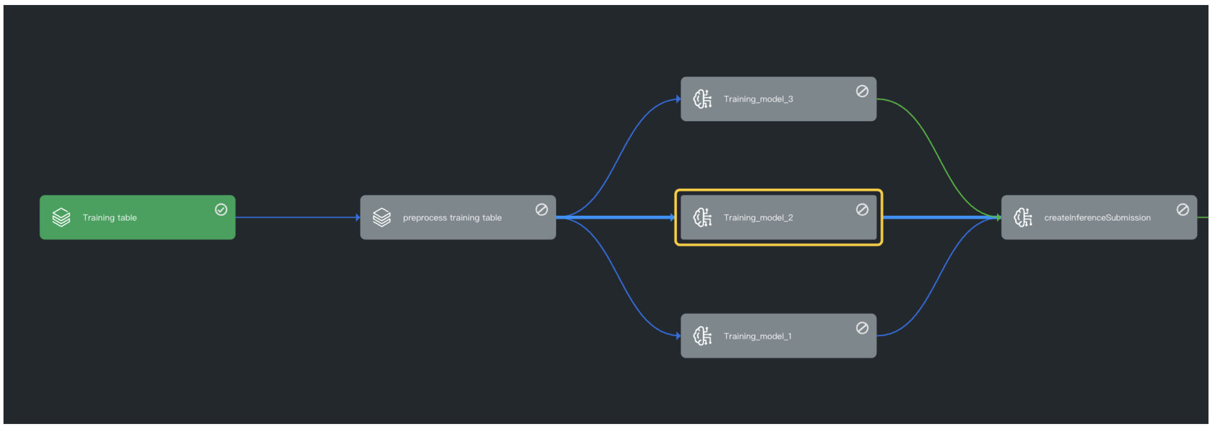
Training Table Design

1.Overall pipeline design



Pipeline Design (4 Stages):

1. Training Table Task – Join and integrate trainingcontext, wagondata, and tonnagedata into a unified total_training_table.

2. Preprocess Training Table Task – Perform preprocessing on the wide table (e.g., missing value handling, standard scaling).

