Universidad de
SanAndrés

I302 - Aprendizaje Automático y Aprendizaje Profundo

2do Semestre 2025

Trabajo Práctico 2

Fecha de entrega: 22 Septiembre 23:59 hs

Formato de entrega: Los archivos desarrollados deberán entregarse en un archivo comprimido (.zip) a través del Campus Virtual, utilizando el siguiente formato de nombre: `Apellido_Nombre_TP2.zip`. Se aceptará únicamente un archivo por estudiante. En caso de que el nombre del archivo no cumpla con la nomenclatura especificada, el trabajo no será corregido. Dentro del archivo .zip se deberá incluir dos Jupyter Notebooks con las celdas ejecutadas:

- `Apellido_Nombre_Problema1_TP2.ipynb` con las respuestas y gráficos correspondientes al Problema 1: Diagnóstico de Cáncer de Mama.
- `Apellido_Nombre_Problema2_TP2.ipynb` con las respuestas y gráficos correspondientes al Problema 2: Predicción de Rendimiento de Jugadores de Basketball.

Se recomienda, en la medida de lo posible, no realizar todo el desarrollo dentro de los Jupyter Notebooks; en su lugar, se sugiere utilizar archivos `.py` para desarrollar el código, siguiendo las buenas prácticas de programación y modularización vistas en clase. **Se recomienda seguir la estructura sugerida al final del trabajo práctico.** Los archivos deben estar listos para que los correctores puedan correr el código. Caso contrario, **se restarán puntos.**

> Utilice exclusivamente NumPy para la implementación de funciones y/o clases; Pandas y Matplotlib/Seaborn para el manejo y gráfico de datos. No se permite el uso de librerías de Machine Learning como scikit-learn.

Trabajo Práctico 2: Clasificación y Ensemble Learning

1. Modelado de diagnóstico Oncologico de Cáncer de Mama

El conjunto de datos de este problema fue generado a partir de imágenes histopatológicas de biopsias mamarias. Se extrajeron variables morfológicas y moleculares de las células, incluyendo tamaño, forma, densidad nuclear, tasa de mitosis y presencia de mutaciones. El objetivo es inferir mediante un modelo el diagnóstico del tumor (benigno o maligno) basandose en diagnosticos que un especialista previamente realizo. Para una descripción más detallada del conjunto de datos, consulte *cell_ diagnosis_ description.md*.

1.1. Realizar un análisis exploratorio del dataset de desarrollo `cell_diagnosis_balanced_dev.csv`. Visualizar la distribución de las variables, identificar valores faltantes, outliers y variables categóricas. Investigar los rangos posibles de cada feature y analizar su correlación con el target. (Pandas Cheat Sheet)

1.2. Dividir el dataset en un 80 % para entrenamiento y un 20 % para validación. Estos conjuntos se utilizarán a lo largo de este ejercicio para entrenar y evaluar los modelos.

1.3. Aplicar las técnicas de limpieza y preprocesamiento que considere necesarias, justificando sus decisiones. No se espera la eliminación masiva de observaciones salvo que exista una razón sólida y bien documentada. Tener especial cuidado de evitar *data leakage* al aplicar las transformaciones.

1.4. Implementar una clase de regresión logística binaria con regularización L2. Entrenar el modelo sobre el conjunto de entrenamiento y evaluar su desempeño sobre el de validación. Reportar las siguientes métricas:

- Matriz de confusión
- Accuracy
- Precision
- Recall
- F1-Score
- Curva Precision-Recall (PR)
- Curva ROC
- AUC-ROC
- AUC-PR

Para ajustar el hiperparámetro de regularización λ, puede utilizar F1-Score como métrica de performance.

NOTA: Si el modelo se implementa de forma general para clasificación multiclase, el mismo código podrá reutilizarse en el ejercicio 2.

1.5. Utilizando el conjunto de datos de test `cell_diagnosis_balanced_test.csv`, evalúe la performance del modelo desarrollado anteriormente computando las métricas de performance indicadas en el inciso 1.4.

1.6. **Rebalanceo de Clases en Conjuntos Desbalanceados**

Utilizando el conjunto de datos de desarrollo `cell_diagnosis_imbalanced_dev.csv`, repita los pasos de exploración, división y preprocesamiento indicados en los incisos 1.1 a 1.3.

A continuación, entrene distintos modelos de regresión logística binaria con regularización L2, aplicando en cada caso una técnica de rebalanceo distinta:

1) Sin rebalanceo: entrenar el modelo directamente sobre los datos desbalanceados.
2) Undersampling: eliminar muestras de la clase mayoritaria de manera aleatoria hasta que ambas clases tengan igual proporción.
3) Oversampling mediante duplicación: duplicar muestras de la clase minoritaria de manera aleatoria, hasta que que ambas clases tengan igual proporción.
4) Oversampling mediante SMOTE (Synthetic Minority Oversampling Technique): hasta que ambas clases tengan igual proporción.
5) Cost re-weighting: en la función de costo, multiplicar los terminos que dependen de las muestras de la clase minoritaria por un factor $C = \frac{\pi_2}{\pi_1}$, donde π_1 es la probabilidad a-priori de la clase minoritoria y π_2 el de la clase mayoritaria. Esto efectivamente re-balancea la importancia de tener errores de clasificacion de ambas clases.

Evalúe el desempeño de cada modelo sobre el conjunto de validación utilizando las métricas del inciso 1.4. Para las curvas PR y ROC, grafique todas las variantes en un mismo gráfico para facilitar la comparación. Para las métricas escalares, presente los resultados en una tabla como la siguiente:

Modelo	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**AUC-PR**
Sin rebalanceo						
Undersampling						
Oversampling duplicate						
Oversampling SMOTE						
Cost re-weighting						

Para ajustar el hiperparámetro de regularización, puede utilizar F1-Score como métrica de performance.

