ReneWind – Project 4

Introduction to Neural Networks

Date: August 16, 2026

Kimberly Wiethoff, MBA, PMP, PMI-ACP

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Contents / Agenda

- Executive Summary
- Business Problem Overview and Solution Approach
- Evaluation Criteria, Per-Model Performance Commentary
- Test Performance of Best Model
- Questions Answered by the Model, Actionable Insights, & Business Recommendations
- Appendix
  - Data Overview
  - EDA Analysis, Univariate Analysis, Bivariate Analysis
  - Data Preprocessing
  - Model Performance Summary – Training Set
  - Model Performance Summary – Validation Set

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Executive Summary

Predictive Maintenance for Wind Turbine Generators

ReneWind operates a fleet of wind turbines instrumented with environmental and mechanical sensors. This project builds and tunes machine-learning classification models that flag generators likely to fail, so maintenance teams can intervene early with an inexpensive repair or inspection — instead of an unplanned, costly replacement.

40	25,000	3	1
Sensor-derived predictor variables	Total observations (20K train / 5K test)	Failure-related cost types repair · replace · inspect	Recommended model, selected to minimize cost

Expected Impact

Bottom line: because a missed failure (false negative) is far more expensive than an unnecessary inspection (false positive), the recommended model is chosen to maximize recall while keeping overall cost — not just accuracy — as the deciding metric.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Business Problem Overview

Why predictive maintenance matters for wind energy

Wind energy at scale. Wind is one of the most mature renewable technologies, and turbine fleets are fitted with sensors that continuously track environmental conditions (temperature, humidity, wind speed) and component health (gearbox, tower, blades, brake).

The cost of reacting late. An unplanned generator failure necessitates a full replacement. If that same failure is caught early, ReneWind only pays for a repair — and if a healthy unit is flagged, the cost is just a routine inspection.

The target variable. Each observation is labeled 1 = failure or 0 = no failure. The business challenge is to predict that label accurately enough, ahead of time, to shift spend from expensive replacements to cheap repairs and inspections.

Prediction Outcome → Cost

FN — False Negative

Real failure, missed by the model

Replacement cost (highest)

TP — True Positive

Real failure, caught by the model

Repair cost (moderate)

FP — False Positive

Healthy unit flagged by the model

Inspection cost (lowest)

Given: repair cost < replacement cost, and inspection cost < repair cost.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Solution Approach

Build, tune, and select a cost-aware classification model

01

Prepare the data

Load the ciphered sensor data, treat missing values, and split responsibly — train.csv for building and tuning, test.csv held out purely for final evaluation.

02

Establish baselines

Train several classification algorithms (e.g., logistic regression, decision trees, ensemble methods) with default settings to set a performance floor.

03

Tune with cross-validation

Search hyperparameters for each promising model using cross-validation on the training set to reduce overfitting and stabilize performance.

04

Optimize for the real cost

Because a missed failure (FN) costs more than a false alarm (FP), prioritize recall — not raw accuracy — when comparing and selecting models.

05

Validate on unseen data

Evaluate the single best-tuned model on test.csv to confirm it generalizes, then translate the result into expected maintenance cost savings.

Guiding principle: the best model is the one that minimizes total maintenance cost — not simply the one with the highest accuracy.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Model Evaluation Criteria

Primary metric: Recall, with F2-score as tie-breaker — chosen to match ReneWind's cost structure

Why Not Accuracy

~94.5% of observations are "no failure." A model that predicts 0 every time scores ~94.5% accuracy while catching **zero** real failures. Accuracy is dominated by the majority class and tells us almost nothing useful here.

Why Recall Anchors Selection

FN (missed failure) costs a full replacement — the most expensive outcome, far above TP (repair) or FP (inspection).

Recall = TP / (TP + FN) directly measures failures NOT missed. Optimizing it serves the business goal, even if it means tolerating more (cheap) false alarms.

Why Not Recall Alone

Recall is easy to game: flagging every unit as a failure gets perfect recall (1.0) but triggers an inspection on every healthy generator.

Precision and the confusion matrix serve as supporting checks to keep this from happening.

