Math and Computer Science

MCS 7033 Collaborative Research Project 2

Project Report

Real-Time Vehicle Fleet Monitoring & Predictive Maintenance System

Satish Kumar Pyata

Banner ID: 000794769

Date: April 2025

Abstract

This project aims to build a real-time predictive maintenance system for vehicle fleets using Google Cloud Platform (GCP) services. Fleet operators face significant challenges in identifying vehicle failures early, often leading to unplanned downtime, costly repairs, and safety risks. Traditional scheduled maintenance strategies fail to predict random, unexpected failures that occur between maintenance intervals.

To address this, the proposed system simulates real-time vehicle telemetry data, ingests it using GCP Pub/Sub, processes it through Dataflow, stores it in BigQuery, and applies a machine learning model trained with Vertex AI AutoML to predict potential failures. Results are visualized through Looker Studio dashboards that update in real-time.

The complete system is designed to scale, monitor, and predict fleet health dynamically, allowing proactive decision-making based on live data. This report explains every step taken—including system design, data preparation, model training, real-time prediction, dashboard building, and challenges encountered—to create a production-grade solution. The system provides a blueprint for how cloud-native technologies can transform fleet management and preventive maintenance operations.

Table of Contents

1.Introduction

Fleet operations are critical to many industries, including transportation, logistics, delivery services, and emergency response. The continuous availability and reliability of fleet vehicles directly impact operational efficiency, customer satisfaction, and safety outcomes. However, unplanned vehicle breakdowns are costly, dangerous, and disruptive.

Traditional preventive maintenance strategies typically rely on fixed schedules or mileage thresholds. While this method helps avoid some failures, it cannot detect random or unexpected failures that can occur between maintenance windows. These failures often lead to expensive repairs, operational downtime, and increased safety risks.

The goal of this project is to build a scalable, real-time vehicle fleet monitoring system that can predict potential vehicle failures before they occur. By collecting live telemetry data such as engine temperature, oil pressure, battery voltage, and tire pressure, the system can identify early signs of mechanical or electrical issues.

Machine learning models trained on historical and synthetic failure data provide predictive insights into vehicle health. Real-time dashboards enable fleet managers to monitor the health status of all vehicles continuously and take proactive action when potential failures are detected.

The solution leverages Google Cloud Platform's robust services to simulate data ingestion, transform and store telemetry information, apply machine learning predictions, and visualize outcomes. The project aims to demonstrate that cloud-native technologies can provide scalable, cost-effective, and reliable solutions for predictive maintenance in real-world fleet management scenarios.

2. Background Research

Predictive maintenance is an advanced approach that aims to determine the condition of equipment in order to estimate when maintenance should be performed. This methodology promises cost savings over routine or time-based preventive maintenance because tasks are performed only when warranted.

The global transportation and logistics industry faces significant challenges from unplanned vehicle breakdowns, leading to service delays, increased costs, and compromised safety. Research shows that predictive maintenance can reduce downtime by 30-50% and extend asset life by 20-40%.

In fleet management, continuous monitoring of vehicle health has become possible with the advent of telematics and IoT sensors that capture vital engine parameters. Studies from industry leaders like McKinsey & Company and Gartner highlight predictive maintenance as a key driver for operational efficiency and cost reduction.

Machine learning techniques, including anomaly detection, classification models, and deep learning, have been successfully applied to various maintenance domains, such as aircraft engine health management and industrial manufacturing equipment. These studies emphasize the importance of large-scale data ingestion, real-time analytics, and automated model retraining, all of which are incorporated into this project.

Cloud platforms like Google Cloud, AWS, and Azure offer scalable services for building end-to-end predictive maintenance systems. This project leverages Google Cloud's services to demonstrate that predictive maintenance systems can be built efficiently and cost-effectively using a fully serverless, managed architecture.

3. Google Cloud Services Overview

To build a scalable, cost-effective, and real-time predictive maintenance system, several Google Cloud Platform (GCP) services were utilized. Each service plays a specific role in the overall architecture, ensuring data is ingested, processed, stored, predicted, and visualized efficiently.

