UNIVERSIDAD
Panamericana

Alumno: Adriana Leticia Martinez Estrada

Trabajo: Proyecto final Github Pull Requests

Materia: Grandes Datos

Profesor: Jaime Ulises Jiménez Cardoso

Fecha de entrega: 6 de marzo de 2024

GitHub Pull Requests

Resumen Ejecutivo

En esta era digital, el desarrollo de software no es solo una tarea aislada de programadores individuales; se ha transformado en una vasta red de colaboración global. Plataformas como GitHub han emergido como epicentros de este paradigma, permitiendo a desarrolladores de todo el mundo contribuir a proyectos comunes, compartir código y fomentar la innovación. GitHub, un repositorio de código y una plataforma que facilita el versionado y la colaboración en proyectos de software, ofreciendo herramientas esenciales para el manejo eficaz de proyectos complejos.

Uno de los conceptos fundamentales en GitHub es el "pull request" (PR), que es una solicitud enviada por contribuyentes para que los cambios que han implementado en su versión del proyecto sean revisados y potencialmente integrados (merge) en la base de código principal. Los PRs son el corazón de la colaboración en GitHub, permitiendo que el código sea discutido, revisado y mejorado antes de su incorporación final, asegurando así la calidad y la coherencia del software.[1]

Este proyecto se centra en el análisis de Pull Requests de GitHub para explorar cómo las contribuciones individuales afectan la calidad general del desarrollo de software. Con el uso de un conjunto de datos extenso, superior a 20 GB, proveniente de GHTorrent —una base de datos que indexa la actividad pública de GitHub—, buscamos identificar patrones y tendencias en las contribuciones que podrían indicar la calidad de los aportes.

La meta es aplicar técnicas de Machine Learning para desarrollar un modelo capaz de predecir la presencia de errores en los PRs. Este modelo no solo podría ser una herramienta invaluable para los mantenedores de proyectos, facilitando la revisión y aceptación de contribuciones, sino que también ofrecería insights sobre prácticas de desarrollo que maximizan la calidad del código.

Desarrollo de la Solución

Utilizando Google Cloud Storage, Dataproc y BigQuery, se extrajeron y transformaron datos significativos de cinco archivos con un peso total de 69 GB, aplicando un proceso ETL detallado para preparar un conjunto de datos para el análisis y la modelación. Las herramientas de Big Data facilitaron este proceso, permitiendo la manipulación eficiente de un gran volumen de información.

Aplicación del Modelo

Se implementó un modelo en BigQuery ML, que, basado en expresiones regulares, identifica y clasifica los comentarios de los commits por la presencia de términos clave. Esta clasificación ayudará a predecir si un pull request contiene errores, lo que puede interpretarse como un indicador de la necesidad de mejora.

Beneficios y Clientela Potencial

El proyecto no solo aporta a la mejora del proceso de revisión de código, sino que también beneficia a los gestores de repositorios y líderes de proyecto al proporcionar un método sistemático para evaluar la calidad de las contribuciones así como la cantidad de las mismas. Los equipos de desarrollo que buscan optimizar sus flujos de trabajo de revisión y contribución son también clientes ideales de este producto.

Visión General del Desarrollo

Solución Actual

Se aborda el desafío de analizar y clasificar las contribuciones a repositorios en GitHub mediante Pull Requests (PRs), utilizando un conjunto de datos de gran volumen (aproximadamente 69 GB). A través de un proceso de ETL (Extracción, Transformación y Carga), hemos procesado y almacenado con éxito estos datos en Google Cloud Storage. Utilizando las capacidades de procesamiento de Dataproc y la consulta de datos de BigQuery, hemos creado vistas y tablas para la segmentación de los datos, y hemos desarrollado un modelo predictivo con BigQuery ML para evaluar la calidad de los PRs basándonos en comentarios y otros metadatos relevantes.

Limitaciones de la Solución

La solución depende de la precisión de las expresiones regulares para identificar comentarios significativos dentro de los PRs y de esta forma generar una categorización inicial. La identificación de errores, buen trabajo y nuevas características está sujeta a las limitaciones inherentes al análisis de texto simple y puede no capturar la complejidad o el contexto completo de las conversaciones en GitHub.

Propósito, Uso y Alcances

El propósito de esta herramienta es proporcionar a los gestores de repositorios y a los equipos de desarrollo insights prácticos y accionables para mejorar la colaboración y la calidad del código en proyectos de software. Los indicadores clave y el modelo predictivo buscan identificar patrones de éxito y áreas de mejora en las contribuciones de código, con el objetivo de incrementar la eficiencia y efectividad de la revisión y aceptación de PRs. El modelo y las visualizaciones resultantes ofrecen una perspectiva valiosa que puede mejorar

las prácticas de desarrollo de software en comunidades de código abierto y en entornos profesionales.

La herramienta tiene un alcance amplio, ofreciendo beneficios a pequeños proyectos independientes así como a grandes organizaciones que gestionan múltiples contribuciones en diversas bases de código. Con su API expuesta, la herramienta puede facilitar la integración en flujos de trabajo existentes.

Revisión y uso de datos

Orígenes de Datos y Control de Datos

La base de datos sobre la que se ha trabajado fue extraída de Kaggle, específicamente del conjunto de datos de GitHub Pull Requests provisto por GHTorrent. Esta extracción se llevó a cabo a través de una máquina virtual configurada en Google Cloud Compute Engine, conectada a la API de Kaggle, donde se ejecutó el comando de descarga y se procesaron los archivos correspondientes.

Preparación de Datos

Los datos, una vez extraídos y descomprimidos en la máquina virtual, fueron almacenados en un Bucket de Google Cloud Storage, designado para su almacenamiento intermedio y accesibilidad. Esta etapa fue esencial para preparar el terreno para el procesamiento y análisis subsiguiente.

