Boosting Algorithms for Credit Card Fraud Detection Across Varied Datasets

Justs Viduss

Transport and Telecommunication Institute

Why This Research Matters

- **Addressing a Major Challenge**: This research directly addresses the critical need for more effective fraud detection systems in the banking sector, particularly in handling the complexity and volume of modern financial transactions.
- **Addressing Real-World Data Challenges**: Financial fraud detection faces the challenge of dealing with highly unbalanced datasets where fraudulent transactions are much less frequent than legitimate ones. The research explores the effectiveness of these algorithms in such settings, offering potential solutions to one of the biggest obstacles in fraud detection
- **Improvement Over Existing Methods**: By conducting a thorough comparison of boosting algorithms against traditional methods, this research fills a gap in the existing literature that often lacks detailed comparative analysis.

Objectives

- Review existing research
- Compare boosting and traditional algorithms on three distinct datasets
- Investigate impact of dataset size on model performance
- Evaluate the computational requirements for each algorithm
- Analyze results, make conclusions and recommendations

Outline

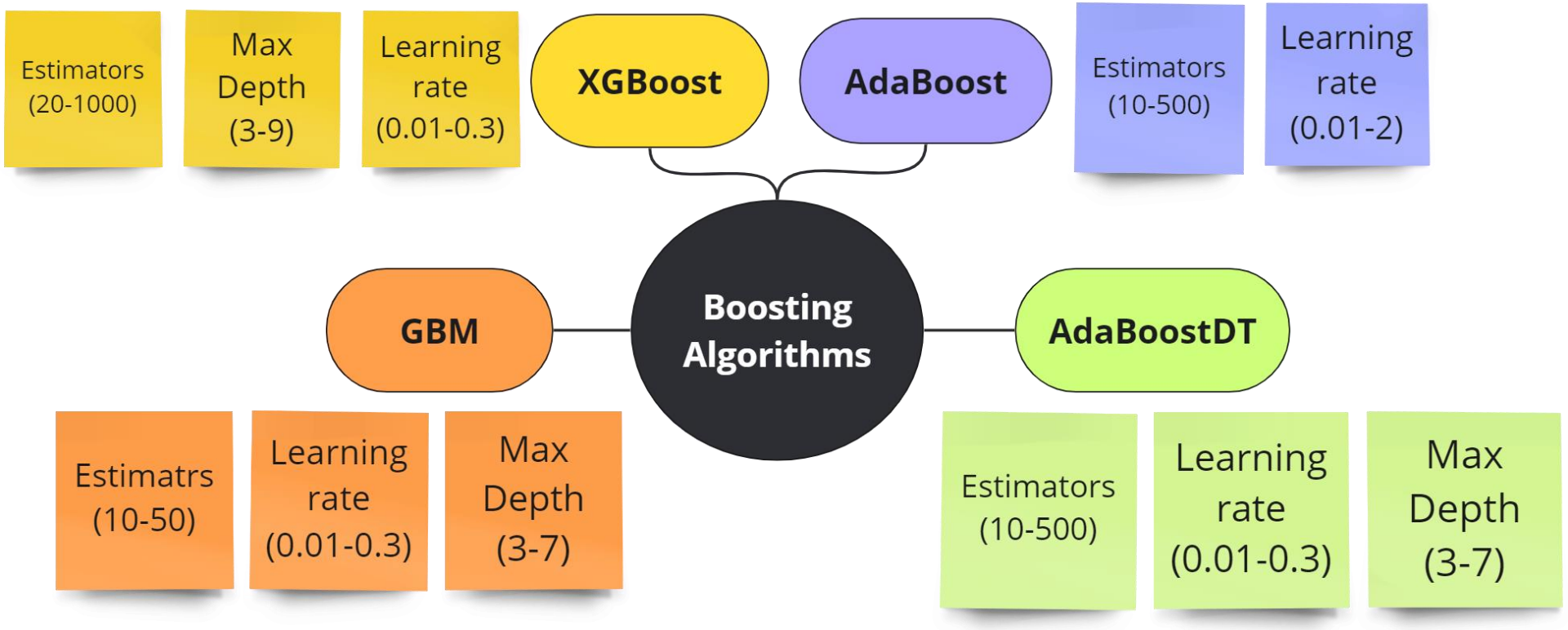
- Methodology
- Algorithms tested
- Datasets
- Implementation
- Algorithm performance analysis
- Conclusions

