Proyecto:_PIA_AA

Versión 1.0

Noel Freire Mahía Iván Hermida Mella

02 de mayo de 2025

Contents:

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

CAPÍTULO 1

Funciones Generales

subir_datos.**cargar_datos**(*path*, *target*)

Carga los datos de un archivo CSV.

Args:

path (str): Ruta de los datos a normalizar. target (str): Nombre de la columna que contiene las etiquetas.

Returns:

X (Dataframe): Datos que se enviarán al modelo. y (Series): Etiquetas de cada dato.

normalizar.**normalizar_datos**(*x*)

Normaliza los datos utilizando StandardScaler (media 0, desviación típica 1) de Scikit-Learn.

Args:

x (Dataframe): Datos a normalizar.

Returns:

x_scaled (Dataframe): Datos normalizados. scalers (dict): Diccionario de scalers utilizados para cada columna.

guardar_metricas.**guardar_metricas**(*k*, *metrics*, *file*, *histories*, *roc*, *n_jobs*)

Guarda las métricas del modelo en un archivo Excel, incluyendo información sobre las predicciones, evolución del entrenamiento y curvas ROC si están disponibles.

El archivo Excel generado puede contener varias hojas: - *k_metrics*: Métricas generales del modelo por cada fold de la validación cruzada. - *y_test*: Clases reales de las imágenes utilizadas en la validación. - *y_pred*: Clases predichas por el modelo. - *histories* (opcional): Evolución del accuracy y la pérdida durante el entrenamiento. - *roc* (opcional): Datos para la curva ROC global y por clases.

Parámetros

- **k** (*int*) – Número de folds en la validación cruzada.
- **metrics** (*dict*) – Diccionario con las métricas del modelo, incluyendo las predicciones y opcionalmente datos de entrenamiento y curvas ROC.
- **file** (*str*) – Nombre del archivo Excel donde se guardarán las métricas.

- **histories** (*bool*) – Indica si *metrics* contiene los datos de evolución del accuracy y pérdida.
- **roc** (*bool*) – Indica si *metrics* contiene los datos para generar curvas ROC.

Devuelve
No retorna ningún valor, solo genera y guarda un archivo Excel con las métricas.

Tipo del valor devuelto
None

metricas_clustering.**guardar_metricas_clustering**(*file*, *metrics*, *etiquetas_real*, *etiquetas_pred*, *centroides=None*)

Guarda las métricas de evaluación, etiquetas reales, etiquetas predichas y, si están disponibles, los centroides del clustering en un archivo Excel.

Args:
file (str): Ruta y nombre del archivo Excel donde se guardarán los resultados. metrics (dict): Diccionario con métricas de evaluación (V-Measure e Índice de Rand ajustado). etiquetas_real (array-like): Etiquetas reales de los datos. etiquetas_pred (array-like): Etiquetas asignadas por el modelo de clustering. centroides (ndarray, optional): Coordenadas de los centroides de los clústers (si aplica).

Returns:
None: Los datos se guardan en el archivo Excel especificado.

comparacion_n_covariables.**comparar_todos_modelos**(*acc*, *modelo*)

Compara todos los modelos usando la prueba de Kruskal-Wallis y la prueba de Tukey si hay diferencias significativas.

Parámetros
acc (*dict*) – Diccionario con las precisiones de los modelos.

Devuelve
El nombre del mejor modelo y su media de precisión.

Tipo del valor devuelto
tuple(str, float)

comparacion_n_covariables.**diagrama_cajas**(*accuracy*)

Genera un diagrama de cajas (boxplot) para comparar la precisión de diferentes modelos.

Parámetros
accuracy (*dict*) – Diccionario que contiene las precisiones de los modelos.

Devuelve
None

comparacion_n_covariables.**inicio_comp**(*Param_default*, *ICA_4*, *ICA_8*, *ICA_11*, *PCA_4*, *PCA_8*, *PCA_11*, *modelo*)

Función principal que inicia el proceso de comparación de modelos.

Lee los archivos de precisión, compara modelos y guarda los resultados.

Devuelve
None

CAPÍTULO 2

Funciones Clustering

clustering.**dbscan**(*x*, *y*)

Aplica el algoritmo DBSCAN, calcula métricas de calidad del clustering y visualiza la distribución de distancias para seleccionar un buen valor de epsilon.

Args:

x (ndarray): Datos de entrada para el clustering. y (array-like): Etiquetas reales de los datos (para evaluación).