Limpieza y Tratamiento de Datos

Utilizando Dataproc y Hive, los datos se sometieron a un riguroso proceso de ETL. Se crearon tablas externas que permitieron la agrupación y organización de los distintos archivos del dataset. Se realizó una limpieza de datos donde se filtraron aquellos registros no esenciales para el análisis, como pull requests sin comentarios, sin repositorio específico o sin lenguaje de programación definido.

Integridad de los Datos

La integridad de los datos se mantuvo mediante consultas de verificación en Hive, asegurándose de que solo los datos válidos y relevantes fueran procesados y analizados. Esto fue vital para garantizar que la información que avanzaba a través del pipeline fuera precisa y confiable.

Limitaciones de Datos

Las limitaciones en el conjunto de datos se abordaron desde el principio. Al centrarnos solo en los registros que proporcionaban información completa, se excluyeron aquellos que podrían introducir ruido o sesgo en el análisis. No obstante, las limitaciones en términos de la variedad y la complejidad de los comentarios en el texto de los commits podrían influir en la capacidad del modelo para generalizar y captar la sutileza de los comentarios humanos.

Proceso de Desarrollo

Metodología

Esta solución se abordó mediante una metodología iterativa, con la flexibilidad de las herramientas de GCP como eje central. La solución implicó el uso de la API de Kaggle para extraer un conjunto de datos significativo de GitHub Pull Requests, proporcionados por GHTorrent y disponibles en Kaggle. Una máquina virtual en GCP sirvió como el punto de partida para la descarga y descompresión de datos, que luego fueron transferidos a Google Cloud Storage. Esta secuencia de acciones no solo evidenció la capacidad de las herramientas de GCP para manejar grandes volúmenes de datos, sino que también mostró la interoperabilidad entre diversos servicios en la nube.

El cluster en Dataproc se utilizó para el procesamiento ETL, aprovechando Hive para realizar la limpieza y organización de los datos, preparándolos para su análisis posterior. Con la data ya procesada, se cargó a BigQuery para realizar análisis más detallados y para alimentar las visualizaciones en Looker Studio.

Una vez en BigQuery, se crearon vistas y tablas con la información necesaria para el entrenamiento y prueba de un modelo predictivo de regresión lineal, utilizando BigQuery ML para evaluar la probabilidad de errores en los commits basándose en los comentarios de los pull requests. El enfoque no solo se limitó al análisis de datos, sino que también se extendió al despliegue de un modelo de Machine Learning a través de un API.

El API fue desarrollado para funcionar inicialmente en un entorno local, permitiendo la ejecución de predicciones y la integración de los resultados en un sitio web creado con Google Sites, haciendo los resultados accesibles y fácilmente interpretables por los usuarios finales. El sitio web actúa como una interfaz de usuario amigable que demuestra el valor práctico del proyecto.

Pruebas

Las pruebas jugaron un papel fundamental en el proyecto. Se emplearon para validar cada paso del proceso ETL, garantizar la integridad de los datos en BigQuery, y confirmar la eficacia del modelo de ML. La precisión del modelo, su recuerdo y su puntuación F1 se calculó utilizando un conjunto de datos de prueba para evaluar su rendimiento y afinar sus parámetros.

Las pruebas no solo se limitaron a la validación de modelos, sino que también se extendieron a la integración del API y su funcionalidad dentro del sitio web. Se realizaron pruebas para verificar la respuesta del API a diversas solicitudes y asegurar que los resultados fueran consistentes y confiables.

Resultados y Conclusiones

Resultados Obtenidos

La implementación de la solución siguió un flujo de trabajo claro y estructurado. Utilizamos Compite engine para la descarga y descompresión de la información, Google Cloud Storage para el almacenamiento inicial de datos así como una herramienta auxiliar para almacenar la informació y poder importarla entre las distintas herramientas utilizadas, Dataproc y Hive para el procesamiento y la limpieza de datos, y BigQuery/BigQueryML para el análisis de datos y la creación de modelos predictivos. La visualización de datos se llevó a cabo con Looker Studio, proporcionando paneles intuitivos que resumen nuestras métricas clave y conclusiones. ravés de un Site de Google para brindar una funcionalidad práctica a esta solución.

A través de este enfoque, pudimos no solo identificar y clasificar comentarios relevantes dentro de los PRs, sino también desarrollar un modelo de Machine Learning que predice la presencia de errores con una precisión, recall, y puntuaciones F1 y AUC significativas. Este modelo se desplegó con las herramientas de AI Platform y Google APIs utilizando un API desarrollado que funcionó inicialmente en un entorno local, y los resultados se hicieron accesibles a través de un sitio web en Google Sites, demostrando la aplicabilidad práctica de nuestra investigación.

Resultados del Modelo de Machine Learning

Esta solución ha culminado en el desarrollo de un modelo de clasificación que demuestra capacidades prometedoras en predecir errores en pull requests de GitHub. La evaluación del modelo reveló una precisión del 79.17%, lo que indica una alta confiabilidad en las predicciones positivas. A pesar de una baja recuperación, lo que sugiere un margen para detectar todos los errores potenciales, el modelo demostró una capacidad significativa para identificar los pull requests más críticos que requieren atención. La pérdida logística mostró una disminución consistente a lo largo de las iteraciones de entrenamiento, señalando un buen ajuste del modelo.

La curva ROC, con un área de 0.6998, y la matriz de confusión respaldan la eficacia del modelo. Estos resultados validan el enfoque analítico adoptado y sientan las bases para aplicaciones más amplias de la tecnología de aprendizaje automático en la mejora del proceso de revisión de software. La tasa de aprendizaje se incrementó progresivamente para permitir que el modelo se adaptara de manera efectiva a la complejidad de los datos. La duración de las iteraciones refleja una optimización eficiente de los recursos computacionales.

