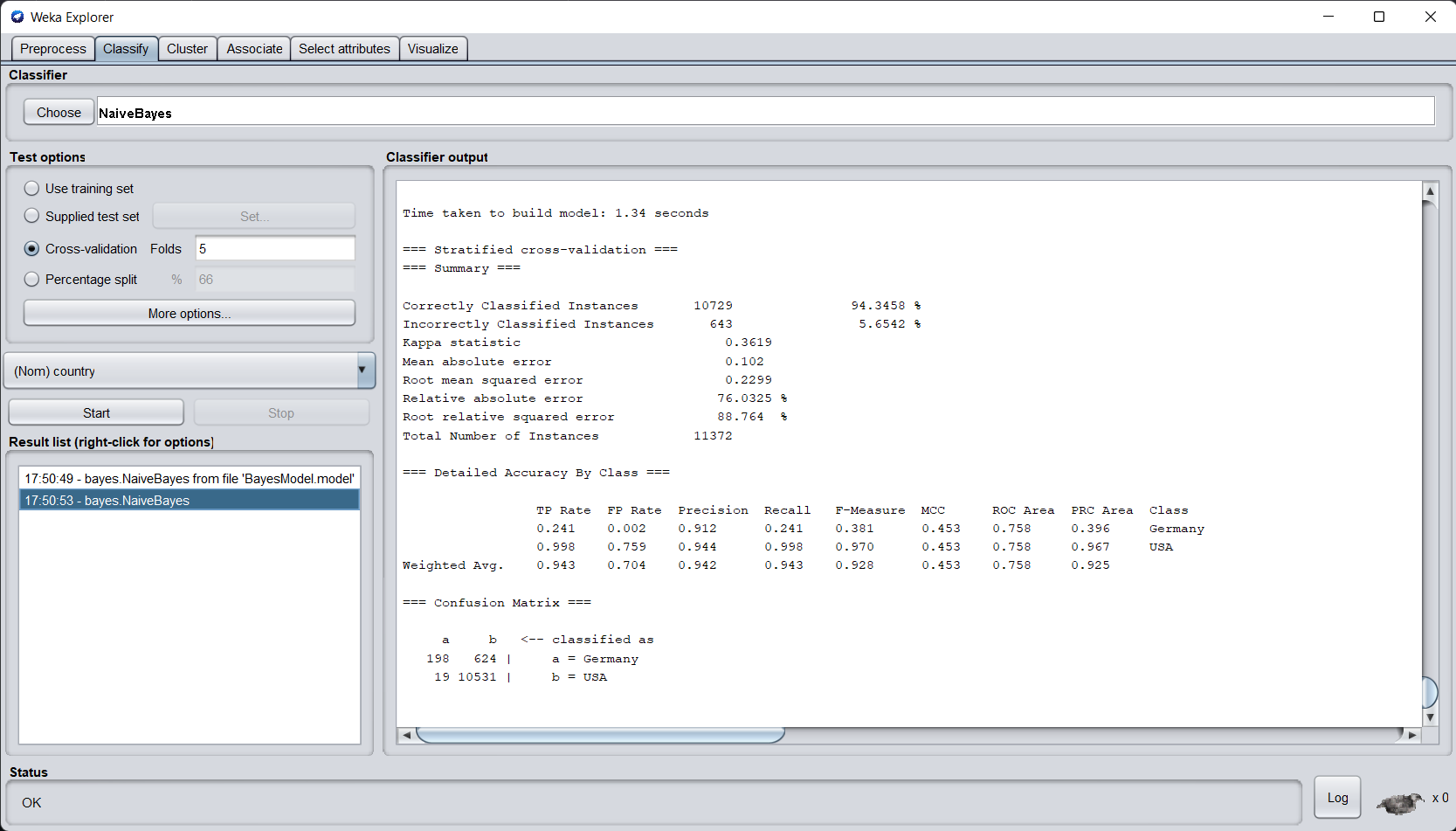


Ilustración 1 Resultados Training con 5 Cross Folds Bayes

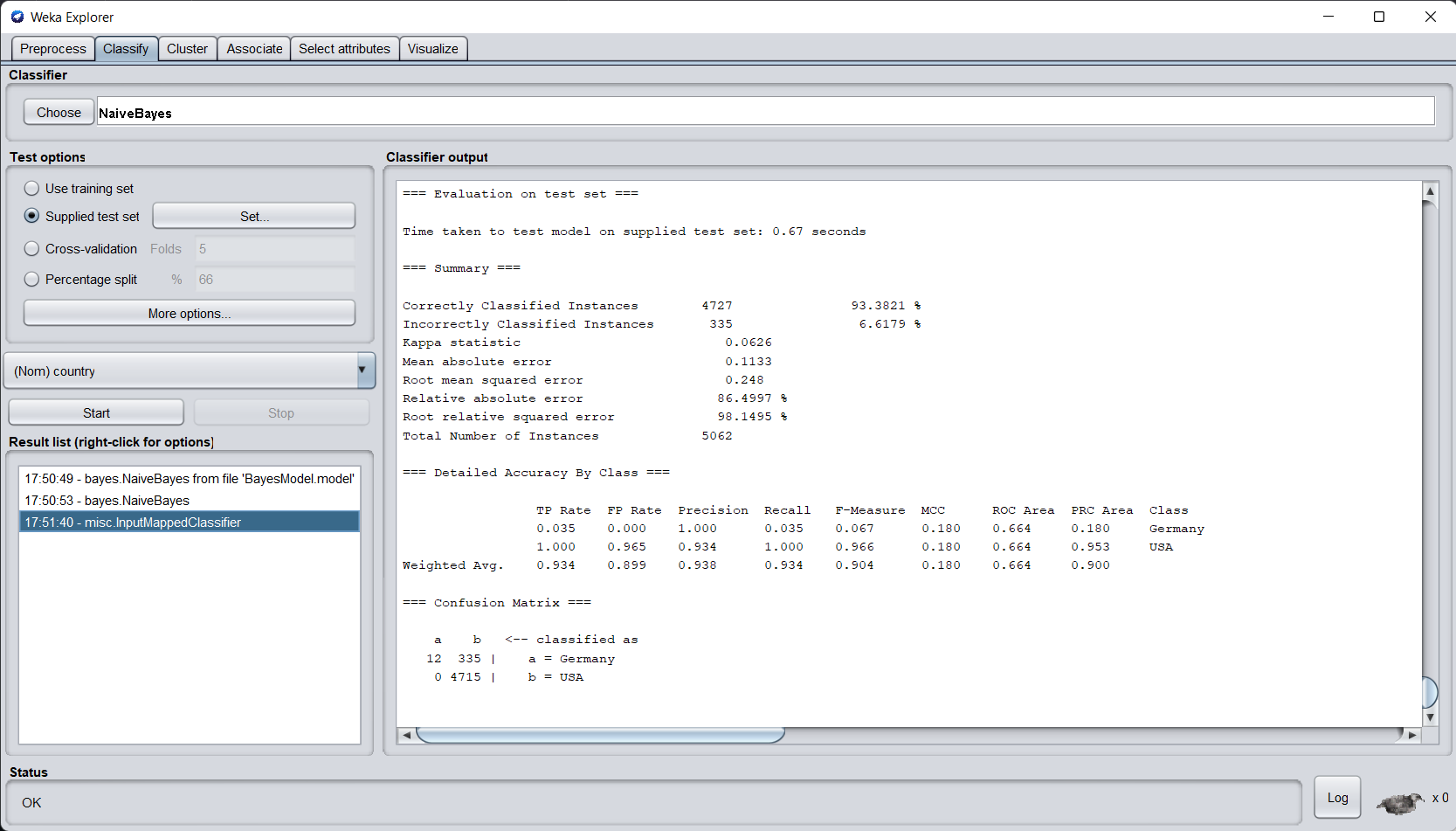


Ilustración 2 Resultados Testing con el modelo de Bayes de Ilustración 1

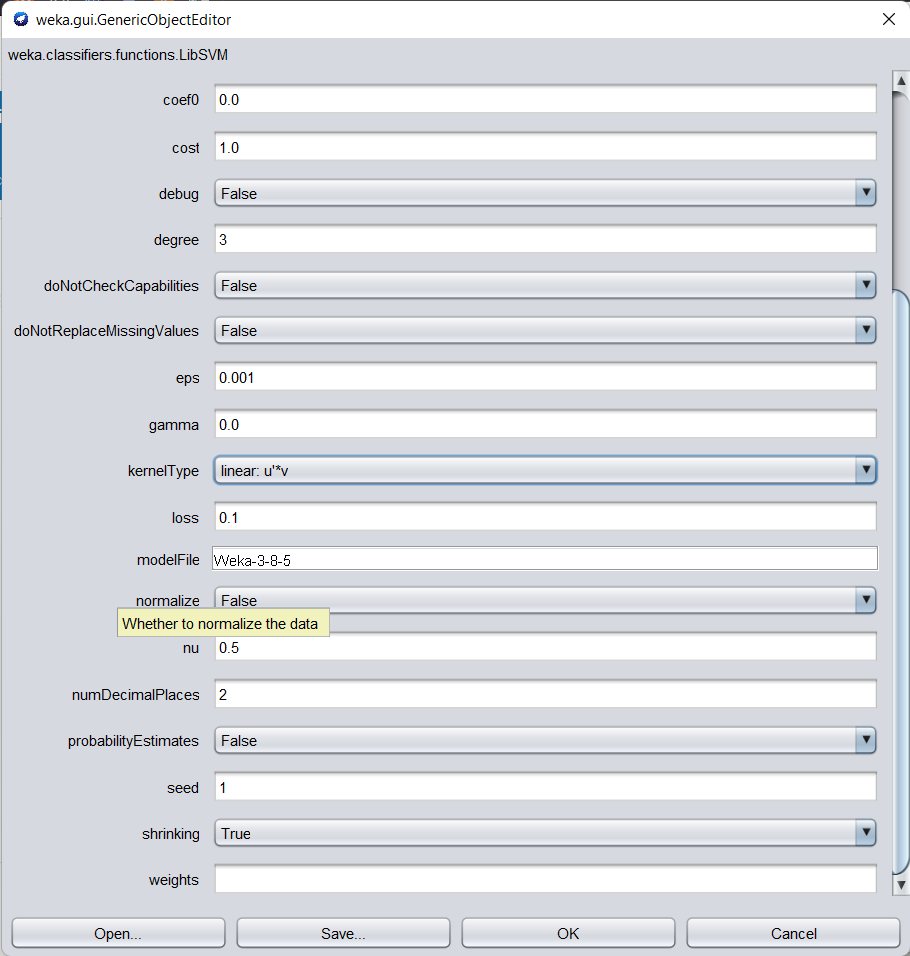


Ilustración 3 Definición SVM con kernel lineal

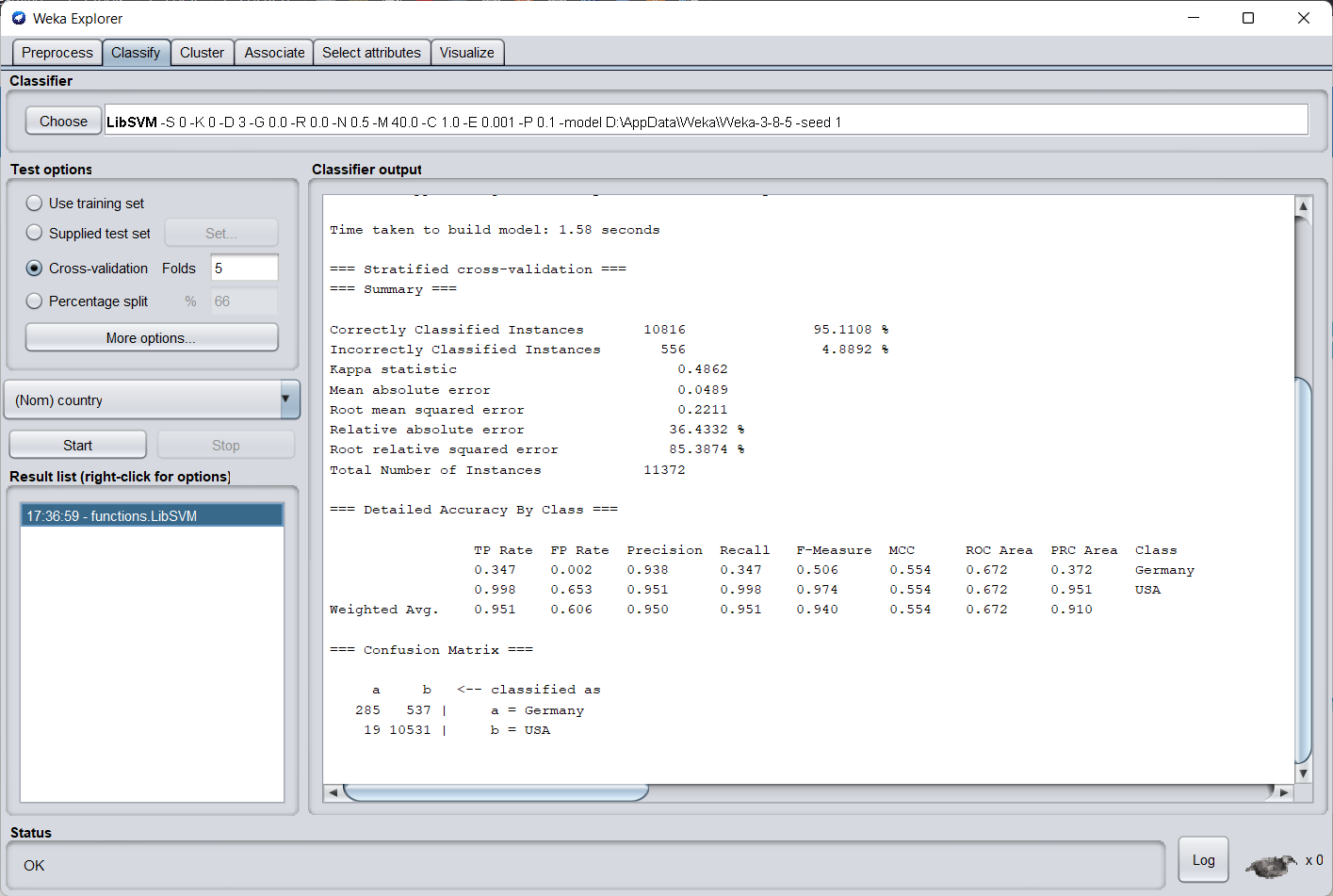


Ilustración 4 Resultados Training con 5 Cross Folds kernel lineal

