Reto Datos

David Guzmán Leyva - A01706417
Enrique Santos Fraire - A01705746
Leonardo Alvarado Menéndez - A01705998
Oscar Enrique Delgadillo Ochoa - A01705935

Índice

Bitácora de cambios

Versión	Fecha	Autor(es)	Modificaciones
1.0	19/10/2022	David Guzmán Leyva Enrique Santos Fraire Leonardo Alvarado Menéndez Oscar Enrique Delgadillo Ochoa	Línea base
1.1	28/11/2022	David Guzmán Leyva Enrique Santos Fraire Leonardo Alvarado Menéndez Oscar Enrique Delgadillo Ochoa	Ajuste al cross validation y enfoque
1.2	30/11/2022	David Guzmán Leyva Enrique Santos Fraire Leonardo Alvarado Menéndez Oscar Enrique Delgadillo Ochoa	Ajuste de Herramientas y tecnologías, modelo de almacenamiento, cross validation y enfoque

Herramientas y tecnologías

A continuación se muestra una tabla general con las herramientas, su función y la fase en la que se empleó, para posteriormente ser explicadas más a detalle:

Herramienta	Función	Características	Fase en que se empleó
Apache Spark	Creación y entrenamiento del modelo	Rapidez RDD Facilidad de uso Uso general Código abierto	Modeling
Pandas	Para la manipulación y análisis de datos	Manipulación de tablas y series numéricas	Data Understanding Data Preparation Modeling
Numpy	Operaciones con los datos	Se usa en conjunto con Pandas para operaciones de tipo matemáticas, lógicas, de ordenación, estadísticas, de entrada y de salida	Data Understanding Data Preparation Modeling
Geopy	Obtención de las comunas	Geocodificación a través de servicios	Data Preparation

Google Colab	Ejecución de scripts del proyecto y su visualización desde otros equipos	Escritura y ejecución de código en el navegador	Data Understanding Data Preparation Modeling
Anaconda	Ejecución local de los scripts	Distribución libre y abierta de los lenguajes Python, R, entre otros	Data Understanding Data Preparation Modeling
GitHub	Alojamiento del repositorio del equipo	Control de versiones Visualización remota	Business Understanding Data Understanding Data Preparation Modeling Evaluation Deployment
Google Drive	Alojamiento de archivos para el proyecto	Actividad colaborativa Acceso remoto	Business Understanding Data Understanding Data Preparation Modeling Evaluation Deployment

La herramienta primaria que se va a utilizar para llevar a cabo el modelado de minería de datos será el framework de Apache Spark, asumiendo que nuestros datos son Big Data. La elección de este fue por sus diversas cualidades, ya que dicho framework tiene las siguientes ventajas:

- Rapidez: Ejecuta las cargas de trabajo 100 veces más rápido que con Hadoop MapReduce. Con Spark, se tiene un alto rendimiento con los datos por lotes y de streaming gracias al programador de grafos acíclicos dirigidos de última generación, al optimizador de consultas y al motor físico de ejecución.
- Manejo de datos: Spark puede hacer uso de Resilient Distributed Datasets (RDD).
- Facilidad de uso: Spark cuenta con más de 80 operadores generales que facilitan el desarrollo de aplicaciones en paralelo. Puede ser utilizado de forma interactiva desde el shell de Scala, Python, R y SQL para escribir aplicaciones rápidamente.
- Uso general: Spark permite usar una pila de bibliotecas que incluye SQL, DataFrame, MLlib y Spark ML para aprendizaje automático, GraphX y Spark Streaming.
- Framework de código abierto: Además de ser gratis, cuenta con potencial colectivo para aportar más ideas, desarrollarlas más rápido y solucionar los problemas en cuanto aparecen.

Asimismo emplearemos algunas librerías de python necesarias para realizar ETL:

- Pandas: Sirve para la manipulación y análisis de datos, además ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales.
- Numpy: Para realizar cálculos u operaciones con los datos.

- Geopy: Será utilizado para la obtención de comunas. Esto debido a que permite localizar las coordenadas de direcciones, ciudades, países y puntos de referencia en todo el mundo mediante geocodificadores de terceros y otras fuentes de datos.

Entornos de ejecución del proyecto:

- Google Colab: Permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador. Es especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación.
- Anaconda: Es una distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático. En ella podemos crear Jupyter notebooks.

