

ML Using GCP Tools - Design and Implementation

Overview

Status	IN PROGRESS
Owner	@ Seethalakshmi Rengaraman
Goals	1. Document Design and Implementation Detail 2. Review and update based on feedback
On this page	<ul style="list-style-type: none">OverviewRequirementsDesign and Implementation DetailOpen items/issues

Requirements

Product requirements

MAML - Ship Date Estimation

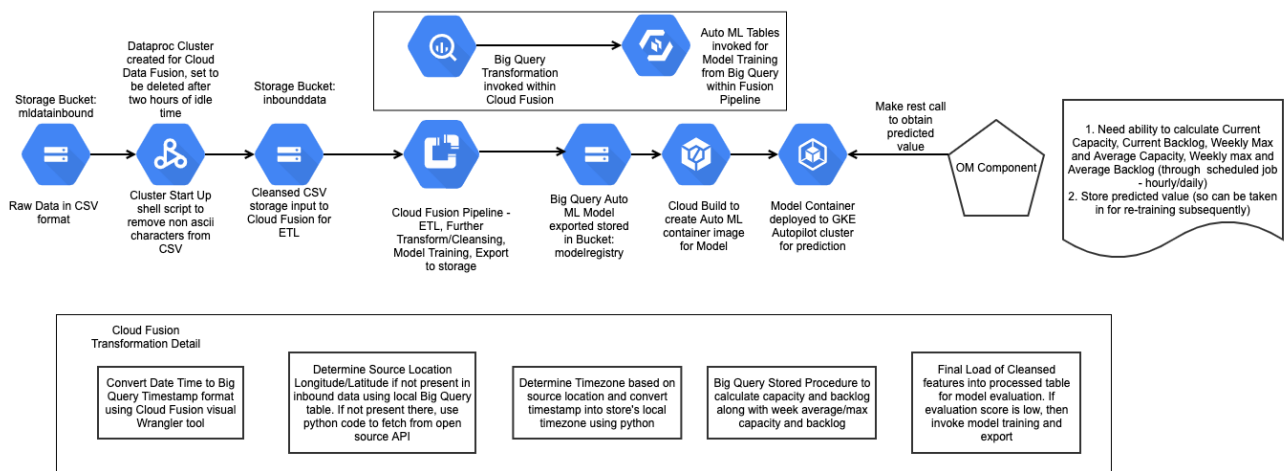
Problem statements

- Data in MA system is the source of truth for many customers. This rich data can help to make intelligent decisions for its customers
- With neural network based Auto ML tools of GCP available, our concentration only needs to be on feeding rich, essential and cleansed feature data to the machine and it will truly learn and take care of identifying the required patterns and predict /classify (minimal code)
- Reduces time to production and operational cost
- More models could be built and deployed within shorter duration

Success metrics

- High Model accuracy - Actual Label close to predicted label 95% of the time (ignoring outliers)
- Deployed model in High Availability secure containerized Kubernetes Cluster for serving through rest API with response time SLA of 30 ms per request, 1000 transactions/sec, 99% of the time across multiple customers
- Implemented near fully-automated CI-CD pipeline ("Near" since model re-training needs manual involvement to review model accuracy before deployment, cannot rely on statistical scores only <https://www.kdnuggets.com/2020/07/r-squared-predictive-capacity-statistical-adequacy.html>)

Design and Implementation Detail



- Data is originally obtained from production into "mldatainbound" bucket as a zipped csv file

- Create a new Data proc cluster for running Cloud Data Fusion pipeline. Data proc cluster comes up with initialization action - startup script to Cleanse data for ASCII characters and move to "inbounddata" bucket for further processing. Also Move processed files in mldatainbound bucket to archive. Additionally, data proc cluster also includes pyspark connector with big query if required to perform transformations using pyspark and store in big query.

```
gcloud beta dataproc clusters create mldataproccluster --bucket dataproclogs --region us-east1 --zone us-east1-b --master-machine-type c2-standard-4 --master-boot-disk-type pd-ssd --master-boot-disk-size 500 --num-workers 2 --worker-machine-type c2-standard-4 --worker-boot-disk-type pd-ssd --worker-boot-disk-size 500 --image-version 2.0-debian10 --max-idle 7200s --project gcproject1 --initialization-actions 'gs://mlprojectstartupscripts/startupscript.sh',gs://goog-dataproc-initialization-actions-us-east1/connectors/connectors.sh --metadata gcs-connector-version=2.2.0 --metadata bigquery-connector-version=1.2.0 --metadata spark-bigquery-connector-version=0.20.0
```

Start up script contents:

```
#!/bin/bash

##Remove existing files
sudo gsutil -m rm gs://inbounddata/*SHIP*DATE*
sudo gsutil -m rm gs://inbounddata/*CAPACITY*
##Create Directory and Copy data to directory
cd /home
sudo mkdir inbounddata
cd /home/inbounddata
sudo gsutil -m cp -r gs://mldatainbound/*SHIP*DATE* .
sudo gsutil -m cp -r gs://mldatainbound/*CAPACITY* .
##gunzip the data
sudo gunzip *

##remove non ascii characters
for file in *.csv
do
    charset=$(file -bi "$file" | grep -o 'charset.*'|cut -f2- -d=)
    sudo bash -c "sudo iconv -c -f "${charset}" -t ascii//TRANSLIT
"$file" > "$file.new"
    sudo chmod 775 *
    sudo mv -f "$file.new" "$file"
done

##Move to Cloud Storage
sudo gsutil -m mv *.csv gs://inbounddata

##Remove temporary directory
cd /home
sudo rm -r inbounddata

##Move processed files to archive
gsutil -m mv -r gs://mldatainbound/* gs://mldatainbound/Archive
```

Use Data Fusion to:

- Further transform Date Time field into Big Query compatible timestamp fields along with few data type conversions using Visual Wrangler tool and then Load into Big Query Staging table "RawOrderDetailsStaging"

