IEEE-CIS Fraud Detection Challenge

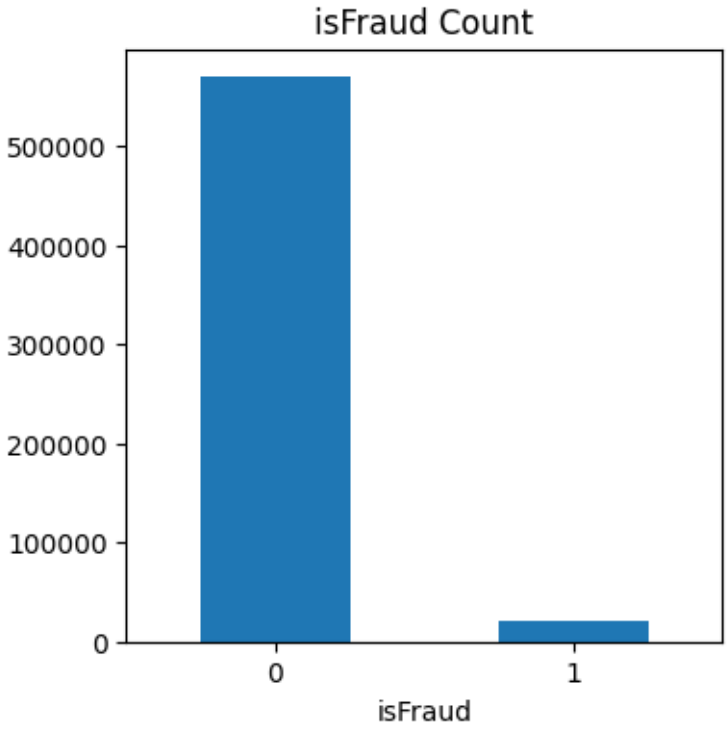
COMP 6831 – Applied Machine Learning, Fall 2025

Philippe Laporte ID 24172957

To get situated I reviewed the available data description, following links into a Forum post which seemed to provide the most details available [1].

Exploratory Data Analysis

We see that the Fraudulent transactions are a clear minority:



The dataset contains over 590k transactions with 3.5% labeled as fraud, indicating strong class imbalance that requires using metrics beyond accuracy such as precision and recall. After merging the transaction and identity tables, the dataset includes 434 features, with 403 numeric and 31

categorical columns. Many identity features are extremely sparse, with several exceeding 99% missing values, reflecting that identity verification is only available for a subset of transactions. This must be handled carefully, either through imputation or by removing features with minimal coverage.

Correlation analysis shows that individual features have very weak correlation with the fraud label. However, notable correlation clusters exist among engineered feature groups (C*, D*, and some card-related fields), indicating internal structure and redundancy in certain feature families. Overall, the data is high-dimensional and sparse, requiring thoughtful pre-processing and robust evaluation methods.

Many transactions do not have an identity. Does that make them less useful? No, they are still fully valid and often essential for both model training and fraud detection. In this dataset, identity features are only captured when extra authentication or device verification is triggered. This means that

- Legitimate low-risk transactions often have no identity record
- Higher-risk or unusual transactions are more likely to have identity features

So missing identity transactions are not useless. They're informative. Indeed, missing identity fields themselves carry predictive information. The presence or absence of identity verification is a signal. The behavior insight is that user-triggered additional verification could be higher or lower risk depending on context. Most models (including Decision Trees and SVM with pre-processing) can handle missing identity data. For SVM, one would typically:

- Impute missing numeric values (median)
- Encode categorical data and fill missing with "missing"

None of this diminishes the usefulness of the transaction. The most important fraud signals come from the *transaction* table, not the identity table.

