UNC Universidad Nacional de Córdoba
FAMAF Facultad de Matemática, Astronomía, Física y Computación

Support Vector Machine en Acción

Angaramo Piñol, Facundo Nicolás

14 de Noviembre de 2023

Resumen

En este estudio, aplicamos el algoritmo de Support Vector Machine (SVM) al conjunto de datos Breast Cancer Wisconsin para resolver un problema de clasificación binaria. Evaluamos el rendimiento del modelo utilizando diferentes kernels (lineal, polinomial y RBF) para identificar el más adecuado. Los resultados indican que SVM ofrece una alta precisión y un alto recall en la separación de las clases, independientemente del kernel empleado.

1. Introducción

En este trabajo, exploramos la aplicación del algoritmo de Support Vector Machine (SVM) en el contexto de la clasificación binaria, utilizando el conjunto de datos Breast Cancer Wisconsin.

2. Un poco de SVM

En términos simples, Support Vector Machine es un tipo de algoritmo de aprendizaje supervisado que se utiliza para clasificación o regresión. La idea principal detrás de SVM es encontrar un hiperplano optimo en el espacio de características que mejor separe las diferentes clases en los datos.

Conceptos Claves:

- **Hiperplano:** En un espacio unidimensional, un hiperplano es un punto que divide una recta en dos segmentos. En un espacio bidimensional, un hiperplano es una recta que divide el plano en dos mitades. En un espacio tridimensional, un hiperplano es un plano que divide el espacio tridimensional en dos partes. Este concepto también puede ser aplicado a espacios de cuatro dimensiones y más, donde estos objetos divisores se llaman simplemente hiperplanos. En SVM, buscamos el hiperplano que maximiza la distancia entre las clases.
- **Vectores de Soporte:** Son los puntos de datos más cercanos al hiperplano. Estos puntos son cruciales para hallar el hiperplano óptimo, ya que determinan la posición y orientación del hiperplano.
- **Margen:** El margen es la distancia entre el hiperplano y los vectores de soporte. SVM busca maximizar este margen, lo que ayuda a mejorar la generalización del modelo.

- **Kernel Trick:** Es una técnica que permite trabajar con conjuntos de datos que no son linealmente separables en su espacio original al proyectarlos a un espacio de mayor dimensión donde la separación lineal es posible, sin tener que realizar explícitamente esta proyección.

En resumen, SVM trata de encontrar el mejor hiperplano que divide las clases en los datos de manera óptima, maximizando el margen y utilizando vectores de soporte [Figura 1]. Esto lo hace muy efectivo, especialmente en casos donde los datos no son linealmente separables, gracias al kernel trick que permite trabajar en espacios de características más complejos.

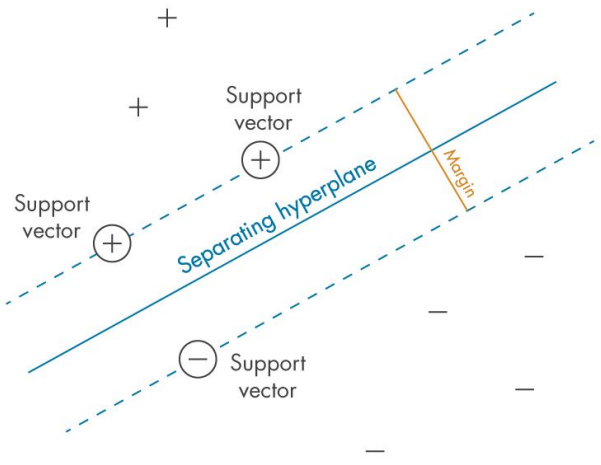


Figura 1: Support Vector Machine

3. Conjunto de Datos: Breast Cancer Wisconsin

Como ya mencionamos, utilizamos el conjunto de datos Breast Cancer Wisconsin para explorar el algoritmo Support Vector Machine (SVM) en el contexto de la clasificación binaria. Este conjunto de datos, ampliamente utilizado en la comunidad de aprendizaje automático, se centra en características derivadas de imágenes digitalizadas de biopsias de mama y tiene como objetivo la clasificación de tumores en malignos o benignos.