- Once Data is in staging table, it is determined if there are any source locations with latitude/longitude missing and corresponding timezone is missing. If missing, these are updated from local table in big query TimeZoneFetch. If TimeZoneFetch doesn't have the details, this is fetched using Python code sample below.

```

• # Import required modules
import requests
from google.cloud import bigquery
from timezonefinder import TimezoneFinder

# Construct a BigQuery client object.
client = bigquery.Client()
#Query for all records that require latitude, longitude and
corresponding timezone
query = """
        SELECT SOURCE_CITY,POSTALCODE_CORRECTED SOURCE_POSTALCODE,
SOURCE_COUNTRY FROM `ma-rd-devils-sren.OrderData.TimeZoneFetch`
WHERE SOURCE_LATITUDE IS NULL OR SOURCE_LONGITUDE IS NULL OR
TIMEZONE IS NULL;
    """

latitude = ''
longitude = ''
query_job = client.query(query) # Make an API request to run the
query
print("The query data:")
for row in query_job:
    print("SOURCE_CITY={}, SOURCE_POSTALCODE={}".format(row
[0], row["SOURCE_POSTALCODE"]))
    #For each postal code and country fetch latitude and
longitude
    try:
        url = "https://nominatim.openstreetmap.org/search/?
postalcode={}&country={}&format=json".format(row
["SOURCE_POSTALCODE"],row["SOURCE_COUNTRY"])
        response = requests.get(url).json()
        latitude = response[0]["lat"]
        longitude = response[0]["lon"]
    except Exception as e:
        #If unable to fetch latitude and longitude by postal code
and country, fetch by city and country
        try:
            url = "https://nominatim.openstreetmap.org
/search/?city='{}'&country='{}'&format=json".format(row
["SOURCE_CITY"],row["SOURCE_COUNTRY"])
            response = requests.get(url).json()
            latitude = response[0]["lat"]
            longitude = response[0]["lon"]
        except Exception as e:
            print("Unexpected error occurred during lat
/long fetch", e )
        if bool(latitude):
            try:

```

```

#Fetch Time zone by latitude and longitude
tf = TimezoneFinder()
timezone = tf.timezone_at(lng=float
(longitude), lat=float(latitude))
print(timezone)
#Update latitude, longitude and timezone
query = """

                                UPDATE `ma-rd-devils-sren.
OrderData.TimeZoneFetch`
                                SET
SOURCE_LATITUDE = @latitude, SOURCE_LONGITUDE = @longitude,
                                TIMEZONE =
@timezone
                                WHERE
POSTALCODE_CORRECTED = @postalcode
                                AND SOURCE_COUNTRY
= @country;

                                """
job_config = bigquery.QueryJobConfig(
query_parameters=[
bigquery.ScalarQueryParameter("latitude",
"STRING", latitude),
bigquery.ScalarQueryParameter("longitude",
"STRING", longitude),
bigquery.ScalarQueryParameter("timezone",
"STRING", timezone),
bigquery.ScalarQueryParameter
("postalcode", "STRING", row["SOURCE_POSTALCODE"]),
bigquery.ScalarQueryParameter("country",
"STRING", row["SOURCE_COUNTRY"]),
]
)
query_job = client.query(query,
job_config=job_config)
print(f'Result:{query_job.result()}')
')
except Exception as e:
    print("Unexpected error occurred during
update", e )

```

- Once timezone is fetched, RawOrderDetailsStaging is updated to store all date times in store's local time zone.
- Calculate Current Backlog, Current Capacity, max capacity and load to capacity table.

❌ Current Backlog/Utilization Capacity is obtained from production but it doesn't seem to reflect the right numbers and hence this calculation. This calculation references [Store Capacity Utilization Updates](#)

- Sample Big Query script for this calculation:

```

BEGIN

```

```

TRUNCATE TABLE `gcpproject1.OrderData.
RawCapacityUtilization`;
CREATE TEMPORARY TABLE rawcap
AS
WITH ordercreatefulfill AS
    (SELECT count(ORDER_ID) ordercount,0 fulfillcount,
ORG_ID,LOCATION,LOCATION_TYPE_ID,
        EXTRACT(DAYOFWEEK FROM
ORDER_LINE_CREATED_TIMESTAMP) EVENT_DAY_OF_WEEK,FORMAT_TIMESTAMP('%
Y-%U',ORDER_LINE_CREATED_TIMESTAMP) EVENT_YEAR_WEEK,
        CASE WHEN EXTRACT(HOUR FROM
TIMESTAMP_TRUNC(ORDER_LINE_CREATED_TIMESTAMP,HOUR)) = 23 THEN
TIMESTAMP_ADD(TIMESTAMP_TRUNC(ORDER_LINE_CREATED_TIMESTAMP,HOUR),
INTERVAL 58 MINUTE)
ELSE TIMESTAMP_ADD(TIMESTAMP_TRUNC(ORDER_LINE_CREATED_TIMESTAMP,
HOUR),INTERVAL 60 MINUTE) END QUERY_TIME
FROM `gcpproject1.OrderData.RawOrderDetails`
/*WHERE LOCATION ='3755'
AND DATE(ORDER_LINE_CREATED_TIMESTAMP) >= '2019-03-14'
AND DATE(ORDER_LINE_CREATED_TIMESTAMP) <= '2019-03-18'*/
GROUP BY ORG_ID,LOCATION,LOCATION_TYPE_ID,
EXTRACT(DAYOFWEEK FROM ORDER_LINE_CREATED_TIMESTAMP),
FORMAT_TIMESTAMP('%Y-%U',ORDER_LINE_CREATED_TIMESTAMP),CASE WHEN
EXTRACT(HOUR FROM TIMESTAMP_TRUNC(ORDER_LINE_CREATED_TIMESTAMP,
HOUR)) = 23 THEN TIMESTAMP_ADD(TIMESTAMP_TRUNC
(ORDER_LINE_CREATED_TIMESTAMP,HOUR),INTERVAL 58 MINUTE)
ELSE TIMESTAMP_ADD(TIMESTAMP_TRUNC(ORDER_LINE_CREATED_TIMESTAMP,
HOUR),INTERVAL 60 MINUTE) END
UNION DISTINCT
SELECT 0 ordercount,count(ORDER_ID) fulfillcount,ORG_ID,LOCATION,
LOCATION_TYPE_ID,
EXTRACT(DAYOFWEEK FROM SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP))
EVENT_DAY_OF_WEEK,FORMAT_TIMESTAMP('%Y-%U',SAFE_CAST
(FULFILLMENT_DATE AS TIMESTAMP)) EVENT_YEAR_WEEK,
CASE WHEN EXTRACT(HOUR FROM TIMESTAMP_TRUNC(SAFE_CAST
(FULFILLMENT_DATE AS TIMESTAMP),HOUR)) = 23 THEN TIMESTAMP_ADD
(TIMESTAMP_TRUNC(SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP),HOUR),
INTERVAL 59 MINUTE)
ELSE TIMESTAMP_ADD(TIMESTAMP_TRUNC(SAFE_CAST(FULFILLMENT_DATE AS
TIMESTAMP),HOUR),INTERVAL 61 MINUTE) END QUERY_TIME
FROM `gcpproject1.OrderData.RawOrderDetails`
WHERE FULFILLMENT_DATE NOT LIKE '%\\N%'
/*AND LOCATION ='3755'
AND DATE(SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP)) >= '2019-03-14'
AND DATE(SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP)) <= '2019-03-
18'*/
GROUP BY ORG_ID,LOCATION,LOCATION_TYPE_ID,
EXTRACT(DAYOFWEEK FROM SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP)),
FORMAT_TIMESTAMP('%Y-%U',SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP)),
CASE WHEN EXTRACT(HOUR FROM TIMESTAMP_TRUNC(SAFE_CAST