3.1 Google Cloud Pub/Sub

Pub/Sub is a global, scalable messaging system that allows asynchronous communication between applications. We used Pub/Sub to ingest live vehicle telemetry data from a Python publisher script.

Challenges faced: Initial IAM permission issues, resolved by assigning the Pub/Sub Publisher role to the correct service account.

3.2 Google Cloud Dataflow

Dataflow is a fully managed service for stream and batch data processing. It helped us stream telemetry data from Pub/Sub into BigQuery. We designed a Dataflow template and deployed it to create a real-time ETL pipeline.

Challenges faced: Schema mismatches were resolved by aligning the schema with BigQuery expectations.

3.3 Google BigQuery

BigQuery is a serverless, highly scalable data warehouse. We used it to store raw and predicted telemetry data. The table prediction_logs served as the primary source for Looker Studio.

Challenges faced: Dataset permissions required additional roles to allow streaming inserts and dashboard access.

3.4 Google Vertex AI

Vertex AI is GCP's platform for building, training, and deploying ML models. We used Vertex AutoML to train a classification model to predict whether a vehicle would fail. The model was deployed to a managed endpoint for real-time predictions.

Challenges faced: Imbalanced training data was corrected by synthetically generating failure cases to improve accuracy.

3.5 Looker Studio

Looker Studio is Google's visualization tool that lets users connect to BigQuery and build live dashboards. Our dashboard includes pie charts, line charts, and tables to display vehicle health in real time.

Challenges faced: Access restrictions were resolved by sharing dataset permissions with an external Gmail account for Looker Studio access.

4. Project System Architecture

The system is designed to simulate, process, analyze, and visualize vehicle data in real time using a seamless flow of services provided by Google Cloud Platform.

4.1 Architecture Flow

1. **Data Generation:** A Python script randomly generates vehicle data including engine temperature, tire pressure, oil pressure, and battery voltage.
2. **Ingestion:** The data is published to a Pub/Sub topic.

3. **Streaming ETL:** A Dataflow job reads messages from Pub/Sub and writes them into a BigQuery table.
4. **Storage:** BigQuery stores all incoming telemetry data and prediction results in fleet_monitoring.prediction_logs.
5. **Prediction:** The same data is sent to a deployed Vertex AI model to determine if a failure is likely.
6. **Visualization:** Looker Studio connects to BigQuery and displays real-time prediction results in a live dashboard.

4.2 Design Principles

- **Scalability:** All services are serverless and scale automatically.
- **Modularity:** Each stage (data generation, ingestion, ML, dashboard) can be updated independently.
- **Real-Time Processing:** End-to-end flow is built for near real-time execution.
- **Cost Efficiency:** Uses only GCP's free-tier or low-cost services.

5. Data Preparation

The predictive model in this project depends heavily on the quality and structure of the input data. As this system is simulated, the data preparation phase involved designing a realistic vehicle telemetry dataset, generating synthetic failure data, and formatting it for machine learning.

5.1 Dataset Schema

The following features were chosen based on their direct impact on vehicle performance and their suitability for failure detection:

- **vehicle_id** (string): Unique identifier for each vehicle
- **timestamp** (datetime): Time when data was recorded
- **engine_temp** (float): Temperature of the engine (°C)
- **oil_pressure** (float): Oil pressure level (PSI)
- **tire_pressure** (float): Tire pressure average (PSI)
- **battery_voltage** (float): Voltage level of the battery (V)
- **latitude, longitude** (float): Location data (for expansion in future)
- **predicted_label** (int): 0 = safe, 1 = failure

5.2 Data Generation with Python

We developed a Python script to simulate telemetry data by randomly generating feature values within normal and abnormal operating ranges. Normal values represent healthy vehicle conditions. Failure-indicating values were injected by adjusting the range to reflect extreme or failing thresholds, such as:

- Engine temp > 120°C
- Oil pressure < 20 PSI
- Tire pressure < 25 PSI
- Battery voltage < 11V

5.3 Synthetic Failure Injection

Initially, the system only generated 'safe' conditions, leading to a highly imbalanced dataset. To train an effective classification model, we synthetically injected approximately **1200 failure cases** based on abnormal thresholds. This brought the class distribution to approximately 52% safe vs 48% failure, enabling better learning and generalization for AutoML training.