La implementación del modelo ha aportado valiosos insights sobre la contribución de las variables a la presencia de errores. Esta información puede guiar a los desarrolladores y a los equipos de proyectos para refinar sus prácticas de código y mejorar la calidad general del software. Además, las visualizaciones desarrolladas ofrecen una manera accesible de comprender el desempeño del modelo y las tendencias de los datos.

Prueba datos Testeo

Fila	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.780698	0.02361515386	0.9252535530	0.04584359546	0.2452104565	0.7105834165

Los resultados de la prueba de testeo del modelo con datos de prueba han reforzado la eficacia de nuestra solución. Alcanzamos una precisión de 0.7807, lo que subraya la habilidad del modelo para identificar correctamente los errores con gran exactitud. Aunque el recall de 0.0236 indica que hay espacio para mejorar en la identificación de todos los casos positivos reales, la exactitud general del modelo fue notable, con un valor de 0.9253.

El f1_score de 0.0458, aunque mejorable, refleja el difícil equilibrio entre precisión y recuperación para nuestro conjunto de datos desafiante. La pérdida logística de 0.2452 demuestra la eficiencia del modelo en términos de las probabilidades pronosticadas, mientras que un área ROC de 0.7106 valida la habilidad del modelo para distinguir entre clases de manera efectiva.

Estos resultados enfatizan la promesa de esta metodología y la aplicabilidad del modelo en entornos reales, abriendo caminos para futuras optimizaciones y aplicaciones prácticas.

Herramientas y Beneficios

Las herramientas seleccionadas para este proyecto ofrecieron beneficios específicos:

- **Compute Engine**: Proporcionó un entorno virtual y en la nube que nos permitió la conectividad con APIs externas para la descarga y descompresión de archivos sin necesidad de una descarga local.
- **Google Cloud Storage**: Ofreció una solución escalable para manejar grandes conjuntos de datos, facilitando el acceso y la colaboración.
- **Dataproc y Hive**: Permitieron el procesamiento eficiente de datos estructurados y semi-estructurados a gran escala.
- **BigQuery y BigQueryML**: Proporcionaron una plataforma poderosa para análisis y modelado predictivo con capacidades de Machine Learning integradas.
- **Looker Studio**: Permitió la visualización interactiva y dinámica de datos para interpretar fácilmente las conclusiones del proyecto.
- **AI Platform y Google APIs** (No visto en Clase): Permitieron la productivización y despliegue del modelo desarrollado

Conclusiones Clave

Esta solución destacó la relevancia del análisis de datos para mejorar la calidad y eficiencia del desarrollo de software. La clasificación automática de PRs por calidad puede servir como una herramienta valiosa para los desarrolladores y equipos de proyectos de software, ayudándoles a identificar áreas de mejora y asegurando mejores prácticas en el desarrollo colaborativo.

Trabajo Futuro

Mirando hacia el futuro, reconocemos que el modelo podría mejorarse aún más con la integración de datos adicionales, como la aceptación de los PRs y el número de revisiones que reciben. Tales métricas podrían ofrecer una comprensión más profunda de la productividad y la calidad en el desarrollo de software. En vez de centrarse únicamente en la presencia de comentarios de errores para medir la productividad, estas métricas adicionales podrían proporcionar una visión más holística del proceso de desarrollo y permitir la creación de un modelo más refinado y preciso.

La incorporación de estas métricas en el conjunto de datos y su análisis a través de nuestro modelo podría resultar en una herramienta más robusta para predecir la calidad de los PRs, lo que a su vez podría traducirse en una colaboración más efectiva y en una mayor calidad del código producido en proyectos de código abierto y privados por igual. Con el objetivo de mejorar constantemente, planeamos recopilar estos datos adicionales y actualizar nuestro modelo y visualizaciones en consecuencia, asegurando que este trabajo siga siendo relevante y de vanguardia en el campo de la ciencia de datos.

Referencias

[1] Olmedo, A., Arévalo, G., Cassol, I., Perez, Q., Urtado, C., & Vauttier, S. (2022, April). Pull Requests Integration Process Optimization: An Empirical Study. In *International Conference on Evaluation of Novel Approaches to Software Engineering* (pp. 155-178). Cham: Springer Nature Switzerland.
[2]https://towardsdatascience.com/quickly-transfer-a-kaggle-dataset-into-a-google-bucket-ac21aefceb15
[3] https://cloud.google.com/sdk/docs/install?hl=es-419
[4] https://cloud.google.com/compute/docs/instances/transfer-files?hl=es-419
[5] https://cloud.google.com/bigquery/docs/export-model-tutorial?hl=es-419
[6] https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-create
[7]https://cloud.google.com/bigquery/docs/create-machine-learning-model?hl=es-419
[8] NUNES, T. A. R. (2019). Git dashboard: dashboard de pull requests para o github.

Anexos

Querys

Creación de Tabla externa HIVE

```
CREATE EXTERNAL TABLE IF NOT EXISTS github_pull_requests (
  actor_login STRING,
  actor_id INT,
  comment_id INT,
  comment STRING,
  repo STRING,
  language STRING,
  author_login STRING,
  author_id INT,
  pr_id INT,
  c_id INT,
  commit_date STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'gs://up-pf-ghtorrent-2024-alme/temp_zip_extract';
```

Creación de tabla Auxiliar

```
CREATE TABLE github_pull_requests_clean AS
SELECT actor_login, actor_id, comment_id, comment, repo, language, author_login,
author_id, pr_id, c_id, commit_date
FROM github_pull_requests
WHERE repo IS NOT NULL
AND language IS NOT NULL
AND comment IS NOT NULL
AND TRIM(comment) <> ''
AND comment NOT LIKE '%NULL%'
AND commit_date IS NOT NULL
AND LENGTH(commit_date) = 23;
```

Exportación de información de HIVE a Bucket

```
INSERT OVERWRITE DIRECTORY
'gs://up-pf-ghtorrent-2024-alme/temp_for_bigquery_load/'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT * FROM github_pull_requests_clean;
```

