Apéndice A

Manual de Usuario

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

A.1. Derechos de autor

A.2. Prefacio

Este programa está implementado en Python 3, software de uso libre y gratuito en el marco del proyecto de fin de carrera de Ingeniería Eléctrica y como parte de un proyecto madre denominado BOCOSUR. El programa AUTOBOL tiene 5 funciones implementadas que permiten predecir sobre videos, extraer características para clasificación manual, crear máscara para estaciones nuevas o modificarlas y por último volver a entrenar el modelo de clasificación.

El programa AUTOBOL cuenta con 4 archivos principales y necesarios para su ejecución:

- `autobol.py`: Es el módulo principal con el que se accede a las distintas funciones, en este archivo se definen los requerimientos para ejecutar cada modo implementado.
- `algoritmos.py`: Donde se encuentran todos los algoritmos relacionados a la extracción de características, crear máscara y clasificación.
- `abyo.py`: Módulo que tiene como objetivo separar el input/output de los demás archivos y tenerlos en un mismo lugar.
- `functions.py`: En este archivo se definen los cuatro modos de funcionamiento.

A.3. Propósito

Este documento tiene como objetivo facilitar la tarea de conocimiento, uso y aprendizaje del sistema desarrollado para detección y clasificación de bólidos a partir de cámaras all-sky. Contiene información acerca de todas las operaciones básicas que el programa ofrece, así como imágenes útiles para el seguimiento de la explicación y su uso.

A.4. Audiencia

Este documento está dirigido a personal técnico y operativo, encargados de ejecutar el software desarrollado.

Si bien su operativa es simple se recomienda conocimientos básicos de líneas de comando en consola. Aunque no es un requerimiento excluyente porque se detallaran sus funcionalidades y opciones a continuación.

A.5. Alcance

Se pretende detallar las funciones de forma que cualquier persona encargada del proyecto así como colaboradores puedan manipular y usar las funcionalidades del programa sin conocer los detalles del funcionamiento. No es parte del alcance, explicar como se implementó, estructuró, desarrolló, etc, el software propiamente dicho. Por más información sobre la implementación se puede visitar el github desarrollado (*Link al Enlace en GitHub*).

A.6. Guía de uso

A.6.1. Consola

Lo primero que se debe hacer es abrir una ventana de línea de comandos.

- En Windows:

  Para la versión de Windows en idioma Español se deberá seguir los siguientes pasos:

  - Pulsa a la vez la tecla de Windows (normalmente tiene el logotipo de Microsoft, está ubicada cerca de la tecla `Alt`) y la letra `R`.
  - Se abrirá una pequeña ventana con el título de "Ejecutar". Dentro del rectángulo para introducir texto que se desplegará, escribir `cmd` y pulsar sobre el botón de "Aceptar".
  - La ventana de comandos `cmd.exe` estará iniciada.

  Otra forma es utilizar el buscador del escritorio con las palabras `cmd` o `símbolo de sistema`.

- En Linux:

  Hay varios métodos para acceder a una ventana de línea de comandos: Uno de ellos es pulsa a la vez la siguiente combinación de teclas `Ctrl + Alt + T`.

  Otra de ellas es buscar usando el tablero Ubuntu.

A.6.2. Directorio de trabajo

Lo siguiente a realizar es ir al directorio de trabajo. En la línea de comandos ubicarse en la carpeta donde se tenga el archivo `autobol.py` y sus archivos relacionados.

- Uno de los comandos más esenciales de la consola tanto en Windows como en Linux es: `cd`. Función para cambiar de directorio.
  Sintaxis: `cd <RutaDirectorio>` (donde `RutaDirectorio` es la ubicación a la cual se quiere mover) o `cd..` para salir de una carpeta e ir al nivel superior o carpeta donde estaba alojada.

- Comando para obtener el contenido del directorio o carpeta donde uno se encuentra, mostrando todas las subcarpetas o archivos que posee. Con este comando se podrá saber si el archivo que se busca se encuentra ahí o a qué subcarpeta navegar.