3. Training Model – Implement inference models, including a feature selection module (e.g., Lasso regularization).

4. Create Submission – Store prediction results and submit to the leaderboard platform.

2.Training Table Design

2.1. Training Table Integration

2.1.1 Training Table Integration Design

Integration sql:

```
"""
WITH
tc AS (
  SELECT
    t.BaseCode             AS Tc_BaseCode,
    m.MappedBaseCode       AS Tc_BaseCode_Mapped,
    t.SectionBreakStartKM  AS Tc_SectionBreakStartKM,
    t.break_date           AS Tc_break_date,
    t.last_fail_if_available_otherwise_null  AS
Tc_last_fail_if_available_otherwise_null,
    t.r_date               AS Tc_r_date,
    t.rul                  AS Tc_rul,
    t.p_key                AS Tc_p_key,
    t.partition_col        AS Tc_partition_col,
    t.target               AS Tc_target,
    t.SectionBreakStartKM + 0.02 AS km_range_end,
    year(to_date(t.r_date, 'yyyy-MM-dd')) AS year_partition,
    month(to_date(t.r_date, 'yyyy-MM-dd')) AS month_partition,
    year(add_months(to_date(t.r_date, 'yyyy-MM-dd'), -1))  AS
prev_year_partition,
    month(add_months(to_date(t.r_date, 'yyyy-MM-dd'), -1)) AS
prev_month_partition,
    date_sub(to_date(t.r_date, 'yyyy-MM-dd'), 30) AS Tc_r_date_minus_30days,
    unix_timestamp(to_timestamp(t.r_date, 'dd-MM-yyyy')) AS Tc_r_date_Timestamp
  FROM dev_adlunise.predictive_maintenance_uofa_2025.trainingcontext t
  LEFT JOIN dev_adlunise.predictive_maintenance_uofa_2025.basecodemap m
    ON t.BaseCode = m.BaseCode
  WHERE t.r_date IS NOT NULL
  AND t.SectionBreakStartKM IS NOT NULL
  AND t.r_date >='2018-12-31'
),


w AS (
  SELECT
      w.BaseCode              AS Wagon_BaseCode,
      m.MappedBaseCode        AS Wagon_BaseCode_Mapped,
      w.SectionBreakStartKM   AS Wagon_SectionBreakStartKM,
      w.SectionBreakFinishKM  AS Wagon_SectionBreakFinishKM,
      avg(w.Twist14m)              AS Wagon_Twist14m,
      avg(w.BounceFrt)             AS Wagon_BounceFrt,
      avg(w.BounceRr)              AS Wagon_BounceRr,
      avg(w.BodyRockFrt)           AS Wagon_BodyRockFrt,
      avg(w.BodyRockRr)            AS Wagon_BodyRockRr,
      avg(w.LP1)                   AS Wagon_LP1,
      avg(w.LP2)                   AS Wagon_LP2,
      avg(w.LP3)                   AS Wagon_LP3,
      avg(w.LP4)                  AS Wagon_LP4,
      avg(w.Speed)                 AS Wagon_Speed,
      avg(w.BrakeCylinder)         AS Wagon_BrakeCylinder,
      avg(w.IntrainForce)          AS Wagon_IntrainForce,
      avg(w.Acc1)                  AS Wagon_Acc1,
      avg(w.Acc2)                  AS Wagon_Acc2,
      avg(w.Acc3)                  AS Wagon_Acc3,
      avg(w.Acc4)                  AS Wagon_Acc4,
      avg(w.Twist2m)               AS Wagon_Twist2m,
      avg(w.Acc1_RMS)              AS Wagon_Acc1_RMS,
      avg(w.Acc2_RMS)              AS Wagon_Acc2_RMS,
      avg(w.Acc3_RMS)              AS Wagon_Acc3_RMS,
      avg(w.Acc4_RMS)              AS Wagon_Acc4_RMS,
      avg(w.Rail_Pro_L)            AS Wagon_Rail_Pro_L,