```

```

(FULFILLMENT_DATE AS TIMESTAMP),HOUR)) = 23 THEN TIMESTAMP_ADD
(TIMESTAMP_TRUNC(SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP),HOUR),
INTERVAL 59 MINUTE)
ELSE TIMESTAMP_ADD(TIMESTAMP_TRUNC(SAFE_CAST(FULFILLMENT_DATE AS
TIMESTAMP),HOUR),INTERVAL 61 MINUTE) END
ORDER BY QUERY_TIME),
maxcapacity AS
(SELECT ORG_ID,LOCATION_ID,MAX(SAFE_CAST(MAX_CAPACITY_UTILIZED AS
FLOAT64)) MAX_CAPACITY_UTILIZED
FROM `gcpproject1.OrderData.RawCapacityUtilization1`
WHERE MAX_CAPACITY_UTILIZED NOT LIKE '%\\N%'
GROUP BY ORG_ID,LOCATION_ID),
rawcapacity AS
(SELECT oc.ORG_ID ORG_ID,oc.LOCATION LOCATION_ID,oc.
LOCATION_TYPE_ID LOCATION_TYPE_ID,
oc.EVENT_YEAR_WEEK YEAR_WEEK,oc.EVENT_DAY_OF_WEEK DAY_OF_WEEK,oc.
ordercount ORDER_COUNT,oc.fulfillcount FULFILL_COUNT,
oc.QUERY_TIME, SUM(oc.ordercount) OVER (
PARTITION BY oc.ORG_ID,oc.LOCATION_TYPE_ID,oc.LOCATION,oc.
EVENT_YEAR_WEEK,oc.EVENT_DAY_OF_WEEK
ORDER BY oc.QUERY_TIME
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
) AS TOTAL_CAPACITY_UTILIZED,MAX_CAPACITY_UTILIZED ,
SUM(oc.ordercount-oc.fulfillcount) OVER (
PARTITION BY oc.ORG_ID,oc.LOCATION_TYPE_ID,oc.LOCATION
ORDER BY oc.QUERY_TIME
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS
CURRENT_BACKLOG
FROM ordercreatefulfill oc
LEFT JOIN maxcapacity mc ON oc.ORG_ID = mc.ORG_ID AND oc.LOCATION
= mc.LOCATION_ID)
SELECT ORG_ID,LOCATION_ID,LOCATION_TYPE_ID,QUERY_TIME,DAY_OF_WEEK,
YEAR_WEEK,MAX_CAPACITY_UTILIZED,ORDER_COUNT,FULFILL_COUNT,
CURRENT_BACKLOG,
null MAX_BACKLOG,
COALESCE(TOTAL_CAPACITY_UTILIZED,CURRENT_BACKLOG)
TOTAL_CAPACITY_UTILIZED,CASE WHEN LAST_VALUE(CURRENT_BACKLOG)
OVER (PARTITION BY ORG_ID,LOCATION_TYPE_ID,LOCATION_ID,YEAR_WEEK,
DAY_OF_WEEK
ORDER BY QUERY_TIME ASC,CURRENT_BACKLOG DESC
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) < 0
THEN 0 ELSE
LAST_VALUE(CURRENT_BACKLOG)
OVER (PARTITION BY ORG_ID,LOCATION_TYPE_ID,LOCATION_ID,YEAR_WEEK,
DAY_OF_WEEK
ORDER BY QUERY_TIME ASC,CURRENT_BACKLOG DESC
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
END LAST_BACKLOG_FOR_THE_DAY
FROM rawcapacity
WHERE EXTRACT(DATE FROM QUERY_TIME) < PARSE_DATE("%F", "2021-06-
```

```

25" )
ORDER BY LOCATION_ID, QUERY_TIME;

UPDATE rawcap a
SET TOTAL_CAPACITY_UTILIZED =
TOTAL_CAPACITY_UTILIZED +
COALESCE((SELECT DISTINCT COALESCE(LAST_BACKLOG_FOR_THE_DAY,0)
FROM rawcap WHERE ORG_ID = a.ORG_ID AND LOCATION_ID = a.
LOCATION_ID AND LOCATION_TYPE_ID =a.LOCATION_TYPE_ID
AND EXTRACT(DATE FROM QUERY_TIME) = DATE_SUB(EXTRACT(DATE FROM a.
QUERY_TIME),INTERVAL 1 DAY)),0)
WHERE TOTAL_CAPACITY_UTILIZED >=0;

INSERT INTO `gcpproject1.OrderData.RawCapacityUtilization`
(ORG_ID,LOCATION_ID,LOCATION_TYPE_ID,QUERY_TIME,DAY_OF_WEEK,
YEAR_WEEK,MAX_CAPACITY_UTILIZED,CURRENT_BACKLOG,MAX_BACKLOG,
TOTAL_CAPACITY_UTILIZED)
SELECT ORG_ID,LOCATION_ID,LOCATION_TYPE_ID,QUERY_TIME,DAY_OF_WEEK,
YEAR_WEEK,COALESCE(CAST(MAX_CAPACITY_UTILIZED AS STRING),'\N'),MIN
(CURRENT_BACKLOG),COALESCE(CAST(MAX_BACKLOG AS STRING),'\N'),
MAX(TOTAL_CAPACITY_UTILIZED)
FROM rawcap
GROUP BY ORG_ID,LOCATION_ID,LOCATION_TYPE_ID,QUERY_TIME,
DAY_OF_WEEK,YEAR_WEEK,COALESCE(CAST(MAX_CAPACITY_UTILIZED AS
STRING),'\N'),COALESCE(CAST(MAX_BACKLOG AS STRING),'\N');

END

