CSC649 · SPECIAL TOPICS IN COMPUTER SCIENCE

An Explainable Prediction Model Using Random Forest for KTMB ETS Ridership Surges

Forecasting hourly passenger surges on Malaysia's ETS network — with predictions transit managers can actually trust and explain.

Muhammad Aiman Bin Jamaluddin 2025168045

Muhammad Aidil Haqkim Bin Ibrahim 2024783429

Group T5CDCS2305B1 | Prepared for Dr. Wan Safra Diyana Binti Wan Abdul Ghani

THE PROBLEM

Passenger surges are hard to see coming—and harder to explain

KTMB ETS trains sell out fast during festive periods like Hari Raya, while off-peak services often run near-empty. Ridership data, weather, and holiday calendars all live in separate systems, so station managers rarely see the full picture in time to act.



Ticket shortages

Sudden surges during festive seasons sell out fast, leaving travelers stranded.



Scattered data

Passenger, weather, and calendar data sit in separate systems with no shared view.



Black-box models

Traditional models predict crowding but can't explain why — managers can't act on a guess.



Costly quiet hours

Running full trains during low-demand hours wastes fuel, staff time, and revenue.

RESEARCH OBJECTIVES

Three objectives guide this project

01



Identify requirements

Identify the requirements of using Random Forest and TreeSHAP to predict KTMB ETS passenger surges.

02



Develop a model

Develop a working passenger-surge prediction model built on the Random Forest algorithm.

03



Evaluate accuracy

Evaluate the accuracy of the Random Forest algorithm in detecting transit passenger surges.

METHODOLOGY

From raw records to a live forecast

Offline training pipeline (left) feeds a frozen model into the live Streamlit engine (right).

OFFLINE TRAINING

ets_2026.parquet

Real per-route KTMB ticketing records

Open-Meteo Archive

Real historical temperature & rainfall

Malaysia holidays

Public-holiday calendar (holidays lib)

preprocess_data.py — merge + aggregate to hourly network totals

LIVE STREAMLIT ENGINE

train_model.py

Random Forest Regressor, chronological 80/20 split → ktmb_surge_model.pkl

Live weather fetch

Current temperature & rainfall (OpenWeatherMap)

Predict + TreeSHAP

rf_model.predict() and shap.TreeExplainer() run locally

Dashboard output

Forecast, feature-contribution chart, route explorer, MAE tracker

ALGORITHM

Random Forest, trained the way real transit data behaves

Why Random Forest

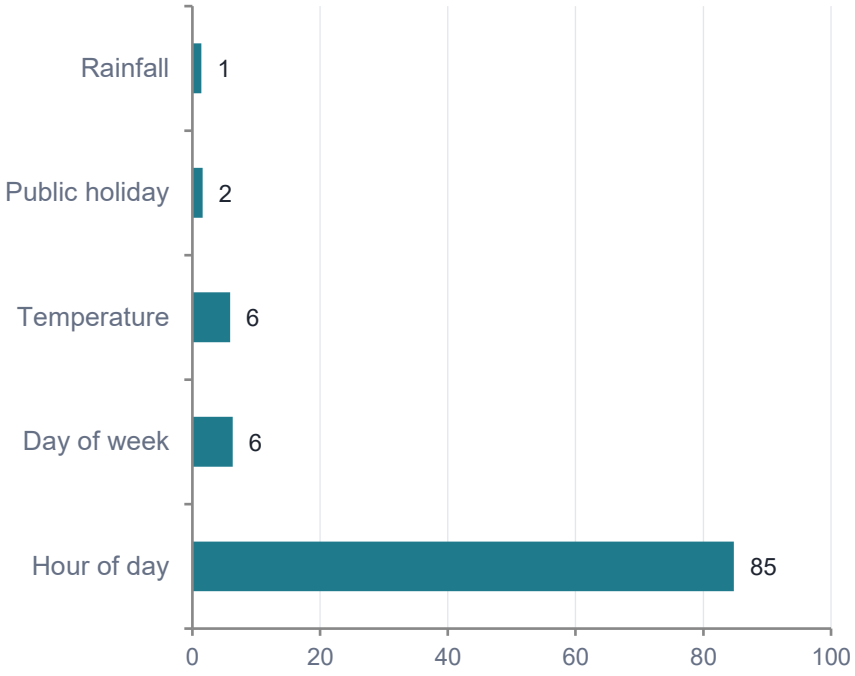
- Ensemble of decision trees averaged together — more trees reduce overfitting and raise accuracy.
- Handles the non-linear interaction of hour, day, weather, and holidays well without heavy tuning.
- Fast to train and pairs naturally with TreeSHAP for exact, model-consistent explanations.
- Trained on 5 real features: hour, day_of_week, is_holiday, temperature, rainfall.