      avg(w.Rail_Pro_R)            AS Wagon_Rail_Pro_R,
      avg(w.SND)                   AS Wagon_SND,
      avg(w.VACC)                  AS Wagon_VACC,
      avg(w.VACC_L)                AS Wagon_VACC_L,
      avg(w.VACC_R)                AS Wagon_VACC_R,
      avg(w.Curvature)             AS Wagon_Curvature,
      avg(w.Track_Offset)          AS Wagon_Track_Offset,
      avg(w.ICWVehicle)            AS Wagon_ICWVehicle,
      to_date(w.RecordingDate, 'yyyy-MM-dd') AS Wagon_RecordingDate_parsed,
      year(to_date(w.RecordingDate, 'yyyy-MM-dd')) AS year_partition,
      month(to_date(w.RecordingDate, 'yyyy-MM-dd')) AS month_partition,
      w.RecordingDate         AS Wagon_RecordingDate,
      avg(w.SND_L)                 AS Wagon_SND_L,
      avg(w.SND_R)                AS Wagon_SND_R
  FROM dev_adlunise.predictive_maintenance_uofa_2025.wagondata w
  LEFT JOIN dev_adlunise.predictive_maintenance_uofa_2025.basecodemap m
    ON w.BaseCode = m.BaseCode
  WHERE w.RecordingDate IS NOT NULL
    AND w.SectionBreakStartKM IS NOT NULL
  GROUP BY
    Wagon_RecordingDate_parsed,
    Wagon_BaseCode,
    Wagon_BaseCode_Mapped,
    Wagon_SectionBreakStartKM,
    Wagon_SectionBreakFinishKM,
    Wagon_RecordingDate,
    year_partition,
    month_partition
),

tng AS (
  SELECT
      BaseCode              AS Tng_BaseCode,
      SectionBreakStartKM   AS Tng_SectionBreakStartKM,
      SectionBreakFinishKM  AS Tng_SectionBreakFinishKM,
      unix_timestamp(to_timestamp(FromDate, 'dd/MM/yyyy'))  AS
Tng_FromDate_Timestamp,
      unix_timestamp(to_timestamp(ToDate,   'dd/MM/yyyy'))  AS
Tng_ToDate_Timestamp,
      FromDate              AS Tng_FromDate,
      ToDate                AS Tng_ToDate,
      Tonnage               AS Tng_Tonnage,
      load_date_utc         AS Tng_load_date_utc
  FROM
`09ad024f-822f-48e4-9d9e-b5e03c1839a2`.predictive_maintenance_uofa_2025.tonnaged
ata
  WHERE FromDate IS NOT NULL
    AND ToDate IS NOT NULL
    AND SectionBreakStartKM IS NOT NULL
),

joined AS (
SELECT
   tc.*,
   w.*,
   tng.Tng_Tonnage as Tng_Tonnage,
   tng.Tng_FromDate as Tng_FromDate,
   tng.Tng_ToDate as Tng_ToDate
 FROM tc
 INNER JOIN w
   ON COALESCE(tc.Tc_BaseCode_Mapped, tc.Tc_BaseCode) =
COALESCE(w.Wagon_BaseCode_Mapped, w.Wagon_BaseCode)
  AND w.Wagon_RecordingDate >=  tc.Tc_r_date_minus_30days
  AND w.Wagon_RecordingDate <= tc.Tc_r_date
  AND w.Wagon_SectionBreakStartKM BETWEEN tc.Tc_SectionBreakStartKM AND
tc.km_range_end
  AND w.year_partition  = tc.year_partition
  AND w.month_partition = tc.month_partition
LEFT JOIN tng
  ON tc.Tc_BaseCode = tng.Tng_BaseCode
  AND w.Wagon_SectionBreakStartKM = tng.Tng_SectionBreakStartKM
  AND w.Wagon_SectionBreakFinishKM = tng.Tng_SectionBreakFinishKM
  AND tc.Tc_r_date_Timestamp BETWEEN tng.Tng_FromDate_Timestamp AND
tng.Tng_ToDate_Timestamp

UNION ALL

SELECT
   tc.*,
   w.*,
   tng.Tng_Tonnage as Tng_Tonnage,
   tng.Tng_FromDate as Tng_FromDate,
    tng.Tng_ToDate as Tng_ToDate
  FROM tc
  INNER JOIN w
    ON COALESCE(tc.Tc_BaseCode_Mapped, tc.Tc_BaseCode) =
