Operaciones de Machine Learning

Proyecto 1
Nivel 0 - Nivel 1

Pontificia Universidad Javeriana
Cristian Javier Diaz Alvarez

19 de febrero de 2025

1. Descripción

Este proyecto busca evaluar la capacidad de **crear** un ambiente de desarrollo de machine learning en el cual sea posible la **ingesta**, **validación** y **transformación** de datos, demostrando capacidad de **versionar código** y **ambiente de desarrollo**.

Específicamente, el estudiante construirá un ambiente de desarrollo que permita realizar la canalización de datos con los siguientes etapas:

- Selección de características
- Ingesta del conjunto de datos
- Generación de las estadísticas del conjunto de datos.
- Creación de un esquema según el conocimiento del dominio.
- Creación de entornos de esquema
- Visualización de las anomalías del conjunto de datos
- Preprocesado, transformación e ingeniería de sus características
- Seguimiento de la procedencia de su flujo de datos utilizando metadatos de ML

Se espera que el ambiente de desarrollo sea construido usando Docker y esté especificado mediante docker-compose. Este archivo de configuración debe estar disponible en el repositorio de código. Este ambiente debe permitir la ejecución de un Jupyter Notebook en donde se describa mediante comentarios y Markdown el código que permita dar seguimiento a cada uno de los siguientes puntos de este proyecto. El trabajo se propone bajo un Dataset y Framework específicos, entorno a los cuales la clase se ha desarrollado. No es obligatorio el uso de ninguno de estos, es decisión del estudiante la definición del dataset a usar y el framework que dará soporte al trabajo, sin embargo, el notebook debe cumplir con cada uno de los puntos expuestos en este documento. Dado el caso que el Framework seleccionado no permita efectuar alguna de las tareas o escenarios expuestos, se debe explicar como se puede suplir cada escenario. (Tip: Usando y explicado otras herramientas)

Se debe versionar el código, para esto se recomienda usar GitHub. En la entrega se tendrá en cuenta, funcionamiento de ambiente de desarrollo **20 %**, correcta ejecución del código **60 %** (distribuido en cada uno de los puntos de este documento), comentarios y explicaciones **10 %**, manejo de repositorios **10 %**.

Tenga en cuenta que, para permitir la reproducibilidad del proyecto, debe proveer un entorno de ejecución aislado (ambiente de desarrollo), con todo las dependencias necesarias y listo para la ejecución. Así como las instrucciones necesarias para su uso.

2. Descripcion del dataset

Se propone utilizar una variante del conjunto de datos Tipo de Cubierta Forestal. Esto se puede utilizar para entrenar un modelo que predice el tipo de cobertura forestal en función de variables cartográficas. Puede leer más sobre el conjunto de datos **original** aquí y describimos las columnas de datos a continuación:

Column Name	Variable Type	Units / Range	Description
Elevation	quantitative	meters	Elevation in meters
Aspect	quantitative	azimuth	Aspect in degrees azimuth
Slope	quantitative	degrees	Slope in degrees
Horizontal_Distance_To_Hydrology	quantitative	meters	Horz Dist to nearest surface water features
Vertical_Distance_To_Hydrology	quantitative	meters	Vert Dist to nearest surface water features
Horizontal_Distance_To_Roadways	quantitative	meters	Horz Dist to nearest roadway
Hillshade_9am	quantitative	0 to 255 index	Hillshade index at 9am, summer solstice
Hillshade_Noon	quantitative	0 to 255 index	Hillshade index at noon, summer soltice
Hillshade_3pm	quantitative	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal_Distance_To_Fire_Points	quantitative	meters	Horz Dist to nearest wildfire ignition points
Wilderness_Area (4 binary columns)	qualitative	0 (absence) or 1 (presence)	Wilderness area designation
Soil_Type (40 binary columns)	qualitative	0 (absence) or 1 (presence)	Soil Type designation
Cover_Type (7 types)	integer	1 to 7	Forest Cover Type designation

Tabla 1: Descripción de variables.