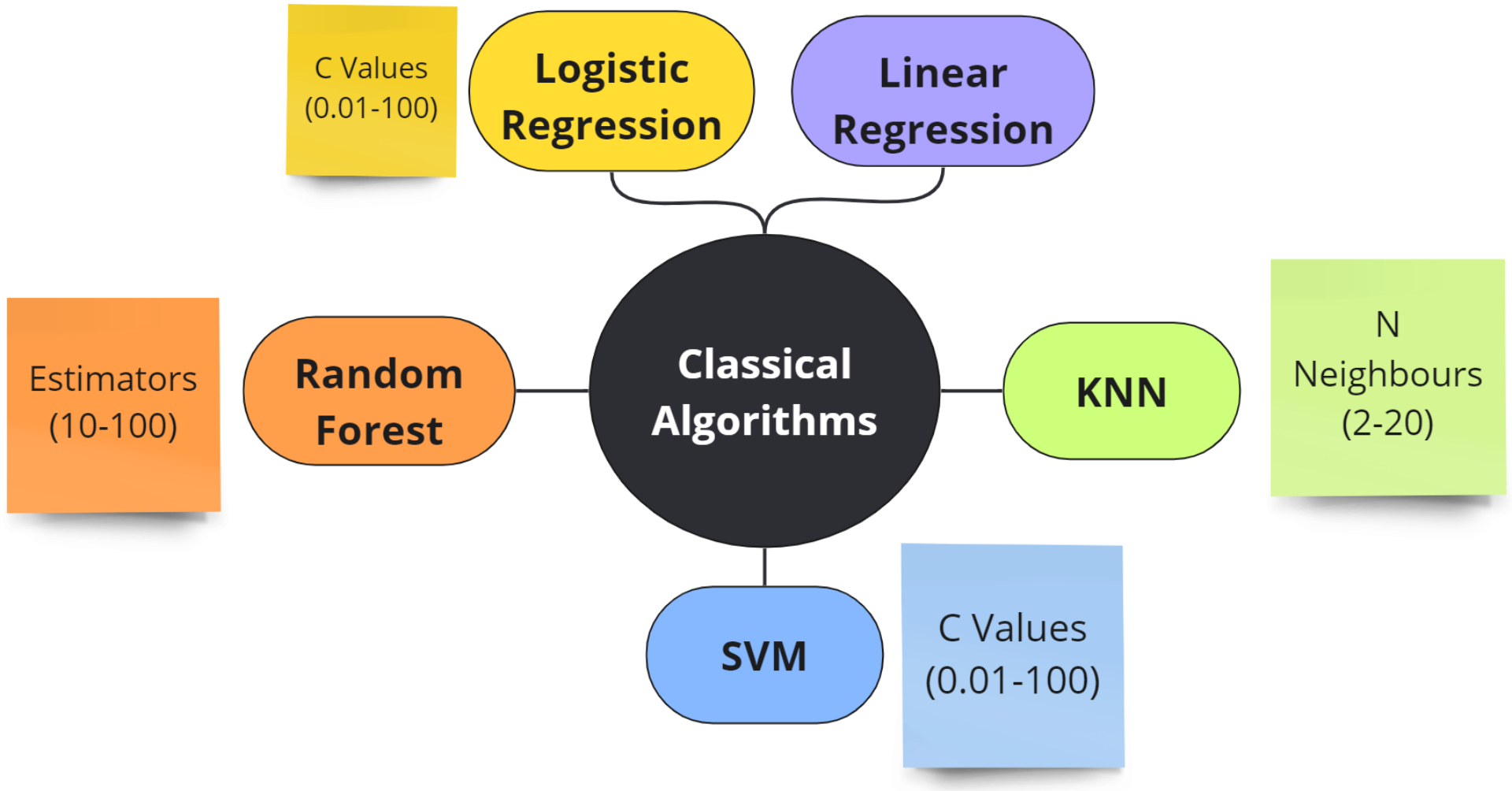
Methodology

- **Materials and Procedures**
  - **Tools:** Python Jupyter Notebooks for algorithm testing, data cleaning, and preparation.
  - **Datasets:** CSV files obtained from Kaggle and other sources, preprocessed and stored in the same format.
  - **Software:** Python for model training, data processing, and result aggregation.
  - **Storage:** All project artifacts, including code, datasets, and results, were saved and made accessible via GitHub.

- **Analysis**
  - **Model Training:** Various algorithms, including boosting algorithms (XGBoost, AdaBoost, Gradient Boosting Machine) and traditional methods (Random Forest, KNN, SVM), were trained and evaluated.
  - **Result Aggregation:** Training results were automatically saved in Excel files using Python. Pivot tables were created to aggregate results and visualize the performance of each model variant.
  - **Performance Metrics:** Key metrics such as Accuracy, F1 Score, Recall, Precision, and Time Taken were recorded for each iteration of parameters.
  - **Documentation:** All results and analyses were documented and uploaded to GitHub for transparency and public access.

