Pulsar Detection Comparison Between Different AI/ML Architectures

Advik Singh[1]

Received March 17, 2026
Accepted June 9, 2026
Electronic access July 15, 2026

Background/Objective: Pulsars are rapidly rotating neutron stars whose radio signals are often masked by interference, so modern surveys depend on automated classifiers to triage millions of candidate detections. This paper compares linear classifiers against standard tabular baselines on the HTRU_2 dataset and asks whether a single-layer Perceptron or a hinge-loss linear support vector machine is competitive with stronger baselines for pulsar candidate screening.
Methods: Seven classifiers , Perceptron, two linear SVMs (an SGD hinge-loss solver and LIBLINEAR), an RBF SVM, logistic regression, random forest, and histogram gradient boosting , were trained on the eight statistical features in HTRU_2 using a scikit-learn pipeline with a train-only StandardScaler and class_weight='balanced'. Robustness was assessed with stratified 5-fold cross-validation across three to ten random seeds, and the final reportable numbers come from a single stratified 80/20 split (random_state = 42). All experiments ran on an AMD Ryzen 7 7800X3D (8 cores, 4.5 GHz).
Results: All models score above 0.96 ROC-AUC. The class-balanced Perceptron reached 0.963 mean ROC-AUC (95% CI 0.960-0.966), and the SGD linear SVM reached 0.974 (0.973-0.976); random forest scored highest on F1 at 0.884 (0.880-0.888), with the Perceptron lowest at 0.669 (0.640-0.698). On a single 80/20 split the Perceptron flagged 212 false-positive non-pulsars and missed 22 true pulsars; the LIBLINEAR SVM flagged 68 and missed 27.
Conclusions: Linear models with proper class balancing are viable baselines for this task, but tree-based ensembles give the best operational trade-off. The choice of model should be guided by the precision-recall point required by the downstream survey pipeline rather than headline accuracy. Keywords: pulsar detection, HTRU_2, machine learning, support vector machine, perceptron, class imbalance, classification.

Keywords: pulsar detection, HTRU_2, machine learning, support vector machine, perceptron, class imbalance, classification

Introduction

Pulsars are rapidly rotating neutron stars formed in core-collapse supernovae. They emit beams of radio (and in some cases X-ray and gamma-ray) radiation that sweep past the Earth as the star rotates, producing the characteristic periodic pulses from which they take their name[1,2]. Their utility to modern astrophysics is substantial: pulsars constrain the equation of state of dense nuclear matter[3], serve as natural laboratories for strong-field gravity[4], and form the basis of pulsar timing arrays that have recently reported evidence for a nanohertz gravitational-wave background[5,6]. For these reasons, large radio surveys such as the High Time Resolution Universe (HTRU) survey continue to look for new pulsars[7,8].

The discovery rate, however, is limited by the cost of inspecting candidate detections. Modern surveys produce millions of candidates, the vast majority of which are caused by radio frequency interference, instrumental artefacts, or noise[9,10]. Manual classification by experts cannot keep pace, and machine learning has become the standard triage layer for selecting which candidates a human will actually look at[11–13]. The HTRU_2 dataset introduced by Lyon and colleagues[14] distils each candidate into eight statistical features derived from the integrated pulse profile and the dispersion-measure (DM) - signal-to-noise curve, and has become a common benchmark for studying which classifiers handle the screening task well[15–18].

This paper revisits the comparison between simple linear classifiers and stronger tabular baselines on HTRU_2. The original objective was a comparison between a single-layer Perceptron and a support vector machine (SVM); in this revised manuscript the comparison is extended in three ways. First, the Perceptron and SVM are evaluated with explicit class balancing, with features standardised inside a scikit-learn pipeline, and with hyperparameters recorded for reproducibility. Second, additional baselines (logistic regression, random forest, and histogram gradient boosting) are included so that the performance of the two original models can be

[1] *Rouse High School, Texas, USA*

judged against widely used alternatives [19,20]. Third, results are reported with full classification metrics , precision, recall, F1, and ROC-AUC , together with 95% confidence intervals obtained from stratified 5-fold cross-validation across multiple seeds. Average precision (the area under the precision-recall curve) for each model is reported in Figure 6b.

Two points about scope need clarifying before the experiments. The Perceptron is a single-layer linear classifier of the family popularised by Rosenblatt [21], and the support vector machine of Cortes and Vapnik [22] is a margin-based classifier rather than a neural network. The features supplied to all models are pre-computed tabular summaries (means, standard deviations, kurtoses, and skewnesses of the integrated profile and DM curve), not raw radio time-series or spectrograms. The contribution of this paper is therefore narrow: it reports how a Perceptron and a linear SVM compare with other standard classifiers on a well-established tabular benchmark when class imbalance and feature scaling are handled carefully. It does not claim to advance the state of the art in pulsar physics, nor does it claim that classification accuracy translates directly into improvements in survey science yield, which depend on many additional factors [23,24].