  - En Windows: `dir`
  - En Linux: `ls`

A.6.3. Módulos de trabajo

En el directorio de trabajo se debe contar con las siguientes carpetas o archivos:

- Carpeta `Mascaras`: Donde se guardaran las mascaras ya guardadas o a crear/modificar.
- Carpeta `<Directorio_Salida>`: Directorio donde se quiere guardar los archivos de salida.
- Archivo `abio.py`.
- Archivo `algoritmos.py`.
- Archivo `functions.py`.
- Un modelo es solo necesario si se desea predecir, en este caso es `model.bin`.

Seguido de ejecutar en consola el archivo `autobol` con `python` y teniendo en cuenta los parámetros de cada módulo de trabajo, que se describirán a continuación:

Train

El módulo `train` es utilizado para reentrenar el modelo `XGBoost`, para mejorar la eficiencia del algoritmo o para entrenar sobre otra base de datos.

Los requerimientos son:

- `-m train`: Se entra a la función `train`.
- `-i <Directorio_Videos>`: Ruta donde se encuentra el archivo .csv con los datos de entrenamiento.

  Observación 1: El archivo `.csv` corresponde al archivo `Clasificacion.csv`, salida del modo `predict verbose`, luego de que haya sido editado por el usuario con las etiquetas 0 o 1.
- `-o <Directorio_Salida>`: Directorio donde se quiere guardar el archivo de salida. Tendrá el nombre por defecto `model_umbral.bin`.

  Siendo `_umbral` el umbral óptimo encontrado donde maximiza el `f2 score` del modelo.

  Observación 2: Este valor es el recomendado a utilizar en el modo `predict` en el parámetro `-u <umbral>`, aunque se puede utilizar cualquiera.
- `-t <%_Test>`: Indica el porcentaje de los datos que se utilizan para `test`, permite evaluar el modelo en datos que no se usaron para el entrenamiento. Donde `<%_Test>` es el índice porcentual, $\%\_Test \in [0, 1]$.

Un ejemplo de ejecución del módulo `train`, se puede ver en la Figura A.1. Donde la línea de comandos ingresada fue:
`python autobol.py -m train -i Clasificacion.csv -o salida-prueba -t 0.15`
Al presionar enter, el software devuelve un mensaje con la cantidad de bólidos

Figura A.1: Ejemplo de ejecución del módulo test.

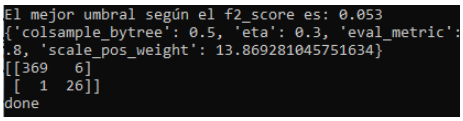


Figura A.2: Salida del ejemplo de ejecución del módulo train con parámetros de entrada vistos.

y no bólidos en los conjuntos de `entrenamiento` y `test`. Preguntará si se desea continuar, esperando una respuesta válida `(Y, y, N o n)`. En caso afirmativo se ingresa `Y` o `y`, y se empezará a ejecutar el entrenamiento. De lo contrario se ingresa `N` o `n` y saldrá del programa.

Observación 3: El módulo de entrenamiento puede llegar a demorar unos minutos. Para el ejemplo mencionado que cuenta con 2122 videos etiquetados como "No Bólido" y unos 153 videos como "Bólido" demoró unos 19 minutos con 30 segundos.

La salida en consola se puede ver en la Figura A.2. Donde se devolverá el valor del mejor umbral que optimice el modelo, junto con la matriz de confusión del modelo aplicado al conjunto de `test`. Estos y otros datos útiles quedarán guardados en un archivo `.txt` en la misma ruta.