3.1. Descripción del Dataset

Número de Instancias

- 569

Features

- **radius** (media de las distancias desde el centro a los puntos del perímetro)
- **texture** (desviación estándar de los valores en escala de grises)
- **perimeter**
- **area**
- **smoothness** (variación local de las longitudes del radio)

- **compactness** ($\frac{\text{perimeter}^2}{\text{área}-1.0}$)
- **concavity** (gravedad de las partes cóncavas del contorno)
- **concave points** (número de porciones cóncavas del contorno)
- **symmetry**
- **fractal dimension** ('aproximación de la línea de costa' - 1)

La media (**mean**), el error estándar (**standard error**) y el peor/mayor (**worst**) de estas características se calcularon para cada imagen, dando como resultado 30 características.

Target

- 0 = Maligno
- 1 = Benigno

3.2. Análisis y Visualización de Datos

De los 569 diagnósticos realizados, 357 corresponden a casos benignos y 212 a casos malignos [Figura 2]. En otras palabras, el 63 % de los diagnósticos indican tumores benignos, mientras que el 37 % señalan tumores malignos.

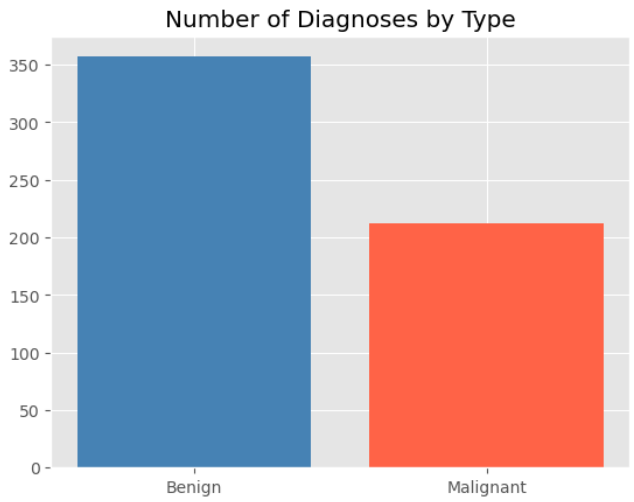


Figura 2: Distribución de Diagnósticos

Al analizar la distribución de las features según el diagnóstico (benigno o maligno) [Figura 3] se identificaron ciertas variables que podrían ser determinantes para la clasificación de tumores en benignos ó malignos. Variables tales como `mean radius`, `mean perimeter` y `worst radius` muestran claras diferencias en sus distribuciones según el diagnóstico, lo que sugiere que son buenos predictores, ya que presentan variaciones significativas entre los dos grupos. En contraste, variables como `mean fractal dimension`, `texture error` y `smoothness error` no exhiben diferencias notables en sus distribuciones respecto al diagnóstico, lo que indica que su capacidad para discriminar entre casos benignos y malignos podría ser limitada.