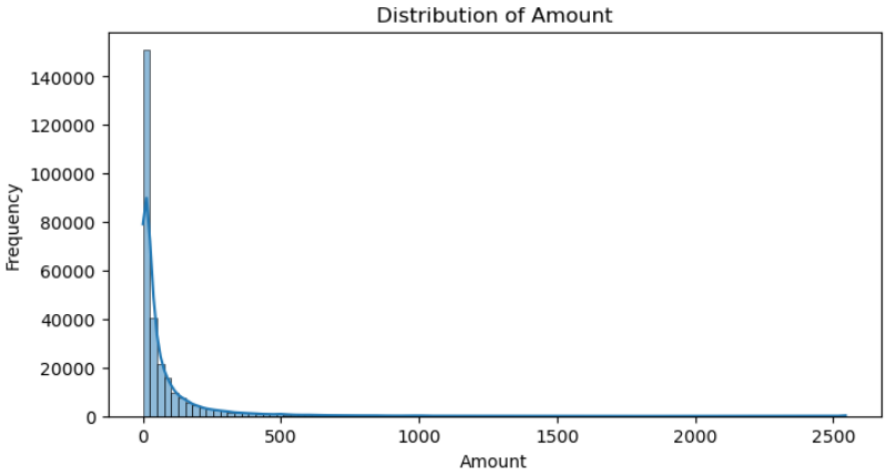
Algorithms tested

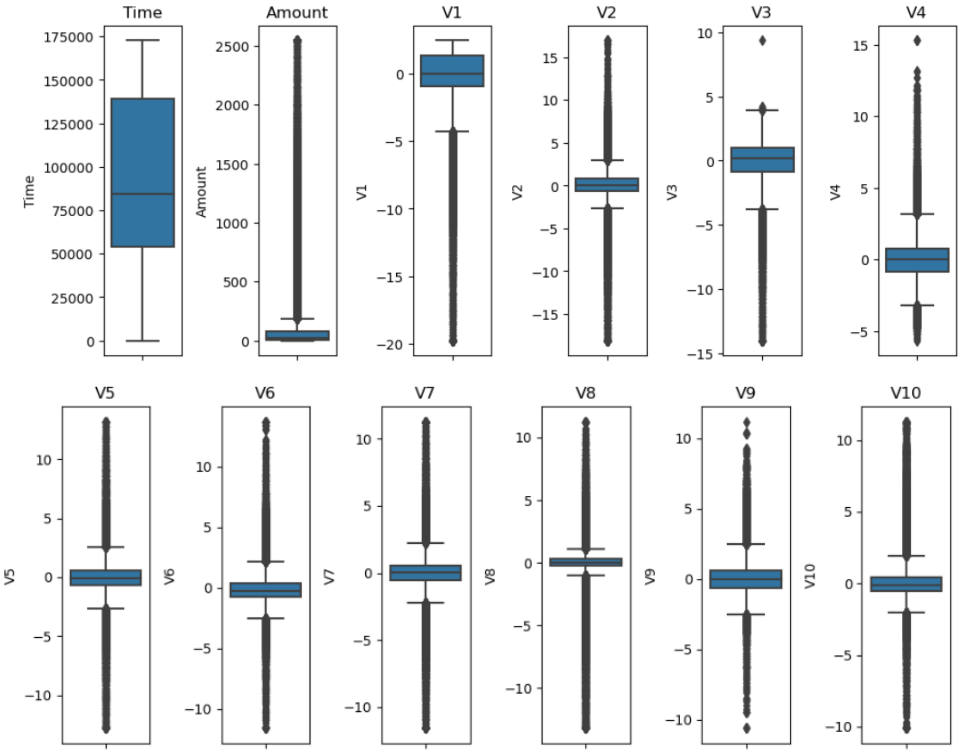



C Values
(0.01-100)
Logistic Regression
Linear Regression
Estimators
(10-100)
Random Forest
Classical Algorithms
KNN
N
Neighbours
(2-20)
SVM
C Values
(0.01-100)

Datasets

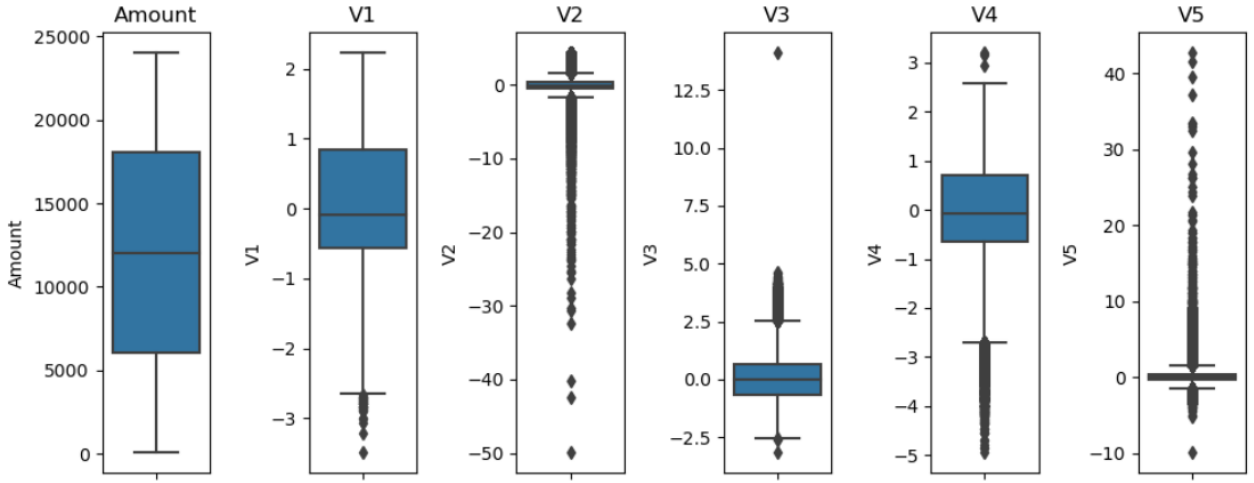
- **Unballanced** dataset: 284k rows, 30 features, 492 fraud cases (0.17%). All features exept time and transaction amount are anonymized using PCA. Source: Kaggle (2018) *transactions made by European cardholders in September 2013*