La matriz de confusión está constituida como:

$$\begin{bmatrix} \text{verdaderos negativos} & \text{falsos positivos} \\ \text{falsos negativos} & \text{verdaderos positivos} \end{bmatrix}$$

Predict

El modulo `predict`, tiene como objetivo clasificar los videos que se encuentran en la ruta especificada entre las categorías "Bólido" y "No Bólido". Los requerimientos son:

- `-m predict`: Se entra a la función `predict`.
- `-i <Directorio_Videos>`: Directorio donde se encuentra/n el/los video/s que se desea/n predecir.
- `-o <Directorio_Salida>`: Directorio donde se quiere guardar el archivo de salida, en este caso un `.csv` con el formato:

  Encabezado: Nombre_Archivo, Probabilidad_De_Ser_Bólido, Etiqueta
  Fila 1: Video1, Probabilidad1, Etiqueta1

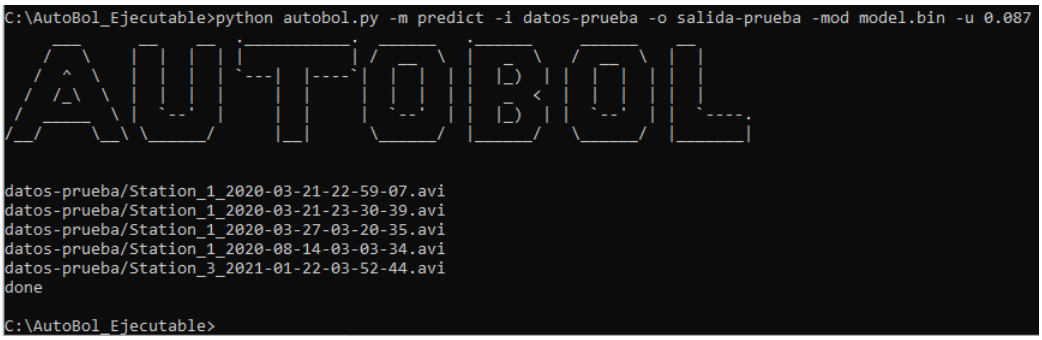


Figura A.3: Ejemplo de ejecución del módulo predict.

Fila 2: Video2, Probabilidad2, Etiqueta2
Fila 3: Video3, Probabilidad3, Etiqueta3

Nombre: Referido a la raíz común de los archivos.
Probabilidad: Valor p absoluto porcentual, $p \in [0, 100]\,\%$.

Etiqueta:

$$Etiqueta = \begin{cases} \text{Bólido} & si \quad p \geq u \\ \text{No Bólido} & si \quad p < u \end{cases} \tag{A.1}$$

Donde u es el umbral que se pasa como parámetro.

Observación 4: Si el archivo `.csv` no está creado en `<Directorio_Salida>`, el software lo crea con el encabezado y formato mencionado, con el nombre `Predicciones.csv`.

Observación 5: Cada vez que se ejecuta el módulo `predict`, se concatenan en el archivo de salida al final del mismo n filas, donde n corresponde a la cantidad de videos en `<Directorio_Videos>`.

- `-mod <Modelo.bin>`: Modelo que se quiere usara para predecir la clasificación de los videos en `<Directorio_Videos>`.
- `-u <Umbral>`: Umbral de decisión u usado para determinar la etiqueta correspondiente como se explicó en A.1. Se recomienda usar el umbral que esta referenciado en el `.txt` del modelo.

Un ejemplo de ejecución del módulo `predict`, se puede ver en la Figura A.3. Donde la línea de comandos ingresada fue:
`python autobol.py -m predict -i datos-prueba -o salida-prueba -mod model.bin -u 0.087`, cuya salida es el archivo `Predicciones.csv` ubicado en `<Directorio_Salida>`.

Predict Verbose

El módulo `predict verbose`, tiene como objetivo obtener las características de los videos indicados en una ruta especificada, de esta manera permite crear nuevos datos de entrenamiento.

	A	B	C
1	Nombre Archivo	Probabilidad de Bólido	Etiqueta
2	datos-prueba/Station_1_2020-03-21-22-59-07.avi	99,96	Bólido
3	datos-prueba/Station_1_2020-03-21-23-30-39.avi	98,78	Bólido
4	datos-prueba/Station_1_2020-03-27-03-20-35.avi	99,98	Bólido
5	datos-prueba/Station_1_2020-08-14-03-03-34.avi	99,99	Bólido
6	datos-prueba/Station_3_2021-01-22-03-52-44.avi	0	No Bólido

Figura A.4: Salida del ejemplo de ejecución del módulo predict con parámetros de entrada vistos.