Creación De Vista en BigQuery

```
CREATE VIEW `bd-final-alme.pf_github.pruebagit_with_flags` AS
SELECT
  actor_login,
  actor_id,
  comment_id,
  comment,
  repo,
  language,
  author_login,
  author_id,
  pr_id,
  c_id,
  commit_date,
  CASE WHEN LOWER(comment) LIKE '%error%' OR LOWER(comment) LIKE '%fix%' OR
LOWER(comment) LIKE '%bug%' THEN TRUE ELSE FALSE END AS contains_error,
  CASE WHEN LOWER(comment) LIKE '%well%' OR LOWER(comment) LIKE '%good%'
OR LOWER(comment) LIKE '%great%'OR LOWER(comment) LIKE '%ok%' THEN TRUE
ELSE FALSE END AS good_job,
  CASE WHEN LOWER(comment) LIKE '%new%' OR LOWER(comment) LIKE '%feature%'
THEN TRUE ELSE FALSE END AS contains_new_feature
FROM
  `bd-final-alme.pf_github.pruebagit`;
```

Creación de tabla auxiliar para segmentación de información en BigQuery

```
CREATE TABLE `bd-final-alme.pf_github.pruebagit-segment`
AS
 SELECT
 actor_login,
 language,
 repo,
 CASE WHEN LOWER(comment) LIKE '%error%' OR LOWER(comment) LIKE '%fix%' OR
LOWER(comment) LIKE '%bug%' THEN TRUE ELSE FALSE END AS contains_error,
  CASE WHEN LOWER(comment) LIKE '%well%' OR LOWER(comment) LIKE '%good%'
OR LOWER(comment) LIKE '%great%'OR LOWER(comment) LIKE '%ok%' THEN TRUE
ELSE FALSE END AS good_job,
 RAND() AS random_number
 FROM `bd-final-alme.pf_github.pruebagit`;
```

Generación de tablas de entrenamiento y pruebas

```
CREATE OR REPLACE TABLE `bd-final-alme.pf_github.tabla_entrenamiento` AS
SELECT *
```

```
FROM `bd-final-alme.pf_github.pruebagit-segment`
WHERE random_number < 0.8;


CREATE OR REPLACE TABLE `bd-final-alme.pf_github.tabla_pruebas` AS
SELECT *
FROM `bd-final-alme.pf_github.pruebagit-segment`
WHERE random_number >= 0.8;
```

Entrenamiento de Modelo BigQueryML

```
CREATE OR REPLACE MODEL `bd-final-alme.pf_github.modelo_contains_error`
OPTIONS(model_type='logistic_reg', input_label_cols=['contains_error']) AS
SELECT
 actor_login,
 language,
 IF(contains_error, 1, 0) AS contains_error
FROM
 `bd-final-alme.pf_github.tabla_entrenamiento`;
```

Evaluación de Modelo BigQueryML

```
SELECT
  *
FROM
  ML.EVALUATE(MODEL `bd-final-alme.pf_github.modelo_contains_error`, (
    SELECT
      actor_login,
      language,
      IF(contains_error, 1, 0) AS contains_error
    FROM
      `bd-final-alme.pf_github.tabla_pruebas`
  ));
```

Comandos configuración de Herramientas

Máquina Virtual

```
gcloud compute instances create vm-pf-kaggle-alme --project=bd-final-alme
--zone=us-central1-c --machine-type=n2-standard-4
--network-interface=network-tier=PREMIUM,stack-type=IPV4_ONLY,subnet=default
--maintenance-policy=MIGRATE --provisioning-model=STANDARD
--service-account=594778691467-compute@developer.gserviceaccount.com
```

--scopes=https://www.googleapis.com/auth/cloud-platform --tags=http-server,https-server --create-disk=auto-delete=yes,boot=yes,device-name=vm-pf-kaggle-alme,image=projects/ubuntu-os-cloud/global/images/ubuntu-2004-focal-v20240229,mode=rw,size=200,type=projects/bd-final-alme/zones/us-central1-c/diskTypes/pd-balanced --no-shielded-secure-boot --shielded-vtpm --shielded-integrity-monitoring --labels=goog-ec-src=vm_add-gcloud --reservation-affinity=any

Conexión máquina local a GCP

gcloud compute ssh --project bd-final-alme --zone us-central1-c vm-pf-kaggle-alme

Transferencia de claves Kaggle

gcloud compute scp Downloads/kaggle.json vm-pf-kaggle-alme:~

Creación Bucket

gsutil mb gs://up-pf-ghtorrent-2024-alme

Descarga de información a través del API de Kaggle

gcloud compute scp Downloads/kaggle.json vm-pf-kaggle-alme:~

Transferencia de indormación de VM a Bucket

gsutil -m cp -r temp_zip_extract gs://up-pf-ghtorrent-2024-alme

Creación de Cluster

```
gcloud dataproc clusters create cluster-pf-alme \
  --enable-component-gateway \
  --region us-central1 \
  --zone us-central1-c \
  --master-machine-type n2-standard-2 \
  --master-boot-disk-size 200 \
  --num-workers 2 \
  --worker-machine-type n2-standard-2 \
  --worker-boot-disk-size 200 \
  --image-version 2.0-ubuntu18 \
  --optional-components JUPYTER  \
  --bucket up-pf-ghtorrent-2024-alme \
  --project bd-final-alme
```

Carga de Información filtrada de Bucket a BigQuery

```
bq load --source_format=CSV --autodetect \
pf_github.pull_requests_clean \
gs://up-pf-ghtorrent-2024-alme/temp_for_bigquery_load/*
```

Exportación de modelo

```
bq extract -m pf_github.modelo_contains_error gs://up-pf-ghtorrent-2024-alme/pred_model
```

Descarga los archivos del modelo exportado en un directorio temporal

```
mkdir tmp_dir
gsutil cp -r gs://up-pf-ghtorrent-2024-alme/pred_model tmp_dir
```