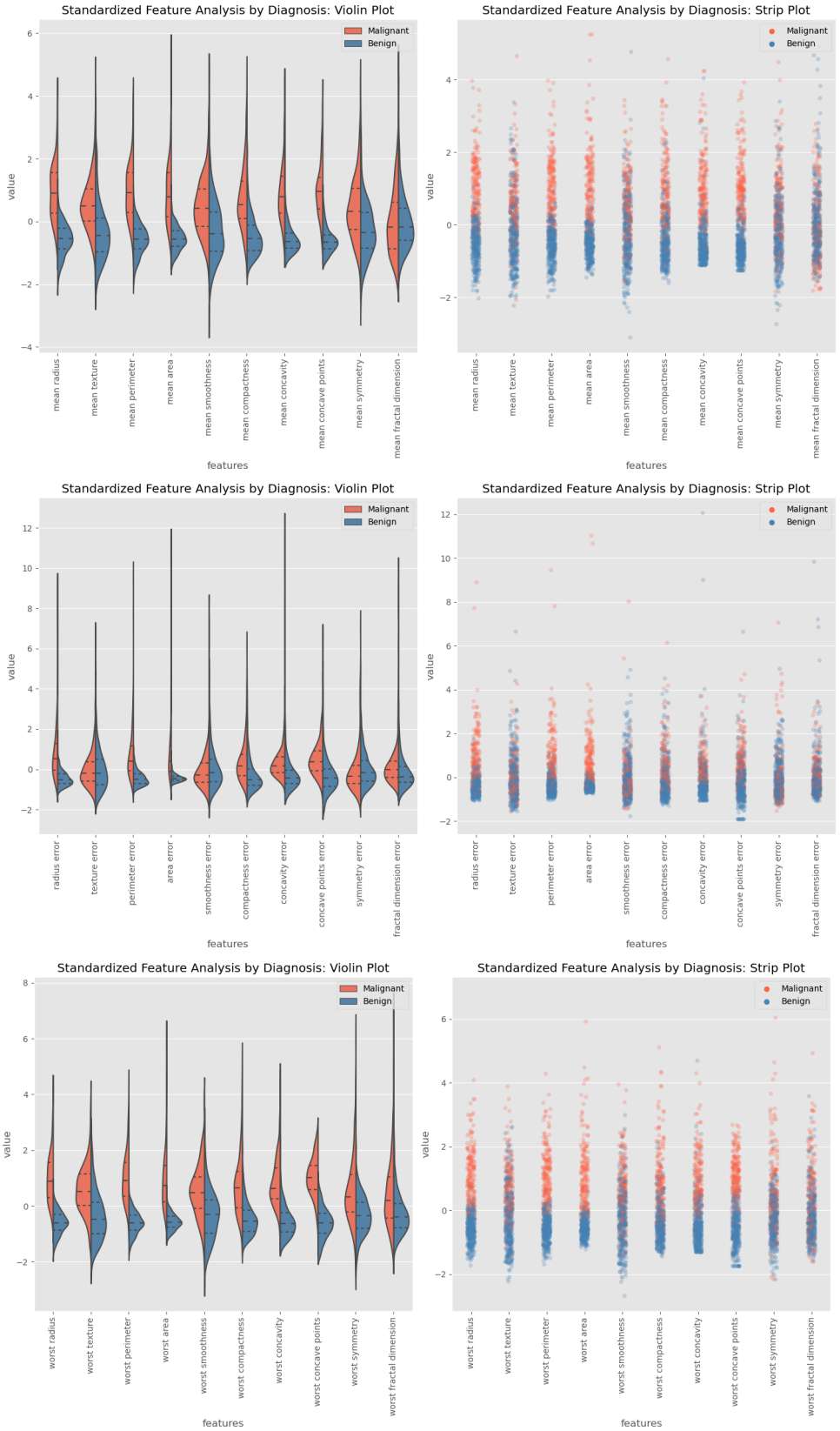


Figura 3: Distribución de las Features según el Diagnóstico

Lo anterior resalta la importancia de realizar un análisis detallado de las variables antes de construir un modelo de clasificación, ya que no todas las características pueden ser igualmente informativas. Es recomendable prestar especial atención a aquellas variables que muestran variaciones sustanciales entre los grupos, ya que podrían desempeñar un papel crucial en la performance del modelo de clasificación.

3.3. Análisis de Correlación

Como podemos observar en la matriz de correlación [Figura 4] muchas de las features se encuetran altamente correlacionada entre si.