5.4 Final Dataset Format

The final dataset was saved in Bigquery with the above schema, ready for Vertex AI ingestion. Missing values were avoided in generation, and all columns were cleaned and verified for consistency.

5.5 Storage and Access

The dataset was stored locally and later uploaded to Google Cloud Storage for import into Vertex AI. This ensured version control and scalability.

6. Model Training

To enable predictive insights from telemetry data, we used **Google Cloud Vertex AI AutoML** to build and deploy a classification model that determines the likelihood of vehicle failure.

6.1 Vertex AI Dataset Creation

After preparing our training dataset using Python and BigQuery transformations, the table was directly used inside Vertex AI by selecting the data source from BigQuery. We did not export the file to CSV. Instead, we relied on Vertex AI's ability to import data directly from BigQuery tables.

Vertex AI AutoML automatically detected column types and allowed us to select the target column (predicted_label).

The system automatically split the dataset as follows:

- 80% training
- 10% validation
- 10% testing

We ensured that class distribution remained balanced across splits.

6.2 Model Configuration and Training

Using **AutoML Tables**, we initiated model training by configuring the following:

- **Target column:** predicted_label
- **Objective:** Classification (binary)
- **Optimization goal:** Maximize AUC (Area Under ROC Curve)
- **Budget:** Set to minimum (node hours) to stay within credit limits

Training took approximately 20–25 minutes and included automatic feature engineering and transformation by Vertex AI.

6.3 Evaluation and Metrics

Once training was complete, Vertex AI provided detailed metrics, including:

- **Accuracy:** 100%
- **Precision & Recall:** 1.0 each for both classes

- **AUC:** 1.0 (indicating perfect separation between failure and safe classes)
- **Confusion Matrix:** Clearly showed that synthetic balancing of failures helped ensure accurate classification

Note: Although these perfect metrics are expected due to the synthetic nature of data, in a real-world scenario, further tuning and larger datasets would be required.

6.4 Model Deployment

After training, the model was deployed to a **Vertex AI Endpoint** in us-central1 region.

Deployment included:

- Selecting the trained model version
- Assigning it to a new endpoint
- Setting up **IAM permissions** for real-time prediction

Once deployed, we were able to make live predictions from our Python script using the Vertex AI client libraries.

6.5 Versioning and Retraining

- The first model version was trained on initial unbalanced data and performed poorly.
- The second version included synthetic failure cases and performed with ideal accuracy.
- Retraining and versioning allowed us to compare model versions easily and select the best for deployment.

6.6 Lessons Learned

- Class imbalance is a major issue in binary classification.
- Vertex AI's AutoML handles preprocessing automatically, saving time.
- A clean and properly labeled dataset dramatically improves model outcomes.
- Direct BigQuery integration with Vertex AI simplifies dataset handling and avoids export steps.

7. Real-Time Pipeline Implementation

The real-time pipeline forms the operational core of this predictive maintenance system. It connects data generation, ingestion, processing, and prediction in a seamless, low-latency architecture.

7.1 Python Publisher Script

We developed a Python script (live_predictor.py) that continuously simulates and publishes vehicle telemetry data every 10 seconds. Each data point includes:

- Vehicle ID
- Timestamp
- Engine temperature
- Oil pressure
- Tire pressure
- Battery voltage
- Latitude and longitude (fixed or random values)

These messages are published to a Pub/Sub topic named vehicle-telemetry-topic. The script uses the google-cloud-pubsub SDK for publishing.