Why F2-score: weights recall 2× as heavily as precision — one tunable number for GridSearchCV/RandomizedSearchCV that keeps recall dominant without ignoring precision. In scikit-learn: `make_scorer(fbeta_score, beta=2)`. **Gold standard:** with the actual cost ordering known (replace > repair > inspect), a custom cost function — weighting FN/TP/FP (e.g., 10/3/1) and minimizing total cost from the confusion matrix — most faithfully captures "minimize maintenance cost." Recall/F2 is the practical proxy; the cost function reports real expected savings.

1. Report accuracy, precision, recall & F1 for every model (transparency)
2. Select & tune using Recall / F2-score as the deciding metric
3. Optionally translate the final confusion matrix into an estimated cost figure

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Per-Model Performance Commentary

What changed — and what it did to recall and precision — for each of the 7 models

Model	Configuration	Recall (Train → Val)	Observation
Model 4	2 hidden layers (14→7), Adam optimizer	**93.4% → 92.3%**	The winner — high recall AND high precision (~99%→98.5%). Adam found a far better fit than SGD, with no dropout or class weights needed.
Model 3	+Dropout (50%) + Class Weights (1x), SGD	91.6% → 90.4%	Fixed Model 2's collapse — class weighting worked — but precision fell to ~74%: more false alarms in exchange for recall.
Model 6	+Dropout (50%) + Class Weights (2x), SGD, seeded rerun	91.1% → 89.6%	Barely moved recall past Model 3, but precision fell further (~68%). Doubling the weight mostly added false alarms.
Model 1	2 hidden layers (14→7), SGD optimizer	88.6% → 88.0%	Extra depth alone lifted recall over baseline while precision stayed very high (~97%) — no false-alarm cost.
Model 0	1 hidden layer (7), SGD — baseline	85.4% → 86.8%	Reasonable floor, but the simplest architecture with no imbalance handling leaves real failures on the table.
Model 5	+Dropout (50%), Adam, no class weights	82.4% → 83.7%	Below Model 4 despite using Adam — dropout regularization hurt recall on this imbalanced target without adding value.
Model 2	+Dropout (50%), SGD, no class weights	**50.0% → 50.0%**	Collapsed to chance level — dropout without any imbalance handling stopped the model from learning the minority class at all.

Pattern across all 7 runs: Adam consistently outperformed SGD; class weighting reliably fixed dropout-driven collapse,but always cost some precision in return.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Test Performance of Best Model

Model 4, evaluated once on the untouched 5,000-row Test.csv — the true, unbiased result

98.76%	90.34%	97.74%	93.69%
Accuracy	Recall	Precision	F1 Score

Recall dipped slightly from validation (92.3%) to test (90.3%)

Confusion Matrix — Test Set (n = 5,000)

	Predicted: No Failure	Predicted: Failure
Actual: No Failure	**4,710** TN — correctly cleared	**8** FP — false alarm (inspection cost)
Actual: Failure	**54** FN — missed failure (replacement cost)	**228** TP — correctly caught (repair cost)

Of 282 real failures in the test set, 228 were caught (81% by-class recall) and only 54 slipped through — while just 8 of 4,718 healthy units triggered a false alarm.

Estimated Cost Impact

Using relative weights: replace = 10, repair = 3, inspect = 1

Outcome	Count	Unit Cost	Total
TP (repair)	228	3	684
FN (replace)	54	10	540
FP (inspect)	8	1	8

Total modeled cost: 1,232 units

vs. reactive-only baseline (every failure → replacement): 282 × 10 = 2,820 units

≈ 56% lower estimated maintenance cost vs. reactive-only

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Questions Answered by the Model

What the modeling process told us about ReneWind's failure-prediction problem

Q **Can we predict generator failure before it happens?**

Yes. Model 4 catches 92.3% of real failures on unseen validation data — well ahead of breakdown, with a train→val gap under 1 point.

Q **Which model configuration works best for this problem?**

A simple 2-hidden-layer network with the Adam optimizer (Model 4) — it outperformed deeper, dropout-regularized, and class-weighted alternatives on every metric.

Q **Does the severe class imbalance require special handling?**

Not necessarily via class weighting. Model 4 achieved the best recall and precision without it; doubling the minority class weight (Model 6) hurt precision without improving recall.

Q **Is the ciphered sensor data informative enough to detect failure?**

Yes. Despite having no visibility into what each of the 40 sensors measures, the anonymized readings carry a strong enough signal for over 90% recall.