Creación de un subdirectorio de la versión

```
mkdir -p serving_dir/pred_model/1
cp -r tmp_dir/pred_model/* serving_dir/pred_model/1
rm -r tmp_dir
```

Extracción de la imagen de Docker

```
docker pull tensorflow/serving
```

Ejecución del contenedor de Docker

```
docker run -p 8500:8500 --network="host" --mount
type=bind,source=`pwd`/serving_dir/pred_model,target=/models/pred_model -e
MODEL_NAME=pred_model -t tensorflow/serving &
```

Ejecución de la predicción

```
curl -d '{"instances": [{"actor_login": "adrianaleticiamartinez", "language": "PHP"}]}' -X POST
http://localhost:8501/v1/models/pred_model:predict
```

Despliegue Online

Creación de modelo

```
MODEL_NAME="PRED_MODEL"
gcloud ai-platform models create $MODEL_NAME
```

Creación de versión

```
MODEL_DIR="gs://up-pf-ghtorrent-2024-alme/pred_model"
VERSION_NAME="v1"
FRAMEWORK="TENSORFLOW"
gcloud ai-platform versions create $VERSION_NAME --model=$MODEL_NAME
--origin=$MODEL_DIR --runtime-version=1.15 --framework=$FRAMEWORK
--region=us-central1 --machine-type=n1-standard-2
```

Repositorio Remoto

Se incluye en un repositorio remoto con acceso público el código, comandos y Querys utilizados
https://github.com/adrianaleticiamartinez/mcd_bigdata

Google Site GitHub Pull Requests

Se incluye en un repositorio remoto con acceso público el código, comandos y Querys utilizados
https://sites.google.com/up.edu.mx/githubpullrequests

Diccionarios de datos

Tabla original

Campo	Tipo de Datos	Descripción
actor_login	STRING	El nombre de usuario del actor que realizó la acción en el Pull Request.
actor_id	INTEGER	El ID del actor que realizó la acción en el Pull Request.
comment_id	INTEGER	El ID único asociado al comentario en el Pull Request.
comment	STRING	El contenido del comentario asociado al Pull Request.
repo	STRING	El nombre del repositorio asociado al Pull Request.
language	STRING	El lenguaje de programación asociado al repositorio.
author_login	STRING	El nombre de usuario del autor del Pull Request.
author_id	INTEGER	El ID del autor del Pull Request.
pr_id	INTEGER	El ID único asociado al Pull Request.
c_id	INTEGER	El ID único asociado al commit (compromiso) relacionado con el Pull Request.
commit_date	TIMESTAMP	La fecha y hora en que se realizó el commit relacionado con el Pull Request.

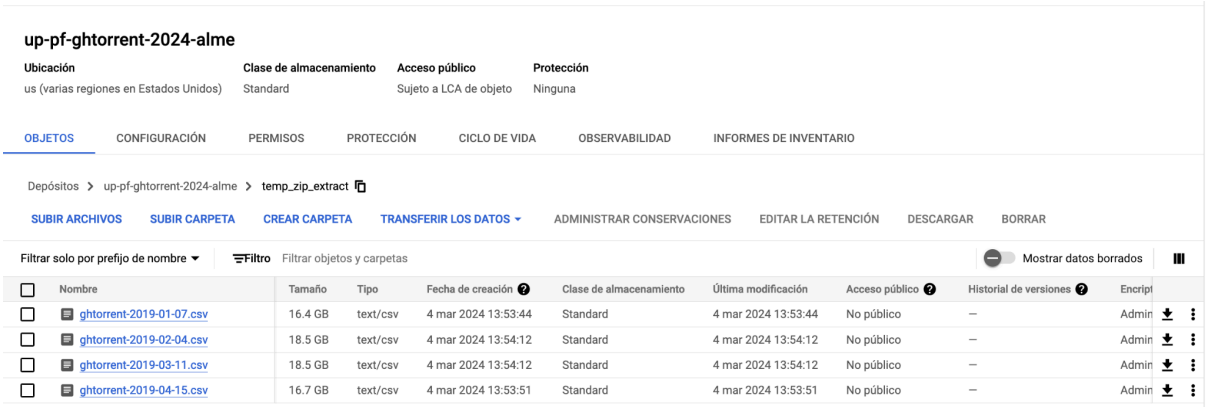
Tabla para modelo

Campo	Tipo de Datos	Descripción
actor_login	STRING	El nombre de usuario del actor que realizó la acción en el Pull Request.
language	STRING	El lenguaje de programación asociado al repositorio.
repo	STRING	El nombre del repositorio asociado al Pull Request.
contains_error	BOOLEAN	Indica si el comentario asociado al Pull Request contiene palabras clave relacionadas con errores (`error`, `fix`, `bug`).
good_job	BOOLEAN	Indica si el comentario asociado al Pull Request contiene palabras clave que reconocen el buen trabajo realizado (`well`, `good`, `great`, `ok`).
random_number	FLOAT	Un número aleatorio generado con el fin de segmentar los datos entre los conjuntos de entrenamiento y prueba.