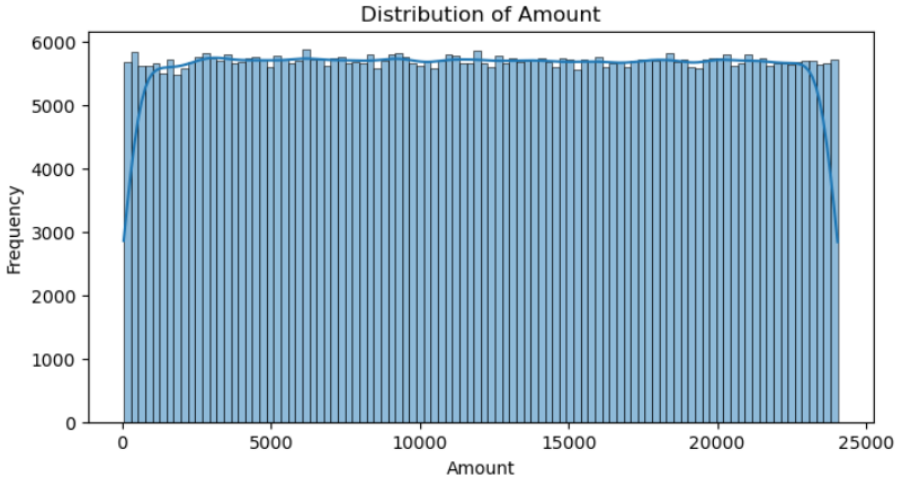


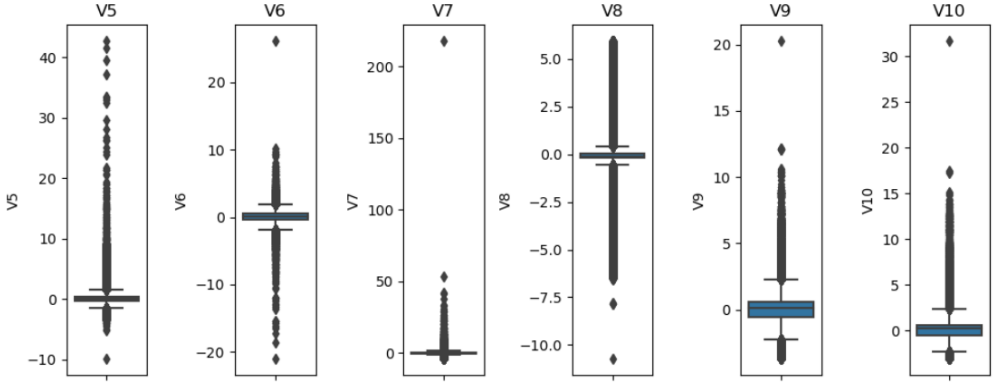


- **Ballanced** dataset: 568k rows, 29 features, 280k fraud cases (50%).

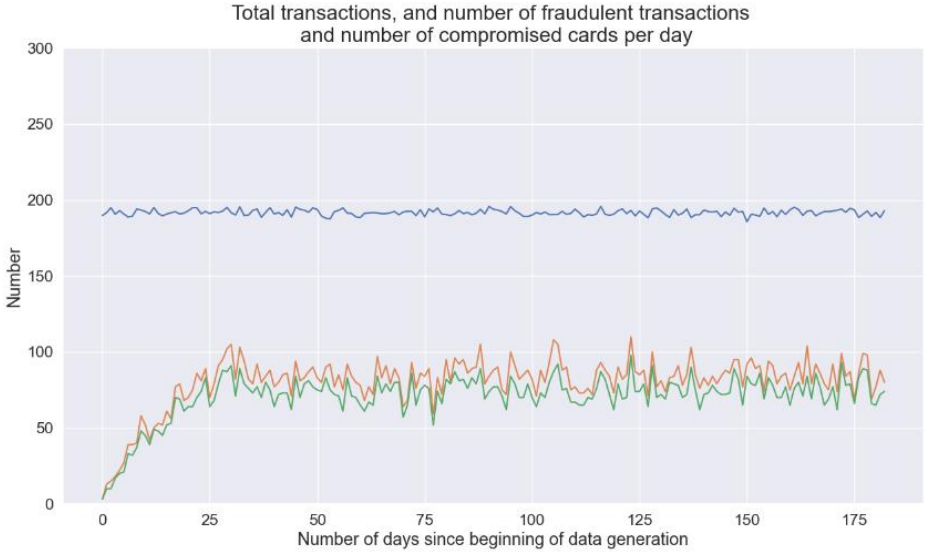
Source: Kaggle (2023). Transactions made by European cardholders in 2023