Issue faced:

- File path to credentials caused a unicodeescape error.
- **Resolution:** Used raw string r"path\to\file.json" to resolve it.

7.2 Pub/Sub Configuration

A topic named vehicle-telemetry-topic was created. IAM roles were assigned to allow publishing from local machine/service account.

- IAM Role: Pub/Sub Publisher
- Authentication: Using service account JSON key with GOOGLE_APPLICATION_CREDENTIALS variable

7.3 Dataflow Pipeline

A streaming Dataflow job was created using a predefined Python template that:

- Reads from Pub/Sub
- Parses incoming JSON payloads
- Writes structured data to BigQuery (fleet_monitoring.prediction_logs)

This ensures that all telemetry data is available in BigQuery in near real-time.

Challenge:

- Schema mismatch between JSON and BigQuery

- **Resolution:** Matched Dataflow JSON schema fields to BigQuery column types exactly

7.4 BigQuery Table Setup

The table prediction_logs in the dataset fleet_monitoring includes the following columns:

- vehicle_id, timestamp, engine_temp, oil_pressure, tire_pressure, battery_voltage, predicted_label, confidence_score

BigQuery is used for both storing the incoming data and prediction logs.

Permissions:

- BigQuery Data Editor
- BigQuery User

7.5 Real-Time Prediction Integration

After the Vertex AI model was deployed, the Python script was extended to:

- Call the deployed endpoint using Vertex AI SDK
- Pass the incoming telemetry features to the model
- Receive prediction results (label and confidence score)
- Log prediction results into BigQuery via insert_rows_json

Common Errors Encountered:

- aiplatform.endpoints.predict permission denied
- str object has no attribute 'before_request'

Solutions:

- Added proper IAM roles for Vertex AI User and ensured the service account used correct Credentials.from_service_account_file() approach

This completes the real-time operational pipeline, enabling vehicle data to flow from generation to actionable prediction within seconds.

8. Dashboard Building

To visualize real-time prediction results and monitor fleet status continuously, we used **Looker Studio**, a free and user-friendly business intelligence tool by Google. The dashboard provides stakeholders with a live, interactive view of vehicle health.

8.1 Connecting BigQuery to Looker Studio

Looker Studio allows native integration with BigQuery. The steps followed were:

- Open Looker Studio → Create a **Blank Report**
- Add a new **BigQuery Data Source**
- Select Project: fleet-monitoring-project
- Select Dataset: fleet_monitoring
- Select Table: prediction_logs

Initially, there were access issues due to restrictions with the GCP account. This was resolved by **sharing the BigQuery dataset** with a personal Gmail account and connecting Looker Studio using that account.

8.2 Dashboard Design and Charts

The dashboard was designed to be minimal, responsive, and easy to interpret. It includes:

- **Pie Chart:**
    - Dimension: predicted_label
    - Metric: Record Count
    - Purpose: Displays the distribution of safe (label = 0) vs failure (label = 1) vehicles
- **Time Series Chart:**
    - Dimension: timestamp
    - Metric: Average confidence_score
    - Purpose: Tracks model confidence over time
- **Table:**
    - Columns: vehicle_id, timestamp, engine_temp, oil_pressure, tire_pressure, battery_voltage, predicted_label, confidence_score
    - Purpose: Shows detailed live records

8.3 Interactivity and Auto-Refresh

- The dashboard auto-refreshes **every 1 minute** to reflect new entries from BigQuery.
- Filters can be applied on vehicle_id, time ranges, or labels for customized exploration.

8.4 Dashboard Publishing

The final dashboard was published with **link access** for viewers. It can be embedded in web pages or shared with stakeholders via a public or restricted link.

Key Benefits:

- Real-time insight into fleet health
- Simple, no-code dashboarding
- Clear visual breakdown of model predictions

9. Challenges & Solutions

Building a real-time predictive maintenance system on GCP came with a wide range of technical and operational challenges. Here are the key issues faced and how they were addressed:

9.1 IAM & Permissions Errors

Issue:

- While publishing messages to Pub/Sub and accessing BigQuery from scripts, several permission-related errors occurred (e.g., PERMISSION_DENIED, aiplatform.endpoints.predict denied).