Como puede notar, los datos cualitativos ya han sido codificados usando representación One-hot (por ejemplo, **'Soil_Type'** tiene 40 columnas binarias donde un "1ïndica la presencia de una característica). Para este proyecto usaremos una versión modificada de este conjunto de datos que muestra un formato más crudo. Esto le permitirá practicar sus habilidades en el manejo de diferentes tipos de datos. Puede ver el código para preparar el conjunto de datos aquí

2.1. Cargar el Dataset

Utilice las siguientes celdas para cargar el conjunto de datos **modificado** en su espacio de trabajo.

```
## download the dataset
# Directory of the raw data files
_data_root = './data/covertype'

# Path to the raw training data
_data_filepath = os.path.join(_data_root, 'covertype_train.csv')

# Download data
os.makedirs(_data_root, exist_ok=True)
if not os.path.isfile(_data_filepath):
    #https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/
    url = 'https://docs.google.com/uc?export= \
    download&confirm={{VALUE}}&id=1lVF1BCWLH4eXXV_YOJzjR7xZjj-wAGj9'
    r = requests.get(url, allow_redirects=True, stream=True)
    open(_data_filepath, 'wb').write(r.content)
```

Si por algún motivo no puede descargar los datos, o detecta algún inconveniente con esta forma de descarga, intente usar el enlace de manera directa, visite la fuente de datos original enlazada previamente, informe al curso en general para encontrar una solución rápida.

3. Selección de características

Reduzca la cantidad de características para alimentar el modelo. Como se mencionó en clase, esto ayudará a reducir la complejidad de su modelo y ahorrará recursos durante el entrenamiento. Supongamos que ya tiene un modelo de referencia que está entrenado con todas las características y desea ver si al reducir la cantidad de variables generará un mejor modelo. Deberá seleccionar un subconjunto que tenga un gran valor predictivo para la etiqueta (en este caso, el 'Cover_Type').

Al observar los tipos de datos de cada columna y la descripción del conjunto de datos, podrá ver que la mayoría de las funciones son numéricas y solo dos no lo son. Esto debe tenerse en cuenta al seleccionar el subconjunto de características porque las características numéricas y categóricas se califican de manera diferente. Cree un subconjunto de datos que solo contenga las características numéricas para que pueda usarlo en las siguientes secciones.

Utilice los módulos integrados de scikit-learn para realizar selección de características univariate-feature-selection en los atributos numéricos de nuestro conjunto de datos. Recuerde que, primero, debe preparar las características de entrada y de destino:

Luego, utilizará SelectKBest para puntuar cada función de entrada frente a la variable objetivo. Tenga en cuenta la función de puntuación que se va a pasar y asegúrese de que sea adecuada para los valores de entrada (numéricos) y objetivo (categóricos).

	Columns	Retain
0	Elevation	True
1	Aspect	False
2	Slope	True
3	$\text{Horizontal}_{D}istance_{T}o_{H}ydrology$	True
4	$\text{Vertical}_{D}istance_{T}o_{H}ydrology$	True
5	$\text{Horizontal}_{D}istance_{T}o_{R}oadways$	True
6	$\text{Hillshade}_{9}am$	True
7	$\text{Hillshade}_{N}oon$	True
8	$\text{Hillshade}_{3}pm$	False
9	$\text{Horizontal}_{D}istance_{T}o_{F}ire_{P}oints$	True

Tabla 2: Resultado esperado de referencia.

4. Data Pipeline

Con el subconjunto seleccionado de características preparado, ahora puede comenzar a construir la canalización de datos. Esto implica ingerir, validar y transformar sus datos. Use los componentes TFX vistos en clase adicionalmente puede buscarlos aquí en la documentación oficial

4.1. Configurar el contexto interactivo

Como de costumbre, primero configurará el contexto interactivo para que pueda ejecutar manualmente los componentes de canalización desde un cuaderno. Guardará la base de datos sqlite en un directorio predefinido en su espacio de trabajo. No modifique esta ruta porque la necesitará en un ejercicio posterior relacionado con los metadatos de ML.

4.2. Generando Ejemplos

El primer paso en la canalización es ingerir los datos. Con ExampleGen, puede convertir datos sin procesar en TFRecords para un cálculo más rápido en las etapas posteriores de la canalización.

Use **ExampleGen** para ingerir el conjunto de datos que cargamdo anteriormente. Algunas cosas a tener en cuenta:

- La entrada está en formato CSV, por lo que deberá usar el tipo apropiado de **ExampleGen** para manejarla.
- Esta función acepta una ruta de **directorio** a los datos de entrenamiento y no la ruta del archivo CSV en sí.

4.3. Estadísticas

Posteriormente calcule las estadísticas de sus datos. Esto le permitirá observar y analizar las características de sus datos a través de visualizaciones proporcionadas por la biblioteca integrada FACETS.