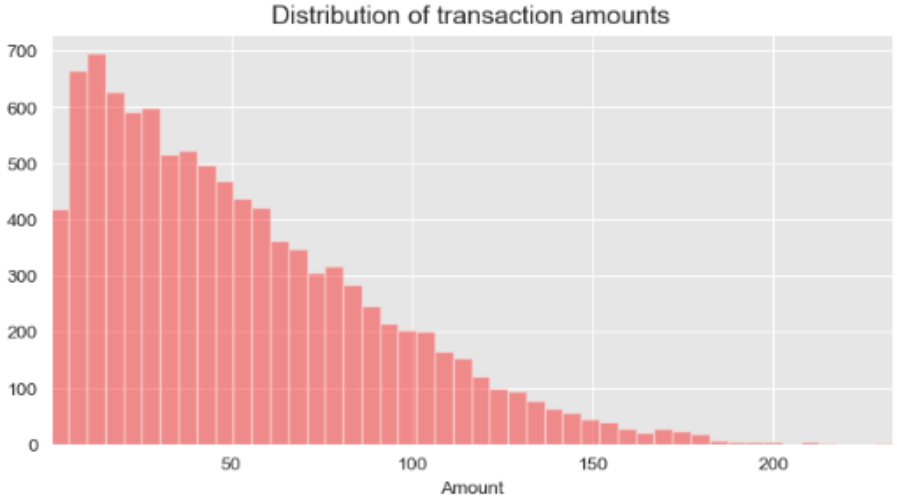


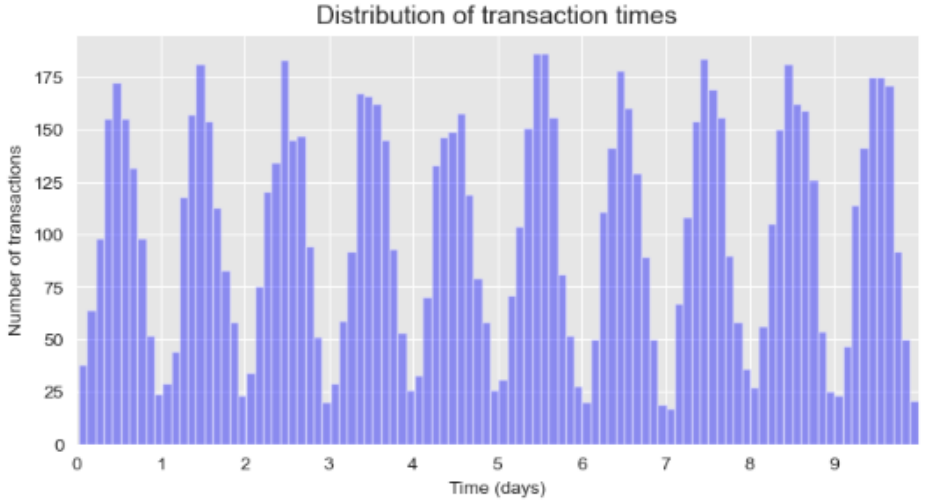




- **Synthetic** data: Generated with python script, 1.7m rows, 5 features, 14k fraud cases (0.84%)