Returns:

None: Los resultados se guardan en un archivo Excel y se muestra una gráfica.

clustering.**kmeans**(*x*, *y*)

Aplica el algoritmo k-Means para encontrar el número óptimo de clústers utilizando las métricas de inercia y coeficiente de silueta. Guarda los resultados y visualiza los clústers con t-SNE.

Args:

x (ndarray): Datos de entrada para el clustering. y (array-like): Etiquetas reales de los datos (para evaluación).

Returns:

None: Los resultados se guardan en un archivo Excel y se muestra una gráfica.

CAPÍTULO 3

Funciones de Reducción de Dimensionalidad

Modelos.**calculo_tiempos**(*k*, *X_scaled*, *y*, *n_componentes*, *reductor*, *modelo*)

Realiza validación cruzada estratificada con un modelo XGBoost y calcula métricas de rendimiento.

Args:

k (int): Número de particiones para la validación cruzada. X_scaled(Dataframe): Lista de características extraídas de las imágenes. y (series): Lista con las etiquetas de las imágenes. n_componentes (int): Número de componentes principales a utilizar. reductor (str): Método de reducción de dimensionalidad ("PCA", "ICA" o ninguno). modelo (str): Tipo de modelo a utilizar ("rfc" o "knn").

Returns:

K_metrics (dict): Diccionario con las métricas de rendimiento.

Modelos.**metricas_ann**(*kfold*, *X_scaled*, *y*, *tiempo_secuencial*, *tiempo_multihilo*, *tiempo_multiproceso*)

Realiza validación cruzada estratificada con un modelo ANN y calcula métricas de rendimiento.

Args:

kfold (StratifiedKFold): Objeto de validación cruzada estratificada. X_scaled (DataFrame): Matriz de datos escalados que serán la entrada al modelo. y (Series): Vector de las variables de salida del modelo. tiempo_secuencial (float): Tiempo de ejecución secuencial. tiempo_multihilo (float): Tiempo de ejecución multihilo. tiempo_multiproceso (float): Tiempo de ejecución multiproceso.

Returns:

K_metrics (dict): Diccionario con las métricas de rendimiento.

Modelos.**metricas_rfc_knn**(*kfold*, *X_scaled*, *y*, *tiempo_secuencial*, *tiempo_multihilo*, *tiempo_multiproceso*, *tiempo_n_jobs*, *metodo*)

Realiza validación cruzada estratificada con un modelo RFC o KNN y calcula métricas de rendimiento.

Args:

kfold (StratifiedKFold): Objeto de validación cruzada estratificada. X_scaled (DataFrame): Matriz de datos escalados que serán la entrada al modelo. y (Series): Vector de las variables de salida del modelo. tiempo_secuencial (float): Tiempo de ejecución secuencial. tiempo_multihilo (float): Tiempo de ejecución multihilo. tiempo_multiproceso (float): Tiempo de ejecución multiproceso. tiempo_n_jobs (float): Tiempo de ejecución con n_jobs. metodo (str): Método a utilizar ("RFC" o "KNN").

Returns:

K_metrics (dict): Diccionario con las métricas de rendimiento.

Modelos.**model**(*X_scaled*, *y*, *n_componentes*, *reductor*, *modelo*)

Entrena y evalúa un modelo XGBoost utilizando características escaladas.

Args:

X_scaled(Dataframe): Matriz de datos escalados que seran la entrada al modelo. y(Series): Vector de las variables de salida del modelo.

entrenar_modelos.**define_model_ann**(*caracteristicas*, *num_clases*)

Define un modelo de red neuronal con arquitectura ANN para clasificación multiclase.

Args:

caracteristicas (int): Número de características de entrada. num_clases (int): Número de clases para la clasificación.

Returns:

model (Sequential): Modelo ANN compilado.

entrenar_modelos.**entrenar_modelo_ann**(*X_scaled*, *train_idx*, *val_idx*, *y*, *mode*)

Función que entrena el modelo ANN utilizando diferentes métodos (secuencial, multihilo, multiproceso).

Args:

X_scaled (array): Datos de entrada escalados. train_idx (array): Índices de entrenamiento. val_idx (array): Índices de validación. y (array): Etiquetas de clase. mode (str): Modo de entrenamiento (“secuencial”, “multihilo”, “multiproceso”, “n_jobs”).