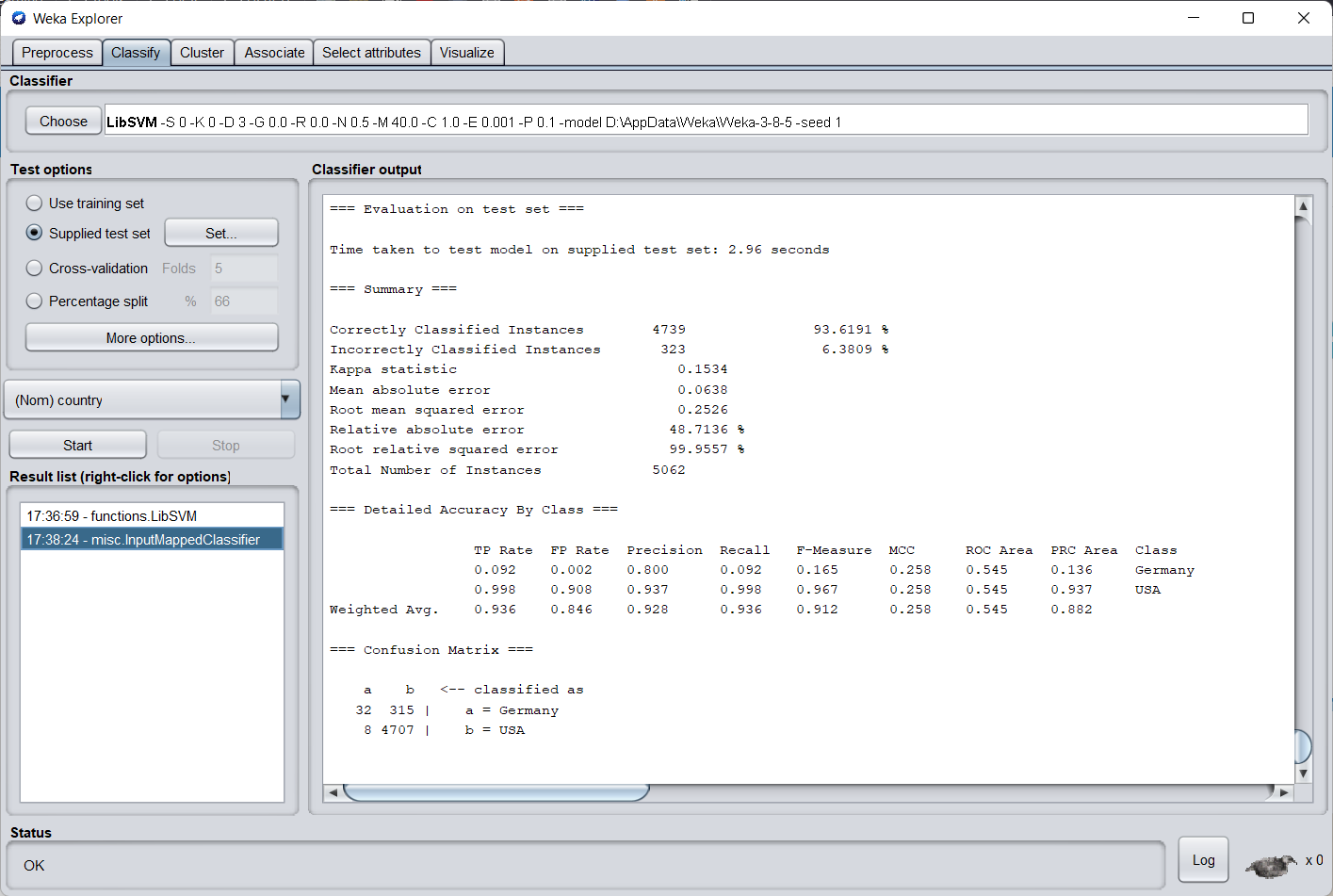


Ilustración 5 Resultados Testing kernel lineal

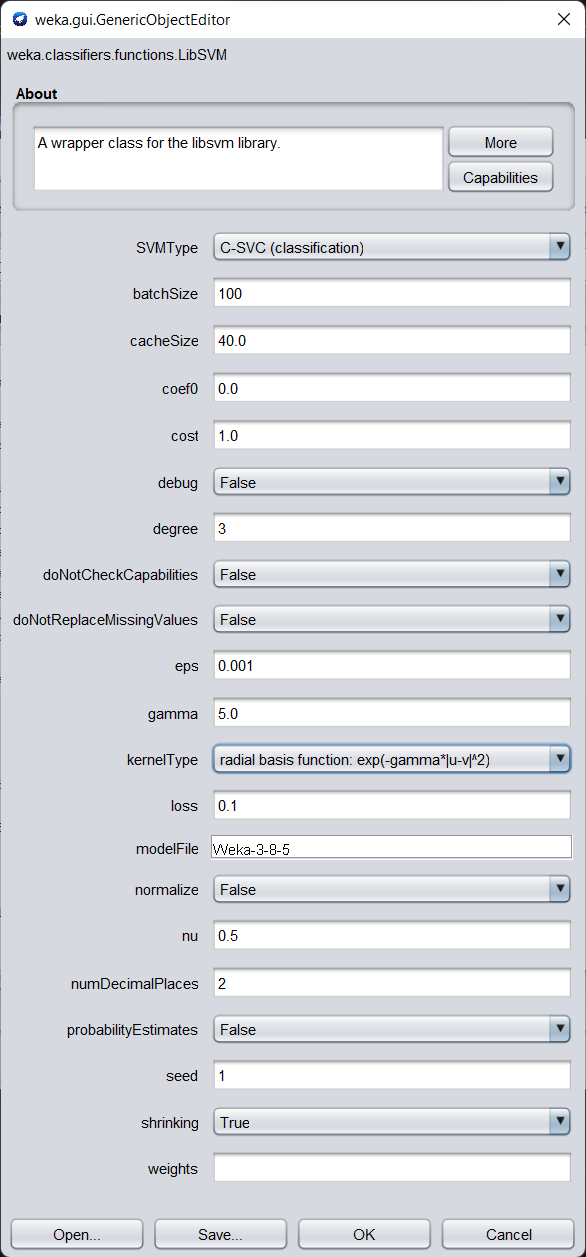


Ilustración 6 Definición SVM con kernel radial y gamma 5

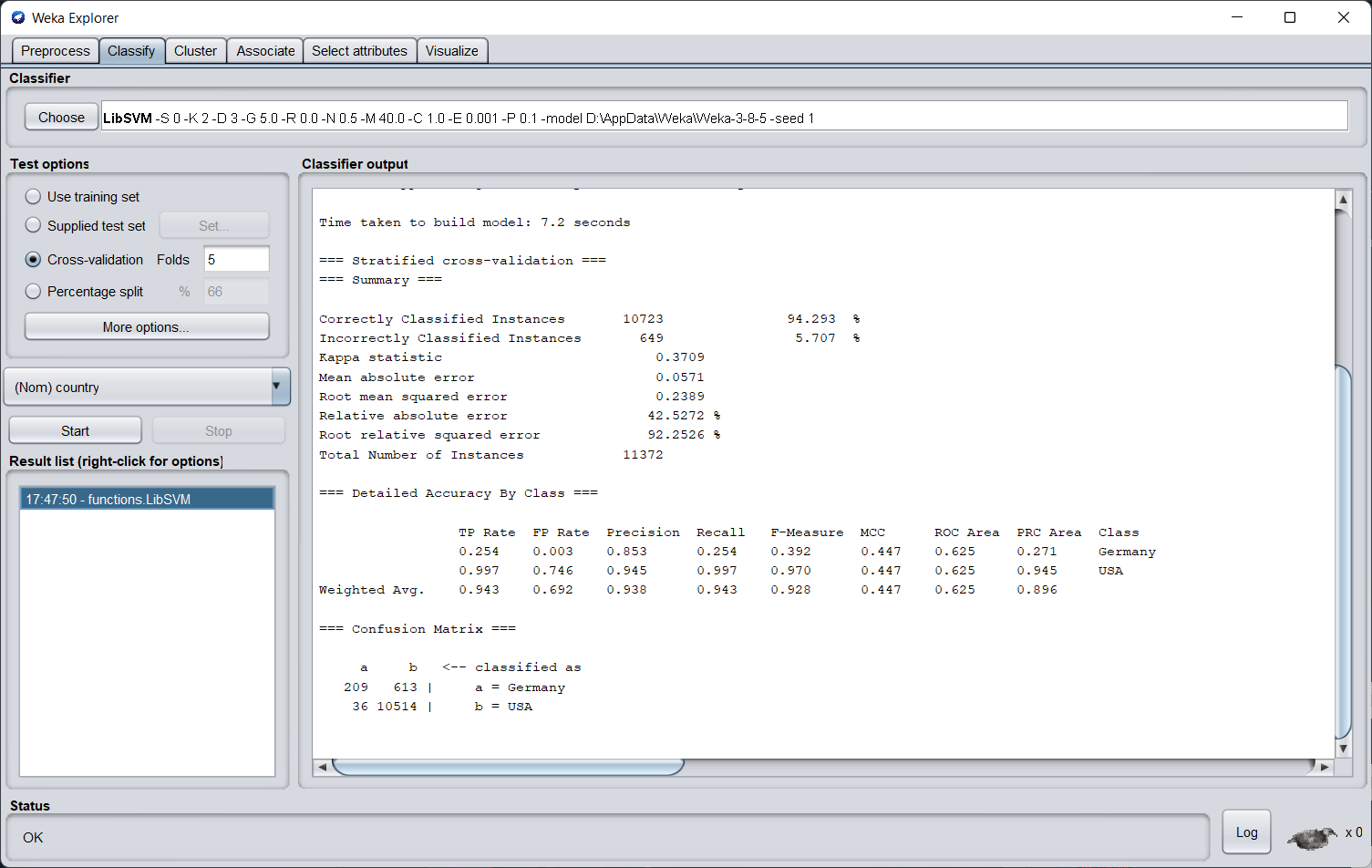


Ilustración 7 Resultados Training con 5 Cross Folds kernel radial

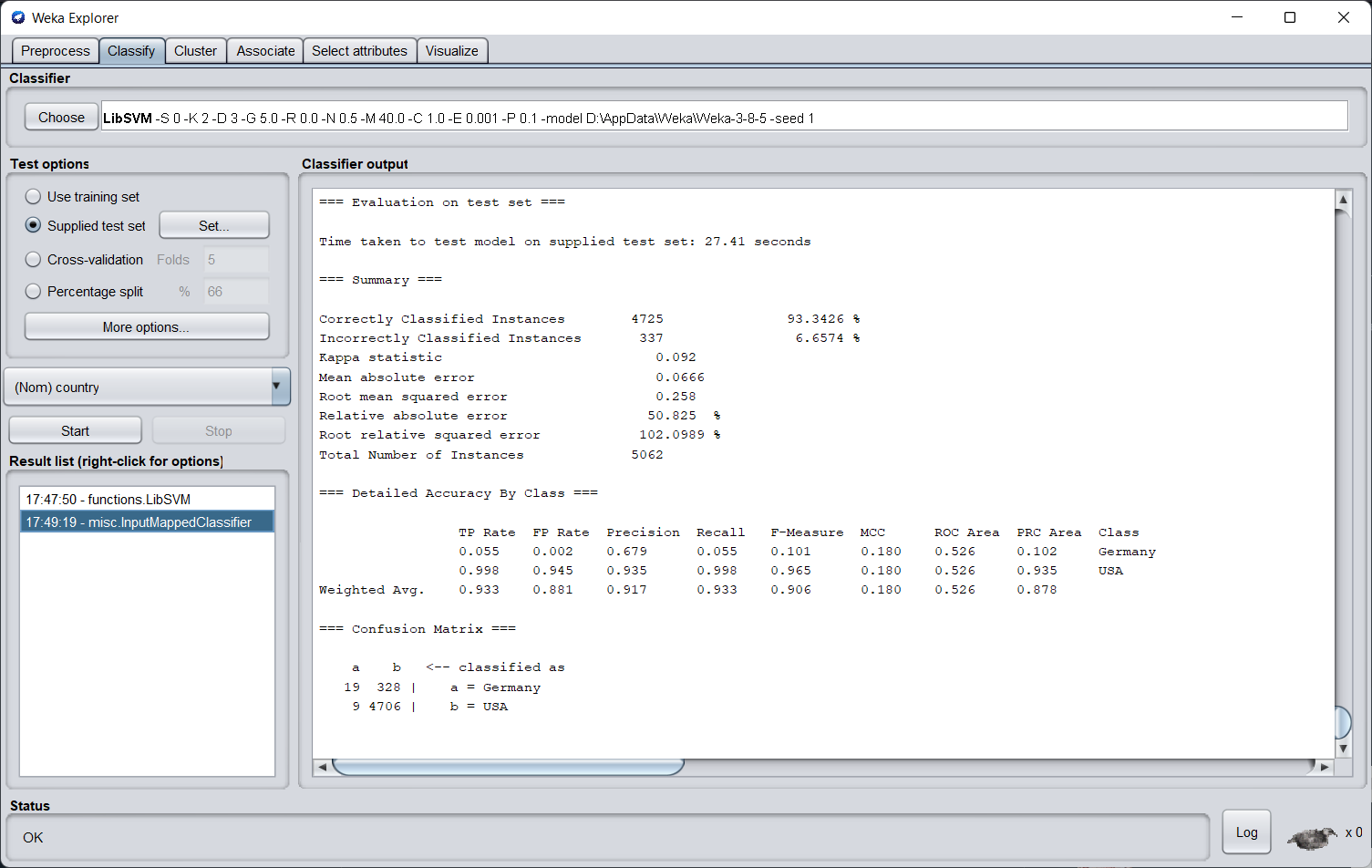


