Universidad de
SanAndrés

I302 - Aprendizaje Automático y Aprendizaje Profundo

1er Semestre 2025

Trabajo Práctico 2

Fecha de entrega: Lunes 14 de abril 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 y debe contener por lo menos los siguientes elementos:

```
Apellido_Nombre_TP2.zip/
|- data/
|- Apellido_Nombre_Informe_TP2.pdf
|- Apellido_Nombre_Notebook_TP2.ipynb
```

- **Informe:** Debe incluir todos los aspectos teóricos, decisiones metodológicas, visualizaciones, análisis y conclusiones. El objetivo es que el informe contenga toda la explicación principal del trabajo. Se puede hacer referencia al notebook con frases como "Ver sección X del notebook para la implementación". El informe debe entregarse utilizando el archivo `template_informe.tex` provisto y no debe exceder las 10 páginas.
- **Notebook:** Debe contener el código utilizado, experimentos, análisis exploratorio, gráficos y el proceso completo de procesamiento y modelado. Sirve como respaldo técnico del informe y debe estar ordenado y bien documentado. Se recomienda modularizar el código en archivos `.py` cuando sea posible.

Para más detalles, consultar la estructura sugerida al final del trabajo práctico.

Trabajo Práctico 2: Clasificación y Ensemble Learning

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.

1. **Diagnóstico 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 predecir el diagnóstico del tumor (benigno o maligno). Para una descripción más detallada del conjunto de datos, consulte *cell_diagnosis_description.md*.

   1.1. Realizar un análisis exploratorio de los datos para visualizar la distribución de las variables e identificar valores faltantes, outliers y variables categóricas. Investigar los rangos posibles de cada feature y analizar su correlación con el target.

   1.2. Implementar una clase de regresión logística binaria con regularización L2. Utilizando el conjunto de datos de desarrollo *cell_diagnosis_balanced_dev.csv*, dividir los datos en 80 % entrenamiento y 20 % validación. Entrenar el modelo sobre el conjunto de entrenamiento y evaluar su desempeño sobre el conjunto de validación. Reportar las siguientes métricas de performance:

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

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

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

   1.3. 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 1.2.

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

   Utilizando el conjunto de datos de desarrollo *cell_diagnosis_imbalanced_dev.csv*, dividir los datos en 80 % entrenamiento y 20 % validación. Entrenar distintos modelos de regresión logística binaria con regularización L2, aplicando una técnica de re-balanceo distinta en cada caso:

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 indicadas en el inciso 1.2. Para las curvas PR y ROC, grafique las curvas de cada modelo sobre el mismo grafico, de manera de poder comparar las características de cada uno. Para las metricas de performance escalares, ponga los resultados de cada modelo en una sola tabla, como se muestra a continuación:

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

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

1.5. Utilizando el conjunto de datos de test *cell_ diagnosis_ imbalanced_ test.csv* evalue la performance de cada uno de los modelos desarrollados anteriormente computando las métricas del inciso 1.4.

1.6. Analizar los resultados obtenidos y discutir cuál de los modelos implementaría en un entorno de producción. Justificar la elección en base a las métricas observadas y al comportamiento del modelo.

2. Predicción de Rendimiento de Jugadores de Basketball

Los archivos *WAR_ class_ dev.csv* y *WAR_ class_ test.csv* tienen los datos de distintos jugadores de *basketball* recopilados a lo largo de varias temporadas. Los features indican diversas métricas individuales para cada jugador. La variable objetivo es *WAR_ class*, donde WAR son las siglas de una métrica llamada *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*, *Null WAR* o *Positive WAR*, que corresponden a las clases 1, 2 y 3 respectivamente.

El resto de las features presentes están explicadas en el archivo *WAR_class.md*.

2.1. Conducir un análisis exploratorio sobre los datos en *WAR_class_dev.csv*. Visualice la distribución de las variables y de la variable objetivo. Verifique si hay valores faltantes (NaN), datos duplicados o desbalanceo entre clases. Analice también la posible presencia de correlaciones fuertes entre los features. (Pandas Cheat Sheet)

2.2. Implementar las siguientes tres arquitecturas, adecuadas para la tarea de clasificación:

1) Análisis discriminante lineal (Linear Discriminant Analysis, LDA).
2) Regresión logística multi-clase. Recuerde que puede agregar regularización.
3) Bosque aleatorio (Random Forest) utilizando la entropía como criterio de división. Se recomienda experimentar con diferentes configuraciones de hiperparámetros (número de árboles, profundidad máxima, etc.). Luego, se podrá seleccionar la mejor combinación en función de los resultados obtenidos.

2.3. Utilizando el conjunto de datos de desarrollo *WAR_class_dev.csv* entrene los modelos mencionados en el inciso anterior. Recuerde que estos modelos, en particular LDA y regresión logística, suelen obtener mejores resultados cuando se entrenan con datos normalizados. Utilice las técnicas de reescalado y data splits que considere más apropiadas. Para cada modelo desarrollado reporte la estimación de las siguientes métricas de performance.

WAR_class_dev.csv:

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

Presentar los resultados de manera compacta para facilitar la comparación entre los modelos.

2.4. Re-entrene los modelos utilizando todo el conjunto de desarrollo para ajustar los parámetros *WAR_class_dev.csv*, y evalúe las métricas de performance usando el conjunto de test *WAR_class_test.csv*.

2.5. Analizar los resultados obtenidos y discutir cuál de los modelos llevaría a producción. Justificar la elección en base a las métricas observadas y al comportamiento del modelo. Además, compare las métricas de performance obtenidas en los incisos 2.3 y 2.4 . ¿Son parecidas? Es decir, ¿la estimación de métricas de error mediante validación/validación cruzada son parecidas a las métricas obtenidas sobre el conjunto de 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

|- Apellido_Nombre_Informe.pdf          # OBLIGATORIO
|- Apellido_Nombre_Notebook_TP2.ipynb   # OBLIGATORIO

|- Problema 1/
    |- data/
        |- raw/                         # Datos originales sin modificar
            |- cell_diagnosis_dev.csv
            |- cell_diagnosis_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/
    |- 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 el informe y notebook con resoluciones. Se debe incluir todo lo necesario para poder ejecutar el código en otra computadora sin problemas.