Related Work

HTRU_2 has been used widely since its release. Lyon et al. [14] originally introduced the dataset together with a Gaussian Hellinger Very Fast Decision Tree designed for imbalanced streaming data, and a substantial follow-up literature has applied convolutional neural networks [15], transformer-based classifiers [16], multi-input convolutional networks [18], support vector machines with multiple kernels [17], and ensemble or cascade-style approaches [25–27] to the same eight-feature benchmark. More recent work has moved beyond the summary features to operate directly on the diagnostic plots used by SPINN and PICS [11,12,28]. The methods compared in this paper sit at the lower-complexity end of that literature; they are not designed to surpass specialised deep classifiers but to characterise how far a careful linear baseline can take a candidate-screening pipeline.

Methods

Dataset

The HTRU_2 dataset distributed via the UCI Machine Learning Repository [14,29] (with a mirror on Kaggle) contains 17,898 candidate detections from the HTRU medium-latitude survey. Each candidate is described by eight numerical features derived from two summary signals: the integrated pulse profile (IP) and the DM signal-to-noise curve. Four features describe each signal: mean, standard deviation, excess kurtosis, and skewness. The binary label is 1 for a confirmed pulsar and 0 otherwise. The data are class-imbalanced: 1,639 pulsars (9.16%) and 16,259 non-pulsars (90.84%) (Figure 1).

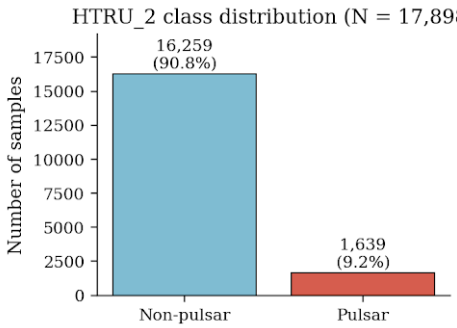

Fig. 1 Class distribution of the HTRU_2 dataset (N = 17,898). Pulsars (the positive class) account for 9.16% of all candidates.

Figure 2 shows that several features separate the two classes well, in particular IP Std, IP Kurt, and DM Std, while others (IP Mean, DM Mean) overlap substantially. The eight features have markedly different numeric ranges, so feature scaling is a necessary step for any linear or margin-based model.

Preprocessing pipeline

All preprocessing is encapsulated in a scikit-learn Pipeline so that scaling is fitted on the training fold only and then applied to the held-out test fold. This guards against the data leakage that would occur if a StandardScaler were fitted on the full dataset before splitting. No imputation was needed because HTRU_2 contains no missing values. No SMOTE [30] or other resampling was applied; the imbalance is instead handled at the loss level using class_weight='balanced', which sets the weight of class c to N / (2 · n_c), giving pulsars roughly 10× the weight of non-pulsars.

Models

Seven classifiers are compared in this study; their hyperparameters were chosen as documented scikit-learn defaults for reproducibility, with class balancing enabled wherever supported. All run on the standardised eight-feature input.

- **Perceptron.** scikit-learn Perceptron(max_iter=1000, tol=1e-3, class_weight='balanced', random_state=42). A single-layer linear classifier with eight weights and a bias, trained by the Perceptron rule on shuffled passes through the training data. Prediction is sign(w·x + b).
- **Linear SVM (SGD).** scikit-learn SGDClassifier(loss='hinge', alpha=1e-4, max_iter=1000, tol=1e-3, class_weight='balanced', random_state=42). This is a