Ilustración 8 Resultados Testing con kernel radial

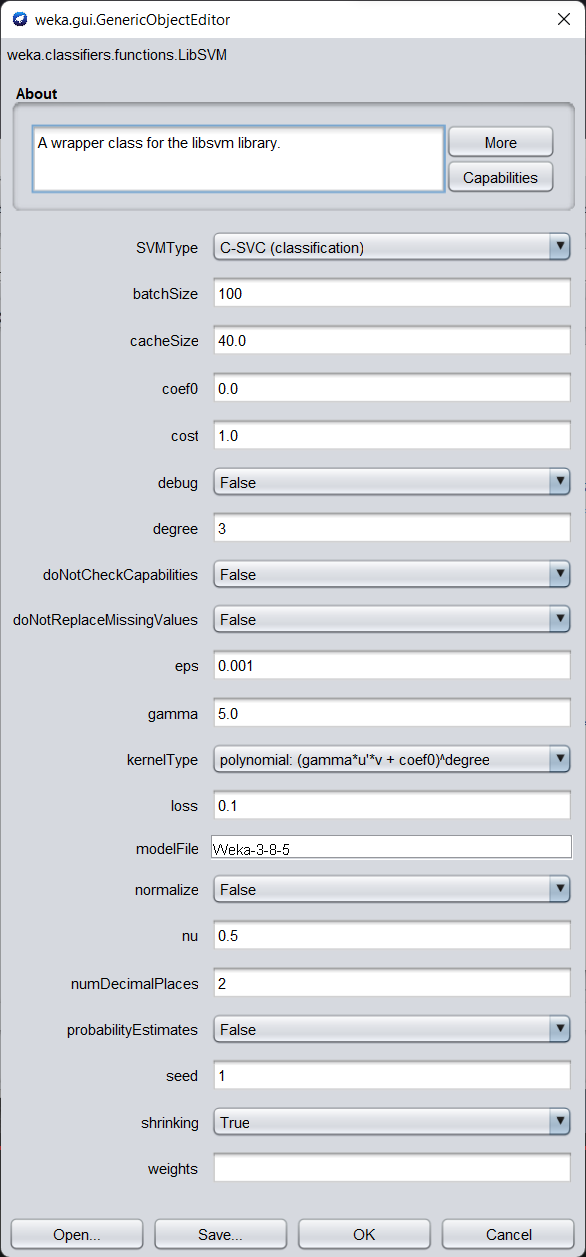


Ilustración 9 Definicion SVM con kernel polinomial y gamma 5

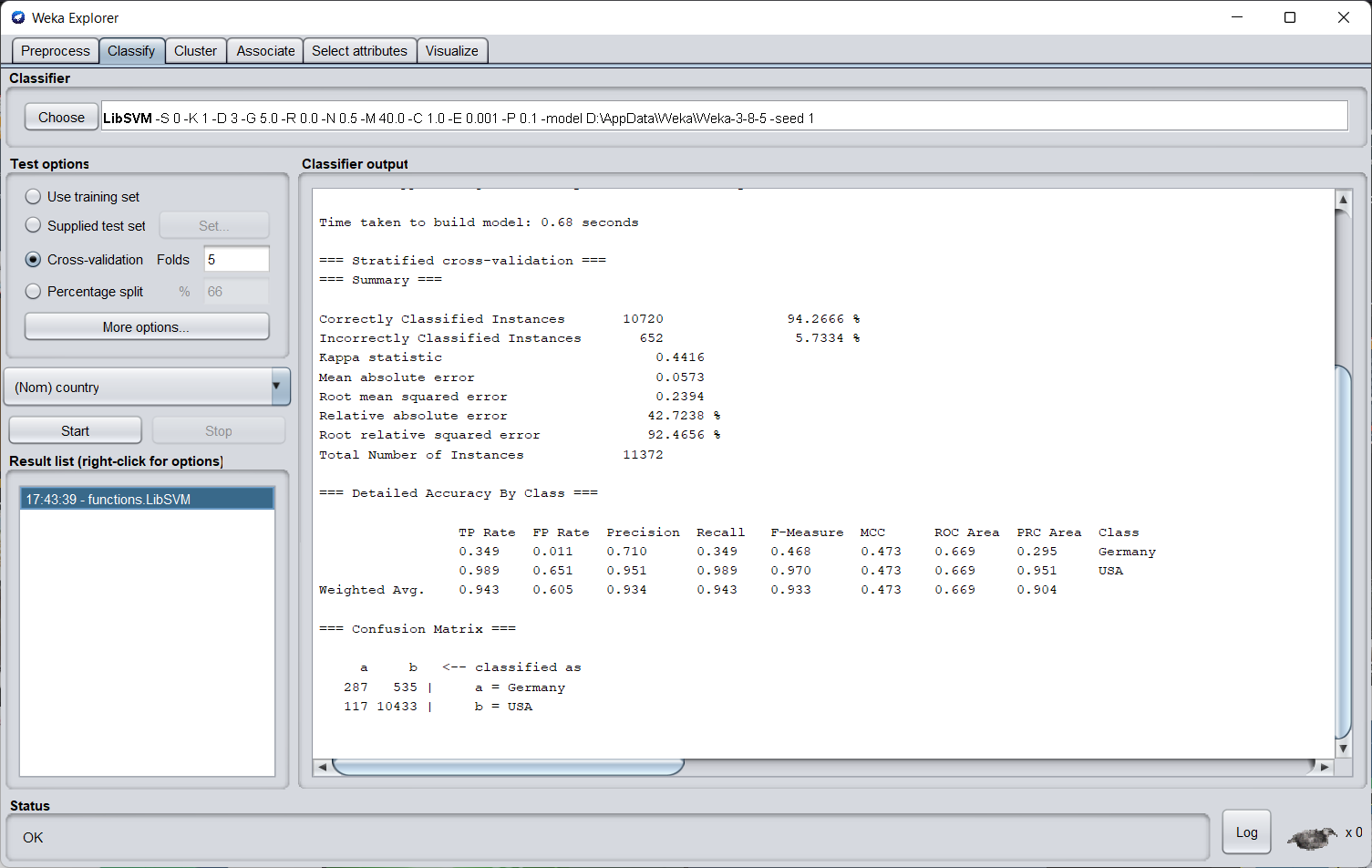


Ilustración 10 Resultados training con kernel polinomial

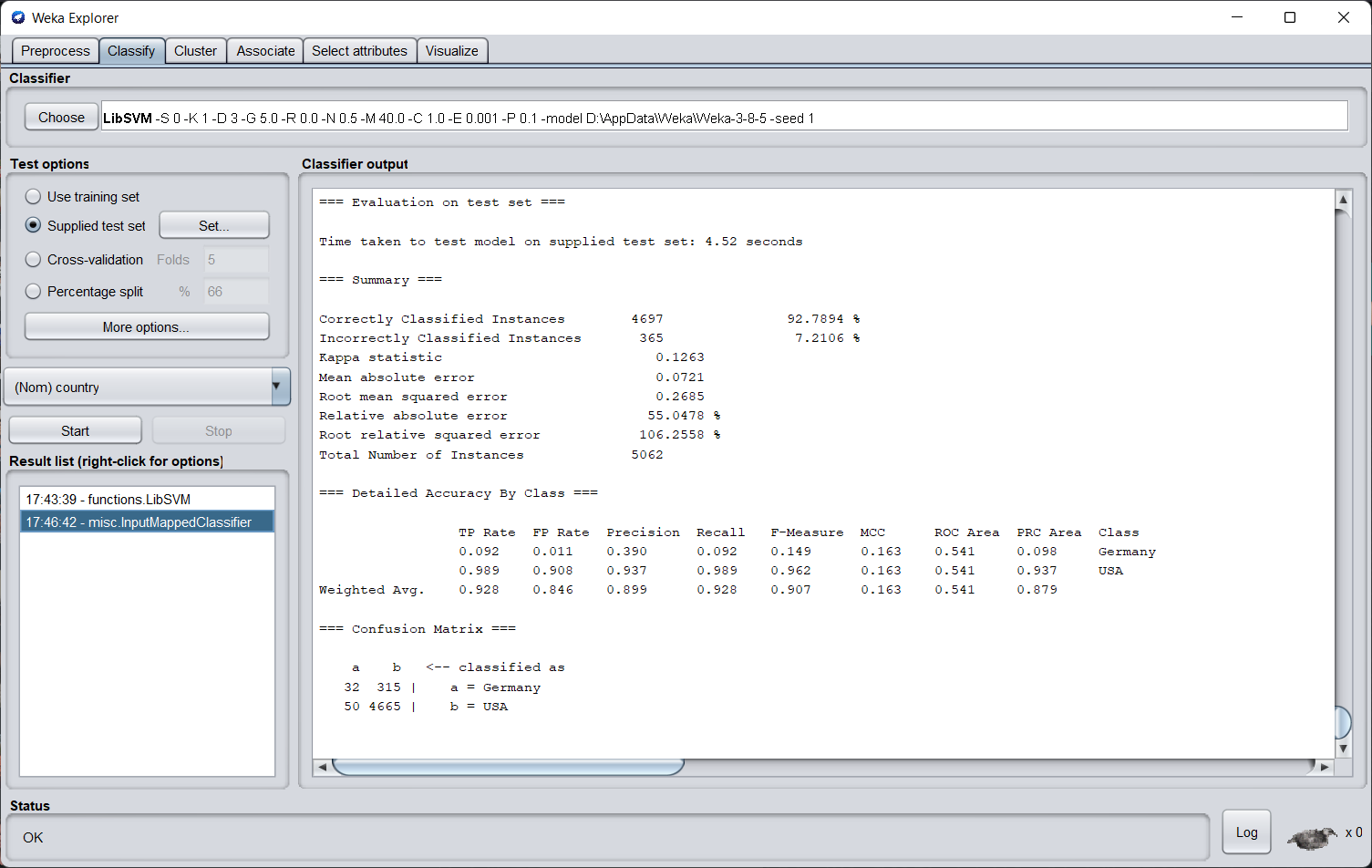


Ilustración 11 Resultados Testing con kernel polinomial

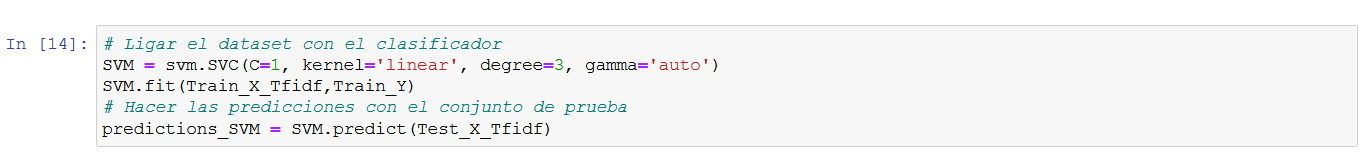