```

- Once capacity data is available, Fetch required features into a “Processed Data” table that will be fed to the model ignoring nulls. This includes:

Feature Name	Description	Data Type
ORG_ID	Organization Id (Eg: Michaels or GroupDynamite)	String
ORDER_TYPE_ID	Order Type (Eg: Web, Available, Damaged)	String
ORDER_LINE_QUANTITY	Quantity on Order Line	Float
IS_GIFT	Determines if item is a gift item (0 or 1)	Integer
DELIVERY_METHOD_ID	Method of Delivery - Whether Ship to Address, Pickup from Store, Ship to Store	String
ORDER_CREATE_TO_ALLOCATION_IN_HOURS	Difference between Order Line Allocation Time stamp and Order Line Creation Time stamp in hours (0 decimal point)	Float
SERVICE_LEVEL	Concatenation of Carrier and Service level Example: “UPS,UPS Ground”	String
SOURCE_LOCATION	Concatenation of Location Id and Location Type (from which order might be fulfilled) Example: “51-Store”	String
SOURCE_STATE	Source Location State	String

SOURCE_COUNTRY	Source Location Country	String
SOURCE_LATITUDE	Source Location Latitude (Useful for determining temperate zones)	Float
SOURCE_LONGITUDE	Source Location Longitude	Float
ORDER_CREATE_TO_RELEASE_IN_HOURS	Difference between Order Line Release Time and Order Line Create Time in Hours (0 decimal point)	Float
CREATION_DAY_OF_WEEK	Order Line Creation Day of Week (from 1 to 7 with 1 being Sunday)	Integer
CREATION_YEAR_WEEK	Order Line Creation Year week where number is a value from 00-53 (Sunday being the first day of the week)	String
CREATION_MONTH	Order Line Creation Month	Integer
CREATION_HOUR	Order Line Creation Hour	Integer
CURRENT_BACKLOG	Current Backlog (Last Hour or two) for the source location	Float
TOTAL_CAPACITY_UTILIZED	Current Utilization capacity (Last Hour or two) for source location	Float
SEVEN_DAY_AVERAGE_CAPACITY	Weekly Average Capacity (Based on above CREATION_YEAR_WEEK) . If weekly average was low but sudden surge on particular days, affects ship date (can be calculated once a day through a job or hourly /bi-hourly along with current backlog /utilization capacity)	Float
SEVEN_DAY_MAX_CAPACITY	Weekly Max Capacity (linked to Weekly average capacity as explained above)	Float
SEVEN_DAY_AVERAGE_BACKLOG	Weekly Average Backlog Based on above CREATION_YEAR_WEEK) . If weekly average was low but sudden surge on particular days, affects ship date (can be calculated once a day through a job or hourly /bi-hourly along with current backlog /utilization capacity)	Float
SEVEN_DAY_MAX_BACKLOG	Weekly Max Backlog (linked to Weekly average backlog as explained above)	Float

Sample Big Query Script used to fetch the details and load into training table “ProcessedOrderDetails”:

```

BEGIN
CREATE OR REPLACE TABLE `OrderData.ProcessedOrderDetails`
AS
WITH orderdetails AS
(
SELECT DISTINCT ORG_ID,
ORDER_TYPE_ID,
ITEM_ID,
ORDER_LINE_QUANTITY,
IS_GIFT,
DELIVERY_METHOD_ID,
IFNULL(SAFE_CAST(ALLOCATION_TIMESTAMP AS TIMESTAMP),
ORDER_LINE_CREATED_TIMESTAMP) ALLOCATION_TIMESTAMP,

```



```

EXTRACT(DAYOFWEEK FROM ORDER_LINE_CREATED_TIMESTAMP)
CREATION_DAY_OF_WEEK,
FORMAT_TIMESTAMP('%Y-%U',ORDER_LINE_CREATED_TIMESTAMP)
CREATION_YEAR_WEEK,
EXTRACT(HOUR FROM ORDER_LINE_CREATED_TIMESTAMP) CREATION_HOUR,
EXTRACT(MONTH FROM ORDER_LINE_CREATED_TIMESTAMP) CREATION_MONTH,
IF(REQUESTED_CARRIER = '\\N',SHIPPED_CARRIER,REQUESTED_CARRIER)||','||IF
(REQUESTED_SERVICE_LEVEL = '\\N',SHIPPED_SERVICE_LEVEL,
REQUESTED_SERVICE_LEVEL) SERVICE_LEVEL,
LOCATION,
LOCATION_TYPE_ID,
IF(SOURCE_STATE != '\\N',UPPER(SOURCE_STATE),SOURCE_STATE) SOURCE_STATE,
SOURCE_COUNTRY,
SAFE_CAST(SOURCE_LATITUDE AS FLOAT64) SOURCE_LATITUDE,
SAFE_CAST(SOURCE_LONGITUDE AS FLOAT64) SOURCE_LONGITUDE,
IFNULL(SAFE_CAST(RELEASE_TIMESTAMP AS TIMESTAMP),
ORDER_LINE_CREATED_TIMESTAMP) RELEASE_TIMESTAMP,
SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP) FULFILLMENT_DATE,
EXTRACT(DAYOFWEEK FROM SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP))
FULFIL_DAY_OF_WEEK,
FORMAT_TIMESTAMP('%Y-%U',SAFE_CAST(FULFILLMENT_DATE AS TIMESTAMP))
FULFIL_YEAR_WEEK,
ORDER_LINE_CREATED_TIMESTAMP
FROM `gcpproject1.OrderData.RawOrderDetails`
--WHERE LOCATION = '2020'
),
capacity AS
(SELECT DISTINCT ORG_ID CAPACITY_ORG_ID,LOCATION_ID,DAY_OF_WEEK,
YEAR_WEEK,
TOTAL_CAPACITY_UTILIZED,CURRENT_BACKLOG,TIMESTAMP_TRUNC(QUERY_TIME,
HOUR) QUERY_TIME
FROM `gcpproject1.OrderData.RawCapacityUtilization`
--WHERE LOCATION_ID = '2020'
)
SELECT
DISTINCT ORG_ID,
ORDER_TYPE_ID,
ORDER_LINE_QUANTITY,
IS_GIFT,
DELIVERY_METHOD_ID,
CASE WHEN TIMESTAMP_DIFF(ALLOCATION_TIMESTAMP,
ORDER_LINE_CREATED_TIMESTAMP,MINUTE)/60 < 0
THEN 0
WHEN TIMESTAMP_DIFF(ALLOCATION_TIMESTAMP,ORDER_LINE_CREATED_TIMESTAMP,
MINUTE)/60 < 1
THEN 1
ELSE ROUND(TIMESTAMP_DIFF(ALLOCATION_TIMESTAMP,
ORDER_LINE_CREATED_TIMESTAMP,MINUTE)/60,0) END
ORDER_CREATE_TO_ALLOCATION_IN_HOURS,
CASE WHEN LOWER(DELIVERY_METHOD_ID) LIKE '%pick%store%' THEN

