CS 441 - Final Project - Apply ML to Established Benchmark

You only need to complete and submit this once for the group. Be sure to add the other group members to the submission in Gradescope. List them here also.

Group Member Names	**NetIDs**
Vira Kasprova	vkaspr2

Complete the sections below. You do not need to fill out the checklist.

Total Points Available	**[ ] / 150**
1. Problem description	[ ] / 20
2. Model comparison	
a. Which models / hyperparameters	[ ] / 15
b. Train/val experiments	[ ] / 45
c. Best model/parameters result	[ ] / 20
3. Stretch Goal: Innovation	
a. Approach description, experiments, innovation	[ ] / 25
b. Publishable with justification	[ ] / 25
Attribution / Group Contributions	[ ] -5 if incomplete (or page not selected)

1. Problem Description

Give an overview of the problem/benchmark you are trying to solve.

The task is to build a machine learning model to detect fraudulent credit card transactions using the highly imbalanced Credit Card Fraud Detection dataset. The goal is to distinguish fraudulent transactions (positive class) from legitimate ones (negative class) while addressing challenges posed by class imbalance and ensuring high precision and recall, which are critical in fraud detection.

Explain what you are trying to predict, and what inputs (features) are available for prediction.

Inputs (Features): The dataset contains 30 features derived from PCA transformations (V1 to V28), along with Time (elapsed time since the first transaction) and Amount (transaction value). Feature engineering added two interaction features: V14_V4 (multiplicative interaction) and V14_div_V12 (division interaction).

Target: The binary variable Class, where 1 indicates fraud and 0 indicates legitimate transactions.

Explain the experimental setup. How many examples for train, val, test? What are the metrics?

Data Split:
- Training set: 80% of resampled data (after applying SMOTE).
- Validation set: 10%.
- Test set: 10%.

Metrics:
- Precision: Measure of false positives.
- Recall: Measure of false negatives.
- F1-Score: Balance between precision and recall.
- AUC-PR: Area Under the Precision-Recall Curve, suitable for imbalanced datasets.

What are some of the challenges in solving this problem?

Class Imbalance: Fraudulent transactions represent a small fraction of the dataset (~0.17%), making it hard for models to learn patterns for the minority class.

Overfitting Risk: Synthetic data generated by SMOTE might introduce noise, potentially leading to overfitting.

Feature Interpretability: PCA-transformed features (V1 to V28) lack intuitive meaning, making explainability difficult.

Precision-Recall Trade-off: Balancing false positives and false negatives is critical, as both have significant real-world implications.

Scalability: Ensuring the model generalizes well on unseen data despite the imbalance and limited fraud examples.

2. Model Comparison

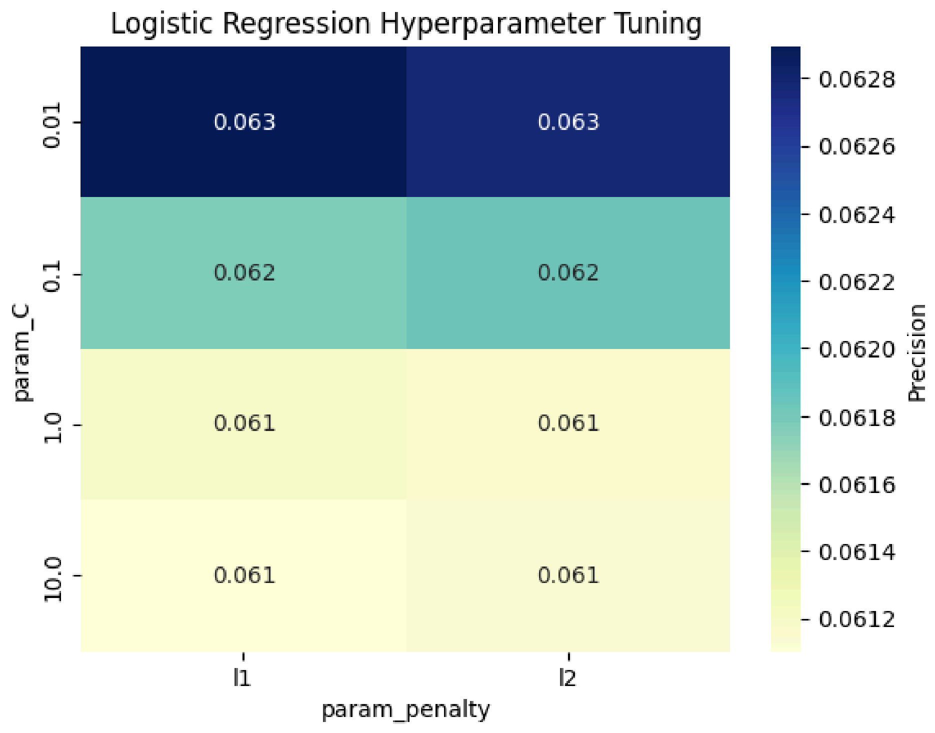
Which three models will you compare? Which hyperparameter(s) will you test for each one?