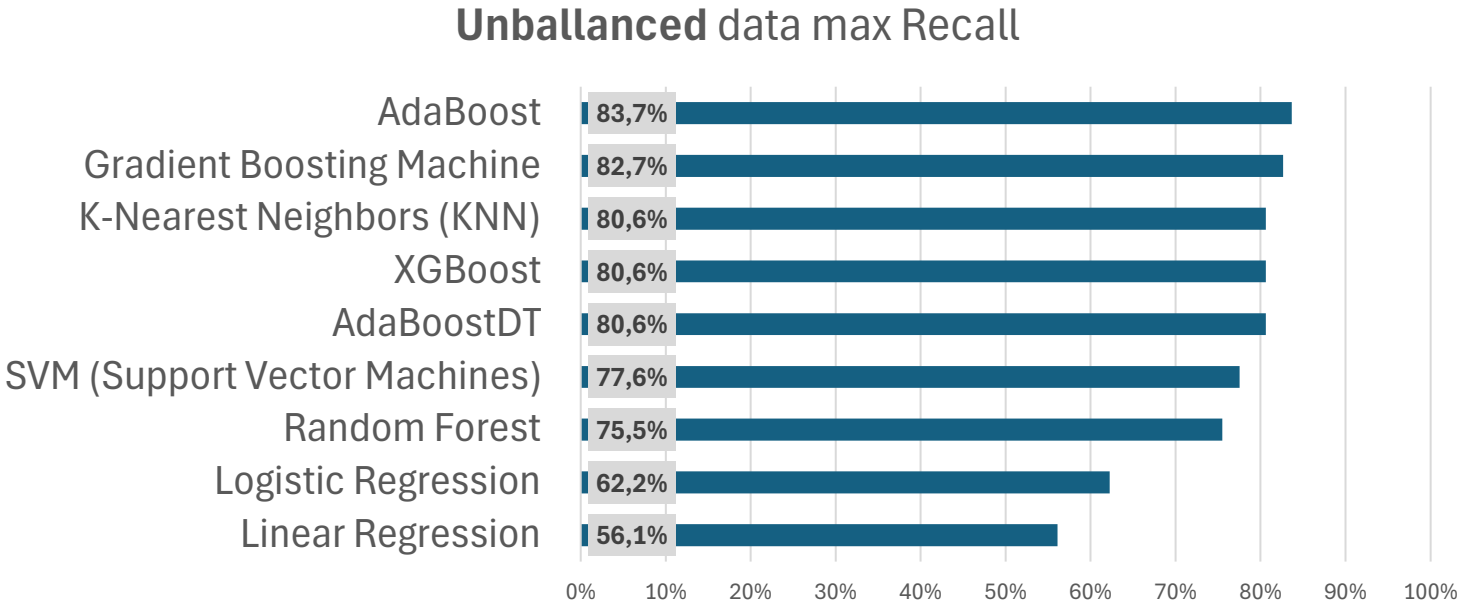


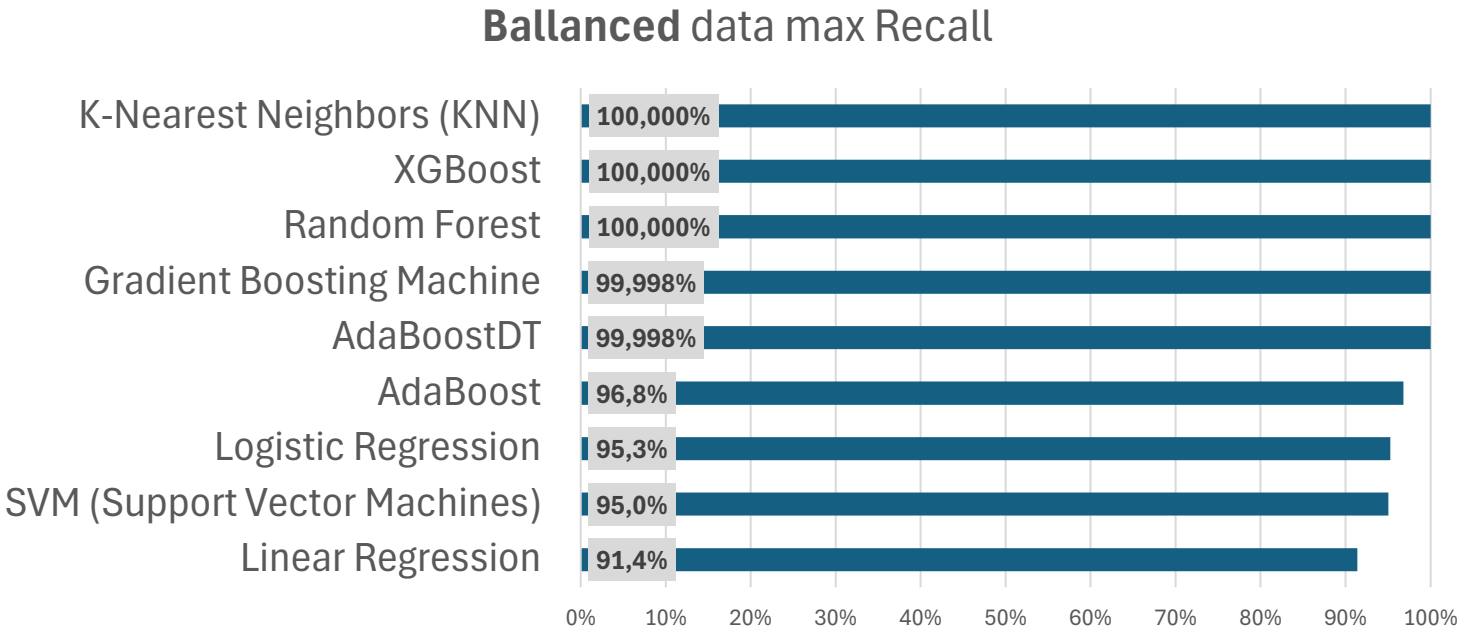
Algorithm performance analysis

Why each metric was chosen for tests:

- **Recall** when it is critical to catch as many frauds as possible (minimizing false negatives)
- **Precision** when it is crucial to be as accurate as possible in your fraud predictions (minimizing false positives)
- **F1 Score** when you need a balance between precision and recall, and both types of errors are similarly costly
- **Accuracy** only when the classes are somewhat balanced or when you want a general idea of the model's performance across all predictions

Recall


Unballanced data max Recall
AdaBoost
Gradient Boosting Machine
K-Nearest Neighbors (KNN)
XGBoost
AdaBoostDT
SVM (Support Vector Machines)
Random Forest
Logistic Regression
Linear Regression
83,7%
82,7%
80,6%
80,6%
80,6%
77,6%
75,5%
62,2%
56,1%
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%


Ballanced data max Recall
K-Nearest Neighbors (KNN)
XGBoost
Random Forest
Gradient Boosting Machine
AdaBoostDT
AdaBoost
Logistic Regression
SVM (Support Vector Machines)
Linear Regression
100,000%
100,000%
100,000%
99,998%
99,998%
96,8%
95,3%
95,0%
91,4%
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%

Accuracy

	Max of Accuracy		
	Unballanced	**Ballanced**	**Synthetic**
XGBoost	99,96%	99,99%	99,26%
AdaBoostDT	99,96%	99,98%	99,50%
Random Forest	99,96%	99,99%	99,38%
K-Nearest Neighbors (KNN)	99,95%	99,93%	99,36%
AdaBoost	99,95%	97,62%	99,37%
Gradient Boosting Machine	99,95%	99,96%	99,60%
SVM (Support Vector Machines)	99,94%	96,39%	99,19%
Logistic Regression	99,92%	96,53%	99,27%
Linear Regression	99,91%	94,84%	99,19%