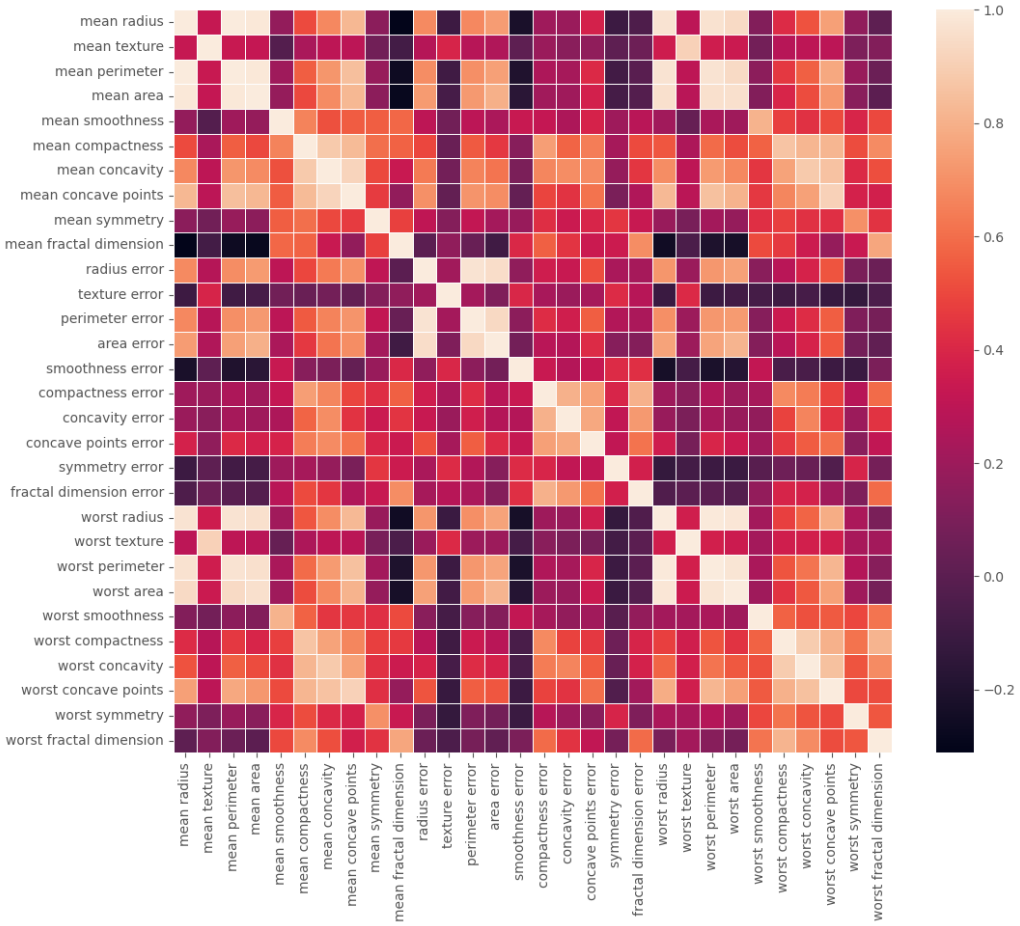


Figura 4: Matriz de Correlación

Esta alta correlación puede tener varias consecuencias para nuestro modelo de clasificación. A continuación, se listan algunas de las consecuencias y sus posibles soluciones:

Consecuencias

- **Multicolinealidad:** La presencia de alta correlación entre variables puede conducir a problemas de multicolinealidad, lo que dificulta la interpretación de la importancia de cada variable en el modelo.

- **Redundancia de Información:** Variables altamente correlacionadas pueden contener información redundante, lo que podría afectar negativamente la capacidad del modelo para generalizar a nuevos datos.

- **Inestabilidad del Modelo:** La variabilidad en los datos puede aumentar debido a la alta correlación, lo que podría hacer que el modelo sea más sensible a pequeñas variaciones en los datos de entrenamiento.

Soluciones

- **Feature Selection:** Considerar la eliminación de variables altamente correlacionadas o utilizar métodos de selección de características para mantener solo las más informativas.

- **Análisis más Profundo:** Realizar un análisis más detallado de la naturaleza de la correlación para entender mejor la relación entre las variables y tomar decisiones informadas sobre cómo manejarlas.

Cabe destacar que estas son solo algunas consideraciones iniciales, y es recomendado realizar un análisis más detallado de la correlación entre features antes de construir el modelo final.

4. Entrenando y Evaluando SVMs

```
# Importar bibliotecas
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import (
    train_test_split, cross_val_score, GridSearchCV
)
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import (
    SelectKBest, f_classif
)
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    confusion_matrix, ConfusionMatrixDisplay, classification_report
)
```

```
# Cargar el dataset 'Breast Cancer Wisconsin'
X, y = load_breast_cancer(return_X_y=True, as_frame=True
```

```
# Dividir el dataset en train y test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=0.8, random_state=42
)
```

```
# Crear un pipeline que incluye escalado, seleccion de features y un clasificador SVM
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('selector', SelectKBest(f_classif, k=10)),
    ('svc', SVC(class_weight='balanced'))
])
```