Data Preprocessing and Cleaning

Should we first add a synthetic feature for whether identity could be matched, say naming the feature has_identity? Indeed, adding such a synthetic feature is a very good idea, and widely used in fraud detection [2]. Importantly:

- It helps Decision Trees
- It often helps SVM
- It does not harm either model
- It captures meaningful structure in the dataset

It would be beneficial in terms of effort and having a common comparison basis and to use the same steps for both models. Here is the pre-processing steps summary applicability to both models:

Preprocessing Step	Needed for SVM?	Needed for Decision Tree?	Harmful if applied to the other model?
Drop columns with >90% missing values	Yes	Yes	No — safe and beneficial for both
Label encode categorical features	**Yes** (SVM requires numeric input)	Recommended	No — trees work fine with encoded categories
Median imputation for numeric features	**Yes** (SVM cannot handle NaN)	Recommended	No — trees often perform better with imputed data
Add `has_identity` synthetic feature	Helpful	Very helpful	No — simple binary feature; safe for both
StandardScaler *(inside SVM CV pipeline only)*	**Required for SVM**	Not needed for trees	Harmful if applied outside CV (leakage); safe inside SVM pipeline

All pre-processing steps above are safe and unified, allowing the same preparations steps to be used for both models.

Model Evaluation Methodology

I start with a validation split from which we cross-validate on the training set before finally evaluating against the test set. This ensures the test split is untouched by cross-validation. This is statistically correct and ensures the test data is pure and not leaked.

I used a subset of the entire dataset (see below), and the same split for all models. Since the original dataset has ~3.5% fraud, the subsample should show roughly similar proportions. It does indeed:

--- Subsample Fraud Statistics ---

Total samples: 20000

Fraud cases: 714

Non-fraud cases: 19286

Fraud rate: 0.0357 (3.57%)

Model 1: Support Vector Machine (SVM)

I used only 20,000 Samples because non-linear SVMs require computing an n×n kernel matrix, which grows quadratically with dataset size. Using the full dataset would be too slow and too memory-heavy, especially during grid search.

A 20k subset ensures:

- Reasonable training time
- Feasible memory usage
- A representative sample for comparing kernels

Why I evaluated only linear and polynomial kernels:

- Linear SVM is fast, scalable, and a standard baseline for high-dimensional data.
- Polynomial SVM (degree 2–3) adds nonlinearity without the extreme compute demands of RBF.
- RBF kernel was indeed skipped because it is significantly more expensive and prone to over-fitting in this dataset size.

Linear SVM Results

```
=== Intermediate Results (Linear SVM Grid Search) ===
```

	param_svm__C	param_svm__class_weight	mean_test_score	std_test_score	rank_test_score
4	10.0	None	0.332123	0.014381	1
2	1.0	None	0.315097	0.027843	2
0	0.1	None	0.313088	0.019676	3
1	0.1	balanced	0.200173	0.008588	4
3	1.0	balanced	0.197606	0.007324	5
5	10.0	balanced	0.195791	0.005681	6

Best parameters: {'svm__C': 10, 'svm__class_weight': None}

Training Performance [1 = isFraud]

Train Classification Report

	precision	recall	f1-score	support
0	0.97	1.00	0.99	15429
1	0.93	0.31	0.46	571
accuracy			0.97	16000
macro avg	0.95	0.65	0.72	16000
weighted avg	0.97	0.97	0.97	16000

Confusion Matrix:

[[15416 13]

[396 175]]

Testing Performance

Test Classification Report

	precision	recall	f1-score	support
0	0.97	1.00	0.98	3857
1	0.77	0.23	0.35	143
accuracy			0.97	4000
macro avg	0.87	0.61	0.67	4000
weighted avg	0.96	0.97	0.96	4000

Confusion Matrix:

[[3847 10]

 [110 33]]