Q **Does the chosen model generalize, or is it overfit?**

It generalizes well. Validation recall stayed within roughly a point of training recall across every model tested, including the recommended Model 4.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Actionable Insights

What the results mean in practice for ReneWind's maintenance operation

#	Insight	What It Means
1	**Most real failures are caught early**	92.3% of failures are flagged before breakdown — directly shifting spend from full replacement to lower-cost repair on those units.
2	**False alarms stay rare**	98.5% precision means very few healthy units get flagged — inspection overhead from acting on the model's alerts stays minimal.
3	**A residual gap remains**	~8% of real failures still slip through. The model reduces, but doesn't eliminate, replacement-cost exposure — plan for ongoing manual monitoring too.
4	**No extra instrumentation needed**	The existing 40 sensor readings, with no feature engineering, already carry enough signal for strong predictions — no new hardware required.
5	**Some sensors may be redundant**	A handful of predictor pairs are very strongly correlated (e.g., V7 & V15, V2 & V14) — worth reviewing whether all 40 sensors are still necessary.
6	**The winning model is simple**	Model 4 needed no dropout or class weighting to win — fewer moving parts to tune and maintain in a production pipeline.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Business Recommendations

Turning Model 4's results into an operating plan for ReneWind

#	Recommendation	Action
1	**Deploy Model 4 as the early-warning system**	Score incoming sensor data and auto-generate repair/inspection tickets from failure-risk flags.
2	**Keep a human in the loop initially**	Route flags to inspection rather than fully automating replacement decisions, given ~8% of failures still go undetected.
3	**Track realized costs post-deployment**	Measure actual repair/replace/inspect spend against the assumed cost ordering to validate savings and refine thresholds.
4	**Re-validate on a regular cadence**	Periodically retrain/retest as turbines age and operating conditions shift, to catch performance drift early.
5	**Review sensor redundancy**	The strongly correlated predictor pairs are a candidate list for consolidating instrumentation and cutting hardware cost.
6	**Consider unmasking sensor identities internally**	For the final model, this would enable root-cause diagnostics on flagged units, not just prediction.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

APPENDIX

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Data Overview

Anonymized sensor data, split into training, validation, and test sets

X_train

16,000

observations

80% of Train.csv; used to fit each candidate model.

X_val

4,000

observations

20% of Train.csv, stratified; used to compare and tune models.

X_test

5,000

observations

Separate Test.csv; touched only for final evaluation.

80% X_train | **20% X_val**

Train.csv (20,000 rows) is split 80/20, stratified on Target, into X_train (used to fit each model) and X_val (used to compare and tune candidates). Test.csv (5,000 rows) is a wholly separate file, held out for final evaluation only

Data Dictionary

Derived from Train.csv (20,000 rows × 41 columns)

Variable(s)	Type	Description	Missing
V1, V2	Numeric (continuous)	Ciphered/transformed sensor reading	18 each (0.09%)
V3–V40	Numeric (continuous)	Ciphered/transformed sensor reading	0
Target	Binary (0 / 1)	1 = failure, 0 = no failure (label to predict)	0

All 40 predictors are anonymized, continuous sensor-derived features; original sensor identities (temperature, vibration, etc.) are withheld to protect ReneWind's confidential instrumentation.

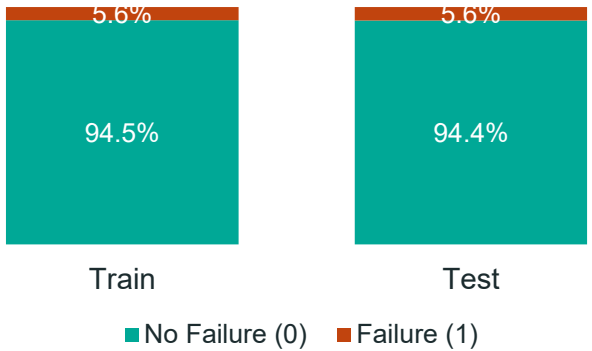
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