Nota: `SelectKBest` es una técnica de selección de características en el ámbito del Machine Learning y Data Mining. Es una herramienta que ayuda a seleccionar las mejores características de un conjunto de datos en función de alguna métrica específica.

```
# Entrenar un modelo basico como baseline usando Cross-Validation
pipeline.named_steps['svc'].set_params(kernel='linear')
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=5)
```

```
print("Cross-Validation Scores:", cv_scores)
print("Mean CV Score:", cv_scores.mean())
```

```
Cross-Validation Scores: [0.95604396 0.92307692 0.96703297 0.95604396 0.94505495]
Mean CV Score: 0.9494505494505494
```

```
# Entrenar un clasificador SVM con kernel lineal
param_grid_linear = {
    'svc__kernel': ['linear'],
    'svc__C': [0.001, 0.1, 1.0, 10.0, 100.0],
}

svc_linear = GridSearchCV(
    pipeline,
    param_grid_linear,
    cv=5
)
svc_linear.fit(X_train, y_train)
```

```
# Entrenar un clasificador SVM con kernel polinomial
param_grid_poly = {
    'svc__kernel': ['poly'],
    'svc__C': [0.001, 0.1, 1.0, 10.0, 100.0],
    'svc__degree': [2, 3, 4, 5, 6, 7]
}

svc_poly = GridSearchCV(
    pipeline,
    param_grid_poly,
    cv=5
)
svc_poly.fit(X_train, y_train)
```

```
# Entrenar un clasificador SVM con kernel RBF
param_grid_rbf = {
    'svc__kernel': ['rbf'],
    'svc__C': [0.001, 0.1, 1.0, 10.0, 100.0],
    'svc__gamma': ['scale', 'auto']
}

svc_rbf = GridSearchCV(
    pipeline,
    param_grid_rbf,
    cv=5
)
svc_rbf.fit(X_train, y_train)
```

- `C`: Controla el grado de penalización por errores de clasificación en los datos de entrenamiento. Un valor alto de `C` intenta minimizar los errores, lo que puede llevar a un modelo más ajustado pero con menor capacidad de generalización.

- `kernel`: Especifica la función de transformación que proyecta los datos en un espacio de mayor dimensión, donde las clases pueden ser separables. Los kernels más comunes son `linear` (lineal), `poly` (polinómico) y `rbf` (función de base radial o gaussiana).
- `gamma`: Define el alcance de la influencia de un solo punto de entrenamiento. Valores pequeños implican una influencia más amplia (modelos más suaves), mientras que valores grandes hacen que el modelo sea más sensible a puntos individuales.
- `degree`: Se utiliza únicamente con el kernel polinómico. Representa el grado del polinomio aplicado en la transformación. Grados más altos permiten modelar relaciones más complejas, pero aumentan el riesgo de sobreajuste.

Resultados de Entrenamiento

- SVM con Kernel Lineal

C	mean score	std score
0.1	0.942857	0.012815
1.0	0.942857	0.014579
100.0	0.940659	0.029157
10.0	0.936264	0.022413
0.001	0.918681	0.030769

- SVM con Kernel Polinomial

C	degree	mean score	std score
100.0	3	0.936264	0.025441
10.0	3	0.927473	0.025631
100.0	5	0.903297	0.023466
1.0	3	0.901099	0.025059
10.0	5	0.887912	0.028991

- SVM con Kernel RBF

C	gamma	mean score	std score
10.0	scale	0.958242	0.018906
10.0	auto	0.958242	0.018906
100.0	scale	0.951648	0.017855
100.0	auto	0.951648	0.017855
1.0	scale	0.942857	0.008223

Para evaluar el desempeño del modelo, utilizaremos las funciones `confusion_matrix` y `classification_report` de `scikit-learn`.

Matriz de Confusión

La matriz de confusión es una tabla que se utiliza en problemas de clasificación para describir el rendimiento de un modelo. En el contexto de una clasificación binaria (dos clases), la matriz tiene cuatro entradas:

- **True Positives (TP):** Son los casos en los que el modelo predijo correctamente que la instancia pertenece a la clase positiva.

- **True Negatives (TN):** Representa los casos en los que el modelo predijo correctamente que la instancia no pertenece a la clase positiva.
- **False Positives (FP):** Estos son los casos en los que el modelo predijo incorrectamente que la instancia pertenece a la clase positiva cuando, en realidad, no lo hace. También se le conoce como "*Error de Tipo I*".
- **False Negatives (FN):** Son los casos en los que el modelo predijo incorrectamente que la instancia no pertenece a la clase positiva cuando, en realidad, sí lo hace. También se le conoce como "*Error de Tipo II*".