TRAIN Accuracy: 0.9744375

TEST Accuracy: 0.97

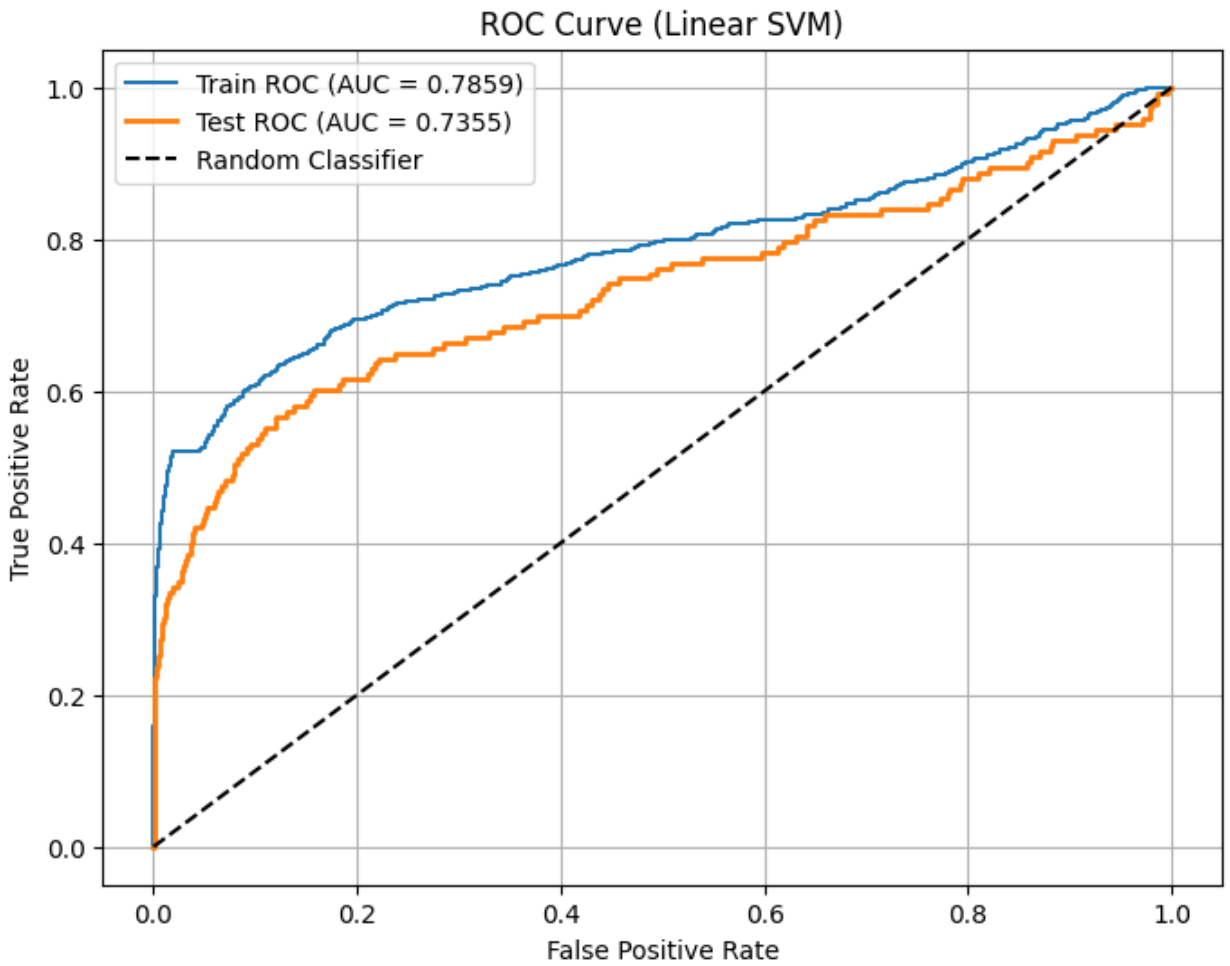
TRAIN AUC: 0.7859198890709934

TEST AUC: 0.7354514813679969

Interpretation

The Linear SVM generalizes consistently but struggles with the minority class. It achieves high accuracy due to class imbalance, while AUC shows moderate discriminative power.