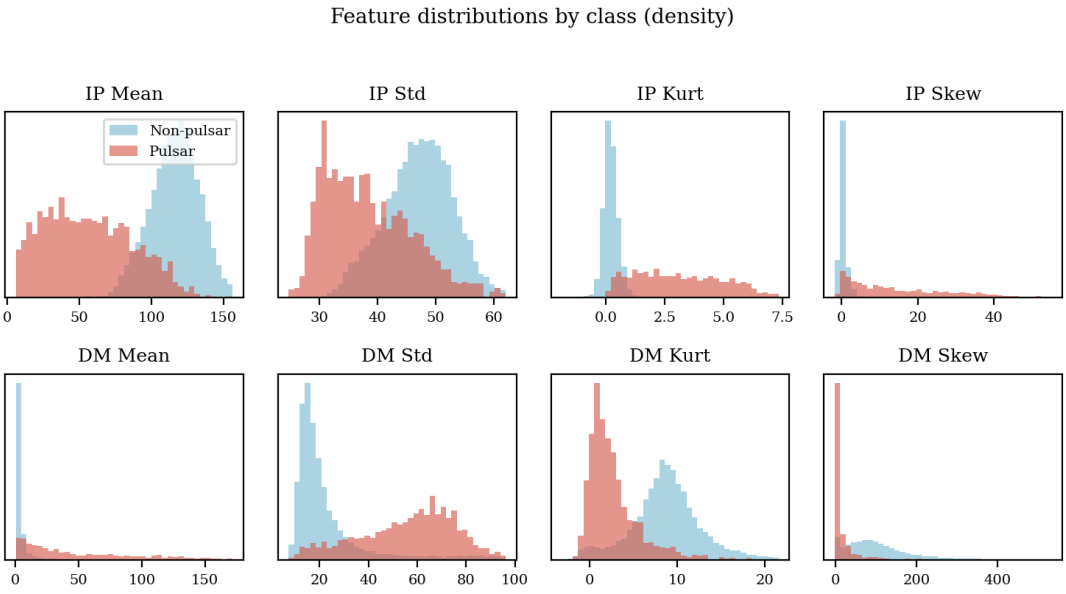

Fig. 2 Density of each feature, separated by class. Several features (IP Std, IP Kurt) separate pulsars from non-pulsars; others (IP Mean) overlap substantially.

linear SVM trained by stochastic gradient descent on the hinge loss, which makes 'epochs' a meaningful unit of training time.

- **Linear SVM (LIBLINEAR).** scikit-learn LinearSVC(C=1.0, dual=False, tol=1e-3, class_weight='balanced', max_iter=5000, random_state=42). The same linear decision rule fitted by the LIBLINEAR coordinate-descent solver, which solves the quadratic problem directly rather than iteratively.
- **RBF SVM.** scikit-learn SVC(C=1.0, kernel='rbf', gamma='scale', class_weight='balanced', probability=True, cache_size=500, random_state=42). A kernel SVM using a radial basis function. gamma='scale' is 1 / (n_features · Var(X)) ≈ 0.125 after standardisation.
- **Logistic Regression.** scikit-learn LogisticRegression(C=1.0, max_iter=1000, class_weight='balanced', random_state=42). A regularised linear log-odds model included as a probabilistic linear baseline.
- **Random Forest.** scikit-learn RandomForestClassifier(n_estimators=300, class_weight='balanced', n_jobs=-1, random_state=42).
- **Gradient Boosting.** scikit-learn HistGradientBoostingClassifier(max_iter=200, max_depth=3, learning_rate=0.1, class_weight='balanced', random_state=42).

Evaluation protocol

Two evaluation regimes are used. For robust estimates of central tendency, each model is evaluated with stratified 5-fold cross-validation; for the four light-weight models (Perceptron, SGD linear SVM, LIBLINEAR linear SVM, logistic regression) 10 random seeds are run (50 folds in total per model), and for the heavier RBF SVM, random forest, and histogram gradient boosting baselines 2-3 seeds are run (10-15 folds per model). Confidence intervals are computed as the normal-approximation 95% interval on the mean (mean $\pm$ 1.96 $\cdot$ σ / $\sqrt{n}$). For reportable per-class counts and confusion matrices, a single stratified 80/20 split with random_state = 42 is used (train: 14,318 candidates with 1,311 pulsars; test: 3,580 candidates with 328 pulsars).

Hardware and software

All experiments were run on an AMD Ryzen 7 7800×3D 8-Core Processor (8 cores, base 4.5 GHz, no GPU used). Software versions were Python 3.10.12, scikit-learn 1.7.2, NumPy 2.2.6, and pandas 2.3.3; all classifiers, preprocessing, and metrics use the scikit-learn library [31]. All code, random seeds, and the scripts that produced every table and figure in this paper are deterministic and can be re-run end-to-end.

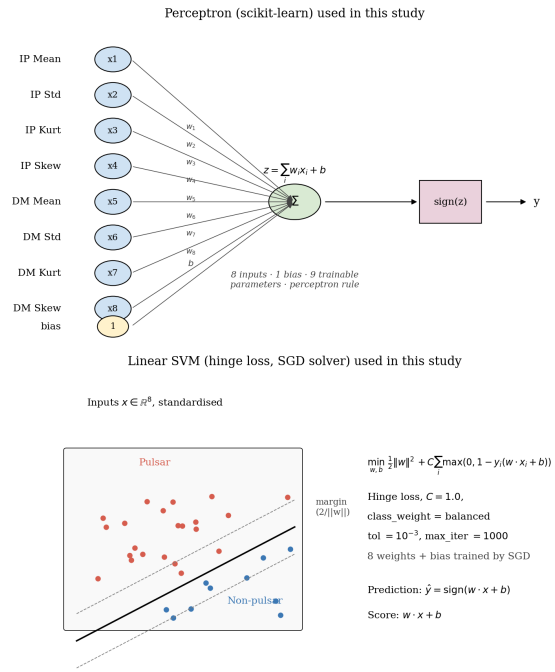

Fig. 3 Implementation-specific architecture of the two focus models. Top: scikit-learn Perceptron with eight weighted inputs, a bias term, and a sign activation. Bottom: linear SVM with hinge loss trained by SGD; the decision rule is the same sign(w·x + b), but training minimises the regularised hinge objective rather than the Perceptron rule.