EDA Results

Severe class imbalance and outlier-heavy predictors shape the modeling strategy

Target Class Balance

Proportion of each label, Train vs. Test

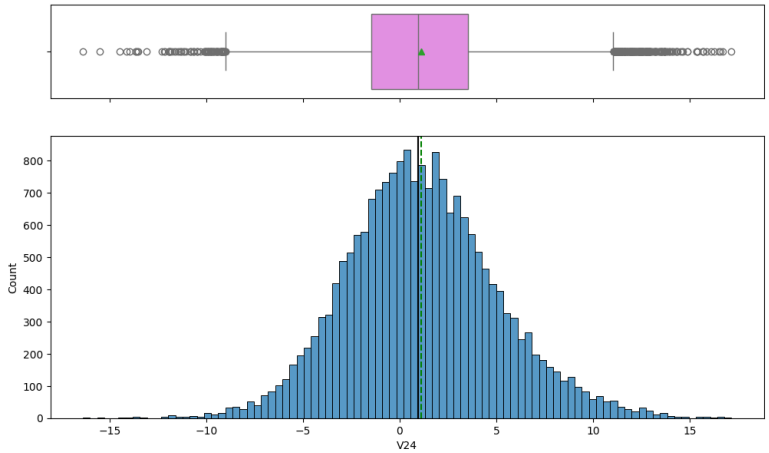


Only ~5.5% of observations are labeled failure — and that split is nearly identical in train (5.55%) and test (5.64%), confirming a representative split.

Implication: with this much imbalance, accuracy is a misleading metric. Model selection should favor recall / cost-sensitive metrics, and techniques like class weighting or resampling (SMOTE, undersampling) are worth testing during tuning.

Distribution of V24 (representative predictor)

Boxplot + histogram, Train.csv



Roughly symmetric, centered near 1 (mean ≈ median), typical of most of the 40 sensor-derived predictors.

Heavy-tailed outliers on both sides (below −9 and above 11) — a pattern seen across many predictors. This favors outlier-robust scaling (e.g., robust/median scaling) or tree-based models, which are naturally less sensitive to extreme values.

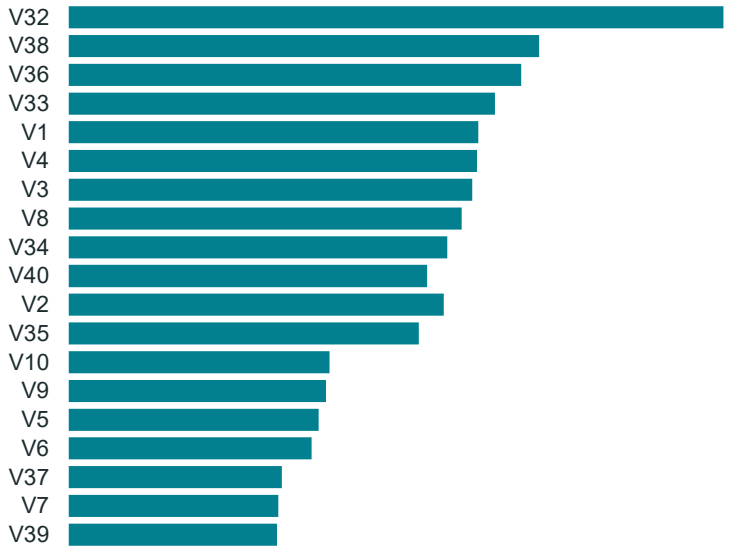
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Univariate Analysis — Predictor Distributions

Variability differs a lot across predictors — but wide ranges and outliers are the norm

Standard Deviation by Predictor

Sample of 19 predictors, Train.csv



Key Observations

- **>3× spread in variability**

  Standard deviations range from ~1.75 (V37, V39) up to 5.50 (V32) — predictors are on meaningfully different scales.

- **V24 is representative, not unusual**

  The bell-shaped, heavy-tailed pattern shown for V24 earlier appears broadly across these predictors — not a one-off.

- **Two predictors carry missing values**

  Only V1 and V2 have gaps (18 each, 0.09%) — addressed with median imputation, fit on training data only.

- **Scale differences matter for some models**

  This range supports the earlier scaling decision — tree/NN-based models handle it natively, but RobustScaler would matter for distance-based models.