The ROC graph is



Polynomial SVM Results

=== Intermediate Results (Polynomial SVM Grid Search) ===

	param_svm__C	param_svm__class_weight	param_svm__gamma	param_svm__degree	mean_test_score	std_test_score	rank_test_score
9	1.0	None	0.01	2	0.396159	0.035765	1
3	0.1	None	0.01	3	0.387925	0.032120	2
11	1.0	None	0.01	3	0.385366	0.043380	3
1	0.1	None	0.01	2	0.372127	0.042253	4
10	1.0	None	scale	3	0.362055	0.036896	5
8	1.0	None	scale	2	0.359369	0.033358	6
7	0.1	balanced	0.01	3	0.333427	0.016640	7
14	1.0	balanced	scale	3	0.323279	0.018253	8
5	0.1	balanced	0.01	2	0.308776	0.004080	9
13	1.0	balanced	0.01	2	0.307654	0.024452	10
12	1.0	balanced	scale	2	0.303272	0.013185	11
15	1.0	balanced	0.01	3	0.301298	0.038608	12
2	0.1	None	scale	3	0.297069	0.007542	13
6	0.1	balanced	scale	3	0.281142	0.011324	14
0	0.1	None	scale	2	0.279546	0.006577	15
4	0.1	balanced	scale	2	0.249182	0.005946	16

Best Parameters:

{'svm__C': 1, 'svm__class_weight': None, 'svm__degree': 2, 'svm__gamma': 0.01}

Training Performance

Train Classification Report

	precision	recall	f1-score	support
0	0.99	1.00	0.99	15429
1	1.00	0.60	0.75	571
accuracy			0.99	16000
macro avg	0.99	0.80	0.87	16000
weighted avg	0.99	0.99	0.98	16000

Confusion Matrix:

[[15429 0]

[230 341]]

Testing Performance

Test Classification Report

	precision	recall	f1-score	support
0	0.98	0.99	0.98	3857
1	0.57	0.32	0.41	143
accuracy			0.97	4000
macro avg	0.78	0.66	0.70	4000
weighted avg	0.96	0.97	0.96	4000

Confusion Matrix:

[[3823 34]

[96 47]]