COALESCE(w.Wagon_BaseCode_Mapped, w.Wagon_BaseCode)
  AND w.Wagon_RecordingDate >=  tc.Tc_r_date_minus_30days
  AND w.Wagon_RecordingDate <= tc.Tc_r_date
  AND w.Wagon_SectionBreakStartKM BETWEEN tc.Tc_SectionBreakStartKM AND
tc.km_range_end
  AND w.year_partition  = tc.prev_year_partition
  AND w.month_partition = tc.prev_month_partition
LEFT JOIN tng
  ON tc.Tc_BaseCode = tng.Tng_BaseCode
  AND w.Wagon_SectionBreakStartKM = tng.Tng_SectionBreakStartKM
  AND w.Wagon_SectionBreakFinishKM = tng.Tng_SectionBreakFinishKM
  AND tc.Tc_r_date_Timestamp BETWEEN tng.Tng_FromDate_Timestamp AND
tng.Tng_ToDate_Timestamp
)

SELECT
    joined.Tc_BaseCode             AS Tc_BaseCode,
    joined.Tc_BaseCode_Mapped       AS Tc_BaseCode_Mapped,
    joined.Tc_SectionBreakStartKM  AS Tc_SectionBreakStartKM,
    joined.Tc_break_date           AS Tc_break_date,
    joined.Tc_last_fail_if_available_otherwise_null  AS
Tc_last_fail_if_available_otherwise_null,
    joined.Tc_r_date               AS Tc_r_date,
    joined.Tc_rul                  AS Tc_rul,
    joined.Tc_target               AS Tc_target,
    joined.Wagon_RecordingDate AS Wagon_RecordingDate,
    AVG(joined.Wagon_Twist14m) AS Wagon_Twist14m,
    AVG(joined.Wagon_BounceFrt) AS Wagon_BounceFrt,
    AVG(joined.Wagon_BounceRr) AS Wagon_BounceRr,
    AVG(joined.Wagon_BodyRockFrt) AS Wagon_BodyRockFrt,
    AVG(joined.Wagon_BodyRockRr) AS Wagon_BodyRockRr,
    AVG(joined.Wagon_LP1) AS Wagon_LP1,
    AVG(joined.Wagon_LP2) AS Wagon_LP2,
    AVG(joined.Wagon_LP3) AS Wagon_LP3,
    AVG(joined.Wagon_LP4) AS Wagon_LP4,
    AVG(joined.Wagon_Speed) AS Wagon_Speed,
    AVG(joined.Wagon_BrakeCylinder) AS Wagon_BrakeCylinder,
    AVG(joined.Wagon_IntrainForce) AS Wagon_IntrainForce,
    AVG(joined.Wagon_Acc1) AS Wagon_Acc1,
    AVG(joined.Wagon_Acc2) AS Wagon_Acc2,
    AVG(joined.Wagon_Acc3) AS Wagon_Acc3,
    AVG(joined.Wagon_Acc4) AS Wagon_Acc4,
    AVG(joined.Wagon_Twist2m) AS Wagon_Twist2m,
    AVG(joined.Wagon_Acc1_RMS) AS Wagon_Acc1_RMS,
    AVG(joined.Wagon_Acc2_RMS) AS Wagon_Acc2_RMS,
    AVG(joined.Wagon_Acc3_RMS) AS Wagon_Acc3_RMS,
    AVG(joined.Wagon_Acc4_RMS) AS Wagon_Acc4_RMS,
    AVG(joined.Wagon_Rail_Pro_L) AS Wagon_Rail_Pro_L,
    AVG(joined.Wagon_Rail_Pro_R) AS Wagon_Rail_Pro_R,
    AVG(joined.Wagon_SND) AS Wagon_SND,
    AVG(joined.Wagon_VACC) AS Wagon_VACC,
    AVG(joined.Wagon_VACC_L) AS Wagon_VACC_L,
    AVG(joined.Wagon_VACC_R) AS Wagon_VACC_R,
    AVG(joined.Wagon_Curvature) AS Wagon_Curvature,
    AVG(joined.Wagon_Track_Offset) AS Wagon_Track_Offset,
    AVG(joined.Wagon_ICWVehicle) AS Wagon_ICWVehicle,
    AVG(joined.Wagon_SND_L)   AS Wagon_SND_L,
    AVG(joined.Wagon_SND_R)   AS Wagon_SND_R,
    COUNT(*)                  AS w_row_count,
    AVG(joined.Tng_Tonnage)   AS Tng_Tonnage
FROM joined
GROUP BY
  Tc_BaseCode,
  Tc_BaseCode_Mapped,
  Tc_SectionBreakStartKM,
  Tc_break_date,
  Tc_last_fail_if_available_otherwise_null,
  Tc_r_date,
  Tc_rul,
  Tc_target,
  Wagon_RecordingDate
""")
```