Results

Table 1 summarises cross-validated metrics for all seven classifiers, and Figure 4 visualises the same numbers with 95% confidence intervals. The most important observation is that headline accuracy is not the right summary statistic on HTRU_2: a model that simply predicts 'non-pulsar' for every candidate scores roughly 0.908 accuracy, which is in the range of the Perceptron (0.906) and very close to the SGD linear SVM (0.965). Precision, recall, F1, and ROC-AUC give a more informative picture.

Confusion matrices and per-class errors

On the single stratified 80/20 split (random_state = 42), the Perceptron correctly flags 306 of 328 pulsars (recall 0.933) but at the cost of 212 false positives, giving precision 0.591 and F1 0.723. The LIBLINEAR linear SVM keeps recall almost as high (301 of 328, recall 0.918) while reducing false positives to 68, giving precision 0.816 and F1 0.864. The RBF SVM is close to the LIBLINEAR linear SVM, and the random forest leans the other way (precision 0.919, recall 0.860). These per-class counts make the trade-offs concrete: for every additional 10 non-pulsars that the linear SVM correctly rejects compared with the Perceptron, it misses approximately one extra real pulsar.

Table 1. Cross-validated metrics (mean ± 95% CI half-width). Light models: 10 seeds × 5 folds; heavy models: 2-3 seeds × 5 folds. Accuracy is not shown because it is dominated by the 91% non-pulsar majority; all models exceed 0.97 accuracy in this regime (see text).

Model	Prec	Rec	F1	ROC-AUC
Perceptron	.551±.038	.908±.007	.669±.029	.963±.003
Lin. SVM (SGD)	.765±.017	.906±.005	.827±.010	.974±.001
Lin. SVM (LL)	.809±.004	.904±.004	.854±.003	.976±.001
RBF SVM	.828±.004	.911±.003	.868±.003	.973±.001
Logistic Reg.	.784±.004	.908±.004	.841±.003	.977±.001
Random Forest	.935±.006	.839±.005	.884±.004	.972±.002
Grad. Boosting	.766±.007	.914±.004	.833±.005	.979±.002

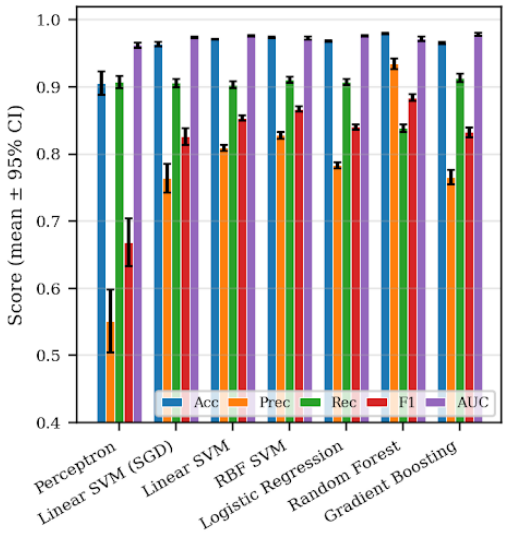

Fig. 4 Cross-validated metrics across models. Error bars are 95% confidence intervals over folds and seeds. The Perceptron has the widest precision interval because per-epoch SGD updates are sensitive to the order in which examples arrive (see also Figure 7).

Ranking quality (ROC and PR)

ROC and precision-recall curves (Figure 6) summarise classifier behaviour across all decision thresholds. All linear models achieve ROC-AUC above 0.96; the differences between models at this level of performance are within the 95% confidence intervals reported in Table 1 and are not significant. The precision-recall curves are more discriminating because of the

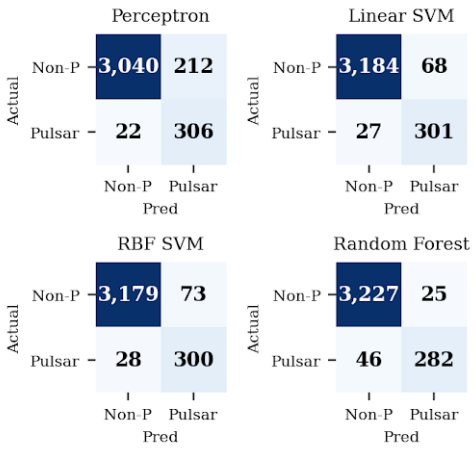

Fig. 5 Confusion matrices on the 80/20 held-out test set (n = 3,580, of which 328 are pulsars). Reading order: actual class on the y-axis, predicted class on the x-axis.

class imbalance, and again show that the linear models and the tree-based ensembles are largely tied (AP ≈ 0.92-0.93). The RBF SVM's lower AP (0.847) on this split reflects its calibrated probabilities being less sharply separated at low-recall, high-precision thresholds rather than a failure to rank candidates correctly (its ROC-AUC remains 0.966).