Los requerimientos son:

- `-m predict -verbose`: Se entra a la función `predict verbose`.
- `-i <Directorio_Videos>`: Directorio donde se encuentran los videos que se desean predecir.
- `-o <Directorio_Salida>`: Directorio donde se quiere guardar el archivo de salida, en este caso un `.csv` con el formato:

  Encabezado: Nombre_Archivo, Característica1,.., Característica20, Clasificación
  Fila 1: Video_1, Característica1_1,..., Característica1_20, NaN
  Fila 2: Video_2, Característica2_1,..., Característica2_20, NaN
  Fila 3: Video_3, Característica3_1,..., Característica3_20, NaN

  Nombre: Referido a la raíz común de los archivos.
  Características: Valor real que abstrae una característica del video.
  Clasificación: Valor por defecto `NaN`.

  Luego el usuario debe editar el archivo y sustituir el `NaN` con el siguiente criterio:

$$Clasificacin = \begin{cases} 1 & si \quad \text{Bólido} \\ 0 & si \quad \text{No Bólido} \end{cases} \tag{A.2}$$

  Observación 6: Si el archivo `.csv` no está creado en `<Directorio_Salida>`, el programa lo crea con el encabezado y formato mencionado, con el nombre `Clasificacion.csv`.

  Observación 7: Cada vez que se ejecuta el módulo `predict verbose` se concatenan en el archivo de salida al final del mismo n filas, donde n corresponde a la cantidad de videos en `<Directorio_Videos>`.

Un ejemplo de ejecución del módulo `predict verbose`, se puede ver en la Figura A.5. Donde la línea de comandos ingresada fue:
`python autobol.py -m predict -verbose -i datos-prueba -o salida-prueba`, cuya salida es el archivo `Clasificacion.csv` ubicado en `<Directorio_Salida>`.

Predict Localize

El módulo `predict localize`, tiene como objetivo brindar una retroalimentación visual al usuario. A la vez de predecir, también devuelve

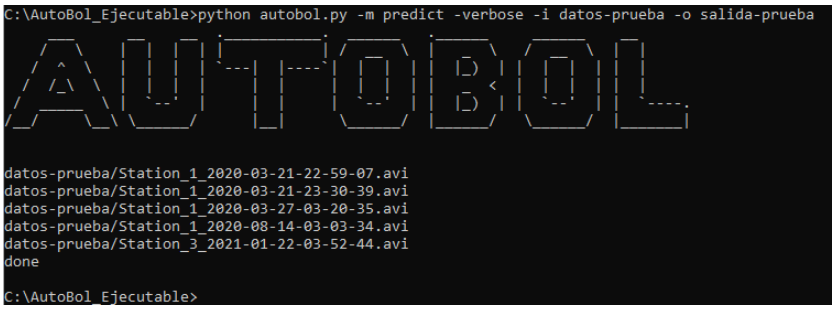


Figura A.5: Ejemplo de ejecución del módulo predict verbose.

	A	B	C	D	T	U	V
1	Video	velocidad media	mean_intensidad	N	loss	recorrido	Clasificación
2	datos-prueba/Station_1_2020-03-21-22-59-07.avi	45,948350019	5104,1621312626	40	350,860844151	57,1054429673	NaN
3	datos-prueba/Station_1_2020-03-21-23-30-39.avi	20,527577229	1148,775	22	97,5008567932	12,0274772255	NaN
4	datos-prueba/Station_1_2020-03-27-03-20-35.avi	41,4488000575	631,430952381	21	105,4922961999	25,3835826398	NaN
5	datos-prueba/Station_1_2020-08-14-03-03-34.avi	39,1987445908	957,4983333333	60	292,2970702623	74,8302540869	NaN
6	datos-prueba/Station_3_2021-01-22-03-52-44.avi	53,9508676283	50850,4440789474	228	338,3185222315	210,7419988516	NaN

Figura A.6: Salida del ejemplo de ejecución del módulo predict verbose con parámetros de entrada vistos.

una imagen con el evento identificado encerrado en un cuadrado, esto permite al usuario ver qué detecta el algoritmo.