The integration is mainly based on trainingcontext as the primary table, with left joins to wagondata and tonnagedata.

- Design of trainingcontext join with wagondata

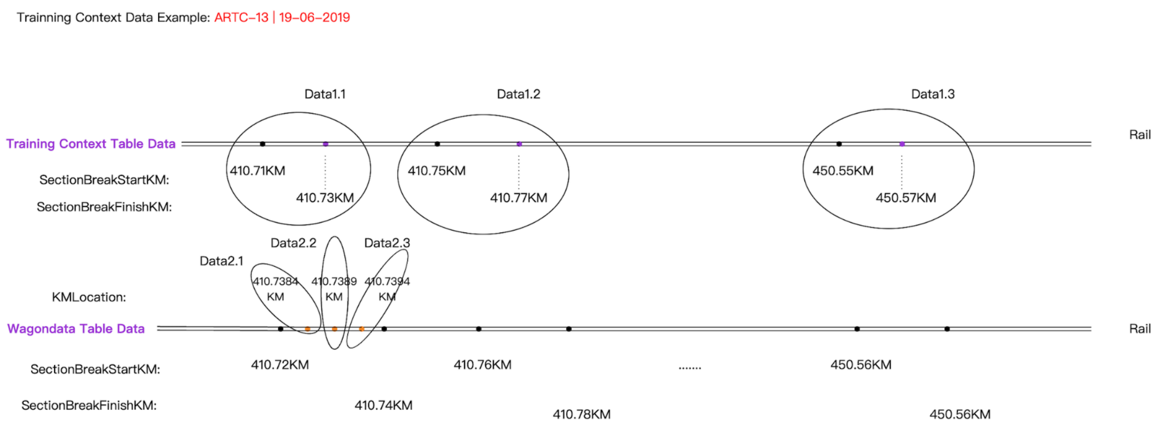
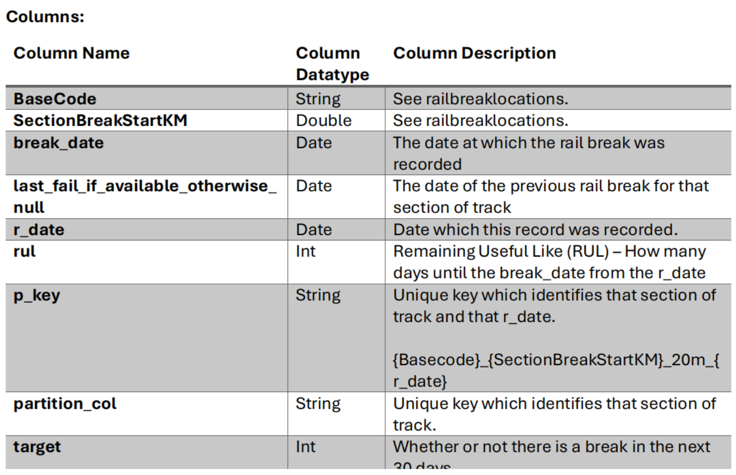


Figure1. Data Examples in Training Context Table and Wagondata Table

Taking ARTC-13 data on 19-06-2019 as an example:

- Training Context Data

This is the training data, where SectionBreakStartKM and SectionBreakFinishKM are used to identify whether a 20m segment of ARTC-13 is broken. Examples include Data1.1, Data1.2, and Data1.3 in Figure 1.



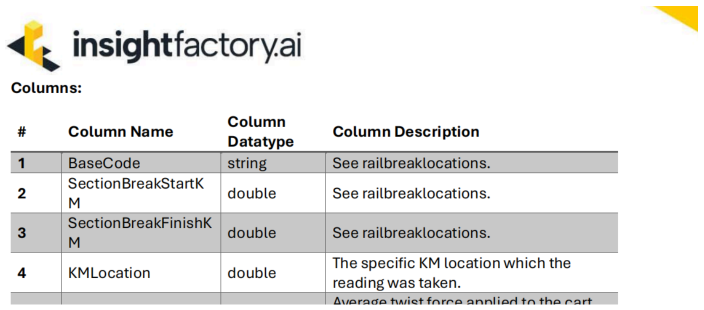
Columns:

Column Name	Column Datatype	Column Description
BaseCode	String	See railbreaklocations.
SectionBreakStartKM	Double	See railbreaklocations.
break_date	Date	The date at which the rail break was recorded
last_fail_if_available_otherwise_null	Date	The date of the previous rail break for that section of track
r_date	Date	Date which this record was recorded.
rul	Int	Remaining Useful Like (RUL) – How many days until the break_date from the r_date
p_key	String	Unique key which identifies that section of track and that r_date. {Basecode}_{SectionBreakStartKM}_20m_{r_date}
partition_col	String	Unique key which identifies that section of track.
target	Int	Whether or not there is a break in the next 30 days