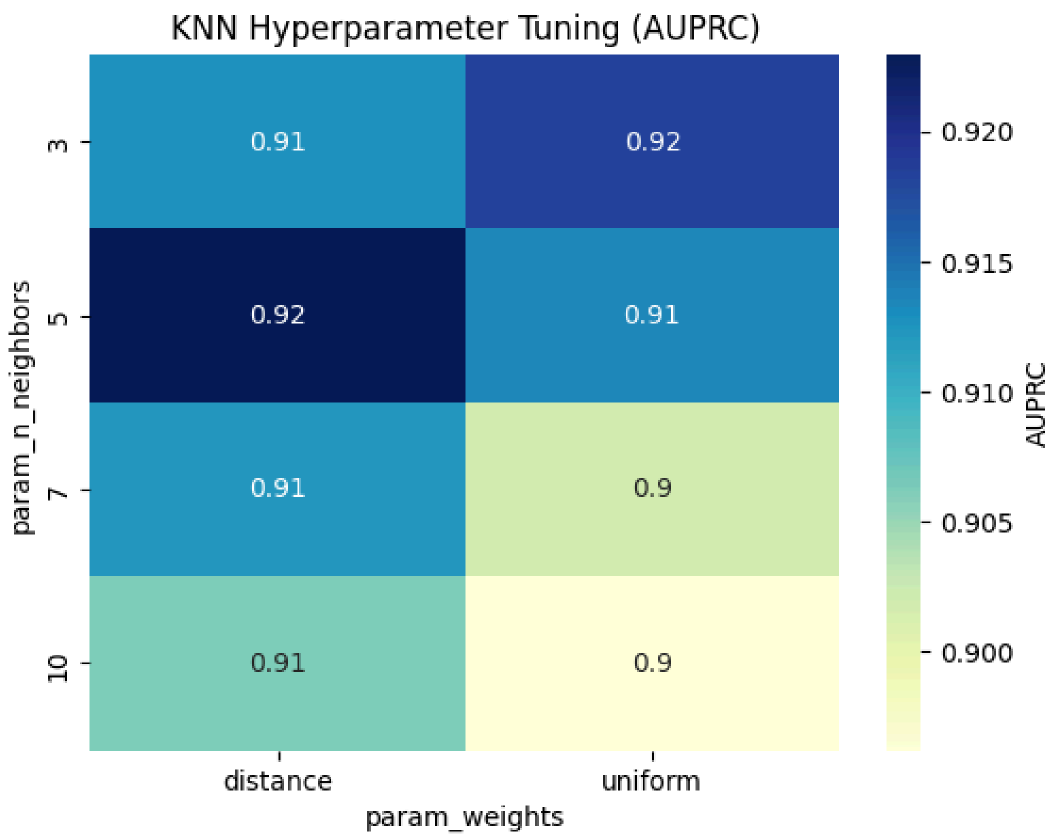
Logistic Regression is a simple, linear baseline model with regularization to handle overfitting and class imbalance. The hyperparameters tested include C, which controls the inverse of regularization strength with values [0.01, 0.1, 1, 10], and penalty, which specifies the type of regularization (l1 for Lasso and l2 for Ridge). The tuning results are visualized in a heatmap showing mean test precision for each combination of C and penalty. Higher values of C may reduce regularization, while the choice of l1 or l2 penalties determines sparsity or overall shrinkage.