Los requerimientos son:

- `-m predict -localize`: Se entra a la función `predict localize`.
- `-i <Directorio_Videos>`: Directorio donde se encuentran los videos que se desean predecir y devolver la imagen.
- `-o <Directorio_Salida>`: Directorio donde se quiere guardar el archivo de salida y la imagen .png.

Ejemplo de ejecución:
`python autobol.py -m predict -u 0.053 -o salida -i datos-prueba -mod model_0.053.bin -localize`, cuya salida es el archivo `Clasificacion.csv` y una imagen ubicados en `<Directorio_Salida>`. jj

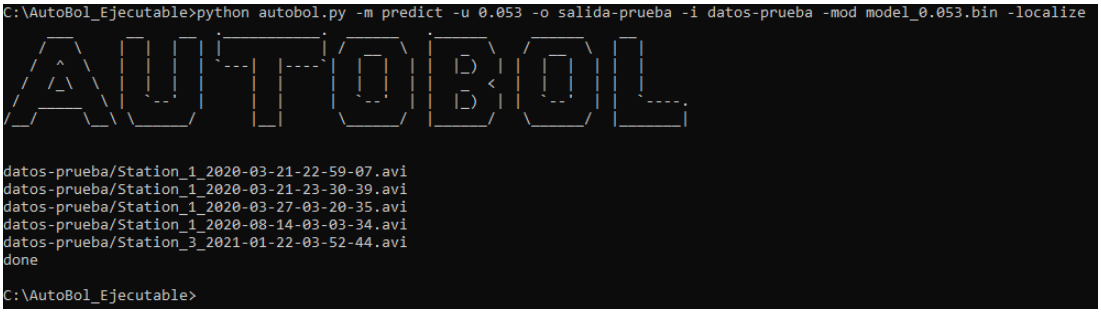


Figura A.7: Ejemplo de ejecución del módulo predict localize.

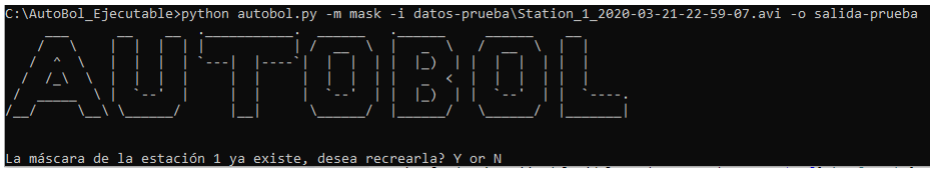


Figura A.8: Ejemplo de ejecución del módulo mask.

Mask

El módulo `mask`, se utiliza para crear la máscara de una estación. Este se ejecuta automáticamente en caso de que se intente procesar un video de una estación que no tenga máscara. Por ejemplo, al ejecutar el módulo `predict` con un video de una estación X, donde no se tiene el archivo `Mascara_Station_X`. También se puede ejecutar este módulo por intervención del usuario si quiere o necesita crear una nueva máscara de una estación ya existente. Los requerimientos para ejecutarlo son:

- `-m mask:` Se entra a la función `mask`.
- `-i <Directorio_Video>`: Directorio donde se encuentra el video de la estación de interés que se quiere crear la máscara.

  El formato a seguir debe de ser:

  `Station_X_Año-Mes-Día-Hora-Minutos-Segundos`
  Donde X es el número de la estación.
  Ejemplo: `Station_1_2020-03-21-22-59-07`

  Recordar las estaciones existentes hasta el momento:

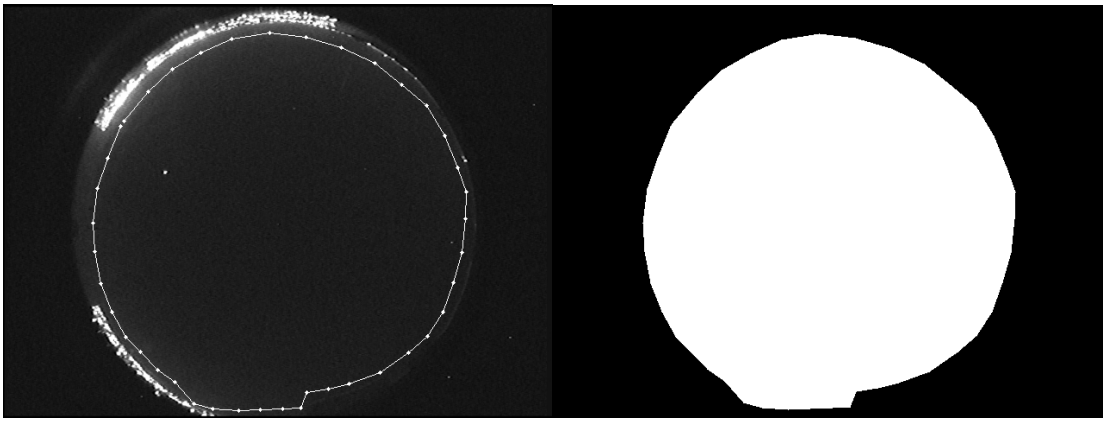
  - Estación 1: Montevideo – Facultad de Ciencias
  - Estación 2: Maldonado –Liceo de San Carlos
  - Estación 3: Florida – Liceo de Casupá

- `-o <Directorio_Salida>`: Directorio donde se guardan las máscaras. Está predefinido a la carpeta Mascaras y se guarda con el nombre `Mascara_Station_E.png`.

Un ejemplo de ejecución del módulo `mask`, se puede ver en la Figura A.8, donde la línea de comandos ingresada fue:
`python autobol.py -m mask -i datos-prueba\Station_1_2020-03-21-22-59-07.avi -o salida-prueba`

Al presionar enter, el software devolverá el mensaje: `La máscara de la estación X ya existe, desea recrearla? Y or N`.

Esperando una respuesta válida `(Y, y, N o n)` por un máximo de 3 intentos. Si no se ingresa una respuesta válida en los primeros 3 intentos o se ingresa `N` o `n`, el software se cerrará y habrá que ejecutar el módulo de nuevo.



(a) Frame con máscara delimitada punto a punto. (b) Máscara generada.

Figura A.9: Ejemplo de ejecución del módulo mask para video de la estación 1.

Si se presiona `Y` o `y`, se abrirá una ventana como se puede ver en la Figura A.9 con un frame aleatorio del video que fue utilizado como input, es decir `<Directorio_Video>`. Una vez que nos aparece el frame, se tiene que ir haciendo click sobre la imagen para delimitar la máscara sobre el horizonte de la toma. Los únicos botones que funcionan en la ventana que está el frame son los siguientes:

- Botón izquierdo del mouse: Agregar punto.
- Botón derecho del mouse: Quitar último punto agregado.
- Tecla `Esc` del teclado: Salir del módulo y guardar la nueva máscara.

Observación 8: Cuando se está delimitando la máscara se pueden ingresar tantos puntos como sean necesarios, se recomienda un polígono convexo.

Observación 9: La cantidad mínima de puntos válidos para crear la máscara es de 3. Si se ingresan una cantidad menor de puntos y se presiona `Esc`, el software saldrá del módulo con el mensaje: `Ingrese una cantidad de puntos superior a 2`.

Observación 10: Se recomienda actualizar la máscara cada vez que se modifique el lugar de la cámara all-sky o aparezcan elementos en la periferia o deformaciones de la cámara que agreguen ruido.

A.7. Solución de problemas

A continuación se listan algunos errores conocidos con su solución:

- Si la respuesta de consola al ejecutar cualquier modulo es:
  `usage: autobol.py [-h] -i INDIR -o OUTDIR -m MODE [-u UMBRAL] [-t TEST_SZ] [-mod MODEL] [-verbose] autobol.py: error: the following arguments are required: Parámetro`

  Con Parámetro:

1. `-i --indir`
2. `-o --outdir`
3. `-m --mode`

Solución: Ejecute nuevamente el módulo agregándole el parámetro faltante:

1. Directorio donde se encuentra/n el/los video/s.
2. Directorio donde se va a guardar la salida.
3. El modulo a utilizar: `predict`, `predict verbose`, `predict localize`, `train` o `mask`.