Figure 2. Training Context field description screenshot

- Wagondata Data

This represents sensor data, with each sensor identified by KMLocation (e.g., Data2.1, Data2.2, Data2.3 in Figure 1). The data for Data2.1 contains not only KMLocation, but also the fields SectionBreakStartKM and SectionBreakFinishKM, which together identify the specific 20m track segment where the sensor is located.



insightfactory.ai

Columns:

#	Column Name	Column Datatype	Column Description
1	BaseCode	string	See railbreaklocations.
2	SectionBreakStartKM	double	See railbreaklocations.
3	SectionBreakFinishKM	double	See railbreaklocations.
4	KMLocation	double	The specific KM location which the reading was taken.
			Average twist force applied to the cart

Figure 3. Wagondata field description screenshot

When joining Training Context with Wagondata:

1. **Original approach:**

Training Context.SectionBreakStartKM = Wagondata.SectionBreakStartKM
However, it was found that many values did not match. As a result, after the join, many Training Context records had no corresponding Wagondata entries.

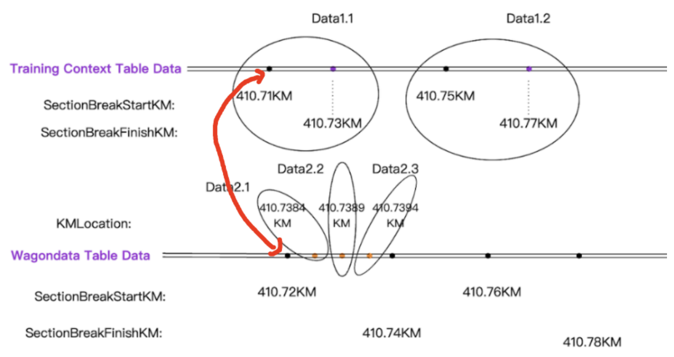


Figure 4. Misalignment between Wagondata and Training Data

2.Adjusted approach:

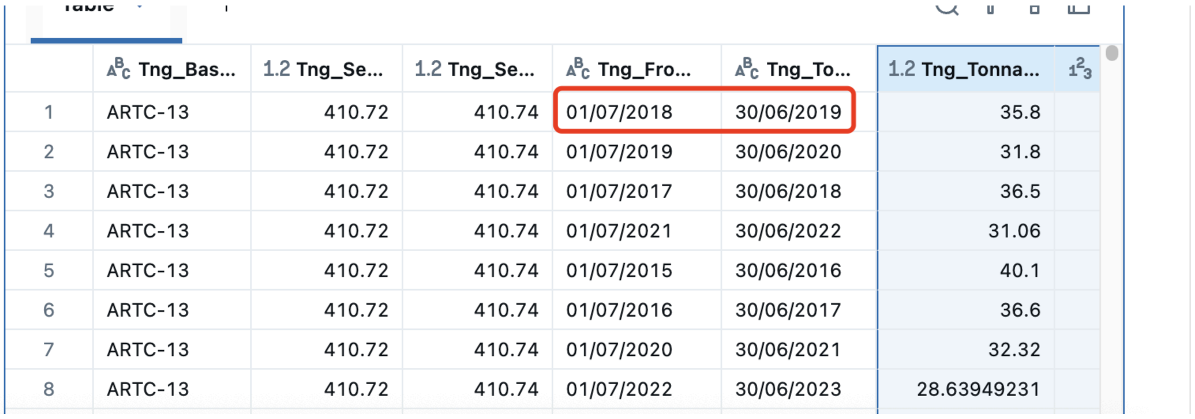
Join Sql:

Wagondata.SectionBreakStartKM
BETWEEN Training Context.SectionBreakStartKM
AND Training Context.SectionBreakStartKM + 0.02

This way, a 20m track segment in the Training Context (e.g., Data1) can be joined with the nearest Wagon sensor data, even if slightly misaligned, to form the training dataset. For example, in Figure 4, Data1.1 joins with Data2.1, Data2.2, Data2.3

- Design of Wagondata join with Tonnagedata

Since Tonnagedata only provides the cumulative freight tonnage for a 20m track segment on a daily basis, it records the cumulative load from a start date to an end date (spanning up to one year).