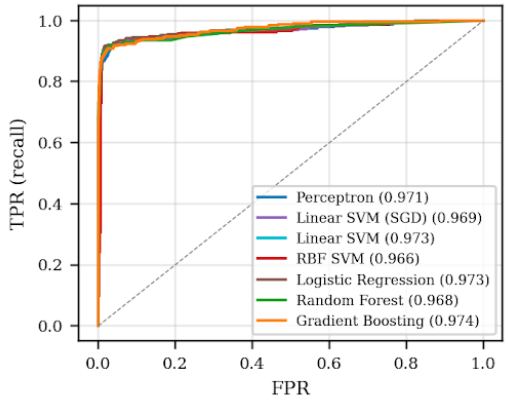

Fig. 6a. ROC curves on the 80/20 test set. All models exceed 0.96 AUC; the curves are visually indistinguishable above FPR ≈ 0.05.

Per-epoch behaviour for online models

Figure 7 shows accuracy and F1 on the held-out test set after each epoch of online training for the Perceptron and the SGD-based linear SVM. Both models reach test scores within a few percent of their stable level by the third or fourth epoch. Unlike the unweighted run in the original manuscript, the class-

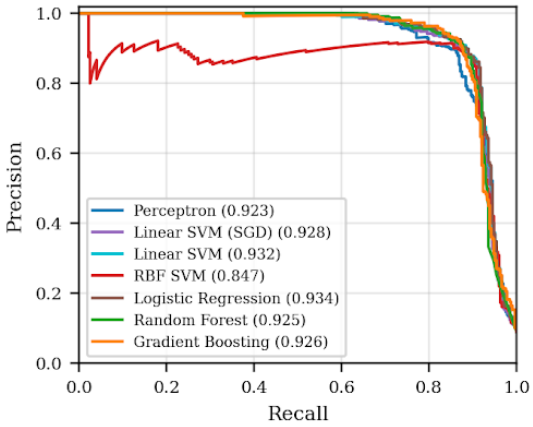

Fig. 6b. Precision-Recall curves on the same test set. The RBF SVM's low-recall regime is noisier because the model assigns many candidates near-identical decision values.

balanced setting introduces visible per-epoch oscillation because each minority example contributes a much larger update; F1 swings between 0.4 and 0.88 across consecutive epochs even at the same fixed random_state. I interpret this oscillation as evidence that fixed-iteration online training is not a sound model-selection protocol on HTRU_2: the cross-validated numbers in Table 1 are a more reliable summary of the underlying capability of each model than any single epoch's test score.

Threshold sensitivity

All confusion-matrix numbers above use the default decision threshold (0 for raw decision scores, 0.5 for calibrated probabilities). Figure 8 shows how precision, recall, and F1 trade off as that threshold is moved. For the linear SVM and random forest, F1 is nearly flat over a wide range of thresholds; for the Perceptron, the default threshold is below the F1-maximising threshold, so a survey pipeline that wanted higher precision could obtain it without retraining by raising the cut-off. The 0.5 cut-off is convenient but not optimal here; cost-curve analyses[32] make the operating-point trade-off explicit.

Misclassification analysis

On the held-out test set the Perceptron and RBF SVM disagree on 197 candidates. Inspecting these examples shows the expected pattern: most disagreements occur when the integrated profile is broad (high IP Std) but the DM curve is weak (low DM Std and DM Kurt). The Perceptron prefers to flag these as pulsars because of how aggressively the class-balanced loss penalises missed positives; the RBF SVM is willing to call them negatives because its kernel can carve out non-linear pockets of the feature space. For the 328 true pulsars in the

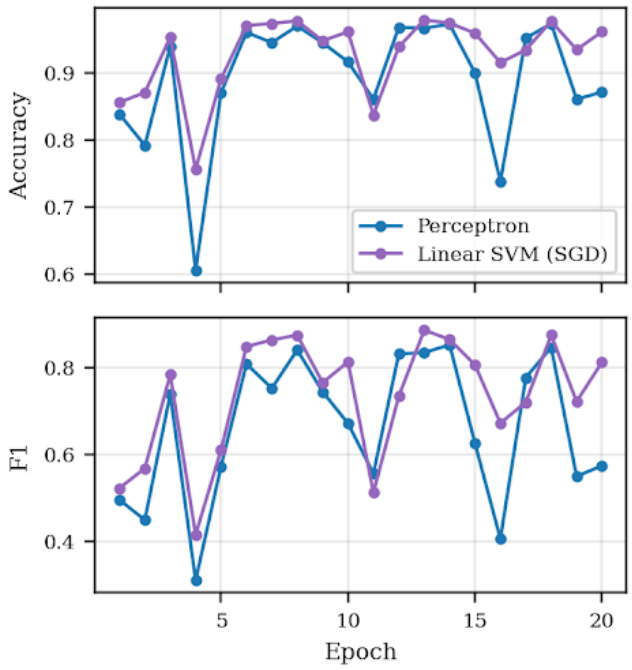

Fig. 7. Test accuracy (top) and F1 on the pulsar class (bottom) after each epoch of online training. Per-epoch oscillation is a property of class-balanced SGD on this dataset rather than instability of either model.