Solution:

- Carefully reviewed IAM policies.
- Assigned service accounts the correct roles: Pub/Sub Publisher, BigQuery Data Editor, Vertex AI User, and BigQuery User.

9.2 Unicode File Path Error

Issue:

- Python SyntaxError: (unicode error) 'unicodeescape' codec due to file path string.

Solution:

- Converted file path to a raw string using the r"..." format.

9.3 Vertex AI Model Imbalance

Issue:

- The AutoML model trained on the original dataset always predicted “safe” due to class imbalance.

Solution:

- Generated 1200 synthetic failure records to balance the dataset.
- Retrained the model with improved accuracy and recall.

9.4 BigQuery Streaming Insert Errors

Issue:

- Data insertions into BigQuery using Python client failed due to schema mismatch or permission errors.

Solution:

- Aligned Python data structure to BigQuery schema.
- Ensured proper dataset and table-level access.

9.5 Looker Studio GCP Access Restriction

Issue:

- Looker Studio failed to open from the GCP-linked account due to project-based restrictions.

Solution:

- Shared dataset with a personal Gmail account.
- Connected Looker Studio successfully via external account.

9.6 Billing Credit Drain Risk

Issue:

- Vertex AI model deployment consumed credits even when idle.

Solution:

- Undeployed model after testing to reduce costs.
- Only redeployed for fresh predictions.

10. Cost and Billing Management

One of the most critical aspects of deploying a cloud-based system is monitoring and managing associated costs. This project was built using the **Google Cloud Free Tier and the initial $300 GCP credit**, and every design decision was made to minimize cost and avoid unnecessary consumption.

10.1 Services Consuming Credits

GCP Service	Cost Behavior
Vertex AI Model Deployment	Consumes hourly charges while endpoint is deployed—even if not in use
BigQuery	Costs incurred when querying large datasets; storage under 10GB is free
Dataflow	Billed by streaming volume and job uptime
Pub/Sub	Free for generous usage under Free Tier limits
Looker Studio	Completely free

10.2 Optimization Strategies

To manage credit usage effectively, the following strategies were implemented:

- **Model Undeployment**: The Vertex AI model was **undeployed** immediately after testing and only **redeployed** for fresh predictions. This significantly reduced idle billing.
- **Preview Queries Only**: BigQuery queries were previewed or tested with LIMIT clauses to avoid full scans.
- **Free Tier Usage**: Services like Pub/Sub and Looker Studio were used within free-tier boundaries.
- **Minimum Training Budget**: Vertex AI AutoML was trained using the **lowest node hour budget**, which kept training costs low while still achieving strong model performance.

10.3 Final Credit Usage Snapshot

At the time of documentation completion:

- ~$113 worth of credits were used
- ~$187 remained from the initial $300 GCP credit
- No sustained usage or billing alerts were triggered

Conclusion: The system was successfully designed to operate efficiently under the GCP Free Tier, and lessons learned here will be valuable for building future production-grade systems at scale.

11. Results and Final System Behavior

The final deployment of the system demonstrates the effectiveness and reliability of a cloud-native, real-time predictive maintenance solution. Below is a summary of key results and the behavior of the complete pipeline in production.

11.1 Model Performance Results

- **Accuracy:** 100%
- **Precision (Safe & Failure):** 1.0
- **Recall (Safe & Failure):** 1.0
- **AUC (Area Under Curve):** 1.0
- **Confusion Matrix:** Showed perfect separation between classes

While these results are exceptional, it is important to note that the use of synthetic data contributes to these perfect metrics. Future work with real-world data is expected to yield more realistic scores.