	Tng_Bas...	Tng_Se...	Tng_Se...	Tng_Fro...	Tng_To...	Tng_Tonna...	
1	ARTC-13	410.72	410.74	01/07/2018	30/06/2019	35.8	
2	ARTC-13	410.72	410.74	01/07/2019	30/06/2020	31.8	
3	ARTC-13	410.72	410.74	01/07/2017	30/06/2018	36.5	
4	ARTC-13	410.72	410.74	01/07/2021	30/06/2022	31.06	
5	ARTC-13	410.72	410.74	01/07/2015	30/06/2016	40.1	
6	ARTC-13	410.72	410.74	01/07/2016	30/06/2017	36.6	
7	ARTC-13	410.72	410.74	01/07/2020	30/06/2021	32.32	
8	ARTC-13	410.72	410.74	01/07/2022	30/06/2023	28.63949231	

Figure 5. Sample of Tonnagedata

The designed join method is as follows:

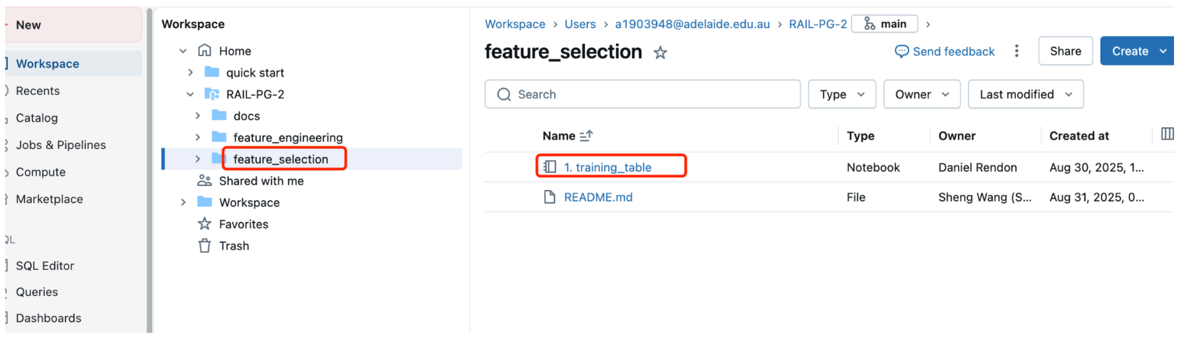
For the same 20m track segment in Wagon and Tonnagedata, when the Wagon.RecordingDate falls within a one-year record range in Tonnagedata, the value is joined into the table. This reflects whether the total cumulative tonnage carried on that 20m track within one year is strongly correlated with the rail break prediction outcome.

Join Sql:
ON tc.Tc_BaseCode = tng.Tng_BaseCode
AND w.Wagon_SectionBreakStartKM = tng.Tng_SectionBreakStartKM
AND w.Wagon_SectionBreakFinishKM = tng.Tng_SectionBreakFinishKM

AND w.Wagon_RecordingDate_Timestamp BETWEEN tng.Tng_FromDate_Timestamp AND tng.Tng_ToDate_Timestamp

2.1.2 Code Repository and Notebook Access

The code logic has been uploaded to Git. It can be viewed in the ***training_table*** notebook under the Databricks Home directory (first make sure that in your Databricks account , our team Git repo have been authorized).



Name	Type	Owner	Created at
1. training_table	Notebook	Daniel Rendon	Aug 30, 2025, 1...
README.md	File	Sheng Wang (S...	Aug 31, 2025, 0...