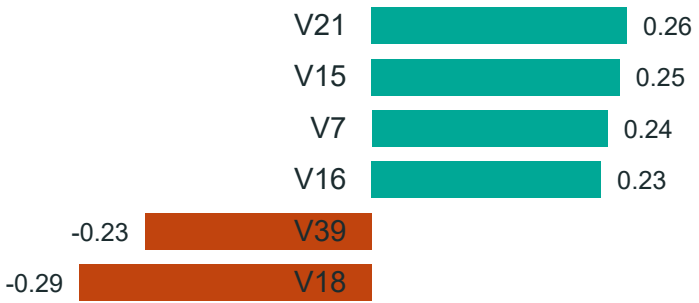
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Bivariate Analysis — Predictors vs. Target

A handful of features shift sharply between healthy and failing units

Correlation with Target

Top 6 predictors by |r|, Train.csv



Observation: no single predictor is strongly correlated with Target on its own — the strongest, V18, sits at only |r| = 0.29. Individually weak but broadly present signal across many features is exactly what a neural network is well suited to combine.

Class Separation — Mean by Target

Same 6 predictors, mean value for failure vs. no failure

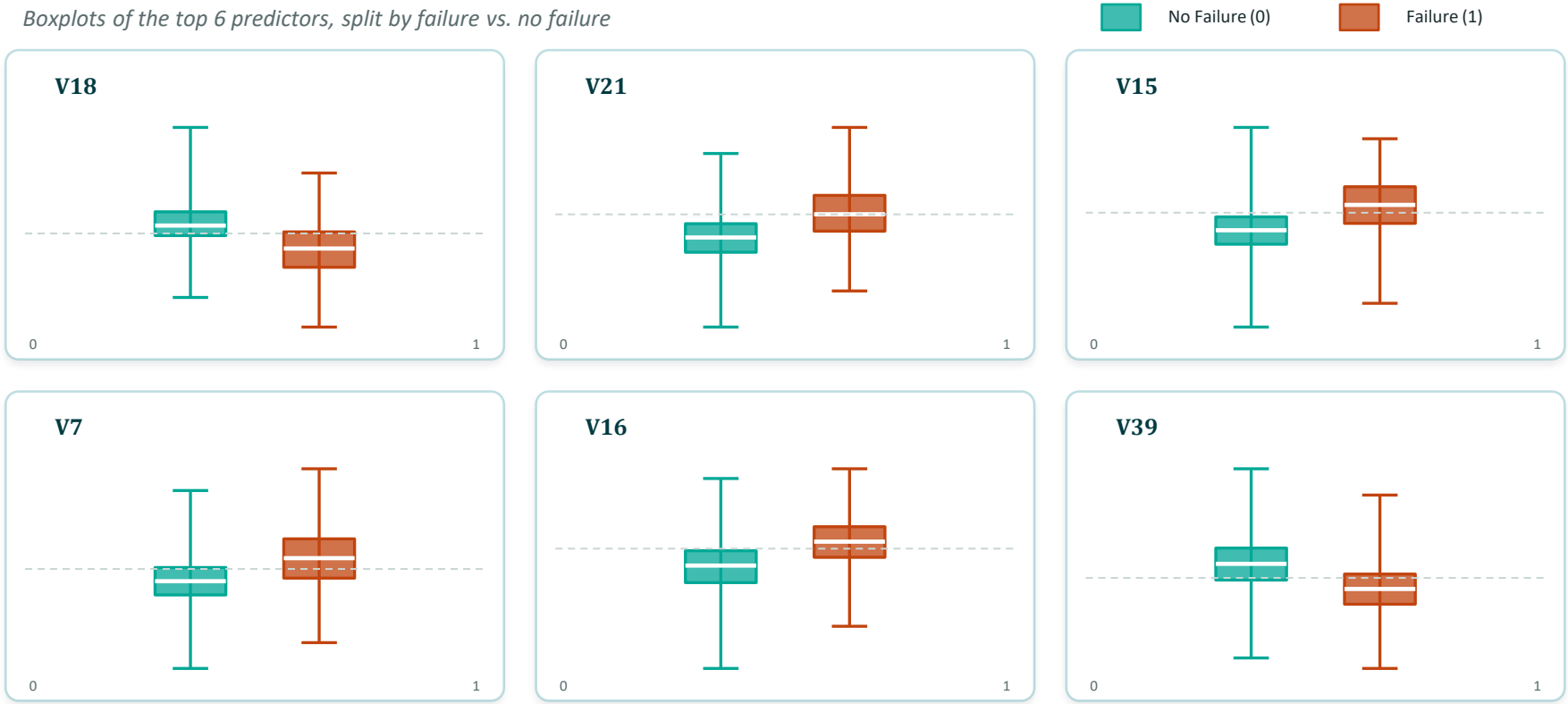
Feature	Mean (No Failure)	Mean (Failure)	Shift
V18	1.37	−1.95	**−3.32**
V39	0.99	−0.75	**−1.74**
V16	−3.16	1.09	**+4.25**
V7	−0.98	0.84	**+1.82**
V15	−2.62	1.03	**+3.65**
V21	−3.83	0.16	**+4.00**