Utilice StatisticsGen para calcular las estadísticas de los ejemplos de salida de **ExampleGen**.

Una vez que haya cargado la pantalla, puede notar que la columna **ceros** para **Cover_type** estara resaltada en rojo. La visualización nos informa que esto podría ser un problema potencial. Sin embargo, en nuestro caso, sabemos que **Cover_type** tiene un rango de [0, 6], por lo que tener ceros en esta columna es algo que esperamos.

4.4. Inferir el esquema

Deberá crear un esquema para validar los conjuntos de datos entrantes durante el entrenamiento y el servicio. Afortunadamente, TFX le permite inferir un primer borrador de este esquema con el componente SchemaGen.

Utilice **SchemaGen** para inferir un esquema basado en las estadísticas calculadas de **Statistics-Gen**

4.5. Curando el esquema

Podrá ver que el esquema inferido puede capturar los tipos de datos correctamente y también puede mostrar los valores esperados para los datos cualitativos (es decir, cadenas). Sin embargo, aún puede ajustar esto. Por ejemplo, tenemos características en las que esperamos un cierto rango:

- **Hillshade_9am**: 0 to 255
- **Hillshade_Noon**: 0 to 255
- **Slope**: 0 to 90
- **Cover_Type**: 0 to 6

Actualice su esquema para tomar nota de estos para que la canalización pueda detectar si se están alimentando valores no válidos al modelo.

Use TFDV para actualizar el esquema inferido para restringir un rango de valores a las caracteristicas mencionadas anteriormente.
Cosas a tener en cuenta:

- Puede usar tfdv.set_domain() para definir valores aceptables para una función en particular.
- Estos deberían seguir siendo tipos **INT** después de realizar los cambios.

- Declare **Cover_Type**“ como una variable **categórica**. A diferencia de las otras cuatro características, los números enteros del 0 al 6 aquí corresponden a una etiqueta designada y no a una medida cuantitativa. Puede mirar las banderas disponibles para **set_domain()** en el documento oficial para saber cómo configurar esto.

4.6. Entornos de esquema

En el aprendizaje supervisado, entrenamos al modelo para que haga predicciones alimentando un conjunto de características con su etiqueta correspondiente. Por lo tanto, nuestro conjunto de datos de entrenamiento tendrá tanto las características de entrada como la etiqueta, y el esquema está configurado para detectarlas.

Sin embargo, después de entrenar y servir el modelo para la inferencia, los datos entrantes ya no tendrán la etiqueta. Esto presentará problemas al validar los datos utilizando la versión actual del esquema. Usted debe simular un conjunto de datos de servicio obteniendo un subconjunto del conjunto de entrenamiento y soltando la columna de la etiqueta (es decir, **Cover_Type**). Luego, validará este conjunto de datos de publicación utilizando el esquema que seleccionó anteriormente.

Se espera que, la columna que falta este marcada. Para solucionar esto, debe configurar el esquema para detectar cuándo se usa para capacitación o para inferencia/servicio. Puede hacerlo configurando entornos de esquema. Al aplicar esta configuración entre entornos debe pasar la validación sin problema y se deberá guardar este esquema seleccionado en un directorio local para poder importarlo a nuestro pipeline de TFX.

Como verificación, debe mostrar el esquema que acaba de guardar y verificar que contiene los cambios que introducidos. Debe mostrar los rangos en la columna **Domain** y debe haber dos entornos disponibles.

4.7. Genere nuevas estadísticas usando el esquema actualizado

Ahora debe calcular las estadísticas utilizando el esquema que acaba de seleccionar. Sin embargo, recuerde que los componentes TFX interactúan entre sí al obtener información de artefactos del almacén de metadatos. Por lo tanto, primero debe importar el archivo de esquema seleccionado a los metadatos de ML. Lo hará mediante un ImportSchemaGen para crear un artefacto que represente el esquema seleccionado. Cree un artefacto **Schema** que apunte a la ruta del archivo de esquema seleccionado.

Con el artefacto creado, ahora puede usar **StatisticsGen** y pasar un parámetro **schema** para usar el esquema curado. Utilice **StatisticsGen** para calcular las estadísticas con el esquema que actualizó en la sección anterior.

Como resultado el gráfico se verá prácticamente igual que en las ejecuciones anteriores, pero puede ver que el **Cover Type** ahora se encuentra debajo de las características categóricas. Eso mostrara que **StatisticsGen** está usando el esquema actualizado.