Returns:

model (Sequential): Modelo entrenado. x_val (array): Datos de validación. y_val (array): Etiquetas de validación. history (History): Historial del entrenamiento.

entrenar_modelos.**entrenar_modelo_knn**(*X_scaled*, *train_idx*, *val_idx*, *y*, *mode*)

Función que entrena el modelo K-Nearest Neighbors utilizando diferentes métodos (secuencial, multihilo, multiproceso).

Args:

X_scaled (array): Datos de entrada escalados. train_idx (array): Índices de entrenamiento. val_idx (array): Índices de validación. y (array): Etiquetas de clase. mode (str): Modo de entrenamiento (“secuencial”, “multihilo”, “multiproceso”, “n_jobs”).

Returns:

modelo_fold (KNeighborsClassifier): Modelo entrenado. x_val (array): Datos de validación. y_val (array): Etiquetas de validación.

entrenar_modelos.**entrenar_modelo_rfc**(*X_scaled*, *train_idx*, *val_idx*, *y*, *mode*)

Función que entrena el modelo de Random Forest utilizando diferentes métodos (secuencial, multihilo, multiproceso).

Args:

X_scaled (array): Datos de entrada escalados. train_idx (array): Índices de entrenamiento. val_idx (array): Índices de validación. y (array): Etiquetas de clase. mode (str): Modo de entrenamiento (“secuencial”, “multihilo”, “multiproceso”, “n_jobs”).

Returns:

modelo_fold (RandomForestClassifier): Modelo entrenado. x_val (array): Datos de validación. y_val (array): Etiquetas de validación.

entrenar_modelos.**param_KNN**()

Función que define los parámetros del modelo KNN.

Returns:
parametros_KNN (dict): Parámetros del modelo KNN.

entrenar_modelos.**param_RFC**()

Función que define los parámetros del modelo Random Forest.

Returns:
parametros_RFC (dict): Parámetros del modelo Random Forest.

Seleccion_ejecucion.**multihilo**(*X_scaled*, *y*, *kfold*, *metodo*)

Entrena un modelo de clasificación secuencialmente. Según el valor metodo llamará a la función de entrenamiento correspondiente. Args:

X_scaled (np.ndarray): Datos de entrada escalados. y (np.ndarray): Etiquetas de los datos. kfold (StratifiedKFold): Objeto de validación cruzada estratificada. metodo (str): Método de entrenamiento a utilizar («RFC», «KNN», «ANN»).

Seleccion_ejecucion.**multiproceso**(*X_scaled*, *y*, *kfold*, *metodo*)

Procede a realizar un entrenamiento secuencial del modelo. Según el valor metodo llamará a la función de entrenamiento correspondiente.

Args:
X_scaled (np.ndarray): Datos de entrada escalados. y (np.ndarray): Etiquetas de los datos. kfold (StratifiedKFold): Objeto de validación cruzada estratificada. metodo (str): Método de entrenamiento a utilizar («RFC», «KNN», «ANN»).

Seleccion_ejecucion.**n_jobs**(*X_scaled*, *y*, *kfold*, *metodo*)

Entrena un modelo de clasificación utilizando múltiples trabajos en paralelo. Solo se utiliza para el método «RFC» o «KNN».

Args:
X_scaled (np.ndarray): Datos de entrada escalados. y (np.ndarray): Etiquetas de los datos. kfold (StratifiedKFold): Objeto de validación cruzada estratificada. metodo (str): Método de entrenamiento a utilizar («RFC», «KNN»).

Seleccion_ejecucion.**single**(*X_scaled*, *y*, *kfold*, *metodo*)

Realiza el entrenamiento del modelo de manera secuencial. Según el valor metodo llamará a la función de entrenamiento correspondiente. Args:

X_scaled (np.ndarray): Datos de entrada escalados. y (np.ndarray): Etiquetas de los datos. kfold (StratifiedKFold): Objeto de validación cruzada estratificada. metodo (str): Método de entrenamiento a utilizar («RFC», «KNN», «ANN»).

CAPÍTULO 4

Funciones Visualizacion

tsne.**tsne**(*x*, *cluster_labels*, *metodo*)

Representa el conjunto de datos en 2D aplicando la reducción de dimensionalidad con t-SNE.

Args:

x (ndarray): Datos originales de entrada. cluster_labels (array-like): Etiquetas de cada dato. metodo (str): Nombre del método de clustering utilizado (para el título del gráfico).

Returns:

None: Muestra un gráfico con la representación t-SNE de los clústers.

leer_metricas.**get_data**(*excel_data*)

Extrae las métricas almacenadas en el diccionario generado por *read_excel_file*, separándolas en distintos DataFrames.