Manejo de versiones del proyecto:

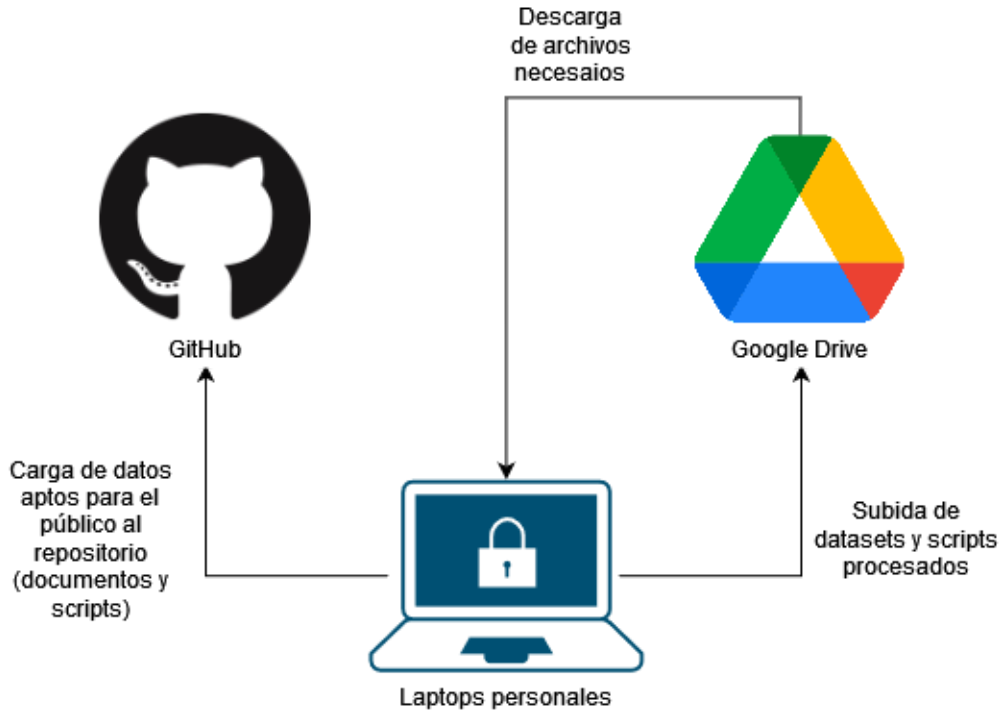
- Github: es una forja para alojar proyectos utilizando el sistema de control de versiones Git.
- Link de repositorio: https://github.com/Davidguzley/Movilidad-Chile

Administración del proyecto:

- Google Drive: Es un servicio de alojamiento de archivos en el cual realizamos planes de trabajo y documentación del proyecto de manera colaborativa.

Modelo de almacenamiento

El modelo de almacenamiento que aplicamos para guardar nuestros datos es del tipo filestore, pues ya está completamente administrado y basado en la nube, esto haciendo uso de google drive. El modelo se ve representado a través del siguiente diagrama:



Contamos con 3 componentes principales para el almacenamiento: GitHub, Google Drive y nuestras laptops personales. Partiendo de Google Drive, es ahí donde tenemos nuestra unidad compartida, a la cual únicamente tienen acceso los miembros del equipo por cuestiones de seguridad, explicadas más a fondo en el documento de seguridad y privacidad de datos. Dentro de la unidad compartida almacenamos los datasets, la documentación y los scripts necesarios para el procesamiento de los datos.

Por otra parte, contamos con nuestras computadoras personales, donde realizamos el procesamiento de los datos y nos conectamos a la unidad compartida, formando el siguiente ciclo:

- Bajamos los datasets de Google Drive
- Procesamos los datos de manera local o en Google Colab dependiendo la situación
- En caso de que se hayan procesado de manera local se suben los nuevos datasets y scripts a la unidad compartida
- Se genera la documentación colaborativa igualmente en Google Drive

Y así sucesivamente de manera iterativa hasta completar la solución del reto.

Finalmente, tenemos la plataforma GitHub, donde se aloja el repositorio del equipo y se realiza el manejo de versiones. Su proceso de uso consiste en que al concluir una iteración o modificaciones significativas se descargan todos los archivos correspondientes de la unidad compartida a la computadora personal y se suben al repositorio.

Cabe destacar que no se suben los datasets al repositorio por cuestiones de privacidad (referenciadas anteriormente), esto ya que GitHub es una plataforma pública y nuestro repositorio puede ser visualizado por cualquier persona.

En un entorno real, fuera del contexto académico y con una cantidad muchísimo mayor de datos, lo ideal sería usar un sistema de almacenamiento de AWS (Amazon Web Services) con AWS RDS. Pues nos permite la escalabilidad en caso aumentar o reducir la cantidad de datos, pudiendo enlazarlo a un entorno de AWS Elastic Beanstalk.