Ilustración 12 Definicion kernel lineal en Python

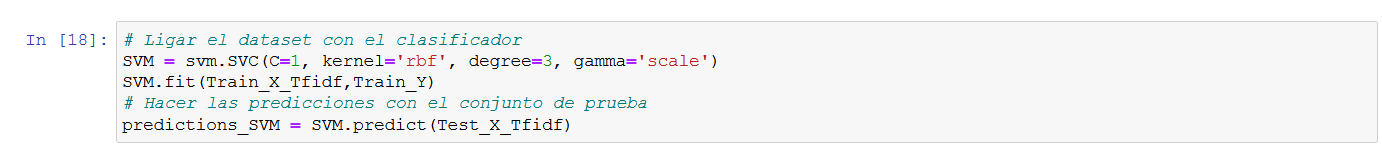


Ilustración 13 Definicion kernal radial en Python

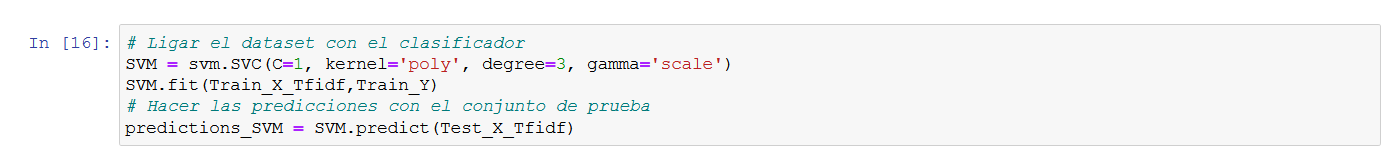


Ilustración 14 Definicion kernel polinomial en Python

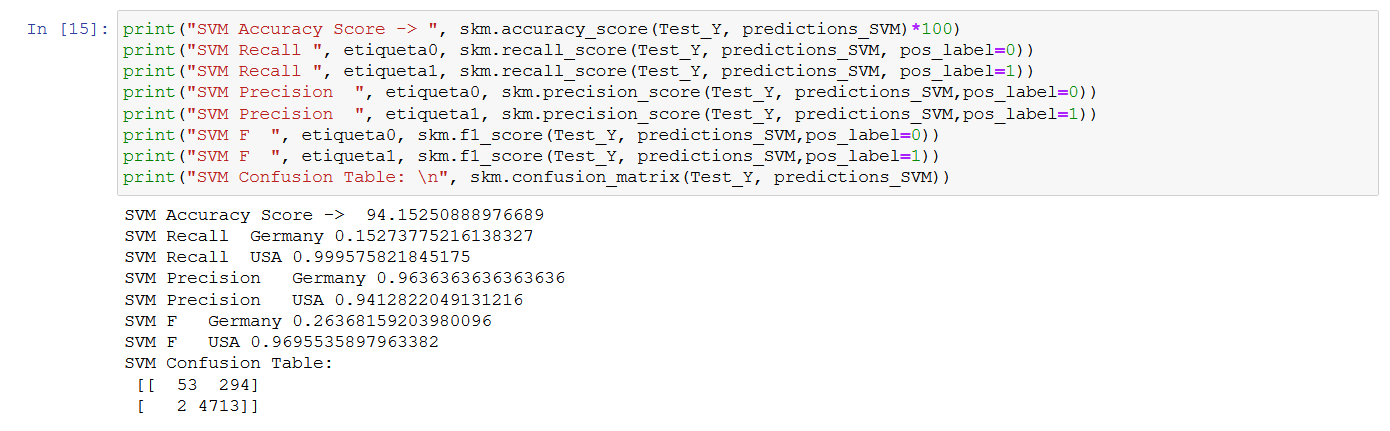


Ilustración 15 Resultados kernel lineal en Python

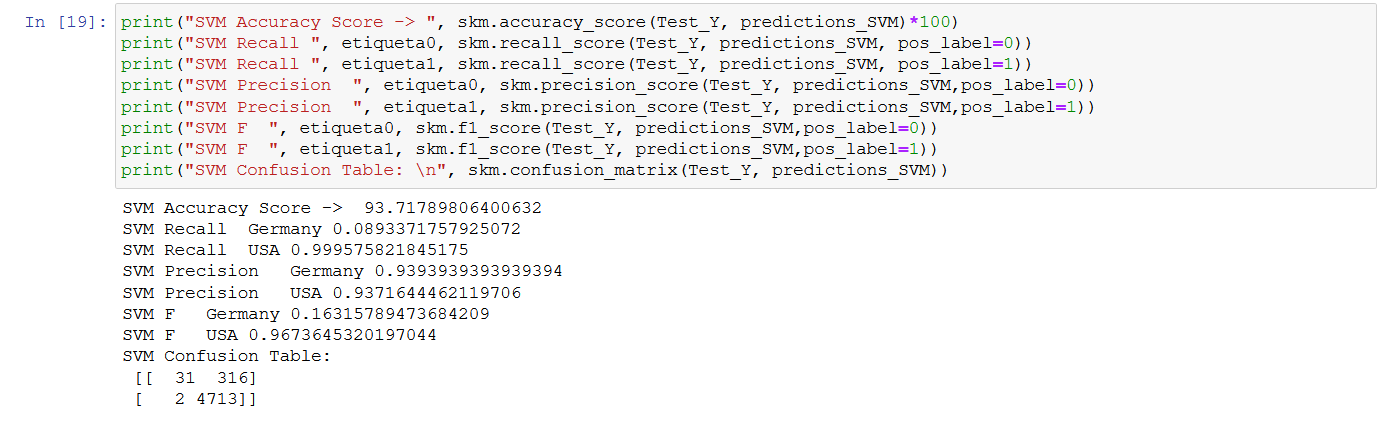


Ilustración 16 Resultados kernel radial en Python