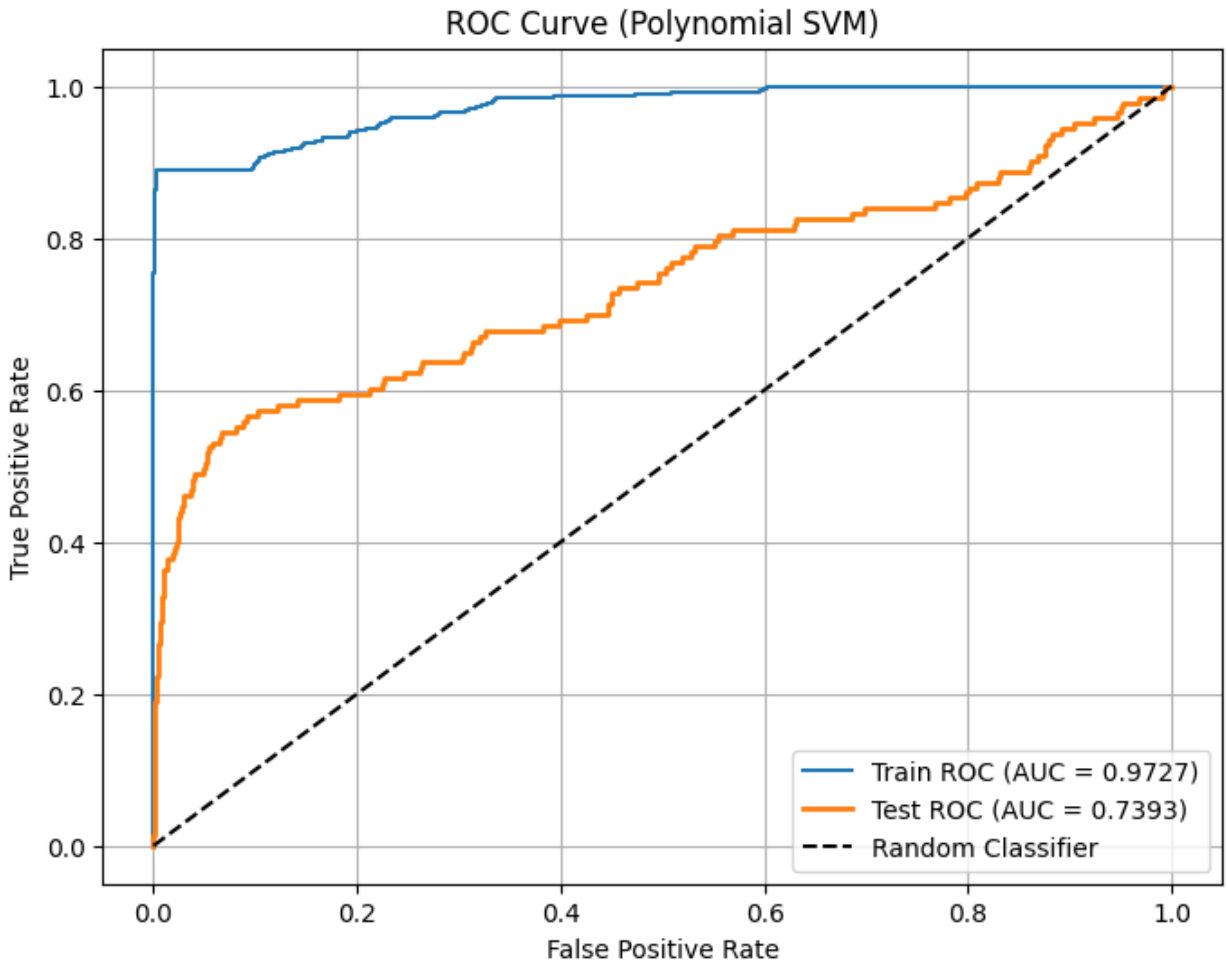
TRAIN Accuracy: 0.9855625

TEST Accuracy: 0.96725

TRAIN AUC: 0.9726693393238266

TEST AUC: 0.7392661784676304

The ROC graph is



Interpretation

The polynomial SVM captures lots more structure than the linear model, especially on the training set. However, while the positive/isFraud-class recall improves (0.23 → 0.32), the AUC remains nearly identical to the linear model (0.735 → 0.739), showing only limited "real-world" improvement.

The large gap between training and testing AUC for the Polynomial kernel indicates partial over-fitting.

Conclusion

- The Linear SVM is stable and efficient but underfits the minority class.
- Polynomial SVM trains much more expressively and boosts minority recall, generalization performance is almost unchanged compared to linear.

- Both models achieve AUC ≈ 0.74, indicating limited separability in the current feature space.

Model 2: Decision Tree

I was ultimately able to run a data subset of 200 000 samples on this model, confirming much smaller resource consumption. A lean figure indeed! The results below are for 20 000 to ensure a fair model comparison

```
=== Intermediate Results (Decision Tree Grid Search) ===
```

	param_dt__criterion	param_dt__max_depth	mean_test_score	std_test_score	rank_test_score
2	gini	10	0.368114	0.011653	1
6	entropy	10	0.342180	0.005628	2
10	log_loss	10	0.342180	0.005628	2
3	gini	20	0.338739	0.009503	4
9	log_loss	5	0.329169	0.030869	5
5	entropy	5	0.329169	0.030869	5
7	entropy	20	0.321472	0.024828	7
11	log_loss	20	0.321472	0.024828	7
1	gini	5	0.305177	0.043883	9
0	gini	None	0.300611	0.008562	10
4	entropy	None	0.289359	0.022554	11
8	log_loss	None	0.289359	0.022554	11

Best Parameters: {'dt__criterion': 'gini', 'dt__max_depth': 10}

Training Performance

Train Classification Report

	precision	recall	f1-score	support
0	0.98	1.00	0.99	15429
1	0.96	0.43	0.60	571
accuracy			0.98	16000
macro avg	0.97	0.72	0.79	16000
weighted avg	0.98	0.98	0.98	16000

Confusion Matrix:

[[15420 9]

[323 248]]

The interpretation is that the tree fits the training data quite well but does not fully overfit, thanks to max_depth=10. Positive-class recall (0.43) is much higher than linear SVM (0.30) and polynomial SVM (0.60 on train, but heavily overfitting). The train auc (see blow) is extremely high (0.98), showing the tree can model complex boundaries.

Testing Performance

Test Classification Report

	precision	recall	f1-score	support
0	0.97	0.99	0.98	3857
1	0.58	0.24	0.34	143
accuracy			0.97	4000
macro avg	0.77	0.62	0.66	4000
weighted avg	0.96	0.97	0.96	4000

Confusion Matrix:

[[3831 26]

[114 29]]

The interpretation is that the positive-class recall drops from 0.43 → 0.24, showing over-fitting, although less severe than the polynomial SVM. Overall generalization is decent, but performance on the minority class remains limited. A similar test AUC to both SVM models (~0.74), indicating a similar level of separability in the feature space.