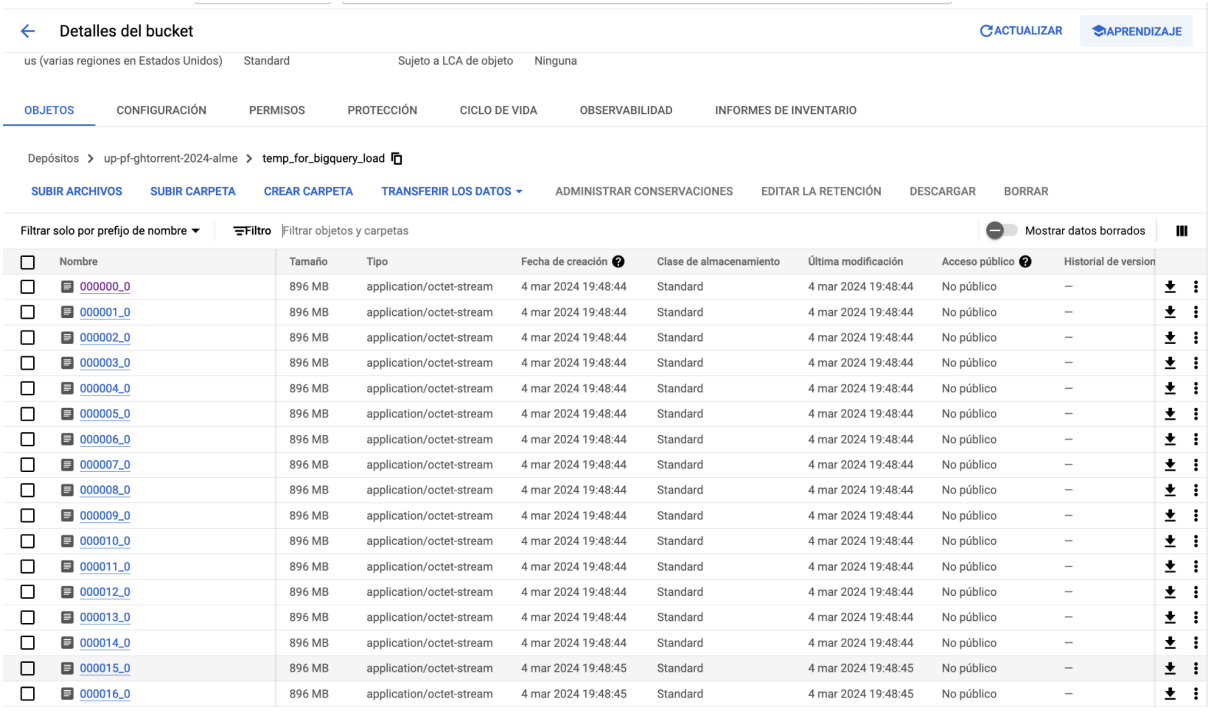
Evidencias

Almacenamiento en Bucket

Archivos descompresión

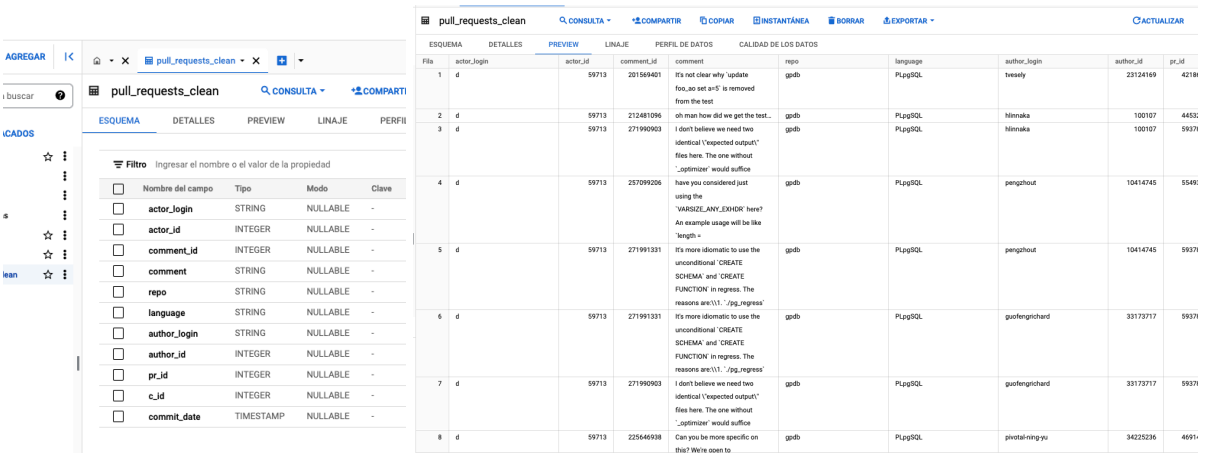


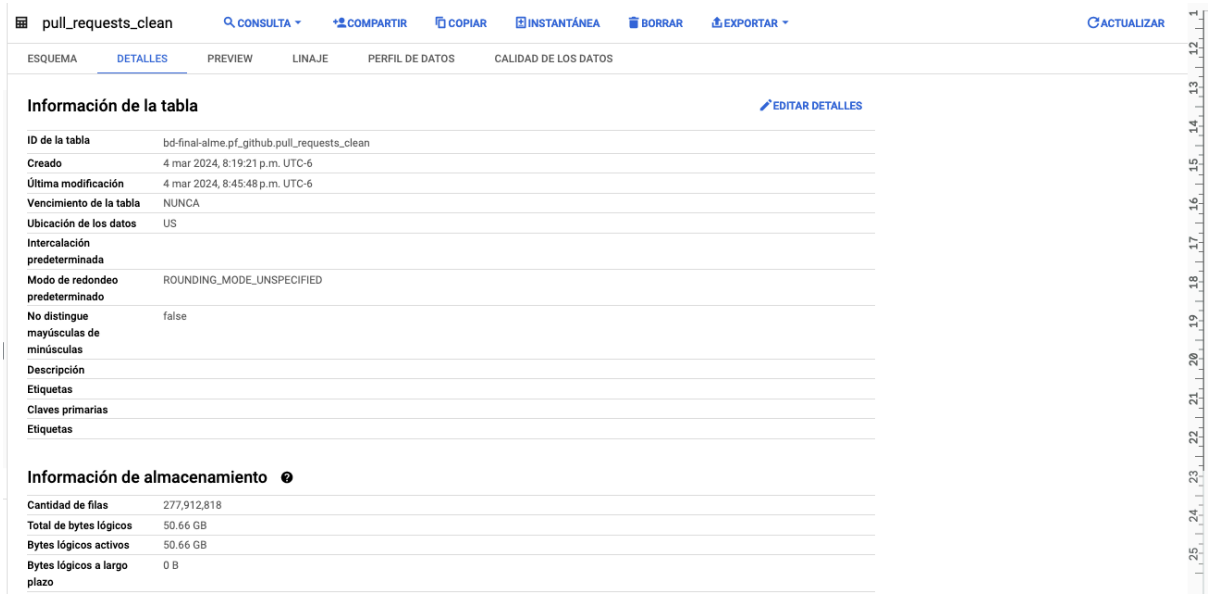
Archivos para carga en BigQuery



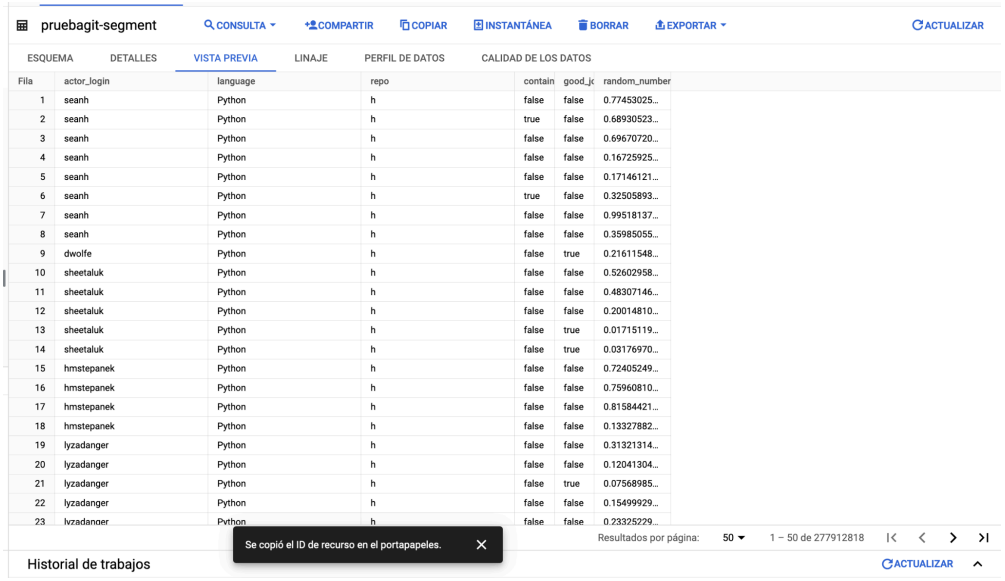
BigQuery

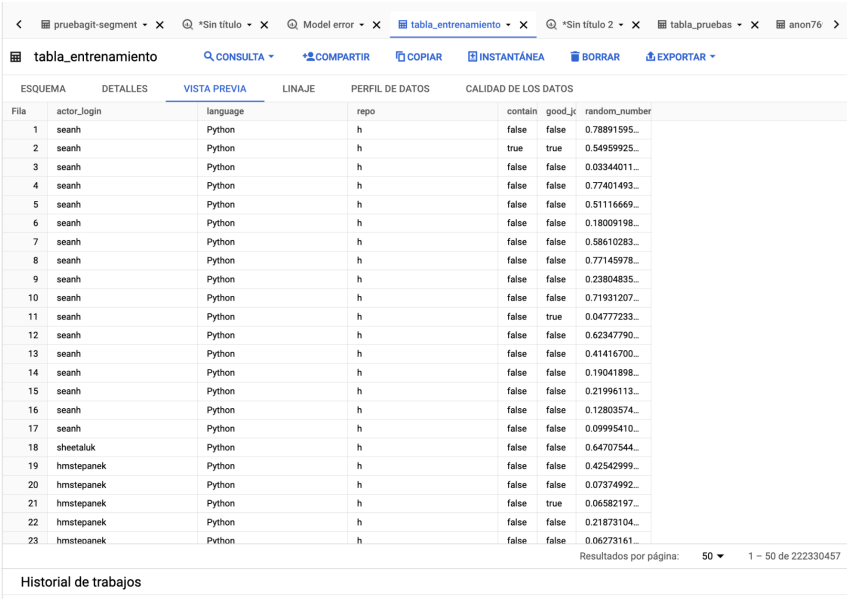
Carga de información

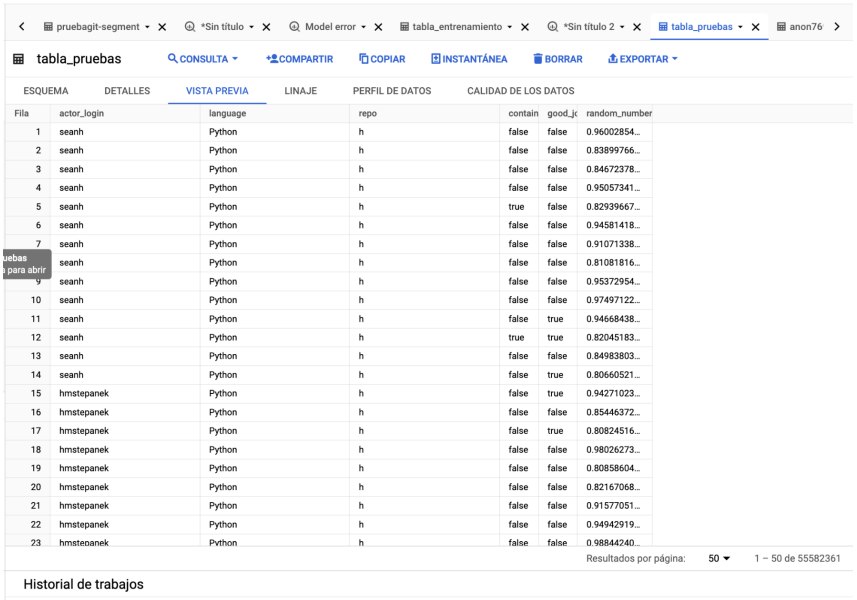




Sets de entrenamiento y prueba

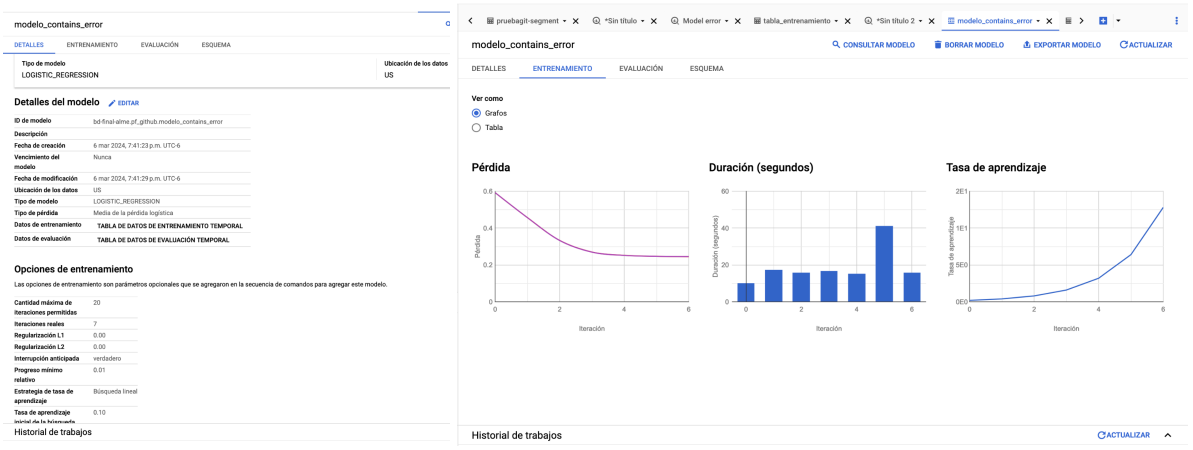


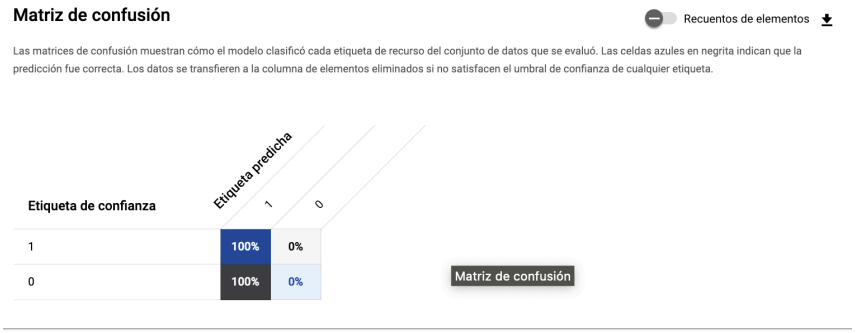




Visualizaciones Looker