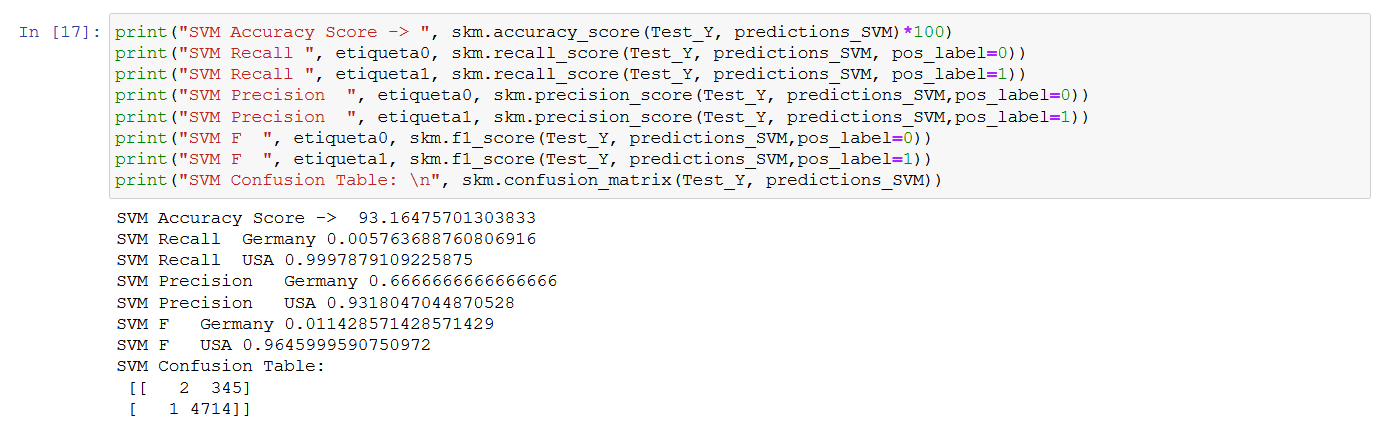


Ilustración 17 Resultados kernel polinomial en Python

**Conclusiones**

Para analizar los resultados hay que tener claro que las maquinas de Soporte Vectorial en Jupyter y en Weka tienen que analizarse sin relacionar los resultados entre estas mismas, ya que, al ser Máquinas de Soporte Vectorial diferentes representan una variable de bloqueo.

**Análisis Weka**

En los rendimientos del modelo, entre las Maquinas de Soporte Vectorial vemos que la que tiene mejor rendimiento es la de kernel lineal, ya que tiene más precisión para predecir las *X* que pertenecen a **Germany.** Dicho esto, hay algo peculiar; por alguna razon, el modelo de Naive Bayes tiene precisión del 100 en el *target variable* **Germany.** Podemos decir que es por la naturaleza de los datos, pero es algo que llama poderosamente la atención, y mas porque acierta todos los *True Negative*.

**Análisis Jupyter**

Para trabajar esta parte, se modificó el código de Python para respetar la separación de los datos de prueba y de entrenamiento. No se separan con los métodos de *sci-kit learn*, si no que se importan como csv aparte (el notebook viene adjunto con esta tarea). Dentro de los resultados que genera *sci-kit learn* vemos que el kernel lineal tiene un mejor resultado, pero marginalmente. De conclusión, para analizar datos con 1 *X* y *1 target variable* con maquinas de soporte vectorial recomendamos utilizar el kerneal líneal, ya que no es tan complejo y parece que su umbral de decisión es más preciso que los otros kernel.