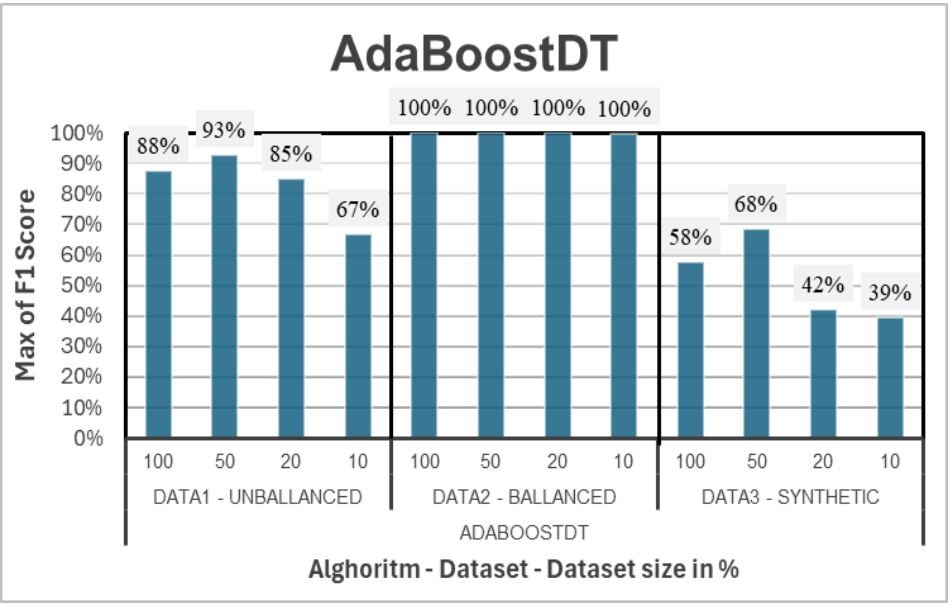
F1 Score

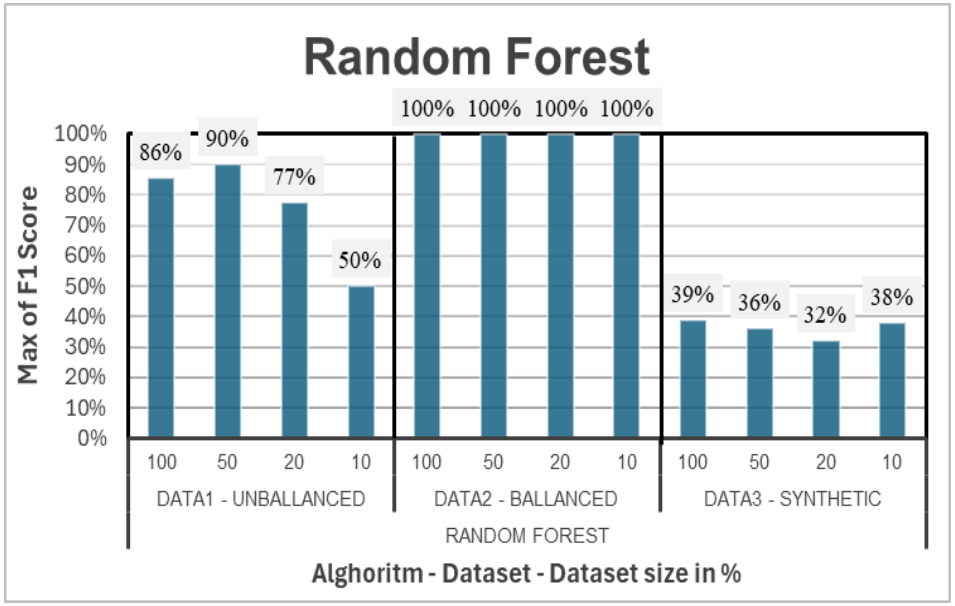
	Max of F1 Score		
	Unballanced	**Ballanced**	**Synthetic**
XGBoost	88,27%	99,99%	26,86%
AdaBoostDT	87,64%	99,98%	57,55%
Random Forest	85,55%	99,99%	38,74%
K-Nearest Neighbors (KNN)	85,39%	99,93%	36,02%
AdaBoost	85,08%	97,61%	36,72%
Gradient Boosting Machine	82,76%	99,96%	72,04%
SVM (Support Vector Machines)	82,16%	96,34%	0,21%
Logistic Regression	72,62%	96,49%	18,47%
Linear Regression	67,48%	94,67%	0,07%

Precision

	Max of Precision		
	Unballanced	**Ballanced**	**Synthetic**
Random Forest	100,0%	100,0%	99,6%
Gradient Boosting Machine	100,0%	99,9%	100,0%
XGBoost	98,7%	100,0%	100,0%
AdaBoostDT	98,7%	100,0%	100,0%
K-Nearest Neighbors (KNN)	95,0%	99,9%	99,2%
AdaBoost	92,8%	98,5%	100,0%
SVM (Support Vector Machines)	89,2%	98,1%	100,0%
Logistic Regression	87,1%	98,2%	100,0%
Linear Regression	84,6%	98,2%	100,0%

Sensitivity to dataset size





Algorithm execution times

	Data1 - Unballanced		Data2 - Ballanced		Data3 - Synthetic	
	Min of Time (s)	Max of Time (s)	Min of Time (s)	Max of Time (s)	Min of Time (s)	Max of Time (s)
AdaBoost	30.9	8401.7	38.4	381.9	24.0	224.1
AdaBoostDT	106.2	1513.7	102.3	1149.7	55.7	601.3
XGBoost	0.8	24.7	2.3	67.1	3.8	86.0
Gradient Boosting Machine	85.2	4479.7	221.4	7014.6	50.1	1825.6
Logistic Regression	2.2	4.5	7.6	9.1	3.5	5.3
Random Forest	22.9	668.1	122.5	1493.5	114.1	1723.5
Linear Regression	0.8	0.8	1.4	1.4	3.2	3.2
K-Nearest Neighbors (KNN)	83.2	119.3	494.4	3476.8	36.2	190.0
SVM (Support Vector Machines)	10.6	785.5	17.3	1135.5	19.2	5998.5

Conclusions

- Review of existing research: boosting algorithms like XGBoost, AdaBoost, and Light Gradient Boosting beat standard methods in a variety of classification situations
- Comparison boosting and traditional algorithms:
  - AdaBoost outscored all others on the unbalanced dataset (83.7% recall)
  - In balanced datasets boosting and traditional algorithms performed outstanding (100% recall)
  - Gradient Boosting Machine performed best on synthetic data

- Sensitivity to Dataset Size:
  - Boosting models performed well on balanced datasets, retaining good accuracy levels across all sizes
  - Traditional models were more sensitive to data size changes
  - All models were extremely sensitive to size drops in imbalanced dataset
- Computational efficiency:
  - In real-world implementations the choice of algorithm may be influenced not just by performance, but also operational constraints

- Analysis of the results, conclusions, and recommendations:
  - No single algorithm performs best in all situations – careful selection based on dataset characteristics and desired outcomes needs to be done
  - Smaller businesses may consider simpler methods like Logistic regression which requires minimal resources
  - Future research may consider feature engineering methodologies, hybrid models or incorporate future machine learning developments

Feedback and Discussion

- The Thesis could benefit from improvement in presenting methodology and definition of research subject
- Thesis did not include seperate section on Research Methodology
- Question: How would the author define traditional methods and what are the criteria for classifying methods as traditional?

Thank you

Justs Vīdušs

viduss.justs@gmail.com