```

```

        'IN_STORE'
ELSE
    SERVICE_LEVEL END SERVICE_LEVEL,
    LOCATION || '-' || LOCATION_TYPE_ID SOURCE_LOCATION,
    TRIM(SOURCE_STATE) SOURCE_STATE,
    TRIM(SOURCE_COUNTRY) SOURCE_COUNTRY,
    SOURCE_LATITUDE,
    SOURCE_LONGITUDE,
    CASE WHEN TIMESTAMP_DIFF(RELEASE_TIMESTAMP,
ORDER_LINE_CREATED_TIMESTAMP, MINUTE)/60 < 0
    THEN 0
    WHEN TIMESTAMP_DIFF(RELEASE_TIMESTAMP, ORDER_LINE_CREATED_TIMESTAMP,
MINUTE)/60 < 1
    THEN 1
    ELSE ROUND(TIMESTAMP_DIFF(RELEASE_TIMESTAMP,
ORDER_LINE_CREATED_TIMESTAMP, MINUTE)/60, 0) END
ORDER_CREATE_TO_RELEASE_IN_HOURS,
    CASE WHEN TIMESTAMP_DIFF(FULFILLMENT_DATE,
ORDER_LINE_CREATED_TIMESTAMP, MINUTE)/60 < 0 THEN 0
    WHEN TIMESTAMP_DIFF(FULFILLMENT_DATE, ORDER_LINE_CREATED_TIMESTAMP,
MINUTE)/60 < 1 THEN 1
    ELSE ROUND(TIMESTAMP_DIFF(FULFILLMENT_DATE,
ORDER_LINE_CREATED_TIMESTAMP, MINUTE)/60, 0) END
FULFILLMENT_DATE_IN_HOURS,
    CREATION_DAY_OF_WEEK,
    CREATION_YEAR_WEEK,
    CREATION_MONTH,
    CREATION_HOUR,
    CASE WHEN CURRENT_BACKLOG < 0 THEN 0 ELSE ROUND(CURRENT_BACKLOG, 0) END
CURRENT_BACKLOG,
    ROUND(TOTAL_CAPACITY_UTILIZED, 0) TOTAL_CAPACITY_UTILIZED,
    0.0 SEVEN_DAY_AVERAGE_CAPACITY,
    0.0 SEVEN_DAY_MAX_CAPACITY,
    0.0 SEVEN_DAY_AVERAGE_BACKLOG,
    0.0 SEVEN_DAY_MAX_BACKLOG
FROM orderdetails o
INNER JOIN CAPACITY c ON o.ORG_ID = c.CAPACITY_ORG_ID
                        AND o.LOCATION = c.LOCATION_ID
                        AND o.CREATION_YEAR_WEEK = c.YEAR_WEEK
                        AND CAST(o.CREATION_DAY_OF_WEEK AS INT64) = c.
DAY_OF_WEEK
                        AND o.ORDER_LINE_CREATED_TIMESTAMP >= TIMESTAMP_SUB
(c.QUERY_TIME, INTERVAL 1 HOUR)
                        AND o.ORDER_LINE_CREATED_TIMESTAMP <= c.QUERY_TIME;
-- AND DATE(o.ORDER_LINE_CREATED_TIMESTAMP) >= '2019-
03-14'
-- AND DATE(o.ORDER_LINE_CREATED_TIMESTAMP) <= '2019-
03-18'

UPDATE `gcpproject1.OrderData.ProcessedOrderDetails` a