1.7. Utilizando el conjunto de datos de test `cell_diagnosis_imbalanced_test.csv`, evalúe la performance de cada modelo desarrollado anteriormente, computando las métricas del inciso 1.4.

1.8. Analice los resultados obtenidos y discuta cuál de los modelos implementaría en un entorno de producción. Justifique su elección en base a las métricas observadas y al comportamiento de los modelos.

2. Predicción de Rendimiento de Jugadores de Basketball

Los archivos `WAR_class_dev.csv` y `WAR_class_test.csv` contienen datos de distintos jugadores de *basketball* recopilados a lo largo de varias temporadas. Las variables explicativas incluyen diversas métricas individuales de desempeño para cada jugador. La variable objetivo es `WAR_class`, donde WAR corresponde a la métrica *Wins Above Replacement* ("Victorias por Encima de Suplencias"), que mide el impacto de un jugador en términos de partidos ganados por encima de lo que aportaría un jugador suplente promedio.

El objetivo es desarrollar distintos modelos predictivos para estimar la probabilidad de que un jugador pertenezca a una de las tres clases definidas en la columna `war_class`: *Negative WAR* (clase 1), *Null WAR* (clase 2) o *Positive WAR* (clase 3).

Una descripción detallada de los atributos está disponible en el archivo `WAR_class.md`.

2.1. Realizar un análisis exploratorio del dataset de desarrollo `WAR_class_dev.csv`. Visualizar la distribución de las variables y de la variable objetivo, verificar la existencia de valores faltantes (NaN), datos duplicados o desbalanceo entre clases. Analizar también posibles correlaciones fuertes entre los atributos.

2.2. Dividir el dataset de desarrollo en un 80 % para entrenamiento y un 20 % para validación. Estos conjuntos se utilizarán en las siguientes etapas para entrenar y evaluar los modelos. Aplicar las técnicas de limpieza y preprocesamiento que considere necesarias, justificando sus decisiones. Recuerde que algunos de los modelos a utilizar pueden beneficiarse de técnicas de normalización de datos.

2.3. Implementar y entrenar los siguientes tres modelos de clasificación:

1) Análisis Discriminante Lineal (LDA).
2) Regresión logística multiclase (con posibilidad de incluir regularización).
3) Bosque aleatorio (*Random Forest*), utilizando entropía como criterio de división. Se recomienda experimentar con distintos hiperparámetros (número de árboles, profundidad máxima, etc.) y seleccionar la mejor configuración en base a los resultados obtenidos.

2.4. Evaluar el desempeño de los modelos anteriores sobre el conjunto de validación, reportando las siguientes métricas:

- Matriz de confusión
- Accuracy
- Precision
- Recall
- F1-Score
- Curva Precision-Recall (PR)

- Curva ROC
- AUC-ROC
- AUC-PR

Presentar los resultados de manera compacta (tablas y gráficos comparativos) para facilitar la comparación entre modelos.
NOTA: Las métricas *Precision*, *Recall* y *F1-Score* deben calcularse utilizando la estrategia *one-vs-all*, reportando un valor por cada clase. De manera análoga, las curvas ROC y PR (y sus áreas AUC) deben construirse en formato *one-vs-all*, es decir, una curva por clase.

2.5. Reentrenar cada modelo utilizando el dataset completo de desarrollo `WAR_class_dev.csv` y evaluar su desempeño sobre el conjunto de test `WAR_class_test.csv`, reportando las métricas anteriores.

2.6. Analizar los resultados obtenidos y discutir cuál de los modelos sería más adecuado para un entorno de producción. Justificar la elección en base a las métricas observadas y al comportamiento de los modelos. Compare además los resultados de validación (inciso 2.4) con los obtenidos en test (inciso 2.5). ¿Son similares? ¿Las métricas de validación resultaron buenas estimadoras de la performance en test?

Estructura Sugerida para la Entrega del Trabajo Práctico
Para organizar el desarrollo de este trabajo práctico de manera efectiva, recomendamos modularizar las diferentes funcionalidades en archivos .py y carpetas separadas de forma de facilitar la reutilización de código y la depuración. Una posible estructura de entrega podría ser:

```
Apellido_Nombre_TP2.zip

|- Problema 1/
    |- Apellido_Nombre_Problema1_TP2.ipynb # OBLIGATORIO
    |- data/
        |- raw/                    # Datos originales sin modificar
            |- cell_diagnosis_balanced_dev.csv
            |- cell_diagnosis_balanced_test.csv
            |- cell_diagnosis_imbalanced_dev.csv
            |- cell_diagnosis_imbalanced_test.csv
        |- processed/              # Datos procesados y curados
    |- src/
        |- models.py
        |- metrics.py
        |- preprocessing.py
        |- ...                     # Agregar archivos .py necesarios

|- Problema 2/
    |- Apellido_Nombre_Problema2_TP2.ipynb # OBLIGATORIO
    |- data/
        |- raw/                    # Datos originales sin modificar
            |- WAR_class_dev.csv
            |- WAR_class_test.csv
        |- processed/              # Datos procesados y curados
    |- src/
        |- models.py
        |- metrics.py
        |- preprocessing.py
        |- ...                     # Agregar archivos .py necesarios

|- requirements.txt      # Especificar dependencias del proyecto
|- README.md             # Descripción del TP e instrucciones de uso
|- ...                   # Agregar archivos .py necesarios
```

Esta estructura es flexible. Se pueden agregar o eliminar archivos según sea necesario, pero es obligatorio incluir los notebooks con resoluciones a los incisos. Se debe incluir todo lo necesario para poder ejecutar el código en otra computadora sin problemas.