Chronological train/test split

Because this is hourly time-series data, a random 80/20 split would let the model peek at neighbouring hours. Instead, the model trains on the earliest dates and is tested on the most recent slice — a more honest read of real-world accuracy.

Feature importance

From the trained Random Forest (train_model.py output)



EXPLAINABLE AI

TreeSHAP: opening the black box

Random Forest predictions alone don't tell a station manager **why** a surge is forecast. TreeSHAP computes each feature's exact, game-theoretic contribution to a single prediction — turning a black box into an auditable decision.



TreeSHAP — chosen approach

- Exact for tree ensembles — exploits Random Forest's internal structure directly.
- Deterministic: the same input always yields the same explanation.
- Fast enough to run locally, per-prediction, inside the live dashboard.
- Outputs feature contributions that sum exactly to the prediction — fully auditable.



LIME — why it was ruled out

- Model-agnostic: approximates any model with a local surrogate — unnecessary overhead for a tree model.
- Approximate, not exact — explanations can shift between runs on the same input.
- Not designed to exploit tree structure, so it's slower and less precise here than TreeSHAP.

LIVE DEMO



Switching to the live application

streamlit run app.py — forecasting today's KTMB ETS surge in real time.

LIVE DEMO

What we'll walk through on screen

1 

Launch the engine

Run launch_ktmb_app.bat — loads ktmb_surge_model.pkl and opens the dashboard in the browser.

2 

Tab 1: live forecast

App fetches live weather + today's holiday status, predicts the hourly surge, and shows the SHAP feature-contribution chart explaining why.

3 

Log the prediction

One click appends the forecast to prediction_logs.csv — the input the accuracy tracker runs on.

4 

Tab 2: route explorer

Pick an origin → destination pair and pull up real historical hourly ridership for that route.

Safety net: if the live weather call fails, the app shows an error instead of guessing — so a flaky connection won't produce a silently wrong forecast on stage.

RESULTS

Model performance on held-out recent data

177 pax

Test MAE

879.8 pax

Test-set mean ridership

5,004 / 751

Train / test rows (chronological)

3,755

Hourly records after aggregation

On a chronological 80/20 split (train on earlier dates, test on the most recent slice), the model's average forecast error is ≈ 20% of mean hourly ridership — a reasonable baseline for a first model, with clear room to tighten as more data accumulates.

What hour dominates the forecast tells us

hour of day alone drives ~85% of the model's predictions, with day-of-week and temperature contributing modestly. This matches domain intuition (commute peaks dominate) but also means the model currently leans heavily on one signal — a target for the next iteration.

DISCUSSION

Challenges met — and how the model was refined

Sourcing the real KTMB ETS ticketing dataset

Getting access to real KTMB ETS ticketing data was assumed to be straightforward, but proved to be one of the more time-consuming parts of the project — requiring direct outreach rather than a simple public download

Population-density feature dropped

The proposal scoped population data near ETS stations as a feature. No reliable historical source could be matched to it, so it was excluded rather than faked with placeholder values that would only add noise.

Security & data-consistency follow-ups

Two items flagged for hardening before wider use: move the live weather API key out of source code and reconcile the live weather provider with the one used for training so both draw from a consistent source.

CONCLUSION

A transparent, working model — with a clear next chapter

What we achieved

- Random Forest model trained on real, chronologically-split ridership, weather, and holiday data.
- TreeSHAP integrated for exact, per-prediction explanations — no black box.
- Live Streamlit dashboard: forecast, explanation chart, route explorer, and MAE accuracy tracker.

Future work

- Align live and training weather sources to remove train/serve mismatch.
- Re-introduce population/DOSM data once a reliable historical source is found.
- Expand the dataset beyond 3,755 hourly records for a more robust test set.
- Move API credentials to environment secrets ahead of any public deployment.



Thank You

Questions & discussion welcome