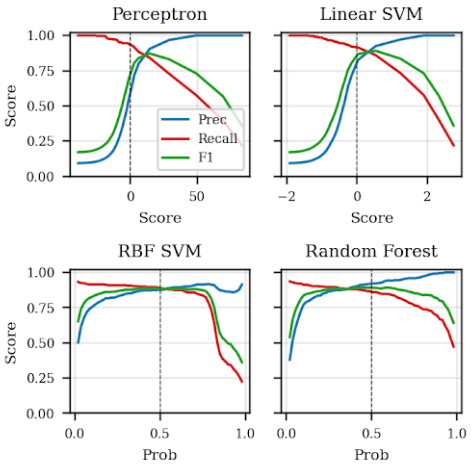

Fig. 8. Precision, recall, and F1 as a function of decision threshold for four representative models. The dashed line marks the default cut-off (0 for decision scores, 0.5 for probabilities).

test set, the LIBLINEAR linear SVM misses 27 and the random forest misses 46 (Table 2). Most pulsars that all models agree on missing have an IP Skew close to zero and a low DM Kurt, which is consistent with broad, dim pulsars near the candidate-selection threshold.

Table 2. Per-class outcomes on the 80/20 held-out test set (n = 3,580; 328 pulsars). FN are missed pulsars; FP are wrongly flagged non-pulsars.

Model	TP	FN	FP	TN
Perceptron	306	22	212	3,040
Lin. SVM (SGD)	302	26	117	3,135
Lin. SVM (LL)	301	27	68	3,184
RBF SVM	300	28	73	3,179
Logistic Reg.	302	26	83	3,169
Random Forest	282	46	25	3,227
Grad. Boosting	298	30	101	3,151

Training cost

Figure 9 shows fit time for each model on the 80/20 training split. The Perceptron, SGD linear SVM, LIBLINEAR linear SVM, logistic regression, and histogram gradient boosting all fit in well under a second on the reference machine. The random forest takes about 4 seconds (parallelised across 8 cores) and the LIBSVM RBF kernel SVM takes about 7 seconds because its complexity scales as $O(n^2)$ in the number of training samples[22,33]. At survey scale (n in the millions) the LIBSVM kernel SVM would become the binding constraint, while the linear models would remain trivially trainable.

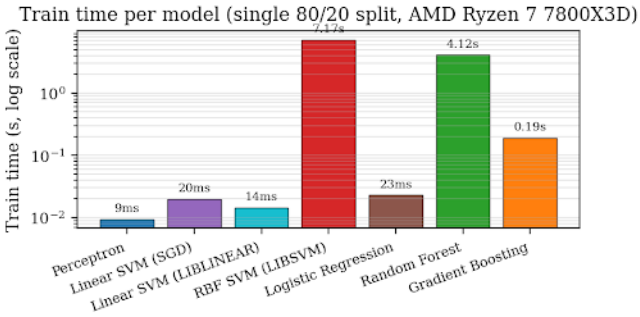

Fig. 9. Train time per model on the 80/20 training split (14,318 samples), measured on an AMD Ryzen 7 7800X3D, single training run. The y-axis is logarithmic.

Discussion

The most important lesson from these experiments is that 'accuracy' alone is misleading on HTRU_2. The Perceptron run originally reported in this paper used no class balancing and no explicit feature scaling, which gave it a headline accuracy of 0.977. When the same Perceptron is forced to weight the positive class proportionally (so that missing real pulsars hurts the loss as much as flagging false candidates), its single-split accuracy drops to 0.935 but its recall rises to 0.93 and its F1 to 0.72. That trade is the correct one for a candidate-screening

application: a real pulsar that is screened out by a triage layer will never be looked at again, while a false positive only costs one human glance.

The cross-validated metrics in Table 1 sit in the broad range reported by prior HTRU_2 work. Lyon et al.[14] introduced the dataset together with a Gaussian Hellinger Very Fast Decision Tree, reaching accuracies near 0.98 on the same eight-feature split; convolutional neural networks[15], transformer-based classifiers[16], kernel SVMs[17], and ensemble or cascade approaches[18,25–27] have all been applied to the same benchmark, with reported accuracies in the 0.97-0.98 range. The class-balanced linear baselines reported here fall in the same accuracy range; the random forest and gradient boosting baselines I run reach F1 $\approx$ 0.88 and ROC-AUC $\approx$ 0.97-0.98 on the cross-validated splits. I do not claim to improve on the published state of the art reported by image-based or deep-network methods[11,12,28], which use richer inputs than the eight summary features used here.