```

```

SET SEVEN_DAY_AVERAGE_CAPACITY = (SELECT ROUND(AVG
(TOTAL_CAPACITY_UTILIZED),0) FROM `ma-rd-devils-sren.OrderData.
RawCapacityUtilization`
WHERE YEAR_WEEK = a.CREATION_YEAR_WEEK
AND LOCATION_ID = SUBSTR(a.SOURCE_LOCATION,1,INSTR(a.
SOURCE_LOCATION,'-')-1)
AND ORG_ID = a.ORG_ID
GROUP BY ORG_ID,LOCATION_ID,YEAR_WEEK)
WHERE SEVEN_DAY_AVERAGE_CAPACITY =0.0;

UPDATE `gcpproject1.OrderData.ProcessedOrderDetails` a
SET SEVEN_DAY_MAX_CAPACITY = (SELECT ROUND(MAX(TOTAL_CAPACITY_UTILIZED),
0) FROM `ma-rd-devils-sren.OrderData.RawCapacityUtilization`
WHERE YEAR_WEEK = a.CREATION_YEAR_WEEK
AND LOCATION_ID = SUBSTR(a.SOURCE_LOCATION,1,INSTR(a.
SOURCE_LOCATION,'-')-1)
AND ORG_ID = a.ORG_ID
GROUP BY ORG_ID,LOCATION_ID,YEAR_WEEK)
WHERE SEVEN_DAY_MAX_CAPACITY =0.0;

UPDATE `gcpproject1.OrderData.ProcessedOrderDetails` a
SET SEVEN_DAY_AVERAGE_BACKLOG = (SELECT CASE WHEN ROUND(AVG
(CURRENT_BACKLOG),2) <0 THEN 0 ELSE ROUND(AVG(CURRENT_BACKLOG),0) END
FROM `ma-rd-devils-sren.OrderData.RawCapacityUtilization`
WHERE YEAR_WEEK = a.CREATION_YEAR_WEEK
AND LOCATION_ID = SUBSTR(a.SOURCE_LOCATION,1,INSTR(a.
SOURCE_LOCATION,'-')-1)
AND ORG_ID = a.ORG_ID
GROUP BY ORG_ID,LOCATION_ID,YEAR_WEEK),
SEVEN_DAY_MAX_BACKLOG = (SELECT CASE WHEN ROUND(MAX(CURRENT_BACKLOG),2)
<0 THEN 0 ELSE ROUND(MAX(CURRENT_BACKLOG),0) END FROM `ma-rd-devils-
sren.OrderData.RawCapacityUtilization`
WHERE YEAR_WEEK = a.CREATION_YEAR_WEEK
AND LOCATION_ID = SUBSTR(a.SOURCE_LOCATION,1,INSTR(a.
SOURCE_LOCATION,'-')-1)
AND ORG_ID = a.ORG_ID
GROUP BY ORG_ID,LOCATION_ID,YEAR_WEEK)
WHERE 1=1;

END;

```

- Auto ML Training is performed

```

CREATE OR REPLACE MODEL
`gcpproject1.OrderData.ShipDateDeterminationBQModel` OPTIONS
(model_type='AUTOML_REGRESSOR',
input_label_cols=['FULFILLMENT_DATE_IN_HOURS'],
budget_hours=5.0,
OPTIMIZATION_OBJECTIVE='MINIMIZE_RMSLE')

```

```
AS
SELECT *
FROM `ma-rd-devils-sren.OrderData.ProcessedOrderDetails`
```

- Evaluate for accuracy by fetching data for testing and processing similar to above steps and inserting into test table "ShipDateInstances" similar to "ProcessedOrderDetails" (Training table). Capacity calculation step is common to training and test data to ensure accurate evaluation

```
SELECT
*
FROM
  ML.EVALUATE(MODEL `gcpproject1.OrderData.
ShipDateDeterminationBQModel`,
  (SELECT
    * EXCEPT (ACTUAL_LABEL_VALUE), ACTUAL_LABEL_VALUE
  FROM
    `gcpproject1.OrderData.ShipDateInstances`))
```

ShipDateDeterminationBQModel

DETAILS

TRAINING

EVALUATION

SCHEMA

Mean absolute error	20.3975
Mean squared error	2,672.7923
Mean squared log error	0.9238
Median absolute error	
R squared	0.2018

<https://www.kdnuggets.com/2020/07/r-squared-predictive-capacity-statistical-adequacy.html> - This very clearly states that R squared alone cannot determine the model accuracy.

Reviewing Results of few locations exported we see predicted value to be very close to actual label mostly and predicted label is stable and varies with restraint unlike outliers seen in the actual label



results-20210623-220545.csv



results-20210620-211504.csv



results-20210625-144220.csv

- After converting Predicted and Actual values from hours to days (Since we need to determine if we can ship in a day or more)
 - Predicted and Actual matches 59.5% of the time and difference of 1 or 2 days at the max for 93.5% of the time as seen below. The number "21323382" is total number of records in the prediction entity.


```
SELECT COUNT(1) ROW_COUNT, ABS(PREDICT_LABEL_IN_DAYS -  
ACTUAL_LABEL_IN_DAYS) PREDICT_ACTUAL_DIFF_IN_DAYS, ROUND((COUNT  
(1)/21323382) * 100, 2) PERCENT_MATCHING  
FROM `ma-rd-devils-sren.OrderData.Predictions`  
WHERE ABS(PREDICT_LABEL_IN_DAYS - ACTUAL_LABEL_IN_DAYS) < 20  
GROUP BY PREDICT_ACTUAL_DIFF_IN_DAYS  
ORDER BY PERCENT_MATCHING DESC
```

