{desafío} latam_ Academia de talentos digitales

Actividad 5 - Clasificación con Máquinas de Soporte Vectorial

- Para poder realizar esta actividad debes haber revisado la lectura correspondiente a la semana.
- Crea una carpeta de trabajo y guarda todos los archivos correspondientes (notebook y csv).
- Una vez terminada la actividad, comprime la carpeta y sube el `.zip` a la sección correspondiente.

Descripción de Actividades

- Para esta sesión trabajaremos con la base de datos sobre cáncer mamario de Wisconsin. El objetivo es desarrollar un Clasificador mediante Máquinas de Soporte de Vectores que predica de forma adecuada en base a una serie de atributos sobre la composición del núcleo de una célula mamaria. Para más detalles técnicos asociados a la base de datos, pueden hacer click en el link.

Desafío 1: Preparar el ambiente de trabajo

- Importe todas las librerías a utilizar.
- Fije los parámetros de los gráficos con `plt.Rcparams`.
- Excluya las columnas `id` y `Unnamed: 32` de la base de datos.
- Recodifique el vector objetivo `diagnosis` a numérico para poder procesarlo posteriormente.

Desafío 2: Visualizando la distribución de los atributos

- Para cada uno de los atributos, grafique los histogramas condicional a cada clase del vector objetivo.
- Agregue las medias correspondientes y reporte a grandes rasgos cuáles son los atributos con una mayor similitud en la distribución.

Desafío 3: Estimando el porcentaje de overlap en los atributos

- Parte de las virtudes de las Máquinas de Soporte Vectorial es la capacidad de lidiar con clases no separables mediante el proceso de kernelización. Resulta que un aspecto importante que muchas veces se obvía es medir la noseparabilidad de los atributos, condicional a cada clase del vector objetivo.
- El procedimiento para estimar el rango de noseparabilidad entre clases se implementa en Python de la siguiente manera:

```
h1, _ = np.histogram(var1, bins=100) # estimamos la frecuencia de obs. en
100 espacios definidos
h2, _ = np.histogram(var2, bins=100) # estimamos la frecuencia de obs. en
100 espacios definidos
get_minima = np.minimum(h1, h2)      # extraemos el mínimo de observaciones
comunes entre h1 y h2
intersection = np.true_divide(np.sum(get_minima), np.sum(tmp_2)) # Estimamos
la intersección de elementos comunes
```

- La intersección devolverá el porcentaje de comunalidad entre ambas clases, donde mayores niveles indican una mayor comunalidad.
- Generalice el código de arriba en una función para evaluar el porcentaje de noseparabilidad entre las clases para cada atributo.
- Posteriormente genere un dataframe donde almacenará el nombre del atributo y su porcentaje. Ordene este dataframe de forma descendente y preserve.

Desafío 3: Selección del modelo por GridSearchCV

- Entrene una serie de modelos `SVC` con los siguientes hiperparámetros:
  - `C: [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]`.
  - `gamma: [0.0000001, 0.0001, 0.001, 0.01, 0.1, 1, 10]`.
  - Validaciones cruzadas: 10.
- Genere un heatmap en base a los puntajes estimados con `GridSearchCV`.

tip: Vea cómo acceder a la llave `mean_test_score` en el diccionario `cv_results_`.

- Reporte en qué rango de cada hiperparámetro el modelo presenta un desempeño eficiente. Reporte la mejor combinación de hiperparámetros y el desempeño en la muestra de entrenamiento.

Desafío 4: Validación del modelo en el hold-out sample

- Genere las predicciones del hold-out sample en base a la mejor combinación de hiperparámetros. Genere un reporte con las métricas de desempeño clásicas para los modelos de clasificación. Comente en qué casos el modelo presenta un desempeño deficiente

Desafío (opcional): Depuración de atributos

- Reentrene el modelo en función de los atributos que presenten un coeficiente de overlap menor a .45.
- Reporte el desempeño del modelo y comente sobre los nuevos hiperparámetros estimados, así como su desempeño en comparación al modelo del ejercicio 4.