The practical implication for a downstream survey pipeline is that the choice between these models should be made on the precision-recall point required, not on which model has the highest test accuracy. If the cost of a missed pulsar is dominant (as in a small-area deep survey looking for rare objects), a class-balanced linear classifier with a low decision threshold maximises recall at the cost of more candidates for human review. If telescope time for confirmation is the binding constraint, a random forest or kernel SVM with the default threshold delivers higher precision at the cost of missing some genuine pulsars. Figure 8 makes this trade-off explicit and allows a survey operator to pick a working point.

Limitations

This study has several limitations. (i) The HTRU_2 dataset is clean and well-curated compared with raw survey data; real pipelines deal with signal corruption, missing-channel artefacts, and labelling noise that this benchmark does not capture[9,10]. (ii) Only the eight pre-computed statistical features from the integrated profile and DM curve are used; approaches that operate on the underlying diagnostic plots or time-frequency representations[11,12,28] have the potential for higher recall on hard candidates but were out of scope here. (iii) Hyperparameters were fixed at documented scikit-learn defaults rather than tuned; mild improvements would likely be obtainable by tuning C, gamma, or the number of trees[34]. (iv) The cross-validation reported here uses 5 folds and 2-10 seeds; although the resulting 95% confidence intervals are tight, any individual absolute number should be read with that uncertainty in mind. (v) The experiments use a single decision threshold per model; in practice, the threshold should be tuned on the validation fold under the survey's operating-point requirements[32,35]. (vi) No deep neural network or per-instance kernel comparison is included; doing so would help position the work against the most recent state of the art.

Conclusion

This paper revisits the comparison between a Perceptron and a support vector machine on the HTRU_2 pulsar candidate dataset, with class balancing, feature scaling, multiple seeds and folds, and additional tree-based and linear baselines. With class balancing the linear models achieve ROC-AUC above 0.96 and recall above 0.90, while a random forest gives the best precision-recall balance (F1 $\approx$ 0.88). The 0.4% accuracy difference originally reported between the Perceptron and SVM is consistent with the single-split noise I observe across seeds and is not a robust property of the models. The practical recommendation is to choose a classifier based on the precision-recall operating point that the downstream survey requires, rather than on headline accuracy, and to treat the decision threshold as a tunable parameter rather than a fixed 0.5. The most important limitation of this work is that it uses only the eight pre-computed statistical features from HTRU_2 and is therefore a candidate-screening study rather than a pulsar-detection pipeline.

Acknowledgment

Thank you for the guidance of Maria Stamatopoulou from the University College of London (UCL) in developing this research paper.

References

1 R. N. Manchester et al. The Australia Telescope National Facility pulsar catalogue. Astronomical Journal. Vol. 129, pg. 1993-2006, 2005, https://doi.org/10.1086/428488.

2 D. R. Lorimer, M. Kramer. Handbook of pulsar astronomy. Cambridge University Press, 2004.

3 J. M. Lattimer, M. Prakash. Neutron star observations: prognosis for equation of state constraints. Physics Reports. Vol. 442, pg. 109-165, 2007, https://doi.org/10.1016/j.physrep.2007.02.003.

4 M. Kramer et al. Tests of general relativity from timing the double pulsar. Science. Vol. 314, pg. 97-102, 2006, https://doi.org/10.1126/science.1132305.

5 G. Agazie et al. (NANOGrav Collaboration). The NANOGrav 15-year data set: evidence for a gravitational-wave background. Astrophysical Journal Letters. Vol. 951, pg. L8, 2023, https://doi.org/10.3847/2041-8213/acdac6.

6 J. Antoniadis, et al. (European Pulsar Timing Array). The second data release from the European Pulsar Timing Array III. Search for gravitational wave signals. Astronomy & Astrophysics. Vol. 678, pg. A50, 2023, https://doi.org/10.1051/0004-6361/202346844.

7 M. J. Keith et al. The High Time Resolution Universe Pulsar Survey - I. System configuration and initial discoveries. MNRAS. Vol. 409, pg. 619-627, 2010, https://doi.org/10.1111/j.1365-2966.2010.17325.x.

8 S. D. Bates et al. The High Time Resolution Universe Pulsar Survey - VI. An artificial neural network and timing of 75 pulsars. MNRAS. Vol. 427, pg. 1052-1065, 2012, https://doi.org/10.1111/j.1365-2966.2012.22042.x.

9 M. A. McLaughlin et al. Transient radio bursts from rotating neutron stars. Nature. Vol. 439, pg. 817-820, 2006, https://doi.org/10.1038/nature04440.

10 D. R. Lorimer et al. A bright millisecond radio burst of extragalactic origin. Science. Vol. 318, pg. 777-780, 2007, https://doi.org/10.1126/science.1147532.

11 V. Morello et al. SPINN: a straightforward machine learning solution to the pulsar candidate selection problem. MNRAS. Vol. 443, pg. 1651-1662, 2014, https://doi.org/10.1093/mnras/stu1188.