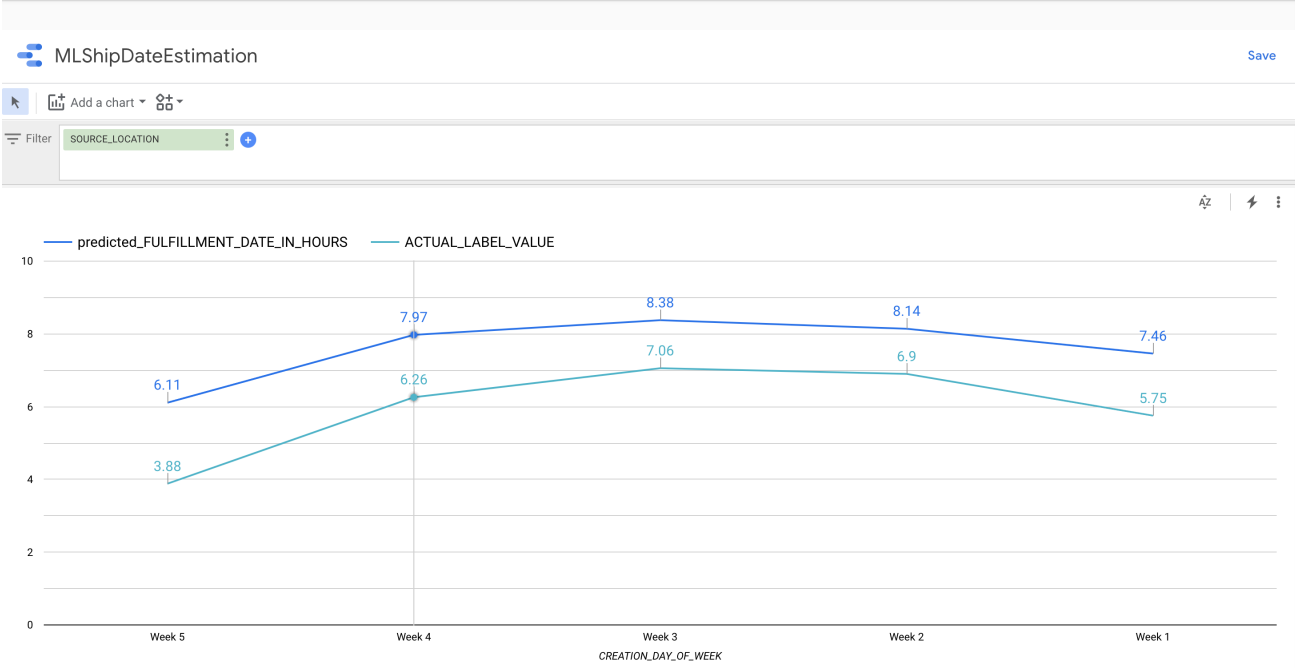
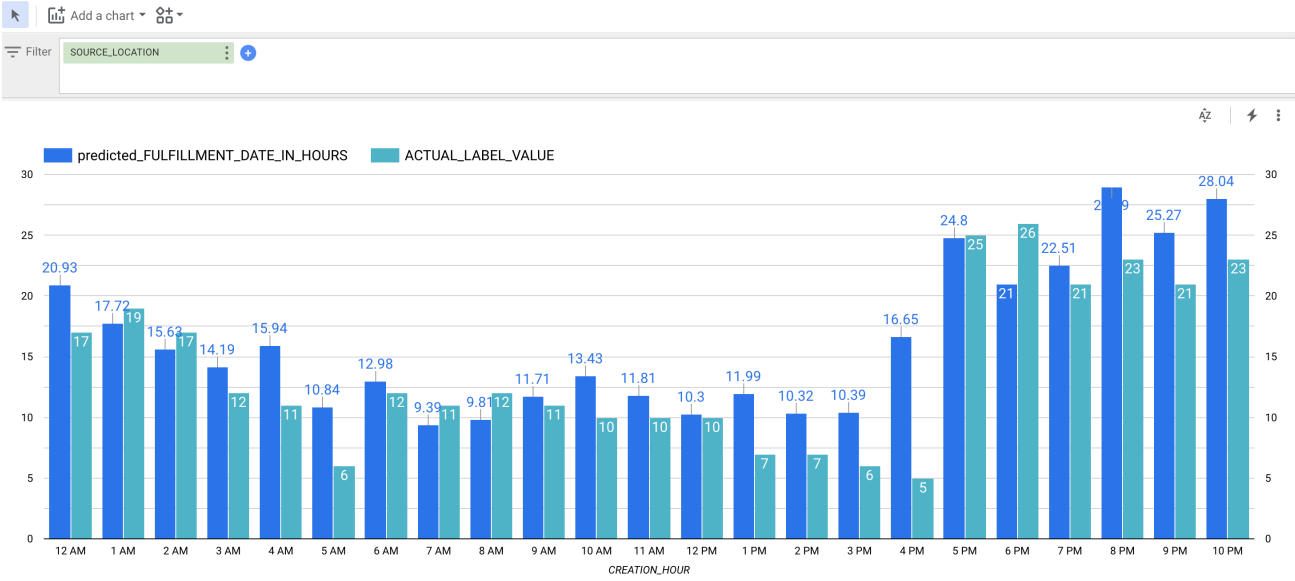
•

32	
33	<code>SELECT COUNT(1) ROW_COUNT, ABS(PREDICT_LABEL_IN_DAYS - ACTUAL_LABEL_IN_DAYS) PREDICT_ACTUAL_DIFF_IN_DAYS, ROUND(((COUNT(1)/21323382) * 100, 2) PERCENT_MATCHING</code>
34	<code>FROM `ma-rd-devils-sren.OrderData.Predictions`</code>
35	<code>WHERE ABS(PREDICT_LABEL_IN_DAYS - ACTUAL_LABEL_IN_DAYS) < 20</code>
36	<code>GROUP BY PREDICT_ACTUAL_DIFF_IN_DAYS</code>
37	<code>ORDER BY PERCENT_MATCHING DESC</code>
38	

Query results		SAVE RESULTS	EXPLORE DATA
Query complete (0.8 sec elapsed, 325.4 MB processed)			
Job information		Results	JSON Execution details
Row	ROW_COUNT	PREDICT_ACTUAL_DIFF_IN_DAYS	PERCENT_MATCHING
1	12692368	0.0	59.52

2	6009279	1.0	28.18
3	1086270	2.0	5.09
4	593188	3.0	2.78
5	371293	4.0	1.74
6	197532	5.0	0.93
7	95997	6.0	0.45
8	88300	7.0	0.41
9	45398	8.0	0.21
10	28381	9.0	0.13
11	19892	10.0	0.09
12	15014	11.0	0.07

- Use Google Data studio to visualize (Actual Label close to predicted label mostly, less time taken during store hours than outside store hours)
-  MLShipDateEstimation Save



- Export Model to model registry

```
EXPORT MODEL `gcpproject1.OrderData.ShipDateDeterminationBQModel`  
OPTIONS(URI = 'gs://modelregistry/ShipDateDetermination')
```

- Build container using gcloud build and Deploy to Kubernetes Cluster for prediction
 - Dockerfile

```
FROM gcr.io/cloud-automl-tables-public/model_server  
ADD ShipDateDetermination /models/default/0000001
```