- Si la respuesta de consola al ejecutar el módulo `predict` es:
  `PermissionError: [Errno 13] Permission denied: 'salida-prueba\\Predicciones.csv'`
  Solución: El archivo `Predicciones.csv` se encuentra abierto, ciérrelo y vuelva a intentarlo.
- Si la respuesta de consola al ejecutar el módulo `predict verbose` es:
  `PermissionError: [Errno 13] Permission denied: 'salida-prueba\\Clasificacion.csv'`
  Solución: El archivo `Clasificacion.csv` se encuentra abierto, ciérrelo y vuelva a intentarlo.

A.8. Preguntas Frecuentes

- ¿En qué sistema operativo se ejecuta el software?
  Se puede ejecutar tanto en Windows como en Linux, teniendo los siguientes requisitos:

  - certifi==2021.10.8
  - charset-normalizer==2.0.7
  - colorama==0.4.4
  - cycler==0.11.0
  - decorator==4.4.2
  - fonttools==4.28.1
  - idna==3.3
  - imageio==2.11.1
  - imageio-ffmpeg==0.4.5
  - joblib==1.1.0
  - kiwisolver==1.3.2
  - matplotlib==3.5.0
  - moviepy==1.0.3
  - networkx==2.6.3
  - numpy==1.21.4
  - opencv-python==4.5.4.58
  - packaging==21.3
  - pandas==1.3.4
  - Pillow==8.4.0
  - proglog==0.1.9
  - pyfiglet==0.8.post1
  - pyparsing==3.0.6
  - python-dateutil==2.8.2
  - python-math==0.0.1
  - pytz==2021.3
  - PyWavelets==1.2.0
  - requests==2.26.0
  - scikit-image==0.18.3
  - scikit-learn==1.0.1
  - scipy==1.7.2
  - setuptools-scm==6.3.2
  - six==1.16.0
  - sklearn==0.0
  - threadpoolctl==3.0.0
  - tifffile==2021.11.2
  - tomli==1.2.2
  - tqdm==4.62.3
  - urllib3==1.26.7
  - xgboost==1.5.0

- ¿Cómo salgo del programa?
  Desde la consola se debe de apretar la combinación de teclas: `Ctrl + C`, esto termina el proceso que se esté ejecutando, útil para recuperar el control de la consola y luego cerrarla.

- ¿Se puede cambiar la carpeta de las máscaras?
  No se puede, la ubicación de las máscaras esta predefinida en la carpeta `Mascaras` ubicada en la raíz junto a los archivos código del software. Para cambiarla hay que modificar el código.
- ¿Es necesario que los parámetros de los módulos especificados en las secciones A.6.3, A.6.3, A.6.3, A.6.3 y A.6.3 tengan el orden descrito?
  No es necesario que se respete el orden, es suficiente con que estén todos los parámetros mencionados en el módulo correspondiente.
- ¿Cómo creo un nuevo modelo?

  1. Disponer de un dataset de videos lo suficientemente grande en una carpeta cuyos nombres tengan el formato:
     `Station_E_Año-Mes-Día-Hora-Minutos-Segundos`.
  2. Ejecutar el modo `predict verbose` con esta carpeta de videos, del cual se obtendrá un archivo `Clasificacion.csv`.
  3. Editar la última columna del archivo `Clasificacion.csv` con el criterio descrito en A.2.
  4. Ejecutar el modo `train` pasandole el archivo `Clasificacion.csv` y se obtendrá un nuevo modelo con el umbral recomendado.

En el caso de que usted quiera conocer más acerca de este proyecto o exista alguna duda, es posible contactarse con el equipo de AutoBol y Bocosur a través de los siguientes e-mail de contacto: juanpballestrino@gmail.com, cdeandraya@gmail.com , cuviedo1@gmail.com.