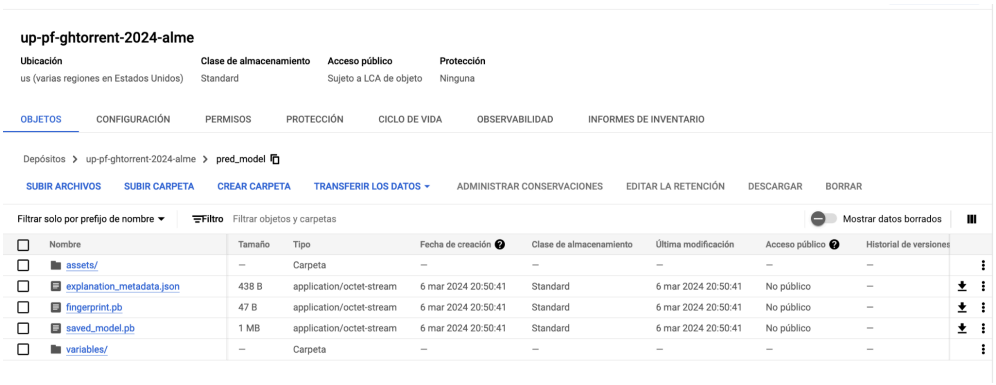
Modelo Regresión BigQueryML



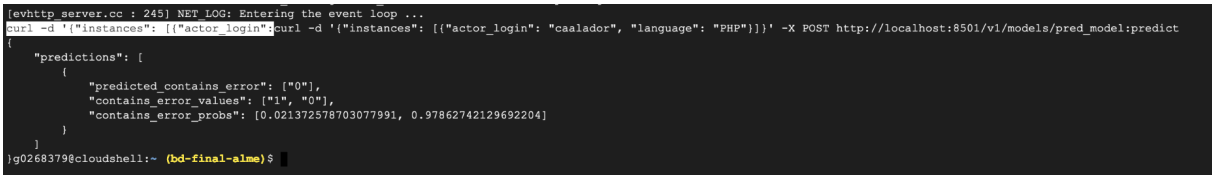


Resultados prueba datos test

Modelo Exportado



Pruebas API



Configuración versión

Pre-built container settings

Python version *
3.7
Select the Python version you used to train the model

Framework
TensorFlow

Framework version
1.15.0

ML runtime version *
1.15

Model URI *
gs:// up-pf-ghtorrent-2024-alme/pred_model EXPLORAR
Cloud Storage path to the entire SavedModel directory. Learn more

Online prediction deployment

Scaling
Auto scaling

Cantidad mínima de nodos
1
Keeping a minimum number of nodes running all the time will avoid dropping requests due to nodes initialization after the service has scaled down. This setting can increase cost, as you pay for the nodes even when no predictions are served.

Machine type *
n1-standard-4, 4 vCPUs, 15 GB memory

Tipo de acelerador
Recuento de aceleradores

Service account
Specifies the service account for resource access control.

GUARDAR BORRAR CANCELAR