Parámetros

excel_data (*dict[str, pandas.DataFrame]*) – Diccionario con los datos del modelo.

Devuelve

- **k_metrics** (*pandas.DataFrame*): Métricas generales del modelo por fold.
- **y_test** (*pandas.DataFrame*): Clases reales de cada imagen.
- **y_pred** (*pandas.DataFrame*): Clases predichas por el modelo.
- **histories** (*pandas.DataFrame*): Evolución del accuracy y pérdida (si está disponible).
- **roc** (*pandas.DataFrame*): Datos para la curva ROC y AUC (si está disponible).

Tipo del valor devuelto

tuple[pandas.DataFrame, pandas.DataFrame, pandas.DataFrame, pandas.DataFrame, pandas.DataFrame]

leer_metricas.**plot_accuracy_evolution**(*histories*)

Muestra la evolución del accuracy en entrenamiento y validación a lo largo de las iteraciones.

Parámetros
histories (*pandas.DataFrame*) – DataFrame con la evolución del accuracy y la pérdida en entrenamiento y validación.

Devuelve
No retorna ningún valor, solo muestra la gráfica de evolución del accuracy.

Tipo del valor devuelto
None

leer_metricas.**plot_classes_roc_curves**(*roc*)

Genera y muestra las curvas ROC individuales para cada clase.

Parámetros
roc (*pandas.DataFrame*) – DataFrame con los datos de la curva ROC por clase.

Devuelve
No retorna ningún valor, solo muestra las curvas ROC por clase.

Tipo del valor devuelto
None

leer_metricas.**plot_confusion_matrix**(*y_test*, *y_pred*)

Genera y muestra una matriz de confusión para cada fold de validación.

Parámetros

- **y_test** (*pandas.DataFrame*) – DataFrame con las clases reales.
- **y_pred** (*pandas.DataFrame*) – DataFrame con las clases predichas.

Devuelve
No retorna ningún valor, solo muestra la matriz de confusión.

Tipo del valor devuelto
None

leer_metricas.**plot_metrics**(*excel_file*)

Genera y muestra visualizaciones clave de las métricas obtenidas durante el entrenamiento de un modelo, incluyendo:

- Matriz de confusión para evaluar el desempeño del modelo en la clasificación.
- Curvas ROC global y por clase para analizar la capacidad de discriminación del modelo.
- Evolución del accuracy en entrenamiento y validación a lo largo de las iteraciones.

Parámetros
excel_file (*str*) – Ruta del archivo Excel que contiene los resultados del modelo.

Devuelve
No retorna ningún valor, solo genera y muestra las visualizaciones correspondientes.

Tipo del valor devuelto
None

leer_metricas.**plot_roc_curves**(*roc*)

Genera y muestra la curva ROC global junto con el área bajo la curva (AUC).

Parámetros
roc (*pandas.DataFrame*) – DataFrame con los datos para la curva ROC global.

Devuelve
No retorna ningún valor, solo muestra la curva ROC.

Tipo del valor devuelto
None

leer_metricas.**read_excel_file**(*excel_file*)

Lee un archivo Excel y devuelve su contenido en un diccionario, donde cada clave representa el nombre de una hoja y su valor correspondiente es un DataFrame con los datos de dicha hoja.

Parámetros
excel_file (*str*) – Ruta del archivo Excel a leer.

Devuelve
Diccionario con los datos de cada hoja en formato DataFrame.

Tipo del valor devuelto
dict[str, pandas.DataFrame]

Matriz_correlacion.**Variables_correlacion**(*X*, *x_pca*, *y*)

Función que calcula la matriz de correlación entre las variables originales y la variable de salida,

y entre las variables transformadas por PCA y la variable de salida. Se muestran las matrices de correlación en dos gráficos diferentes.

Args:
X (Dataframe): Datos no normalizados. x_pca (ndarray): Matriz de características transformadas por PCA. y(Series): Vector de las variables de salida del modelo.

Principales_caracteristicas.**pca**(*x_scaled*)

Realiza PCA sobre los datos escalados y visualiza la varianza explicada por cada componente principal.

Args: x_scaled (DataFrame): Datos escalados.

Returns:
x_pca (ndarray): Datos que se enviarán al modelo.

CAPÍTULO 5

Indices y tablas

- genindex
- modindex
- search

Índice de Módulos Python

Índice

P

R

S

T

V