- Kubernetes model serving file (model_serve.yaml)

```
---  
apiVersion: v1  
kind: Service  
metadata:  
  labels:  
    app: shipdatepredict  
    name: shipdatepredict  
    namespace: default  
spec:  
  ports:  
    - name: model-serving  
      port: 80  
      targetPort: "http-server"  
  selector:  
    app: shipdatepredict  
  type: ClusterIP  
---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  labels:  
    app: shipdatepredict  
    name: shipdatepredict-dep  
    namespace: default  
spec:  
  replicas: 2  
  selector:  
    matchLabels:  
      app: shipdatepredict  
  template:  
    metadata:  
      labels:  
        app: shipdatepredict  
        version: v1  
    spec:
```

```

containers:
- name: shipdatepredict
  image: gcr.io/gcpproject1/shipdateapi
  imagePullPolicy: Always
  livenessProbe:
    initialDelaySeconds: 30
    periodSeconds: 30
    tcpSocket:
      port: 8080
  ports:
  - name: http-server
    containerPort: 8080
  resources:
    limits:
      cpu: "20"
      memory: 20Gi
    requests:
      cpu: "1"
      memory: 1Gi

```

- Deployment shell script - to copy file from cloud to local, invoke gcloud build to build container image and deploy to autopilot GKE cluster "mlgkecluster"

```

#!/bin/bash
project="gcpproject1"
region="us-east1"
gcloud config set project $project
gkecluster=$(gcloud container clusters list |grep
"mlgkecluster")
if [[ ! -z $gkecluster ]];
then
    gcloud container clusters get-credentials mlgkecluster --
region $region --project=$project
    modelexport=""
    modelexport=$(gsutil ls gs://modelregistry
/ShipDateDetermination | grep saved_model)
    if [[ -z $modelexport ]];
    then
        echo "No New Trained Model"
    else
        gsutil -m cp -r gs://modelregistry/ShipDateDetermination .
        image="gcr.io/$project/shipdateapi"
        gcloud builds submit --tag $image .
        kubectl apply -f model_serve.yaml
        kubectl expose deployment shipdatepredict-dep --
name=shipdatepredictsvc --type=LoadBalancer --port 80 --target-
port 8080
    fi
fi

```


- Check if prediction is working after deployment to Kubernetes cluster. Use Instances.json file as below

```
{
  "instances": [
    {
      "ORG_ID": "michaels-us",
      "ORDER_TYPE_ID": "WEB-US",
      "ORDER_LINE_QUANTITY": "1.0",
      "IS_GIFT": "0",
      "DELIVERY_METHOD_ID": "ShipToAddress",
      "ORDER_CREATE_TO_ALLOCATION_IN_HOURS": "1.0",
      "SERVICE_LEVEL": "PROSHIP,TWODAY",
      "SOURCE_LOCATION": "4734-STORE",
      "SOURCE_STATE": "CA",
      "SOURCE_COUNTRY": "US",
      "SOURCE_LATITUDE": "38.0339",
      "SOURCE_LONGITUDE": "-122.5855",
      "ORDER_CREATE_TO_RELEASE_IN_HOURS": "1.0",
      "CREATION_DAY_OF_WEEK": "2",
      "CREATION_YEAR_WEEK": "2021-15",
      "CREATION_MONTH": "4",
      "CREATION_HOUR": "1",
      "CURRENT_BACKLOG": "179.0",
      "TOTAL_CAPACITY_UTILIZED": "246.0",
      "SEVEN_DAY_AVERAGE_CAPACITY": "259.0",
      "SEVEN_DAY_MAX_CAPACITY": "332.0",
      "SEVEN_DAY_AVERAGE_BACKLOG": "178.0",
      "SEVEN_DAY_MAX_BACKLOG": "306.0"
    },
    {
      "ORG_ID": "CAN",
      "ORDER_TYPE_ID": "Web",
      "ORDER_LINE_QUANTITY": "1.0",
      "IS_GIFT": "0",
      "DELIVERY_METHOD_ID": "ShipToStore",
      "ORDER_CREATE_TO_ALLOCATION_IN_HOURS": "17.0",
      "SERVICE_LEVEL": "EZPOST_ECOM,CPOST_EXP_STD",
      "SOURCE_LOCATION": "51-STORE",
      "SOURCE_STATE": "ON",
      "SOURCE_COUNTRY": "CA",
      "SOURCE_LATITUDE": "43.777708",
      "SOURCE_LONGITUDE": "-79.343251",
      "ORDER_CREATE_TO_RELEASE_IN_HOURS": "20.0",
      "CREATION_DAY_OF_WEEK": "5",
      "CREATION_YEAR_WEEK": "2021-17",
      "CREATION_MONTH": "4",
      "CREATION_HOUR": "13",
      "CURRENT_BACKLOG": "13.0",
      "TOTAL_CAPACITY_UTILIZED": "13.0",
    }
  ]
}
```

```

    "SEVEN_DAY_AVERAGE_CAPACITY": "27.0",
    "SEVEN_DAY_MAX_CAPACITY": "57.0",
    "SEVEN_DAY_AVERAGE_BACKLOG": "21.0",
    "SEVEN_DAY_MAX_BACKLOG": "55.0"
  }
]
}

```

- Get the external ip address exposed in kube server using command “kubectl get service” and run the following command to get output as follows (Two instances and hence two results of ship date in hours)

```

curl -X POST --data @./instances.json http://34.139.134.173:80
/predict

```

Output:

```

{"predictions": [18.78145408630371, 104.87823486328125]}

```

Load Test Results (using Locust - Distributed and deployed to same Kubernetes server):

Ran from:

2021/06/25 10:02 PM - 2021/06/26 10:12 PM with 30000+ requests per hour

2021/06/28 06:23 PM - 2021-06-29 07:16 PM with 60000+ requests per hour

Query in Logging between above intervals using:

```

resource.type="k8s_container"
resource.labels.project_id="gcpproject1"
resource.labels.location="us-east1"
resource.labels.cluster_name="mlgkecluster"
resource.labels.namespace_name="default"
labels.k8s-pod/app="shipdatepredict" severity>=DEFAULT
textPayload:predict

```

Sample response time in below report for part of the run (99% of the time 6 ms):



report_1624985...86.854082.html

? Open items/issues

- Once predicted value is obtained in hours, need to either convert to ship date without time or else need to validate if within store processing time, else add hours to get to open store time (may be needed for pickup from store on the same day)
- Need to include the time taken for the product to arrive at the store from manufacturer/warehouse before shipping to customer
- Possibility of improving predictability further using features like:
 - Labor (Number of employees)
 - Holidays/Weather
 - Item properties (Hazmat, product class/department etc)
 - Order Line Requested Delivery date/Hold before Ship date