Ajuste de datos

Data Preparation

Muestreo

- Repositorio
- Acceso a drive

Cross validation

Para realizar validación cruzada utilizaremos CrossValidator de Spark ML, esta validación cruzada de K-fold realiza la selección del modelo al dividir el conjunto de datos en un conjunto de pliegues particionados aleatoriamente que no se superponen y que se utilizan como conjuntos de datos de entrenamiento y prueba separados.

Por ejemplo, con k = 3 pliegues, la validación cruzada de K-fold generará 3 (entrenamiento, prueba) pares de conjuntos de datos, cada uno de los cuales usa 2/3 de los datos para entrenamiento y 1/3 para prueba. Cada pliegue se utiliza como conjunto de prueba exactamente una vez.

Hay cuatro cosas necesarias para utilizar el CrossValidator:

- Estimator: Modelo de aprendizaje automático para predecir valores.
- EstimatorParamMaps: Grid de hiperparametros a probar del estimator definido por ParamGridBuilder().
- NumFolds: Número de folds para cross validation.
- Evaluator: Evaluador específico y compatible para el estimator. Por ejemplo, para un modelo de regresión logística binaria es necesario utilizar BinaryClassificationEvaluator().

En nuestro caso utilizamos la siguiente configuración:

Estimator	EstimatorParamMaps	NumFolds	Evaluator
LinearRegression	maxIter: 5, 10, 20, 50, 100	5	RegressionEvaluator
GBTRegressor	maxDepth: 6, 12, 18, 24, 30 maxBins: 18, 36, 72, 144, 288 maxIter: 5, 10, 20, 50, 100	5	RegressionEvaluator

Justificaciones de configuración:

- Se emplearon valores para maxIter empezando con 5 y aumentando casi siempre al doble hasta llegar al número por default de los modelos que es 100, porque así obtendremos resultados más significativos al tener como número máximo de iteraciones lo más cercano a su valor óptimo para la mayoría de datos utilizados en LinearRegression y GBTRegressor.
- El número máximo permitido para maxDepth es 30, por lo que decidimos utilizar múltiplos de 6 hasta llegar al número máximo para probar con los números posibles con la misma diferencia entre ellos, este valor es muy importante porque representa la profundidad máxima para cada árbol de decisión empleado en el modelo de GBTRegressor.
- El valor mínimo de maxBins es 18 porque es la cantidad de variables utilizadas para predecir el porcentaje de trabajo en casa de un sitio. Por lo tanto, no podemos poner un número menor a éste ya que el propio modelo no podrá realizar el entrenamiento, y aumentamos este valor al doble para obtener mejores resultados que su predecesor.

- Se eligió NumFolds como 5, ya que este es numero multiplo y exponencial para los minimos y maximos valores requeridos para los hiperparametros de nuestros modelos.

Resultados obtenidos gracias a CrossValidator:

Modelo	RMSE	MSE	MAE	R2
LinearRegression	4.631	21.448	3.753	-8.853
GBTRegressor	2.122	4.503	1.713	0.633

El GBTRegression obtiene mejores resultados porque principalmente utiliza boosting, esto quiere decir que emplea modelos de aprendizaje automático muchos más débiles como los árboles de decisiones y cada árbol mejora a su antecesor. Además, en este modelo existen más hiperparámetros que podemos ajustar a las necesidades del proyecto como la profundidad de los árboles de decisión a ocupar en él.

Enfoque Big Data

Debido a la cantidad de datos con los que contamos, y que el número de datos no aumentará a lo largo del reto, de primera mano el dataset puede ser considerado como normal data, ya que no cuenta con ninguna de las 5 características (volumen, velocidad, variedad, veracidad y valor).

Sin embargo, teniendo en consideración nuestro contexto y los recursos que tenemos disponibles, sabemos que estamos en un entorno académico con recursos limitados, dependiendo casi exclusivamente de herramientas gratuitas y nuestras computadoras como poder de procesamiento con un límite de 24 GB de RAM.

Estas características hacen que nuestros datos cuenten con un número de registros considerablemente altos y tiempos de procesamiento de hasta 8 horas. Por lo que para realizar con éxito el reto, debemos extraer, leer, procesar y obtener datos nuevos de la manera más rápida posible.

Es por ello que dado el contexto de nuestra investigación y el tiempo disponible, el dataset es, efectivamente, Big Data. Por lo que el uso de herramientas como Apache Spark para solucionar el reto es recomendable, principalmente con sus modelos para generación de predicciones, ya que es un framework de computación en clúster capaz de dividir y ejecutar las tareas de los algoritmos de aprendizaje automático de manera eficiente gracias a su MapReduce.