4.8. Comprobar anomalías

Ahora debe comprobar si el conjunto de datos tiene alguna anomalía con respecto al esquema. Puede hacerlo fácilmente con el componente ExampleValidator. Compruebe si hay alguna anomalía utilizando **ExampleValidator**. Deberá pasar las estadísticas y el esquema actualizados de las secciones anteriores.

4.9. Ingeniería de características

Ahora procederá a transformar sus características a una forma adecuada para entrenar un modelo. Esto puede incluir varios métodos, como escalar y convertir cadenas en índices de vocabulario. Es importante que estas transformaciones sean consistentes en todos sus datos de entrenamiento y también para los datos de servicio cuando el modelo se implementa para la inferencia. TFX garantiza esto al

generar un gráfico que procesará los datos entrantes durante el entrenamiento y la inferencia. Primero declare las constantes y la función de utilidad que usará para el ejercicio. Recuerde que debe definir **preprocessing_fn** para aplicar transformaciones a las características.

4.10. Función de preprocesamiento

Aquí hay algunos enlaces a los documentos de las funciones que necesitará para completar esta función:

- tft.scale_by_min_max
- tft.scale_to_0_1
- tft.scale_to_z_score
- tft.compute_and_apply_vocabulary
- tft.hash_strings

4.11. Transformar

Utilice el componente de transformación TFX para realizar las transformaciones y generar el gráfico de transformación. Deberá pasar los ejemplos del conjunto de datos, el esquema **curado** y el módulo que contiene la función de preprocesamiento.

Inspeccione algunos ejemplos del conjunto de datos transformado para ver si las transformaciones se realizaron correctamente.

5. Metadatos de aprendizaje automático

TFX usa ML Metadata bajo el capó para mantener registros de los artefactos que usa cada componente. Esto facilita el seguimiento de cómo se ejecuta la canalización para que pueda solucionar problemas si es necesario o desea reproducir los resultados.

En esta sección final, demostrará cómo recorrer este almacén de metadatos para recuperar artefactos relacionados. Esta habilidad es útil cuando desea recordar qué entradas se alimentan a una etapa particular del *pipeline*. Por ejemplo, puede saber dónde ubicar el esquema utilizado para realizar la transformación de características o puede determinar qué conjunto de ejemplos se utilizaron para entrenar un modelo.

Comience importando los módulos relevantes y configurando la conexión al almacén de metadatos. También se proporciona algunas funciones auxiliares para mostrar información de artefactos y puede revisar su código en el módulo externo util.py.

5.1. Acceso a artefactos almacenados

Con la configuración de la conexión, ahora puede interactuar con el almacén de metadatos. Por ejemplo, puede recuperar todos los tipos de artefactos almacenados con la función **get_artifact_types()**. Como referencia, la API está documentada aquí.

También puede obtener una lista de artefactos para un tipo particular para ver si se usan variaciones en la canalización. Deberá mostrar al menos dos filas: una para el esquema inferido y otra para el esquema actualizado. Si ejecutó este cuaderno antes, es posible que vea más filas debido a los diferentes artefactos de esquema guardados en el directorio **./SchemaGen/schema**.

Además, también puede obtener las propiedades de un artefacto en particular. TFX declara algunas propiedades automáticamente para cada uno de sus componentes. Lo más probable es que vea **name**, **state** y **producer_component** para cada tipo de artefacto. Se agregan propiedades adicionales cuando

corresponde. Por ejemplo, se agrega una propiedad **split_names** en los artefactos **ExampleStatistics** para indicar para qué divisiones se generan las estadísticas.

5.2. Seguimiento de artefactos

Finalmente, cree una función para devolver los artefactos principales de uno determinado. Por ejemplo, esto debería poder enumerar los artefactos que se usaron para generar una instancia **TransformGraph** particular.

5.3. Obtener artefactos principales

Rastree las entradas de un artefacto en particular.

Consejos:

- Puede encontrar get_events_by_artifact_ids() y get_events_by_execution_ids() útil aquí.
- Algunos de los métodos de la clase **MetadataStore** (como los dos anteriores) solo aceptan iterables, así que recuerde convertir a una lista (o conjunto) si solo tiene un *int* (por ejemplo, pase **[x]** en lugar de 'x').

Valide el resultado esperado:
Nota: Los números de identificación pueden diferir.

artifact id	type	uri
1	Examples	./CsvExampleGen/examples/1
4	Schema	./ImportSchemaGen/schema/4