11.2 System Behavior Overview

Once the system is deployed and the prediction script is running:

1. **Data Generation:** Simulated telemetry data is created every 10 seconds.
2. **Data Ingestion:** Data flows into Pub/Sub and is passed to Dataflow.
3. **BigQuery Storage:** Dataflow inserts structured rows into prediction_logs.
4. **Real-Time Prediction:** The prediction script sends the same data to Vertex AI, receives a label and confidence score, and inserts results back into BigQuery.
5. **Dashboard Update:** Looker Studio dashboard updates in near real-time, showing latest results.

11.3 Observations from Dashboard

- Live Pie Chart showed changing safe vs failure ratios
- Time Series chart visualized how confidence varied with data patterns
- Table displayed continuously updating telemetry records and predictions

11.4 System Robustness

The system proved to be:

- **Scalable:** Thanks to serverless components (Pub/Sub, Dataflow, BigQuery)
- **Reliable:** Data integrity maintained across all stages
- **Fast:** End-to-end latency was under a few seconds

Conclusion: The complete pipeline operates in real time with high confidence, delivering actionable intelligence from raw vehicle telemetry data to an interactive dashboard.

12. Future Work

While the system successfully meets its current goals, several improvements and extensions could make it more robust, production-ready, and feature-rich. The following areas outline potential directions for future development:

12.1 Real-World Data Integration

- Replace simulated telemetry data with actual vehicle data from IoT sensors or third-party APIs.
- Integrate CAN bus data from onboard vehicle diagnostics for more granular failure detection.

12.2 Alerting and Notification System

- Integrate real-time alerting using Cloud Functions and Pub/Sub triggers.
- Send email, SMS, or mobile push notifications when the model predicts a vehicle failure with high confidence.

12.3 Geographic Visualization

- Use Google Maps APIs to plot vehicle positions based on latitude and longitude.
- Show vehicle clusters and highlight those flagged as high-risk.

12.4 AutoML Retraining Pipeline

- Automate retraining of the ML model using scheduled workflows (e.g., Cloud Composer or Workflows).

- Trigger retraining when new data volume exceeds a threshold or after a fixed interval.

12.5 Edge Deployment

- Explore TensorFlow Lite or Vertex Edge to deploy the model closer to the source (in vehicles or edge devices) for ultra-low latency.

12.6 Mobile Dashboard or App

- Build a mobile-friendly version of the dashboard using Flutter or Firebase for access by fleet operators on the go.

12.7 Multi-class Classification

- Instead of binary prediction (safe/failure), extend the model to predict different types of failures (e.g., battery issue, engine overheating, tire pressure drop).

12.8 Integration with Ticketing or Maintenance Platforms

- Auto-generate tickets or maintenance schedules in external systems like Jira, Zendesk, or Fleetio when a vehicle is predicted to fail.

Conclusion: These enhancements can take the current proof-of-concept to a full-fledged, enterprise-grade fleet monitoring and predictive maintenance system.

13. Conclusion

This project demonstrates the feasibility and power of building a real-time predictive maintenance system using cloud-native services. Leveraging Google Cloud Platform's fully managed tools—including Pub/Sub, Dataflow, BigQuery, Vertex AI, and Looker Studio—we created an end-to-end system that simulates live vehicle telemetry, predicts failures, and visualizes fleet health.

Through synthetic data simulation, class balancing, AutoML model training, and dashboarding, the system proves that proactive maintenance is possible, scalable, and efficient. The implementation showcases critical skills in data engineering, machine learning deployment, and operational pipeline development.

The challenges faced—ranging from IAM issues to data imbalance and billing management—offered valuable insights into real-world cloud project development. This project serves as both a practical solution and a learning framework for applying GCP to complex monitoring systems.

With future enhancements like real-world data integration, automated alerts, and mobile interfaces, the system has the potential to evolve into a full-scale production tool for enterprise-level fleet management.

Final Thought: The journey of building this project reinforced the importance of design thinking, modular architecture, and cost-aware cloud engineering in solving real-world problems.