La matriz de confusión tiene la siguiente forma:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

Reporte de Clasificación

La función `classification_report` proporciona un resumen de varias métricas de rendimiento:

- **Precision:** Mide la proporción de verdaderos positivos entre todas las instancias predichas como positivas. Se calcula como:

$$\text{Precision} = \frac{TP}{TP + FP}$$

  Una alta precisión indica que el modelo tiene menos falsos positivos.

- **Recall:** Mide la proporción de verdaderos positivos entre todas las instancias realmente positivas. Se calcula como:

$$\text{Recall} = \frac{TP}{TP + FN}$$

  Un alto recall indica que el modelo captura la mayoría de los positivos reales.

- **F1-Score:** Media armónica entre precision y recall. Se calcula como:

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Support:** Número real de ocurrencias de la clase en los datos de prueba.
- **Accuracy:** Proporción de instancias correctamente clasificadas. Se calcula como:

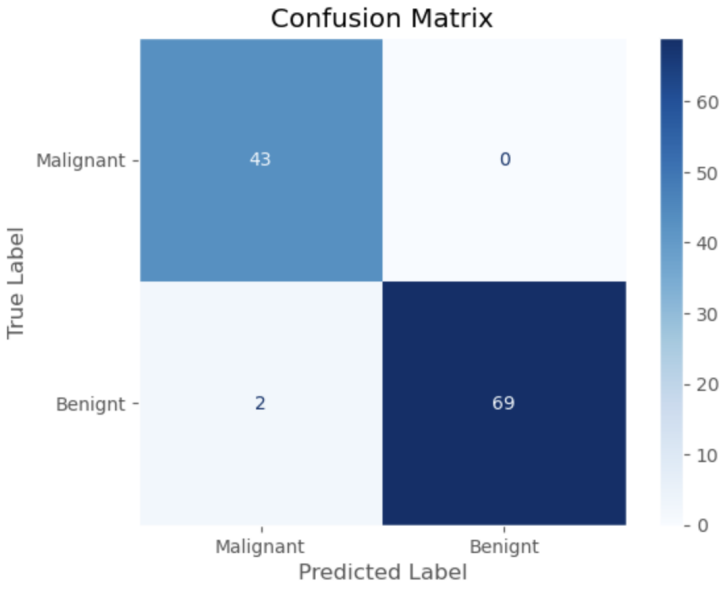
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  El accuracy puede ser engañoso si las clases están desbalanceadas.

- **Macro Avg:** Es el promedio sin ponderar de las métricas para cada clase.
- **Weighted Avg:** Es el promedio ponderado de las métricas para cada clase, según el soporte de cada clase.

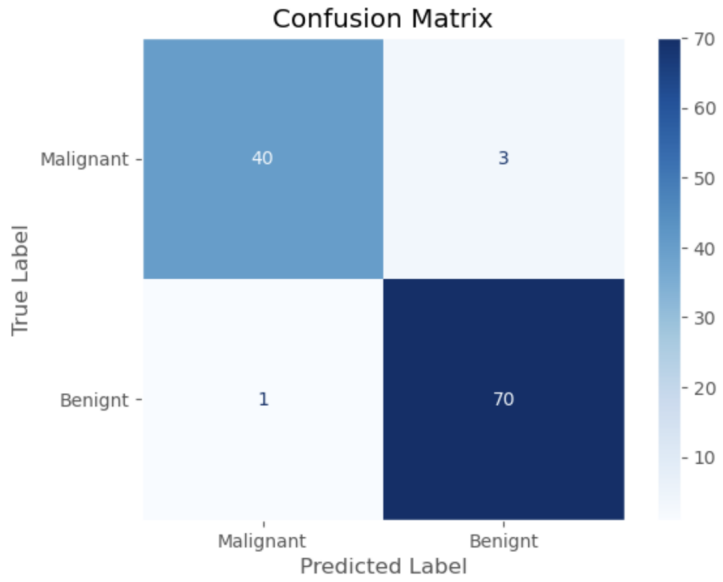
```python
def results(y: np.ndarray, y_pred: np.ndarray) -> None:
    '''
    Evalua los resultados de un modelo de clasificacion binaria
    mostrando una matriz de confusion y un informe de clasificacion.
    '''

    cm = ConfusionMatrixDisplay(
        confusion_matrix=confusion_matrix(y, y_pred),
        display_labels=['Malignant', 'Benignt']
    )
    cm.plot(cmap='Blues')

    plt.title('Confusion Matrix')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')

    plt.show()

    print(classification_report(y, y_pred))
```

```python
# Resultados del clasificador SVM con kernel lineal en el conjunto de test
results(y_test, svc_linear.best_estimator_.predict(X_test))
```