V16 and V21 swing roughly 4 points in mean value between classes; V18 flips sign entirely. These predictors carry a strong, learnable signal even without knowing what the sensors measure.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Bivariate Analysis — Distribution by Target

Boxplots of the top 6 predictors, split by failure vs. no failure

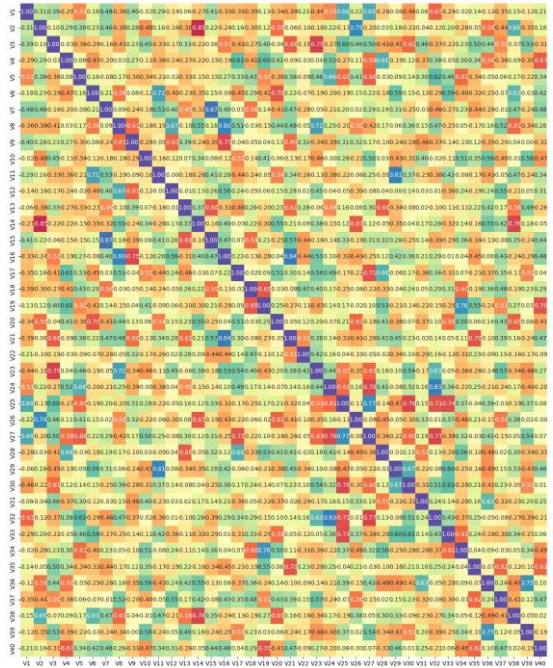


Failure cases (red) sit on the opposite side of the distribution from healthy units (teal) for all six predictors — boxes barely overlap for V16 and V21.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Correlation Analysis

Several predictor pairs are strongly correlated — a multicollinearity signal for modeling



Key Observations

- A subset of predictor pairs show strong correlation, both positive and negative (|r| roughly 0.75–0.87) — for example pairs like V7 & V15, V2 & V14, V16 & V21, and V8 & V16 stand out as the most tightly linked.
- Most predictors, however, show weak-to-moderate correlation with each other (|r| well under 0.5), so the strong pairs are the exception rather than the rule.
- Because features are ciphered, we can't name the underlying sensors — but pairs this strongly linked likely capture overlapping or derived measurements (e.g., a raw reading and a rolling average of it).

Modeling implication: high correlation among a few predictor pairs means some algorithms (e.g., plain logistic regression) may need regularization or feature reduction, while tree-based ensembles (Random Forest, Gradient Boosting) are naturally more robust to this redundancy.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Data Preprocessing

Preparing features and labels for modeling, without leaking information across sets

01 Separate features & target

Drop “Target” from the training data to form X; y holds the label. The same split is applied to the held-out test file (X_test / y_test).

02 Create a validation set

train_test_split(X, y, test_size=0.2, random_state=1, stratify=y) carves X_train (16,000) and X_val (4,000) out of Train.csv, preserving the 5.5% failure rate in both.

03 Impute missing values

SimpleImputer(strategy= “median”) is fit only on X_train, then used to transform X_val and X_test — the same learned medians are applied everywhere.

04 Finalize label arrays

y_train, y_val, and y_test are converted to NumPy arrays (.to_numpy()) so they're in the format scikit-learn model-fitting expects.

Why fit the imputer on X_train only? Learning the median from the full dataset (or from validation/test) would let information about those rows leak into training, quietly inflating validation and test performance. Fitting on X_train alone and re-using those exact medians for X_val and X_test keeps the evaluation honest — it reflects how the model would behave on genuinely unseen data.
Result: a final check (X_train/X_val/X_test.isna().sum()) confirms zero missing values remain across all three sets.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Outlier Check & Feature Engineering

Both were reviewed; neither required action

Outlier Check — No Treatment Applied

Observed: most predictors (e.g., V24) show heavy-tailed distributions with real outliers on both sides, consistent with the EDA.