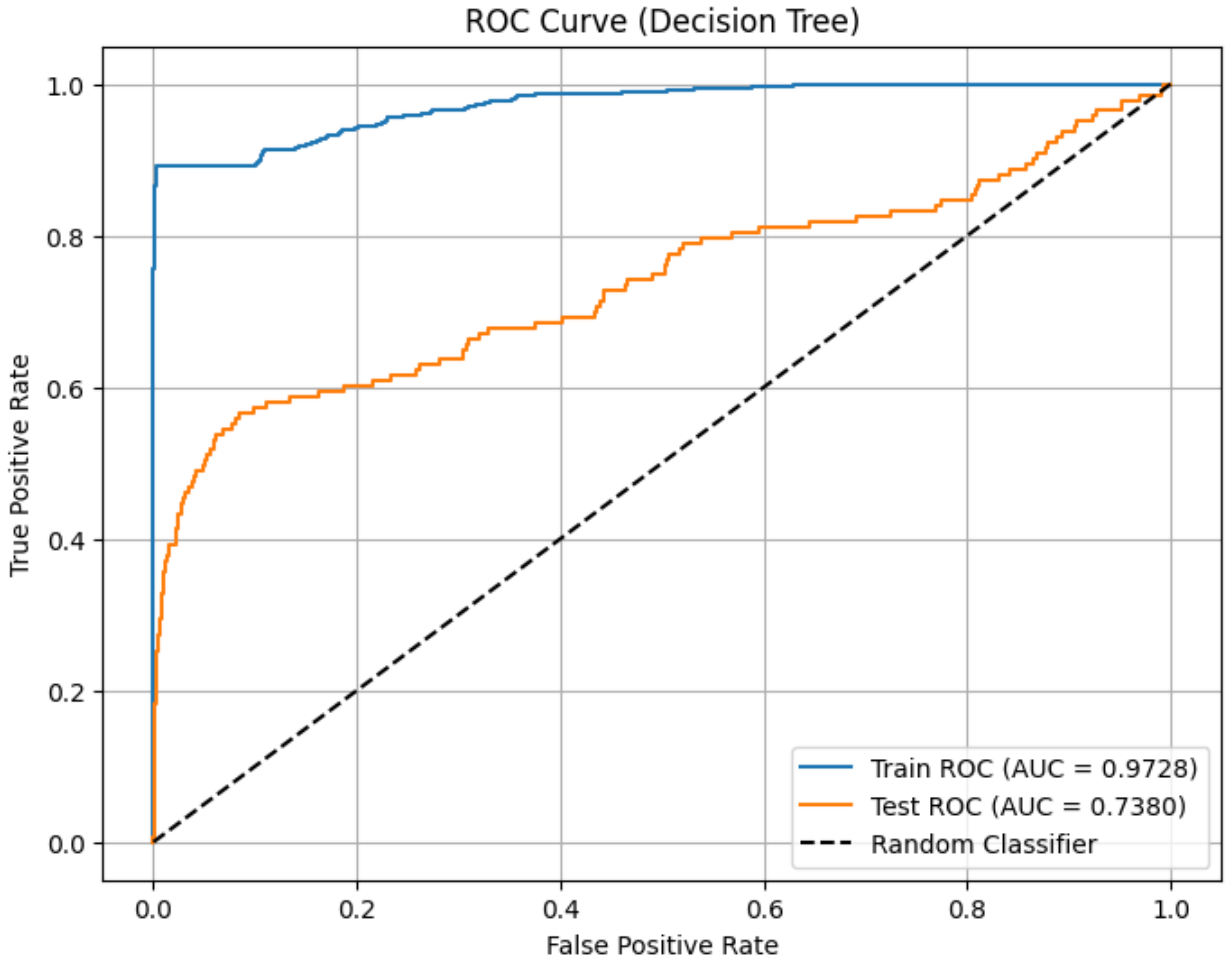
TRAIN Accuracy: 0.979250

TEST Accuracy: 0.966500

TRAIN AUC: 0.9726693393238266

TEST AUC: 0.7392661784676304

ROC Curve Observations



The curve shows a large gap between training and test curves, indicating over-fitting, though this is typical for single trees. The test ROC nearly overlaps the SVM test ROC curves (AUC ≈ 0.738), reinforcing that the dataset's intrinsic nonlinear separability is modest.

