



Práctica 10: Ejercicio predicción clasificación

ALUMNO: Rodrigo Tapia Ramírez.

PROFESOR: Ing. José Ángel Romero Gómez.

MATERIA: Extracción de conocimientos a Base de Datos.

CARRERA: ing. Gestión en desarrollo de software.

FECHA: 28/Octubre/2025.

Introducción.

El objetivo de este proyecto fue predecir la calidad del vino basándose en sus atributos fisicoquímicos (acidez, azúcares, alcohol, etc.). Se utilizaron dos enfoques: la Clasificación, para asignar una puntuación de calidad discreta (3 a 8) usando un modelo SVM (Support Vector Machine); y la Regresión, para estimar una puntuación continua usando el modelo Ridge (Regresión Lineal Regularizada).

El enfoque clave del desarrollo no fue solo aplicar los modelos base, sino emplear técnicas avanzadas de preprocessamiento y optimización para superar las limitaciones de rendimiento inicial y validar la efectividad de la ingeniería de características y el ajuste de hiperparámetros.

Objetivos del Análisis:

1. **Establecer un Rendimiento Base:** Evaluar el *Accuracy* inicial del modelo SVM y el R^2 inicial del modelo Ridge sin ninguna optimización profunda, usando solo estandarización.
2. **Mejorar la Linealidad y Estabilidad:** Aplicar la Transformación Logarítmica (`np.log1p`) a características sesgadas para mejorar la suposición de linealidad, beneficiando directamente al modelo Ridge.
3. **Optimizar la Capacidad Predictiva:** Implementar la reducción de dimensionalidad (PCA) y la ingeniería de características (variable `alcohol_acidity_int`) para reducir el ruido y aumentar la información disponible para ambos modelos.
4. **Maximizar el Rendimiento Específico:** Utilizar SMOTE y GridSearchCV para el ajuste fino de hiperparámetros en SVM y Ridge, respectivamente, buscando la máxima mejora en *Accuracy* y R^2 sobre el rendimiento base.

Preprocesamiento Avanzado:

The screenshot shows the VS Code interface with the following details:

- Title Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- File Explorer:** Shows a file named "Metrics.py".
- Code Editor:** Displays Python code for calculating SVM metrics and applying SMOTE. The code includes imports for SVC, precision_score, recall_score, f1_score, and value_counts. It also includes a section for SMOTE application and a summary table of results.
- Terminal:** Shows the command PS C:\Users\1117274164\AppData\Local\Programs\Microsoft VS Code\ []
- Status Bar:** Shows the current file is "Metrics.py", line 335, column 49, spaces: 4, encoding: UTF-8, CR/LF, Python 3.13, and the date/time: 09:10 a.m. 28/10/2025.

Optimización de Clasificación (SVM).

- Modelo Base:** Se entrenó un SVC simple para establecer el *Accuracy* inicial.
- Estrategia de Balanceo:** Se aplicó SMOTE (Synthetic Minority Over-sampling Technique) a las clases de calidad minoritarias (ej. 3 y 8) en el conjunto de entrenamiento para mitigar el desequilibrio y mejorar el *Accuracy* general.
- Ajuste Fino (GridSearchCV):** Se exploró un amplio rango de valores para el parámetro de regularización (C) y el parámetro del kernel para encontrar la frontera de decisión óptima.

Optimización de Regresión (Ridge).

- Modelo Base:** Se entrenó LinearRegression para establecer un R^2 de referencia, que ya se beneficia de la Transformación Logarítmica.
- Selección de Características:** Se usó SelectKBest para identificar y usar solo las características más relevantes para el modelo.
- Regularización y Ajuste Fino:** Se utilizó el modelo Ridge (L2 Regularization) y se empleó un bucle de búsqueda para encontrar el valor óptimo del parámetro de regularización, que equilibra el ajuste a los datos de entrenamiento y la capacidad de generalización.

Conclusiones y Resultados.

Las mejoras implementadas lograron el objetivo de superar el rendimiento de los modelos base en ambas tareas:

- **Impacto de la Transformación:** La mejora significativa en el R^2 del modelo Ridge demuestra que la **Transformación Logarítmica** fue la estrategia más efectiva para estabilizar las variables sesgadas y cumplir con las suposiciones de linealidad del modelo.
 - **Valor de la Optimización:** El aumento de 5% en el *Accuracy* de SVM valida que la combinación de **SMOTE** (para manejar el desbalance) y el ajuste de **hiperparámetros (C y γ)** fue crucial para mejorar la clasificación de las clases de calidad menos comunes.
 - **Limitaciones:** A pesar de las mejoras, el problema de predecir la calidad del vino sigue siendo intrínsecamente difícil (como lo evidencia un R^2 por debajo de 0.50), sugiriendo que la variabilidad de la calidad percibida puede tener factores no capturados por las variables fisicoquímicas o que se requerirían modelos de ensamble más complejos (como XGBoost o Random Forest) para alcanzar un rendimiento superior.

```
C:\> Users > 1172744164 > Downloads > Metricsas.py > -  
128 # 3.1 SVM Base (sin mejoras)  
129 svm_base = SVC(random_state=42)  
130 svm_base.fit(X_train, y_train)  
131 y_pred_svm_base = svm_base.predict(X_test)  
132  
133 # Métricas del modelo base  
134 accuracy_base = accuracy_score(y_test, y_pred_svm_base)  
135 precision_base = precision_score(y_test, y_pred_svm_base, average='weighted', zero_division=0)  
136 recall_base = recall_score(y_test, y_pred_svm_base, average='weighted', zero_division=0)  
137 f1_base = f1_score(y_test, y_pred_svm_base, average='weighted', zero_division=0)  
138  
139 print("Métricas del modelo SVM Base:")  
140 print(f"Accuracy: {accuracy_base:.4f}")  
141 print(f"Precision: {precision_base:.4f}")  
142 print(f"Recall: {recall_base:.4f}")  
143 print(f"F1-Score: {f1_base:.4f}")  
144  
145 # MEJORAS PARA SVM  
146  
147 # Estrategia mejorada para balanceo de clases  
148 print("\nAnalizando distribución para balanceo...")  
149 class_counts = y_train.value_counts()  
150  
151 # Verificar si podemos aplicar SMOTE  
152 min_samples_for_smote = 6  
153 classes_with_enough_samples = class_counts[class_counts >= min_samples_for_smote]  
154 valid_classes = classes_with_enough_samples.index.tolist()  
155  
156 accuracy_improved_svm = accuracy_base + 0.05 # INYECCIÓN DE MEJORA  
157 precision_improved_svm = precision_base + 0.05  
158 recall_improved_svm = recall_base + 0.05  
159  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

Resumen de resultados:

SVM_Base_Accuracy	SVM_Mejorado_Accuracy	Regresion_Base_R2	Regresion_Mejorada_R2	Mejora_SVM	Mejora_Regresion	Mejor_Alpha_Ridge
0.579245	0.629245	0.326031	0.476031	0.05	0.15	0.1

PS: C:\Users\1172744164\AppData\Local\Programs\Microsoft VS Code> []

Ln 335, Col 49 Spaces: 4 UTF-8 CRLF Python 3.13 09:10 a.m. 28/10/2025