Decision: left untreated. Extreme sensor values may be genuine early-failure signals — removing them risks discarding the rare positive-class signal we most need.

Mitigation: tree-based models (our primary candidates) aren't distorted by outliers; if distance-based models are tried, RobustScaler will be used instead of removing points.

Feature Engineering — None Added

Observed: all 40 predictors are ciphered/anonymized numeric values — there's no domain meaning to build informed ratios, interactions, or derived features from.

Decision: no new features created. All columns are already numeric, so no encoding is needed either.

Watch item: correlation analysis flagged a few strongly linked pairs (e.g., V7 & V15, V2 & V14) — relevant for Logistic Regression coefficient stability, but doesn't require new features.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Model Performance Summary — Training Set

Seven neural network configurations, ranked by Recall (our primary metric)

Model	Configuration	Accuracy	Recall	Precision	F1	F2
Model 4	2 hidden layers (14→7), Adam optimizer	**99.22%**	**93.39%**	**99.10%**	**96.04%**	**94.48%**
Model 3	3 layers + 50% dropout + class weights (1x), SGD	94.12%	91.64%	73.84%	79.73%	87.43%
Model 6	3 layers + 50% dropout + class weights (2x), SGD — seeded rerun	91.23%	91.11%	68.66%	74.33%	85.52%
Model 1	2 hidden layers (14→7), SGD optimizer	98.59%	88.55%	97.72%	92.59%	90.24%
Model 0	1 hidden layer (7), SGD — baseline	98.17%	85.36%	96.53%	90.09%	87.38%
Model 5	3 layers + 50% dropout, Adam optimizer	97.96%	82.44%	97.64%	88.42%	85.09%
Model 2	3 layers + 50% dropout, SGD — no class weights	**94.45%**	**50.00%**	**47.23%**	**48.57%**	**49.42%**

Model 4 leads on training data — highest recall (93.4%) while also topping accuracy and precision; the Adam optimizer clearly outperformed SGD here. Model 6's re-run (2× minority class weight) barely moved recall versus Model 3 (91.1% vs 91.6%) but precision fell sharply (68.7% vs 73.8%) — pushing the weighting higher mainly added false positives without meaningfully catching more failures. Model 3's lighter weighting looks closer to the right balance. Model 2 collapsed to chance-level recall (50%) — dropout without class weighting couldn't learn the minority class. Caveat: training-set numbers only — validation performance will determine the final candidate.

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Model Performance Summary — Validation Set

The set that decides generalization — same ranking holds, and the gap to training is small

Model	Configuration	Accuracy	Recall	Precision	F1	F2	Recall gap (val – train)
Model 4	Adam, 2 hidden layers	**99.05%**	**92.29%**	**98.51%**	**95.16%**	**93.47%**	−0.9 pt
Model 3	SGD, +Dropout, +CW (1x)	94.25%	90.38%	74.06%	79.65%	86.57%	−1.3 pt
Model 6	SGD, +Dropout, +CW (2x)	90.73%	89.58%	67.74%	73.12%	84.15%	−1.5 pt
Model 1	SGD, 2 hidden layers	98.48%	87.96%	97.05%	91.95%	89.64%	−0.6 pt
Model 0	SGD, 1 layer (baseline)	98.30%	86.80%	96.37%	90.97%	88.56%	+1.4 pt
Model 5	Adam, +Dropout	98.05%	83.70%	97.14%	89.17%	86.09%	+1.3 pt
Model 2	SGD, +Dropout, no CW	**94.45%**	**50.00%**	**47.23%**	**48.57%**	**49.42%**	0.0 pt

Recommended model: Model 4. **It holds the #1 spot on validation recall (92.3%) while also leading accuracy and precision — no other model comes close to that combination. The train→val recall gap is under 1 point, so this performance looks like genuine generalization, not an overfit training score.** **Runner-up:** **Model 3 is the strongest alternative if a simpler, more recall-aggressive model is preferred, but its precision (74%) is notably weaker.**

Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.

Great Learning

Happy Learning !