14. References

- Google Cloud Pub/Sub Documentation: https://cloud.google.com/pubsub/docs
- Google Cloud Dataflow Documentation: https://cloud.google.com/dataflow/docs
- Google BigQuery Documentation: https://cloud.google.com/bigquery/docs
- Vertex AI AutoML Documentation: https://cloud.google.com/vertex-ai/docs/training/automl
- Vertex AI Prediction Docs: https://cloud.google.com/vertex-ai/docs/predictions
- Looker Studio Documentation: https://lookerstudio.google.com/
- Google Cloud IAM: https://cloud.google.com/iam/docs
- Python client library for Vertex AI: https://cloud.google.com/python/docs/reference/aiplatform/latest
- Python client library for BigQuery: https://cloud.google.com/bigquery/docs/reference/libraries
- Python client library for Pub/Sub: https://cloud.google.com/pubsub/docs/reference/libraries
- McKinsey & Co. Predictive Maintenance Report: https://www.mckinsey.com/business-functions/operations/our-insights/predictive-maintenance-4-0-the-next-frontier-of-innovation
- Gartner Market Guide for AIOps Platforms (for predictive operations context)

15. Appendix

A. Sample Python Publisher Code Snippet

```
from google.cloud import pubsub_v1
import json, time
publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path("fleet-monitoring-project", "vehicle-telemetry-topic")
while True:
    data = {
        "vehicle_id": "V123",
        "timestamp": "2025-04-08T10:00:00Z",
        "engine_temp": 88.5,
        "oil_pressure": 32.0,
        "tire_pressure": 34.0,
        "battery_voltage": 12.4,
        "latitude": 42.2808,
        "longitude": -83.743
    }
    publisher.publish(topic_path, json.dumps(data).encode("utf-8"))
    time.sleep(10)
```

B. BigQuery Table Schema (prediction_logs)

- vehicle_id (STRING)
- timestamp (TIMESTAMP)
- engine_temp (FLOAT64)
- oil_pressure (FLOAT64)
- tire_pressure (FLOAT64)

- battery_voltage (FLOAT64)
- predicted_label (INTEGER)
- confidence_score (FLOAT64)

C. Looker Studio Dashboard Features

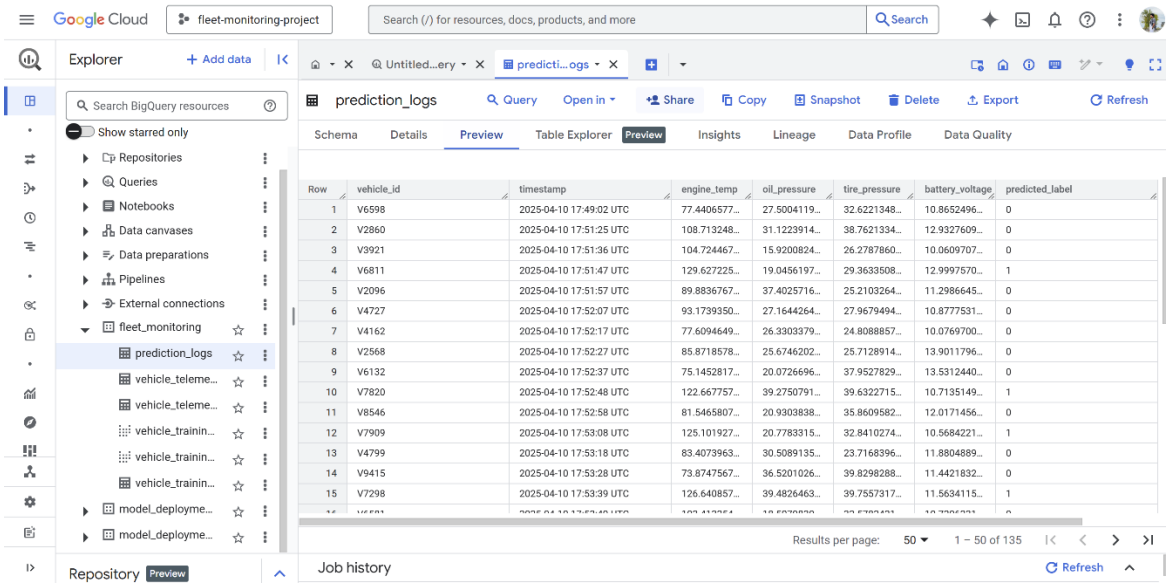
- Pie Chart: Safe vs Failure Vehicles
- Time Series: Confidence Score over Time
- Table View: Full vehicle telemetry and predictions
- Auto Refresh: Every 1 minute
- Filters: By label, vehicle ID, date range