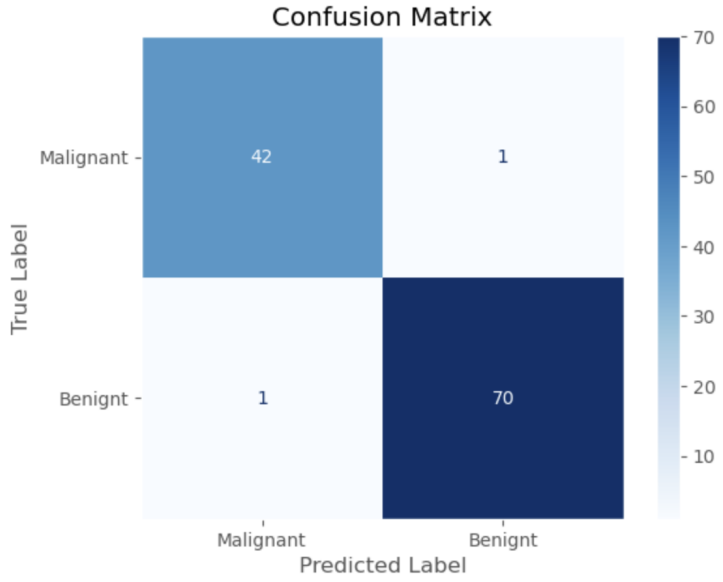
	precision	recall	f1-score	support
0	0.96	1.00	0.98	43
1	1.00	0.97	0.99	71
accuracy			0.98	114
macro avg	0.98	0.99	0.98	114
weighted avg	0.98	0.98	0.98	114

```python
# Resultados del clasificador SVM con kernel polinomial en el conjunto de test
results(y_test, svc_poly.best_estimator_.predict(X_test))
```



	precision	recall	f1-score	support
0	0.98	0.93	0.95	43
1	0.96	0.99	0.97	71
accuracy			0.96	114
macro avg	0.97	0.96	0.96	114
weighted avg	0.97	0.96	0.96	114

```
# Resultados del clasificador SVM con kernel RBF en el conjunto de test
results(y_test, svc_rbf.best_estimator_.predict(X_test))
```



	precision	recall	f1-score	support
0	0.98	0.98	0.98	43
1	0.99	0.99	0.99	71
accuracy			0.98	114
macro avg	0.98	0.98	0.98	114
weighted avg	0.98	0.98	0.98	114

5. Conclusiones

Al aplicar el algoritmo de Support Vector Machine (SVM) al conjunto de datos Breast Cancer Wisconsin para abordar un problema de clasificación binaria, observamos que el rendimiento se mantiene sólido independientemente del tipo de kernel utilizado. Tanto con un kernel lineal, polinómico o radial (RBF), los resultados obtenidos exhiben consistencia.

Este descubrimiento sugiere que el modelo SVM muestra robustez y es capaz de manejar la complejidad intrínseca de este conjunto de datos. Proporciona una notable capacidad de separación entre las clases, lo que subraya su idoneidad para abordar problemas de clasificación en este contexto específico.

A. Código

Puede acceder al notebook con el código utilizado en este proyecto a través del siguiente enlace de Google Colab: breast_cancer.ipynb.

B. Clasificación Multiclase

Puede acceder al siguiente notebook en Google Colab para ver un caso de aplicación del algoritmo SVM en un problema de clasificación multiclase: mnist.ipynb.

Referencias

[1] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.* O'Reilly Media, 2019. ISBN: 9781492032649.

[2] Estefanía Nievas Lio. *Aplicando máquinas de soporte vectorial al análisis de pérdida no técnica de energía eléctrica.* 2016.

[3] scikit-learn. *Support Vector Machines (SVMs).* https://scikit-learn.org/stable/modules/svm.html.