12 W. W. Zhu et al. Searching for pulsars using image pattern recognition. Astrophysical Journal. Vol. 781, pg. 117, 2014, https://doi.org/10.1088/0004-637X/781/2/117.

13 K. J. Lee et al. PEACE: pulsar evaluation algorithm for candidate extraction - a software package for post-analysis processing of pulsar survey candidates. MNRAS. Vol. 433, pg. 688-694, 2013, https://doi.org/10.1093/mnras/stt758.

14 R. J. Lyon et al. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. MNRAS. Vol. 459, pg. 1104-1123, 2016, https://doi.org/10.1093/mnras/stw656.

15 E. M. Mohamed. An efficient pulsar classification model using convolutional neural networks. International Journal of Advanced Computer Science and Applications. Vol. 11, pg. 497-505, 2020, https://doi.org/10.14569/IJACSA.2020.0111160.

16 H. Wang et al. Pulsar candidate identification using advanced transformer-based classifiers. Universe. Vol. 9, pg. 328, 2023, https://doi.org/10.3390/universe9070328.

17 S. Bethapudi, S. Desai. Separation of pulsar signals from noise using supervised machine learning algorithms. Astronomy and Computing. Vol. 23, pg. 15-26, 2018, https://doi.org/10.1016/j.ascom.2018.02.002.

18 H. Wang et al. Pulsar candidate sifting using multi-input convolutional neural networks. Astrophysical Journal. Vol. 887, pg. 149, 2019, https://doi.org/10.3847/1538-4357/ab55ec.

19 L. Breiman. Random forests. Machine Learning. Vol. 45, pg. 5-32, 2001, https://doi.org/10.1023/A:1010933404324.

20 J. H. Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics. Vol. 29, pg. 1189-1232, 2001, https://doi.org/10.1214/aos/1013203451.

21 F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review. Vol. 65, pg. 386-408, 1958, https://doi.org/10.1037/h0042519.

22 C. Cortes, V. Vapnik. Support-vector networks. Machine Learning. Vol. 20, pg. 273-297, 1995, https://doi.org/10.1007/BF00994018.

23 R. P. Eatough et al. Selection of radio pulsar candidates using artificial neural networks. MNRAS. Vol. 407, pg. 2443-2450, 2010, https://doi.org/10.1111/j.1365-2966.2010.17082.x.

24 R. J. Lyon. Why are pulsars hard to find? PhD thesis, University of Manchester, 2016, https://research.manchester.ac.uk/en/studentTheses/why-are-pulsars-hard-to-find.

25 P. Devine, J. Banerjee, M. McCarthy. Machine learning classification of pulsar candidates from the HTRU-S Low Latitude survey. Publications of the Astronomical Society of Australia. Vol. 36, pg. e045, 2019, https://doi.org/10.1017/pasa.2019.36.

26 D. Wang et al. Pulsar candidate selection using ensemble networks for the FAST drift-scan survey. Research in Astronomy and Astrophysics. Vol. 22, pg. 105003, 2022, https://doi.org/10.1088/1674-4527/ac822c.

27 Y. Guo et al. A pulsar candidate selection method based on the cascade-correlation neural network. Astronomy and Computing. Vol. 33, pg. 100412, 2020, https://doi.org/10.1016/j.ascom.2020.100412.

28 R. P. Lin et al. Pulsar candidate classification using image-based attention networks. Monthly Notices of the Royal Astronomical Society. Vol. 519, pg. 2517-2528, 2023, https://doi.org/10.1093/mnras/stac3697.

29 D. Dua, C. Graff. UCI Machine Learning Repository: HTRU2 Data Set. University of California, Irvine, School of Information and Computer Sciences, 2017, https://archive.ics.uci.edu/ml/datasets/HTRU2.

30 N. V. Chawla et al. SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research. Vol. 16, pg. 321-357, 2002, https://doi.org/10.1613/jair.953.

31 F. Pedregosa et al. Scikit-learn: machine learning in Python. Journal of Machine Learning Research. Vol. 12, pg. 2825-2830, 2011, https://www.jmlr.org/papers/v12/pedregosa11a.html.

32 C. Drummond, R. C. Holte. Cost curves: an improved method for visualizing classifier performance. Machine Learning. Vol. 65, pg. 95-130, 2006, https://doi.org/10.1007/s10994-006-8199-5.

33 R.-E. Fan et al. LIBLINEAR: a library for large linear classification. Journal of Machine Learning Research. Vol. 9, pg. 1871-1874, 2008, https://www.jmlr.org/papers/v9/fan08a.html.

34 J. Bergstra, Y. Bengio. Random search for hyper-parameter optimization. Journal of Machine Learning Research. Vol. 13, pg. 281-305, 2012, https://www.jmlr.org/papers/v13/bergstra12a.html.

35 H. He, E. A. Garcia. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering. Vol. 21, pg. 1263-1284, 2009, https://doi.org/10.1109/TKDE.2008.239.