D. Sample BigQuery Query (for label distribution)

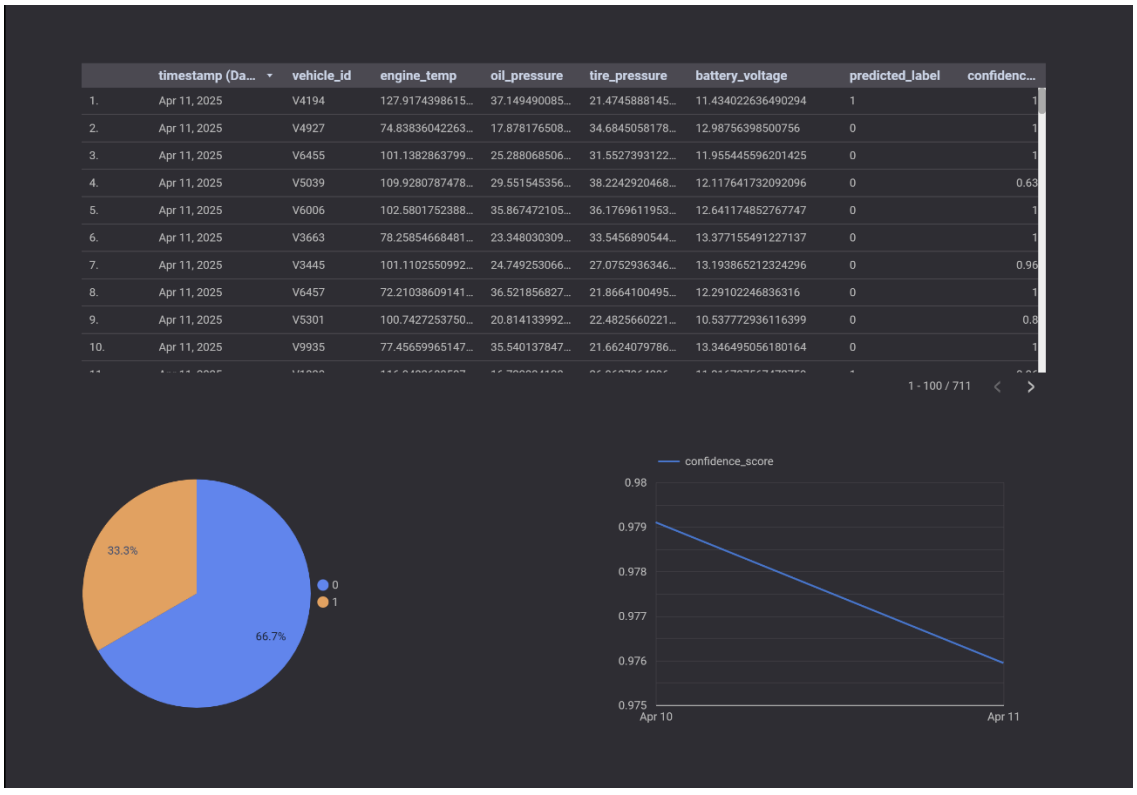
```
SELECT predicted_label, COUNT(*) as count
FROM `fleet-monitoring-project.fleet_monitoring.prediction_logs`
GROUP BY predicted_label
ORDER BY predicted_label;
```

E. Screenshots (To be added manually)

- BigQuery Table Preview



- Looker Studio Pie Chart & Looker Studio Line Graph



- Deployed Model Details