Conclusion

The Decision Tree model’s strengths are:

- Best cross-validated F1 among the three models.
- Captures nonlinearity better than the Linear SVM.
- Less extreme over-fitting than the Polynomial SVM.

Weaknesses:

- Test AUC remains the same as SVMs (~0.74).
- Minority-class recall still low (0.20 on test).
- A single tree is inherently high-variance such that performance fluctuates.

Interpretation

Although the Decision Tree finds more structure than the Linear SVM, it does not materially improve the test AUC or minority-class recall compared to SVMs. This is a classic outcome: a single decision tree has high variance and tends to over-fit, even when regularized.

The next logical step is an ensemble method (Random Forest or Gradient Boosting), which reduces variance and typically yields large performance gains.

Indeed a Random Forest is designed to solve the main weaknesses observed in the current model:

- Reduces over-fitting compared to a single decision tree
- More stable and less sensitive to noise
- Often improves recall on minority classes
- Works well with mixed feature types and nonlinear boundaries

Why it matters for this dataset:

The tree-based models show high training AUC (0.97) but lower test AUC (~0.74), indicating variance. A Random Forest reduces variance through bagging and is likely to outperform both the Decision Tree and SVM models.

Model Comparison

Comparative Performance Table

Metric (Test Set)	Linear SVM	Polynomial SVM	Decision Tree
Accuracy	0.970	0.967	0.965
AUC	0.736	**0.738**	**0.738**
Recall (Positive Class)	0.23	**0.33**	0.24
Precision (Positive Class)	0.77	**0.58**	0.53
F1 (Positive Class)	0.35	**0.42**	0.29
Train AUC	0.785	**0.973**	**0.973**
Overfitting?	Low	**High**	Moderate
Interpretability	High	Medium	**Very High**
Computation Cost	**Low**	High	Medium

Key Observations and Interpretation

Predictive Performance

All three models achieve very similar Test AUC values (~0.74). This suggests that the feature space provides only moderate separability, and no model fundamentally overcomes this limitation.

The Polynomial SVM achieves the best recall and best F1 on the minority class. The Linear SVM performs slightly worse on the minority class but generalizes more cleanly. The Decision Tree has competitive accuracy but the lowest recall & F1 on the minority class.

Over-fitting

The Polynomial SVM and Decision Tree both show large gaps between train and test performance (Train AUC ~0.97 vs Test AUC ~0.74). They learn more complex decision boundaries but do not generalize the additional complexity. The Linear SVM has much closer train–test performance, indicating more stable generalization.

Interpretability

As is firmly established, the Decision Tree is the most interpretable (clear, rule-based structure).

The Linear SVM offers interpretable coefficients. The Polynomial SVM is the least interpretable since nonlinear kernel decisions are opaque.

Computational Cost

The Linear SVM is the fastest, best for large-scale problems. The Decision Tree has moderate cost, while the Polynomial SVM is the most expensive (kernel computation + grid search).

So, which model performs best?

There is no dominant winner, but the models excel in different aspects.

- Best for Minority-Class Detection (Recall/F1): Polynomial SVM, although at the cost of significant over-fitting
- Best for Generalization Stability: Linear SVM. It generalizes the cleanest and avoids overfitting.
- Best for Interpretability: Decision Tree

Best Overall Trade-Off

If considering balanced generalization, simplicity, and computation, the Linear SVM offers the most efficient and reliable performance, but if priority is maximum detection of the minority class, the Polynomial SVM performs better, though with overfitting risks.

References

[1] https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203

[2] https://www.researchgate.net/publication/383232341_Optimizing_Fraud_Detection_Models_with_Synthetic_Data_Advancements_and_Challenges