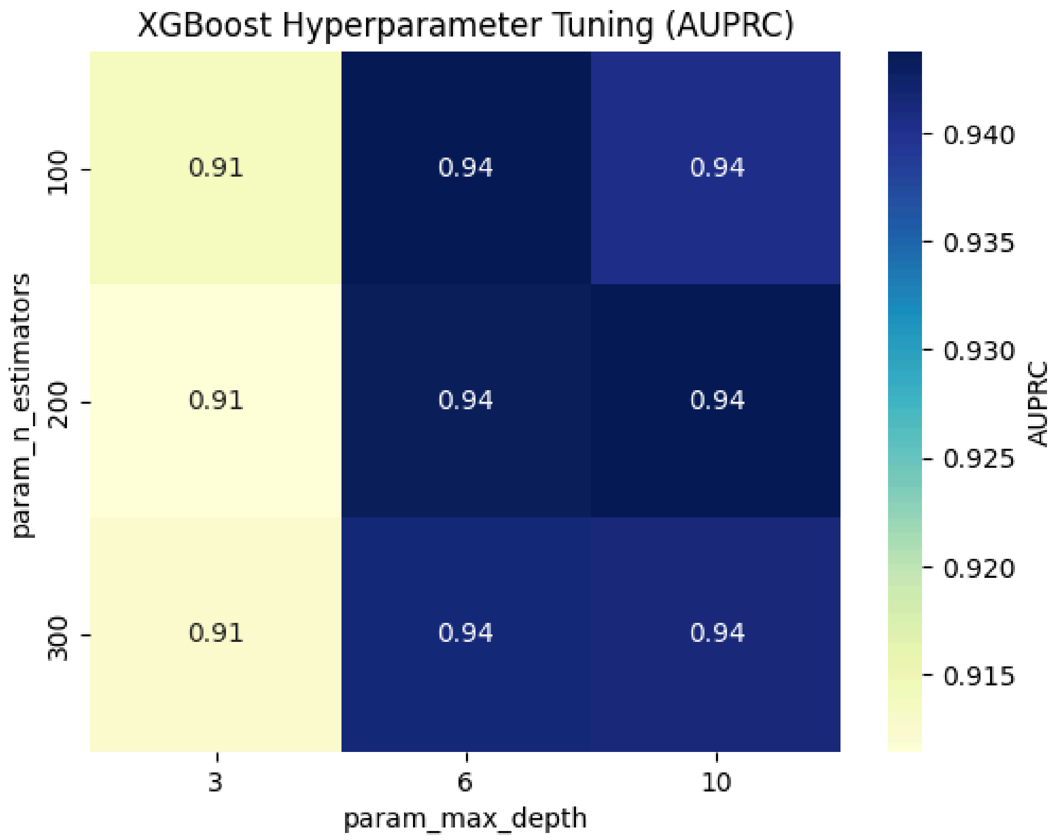
K-Nearest Neighbors (KNN) uses instance-based learning to classify transactions based on proximity in feature space. The hyperparameters tested include n_neighbors, the number of neighbors considered with values [3, 5, 7, 10], and weights, which specifies the voting scheme as either uniform (equal weight) or distance (weight inversely proportional to distance). The tuning results are visualized in a heatmap showing mean test precision (or AUPRC) for each combination of n_neighbors and weights. Smaller values of n_neighbors may overfit, while distance weighting can better handle imbalanced data.

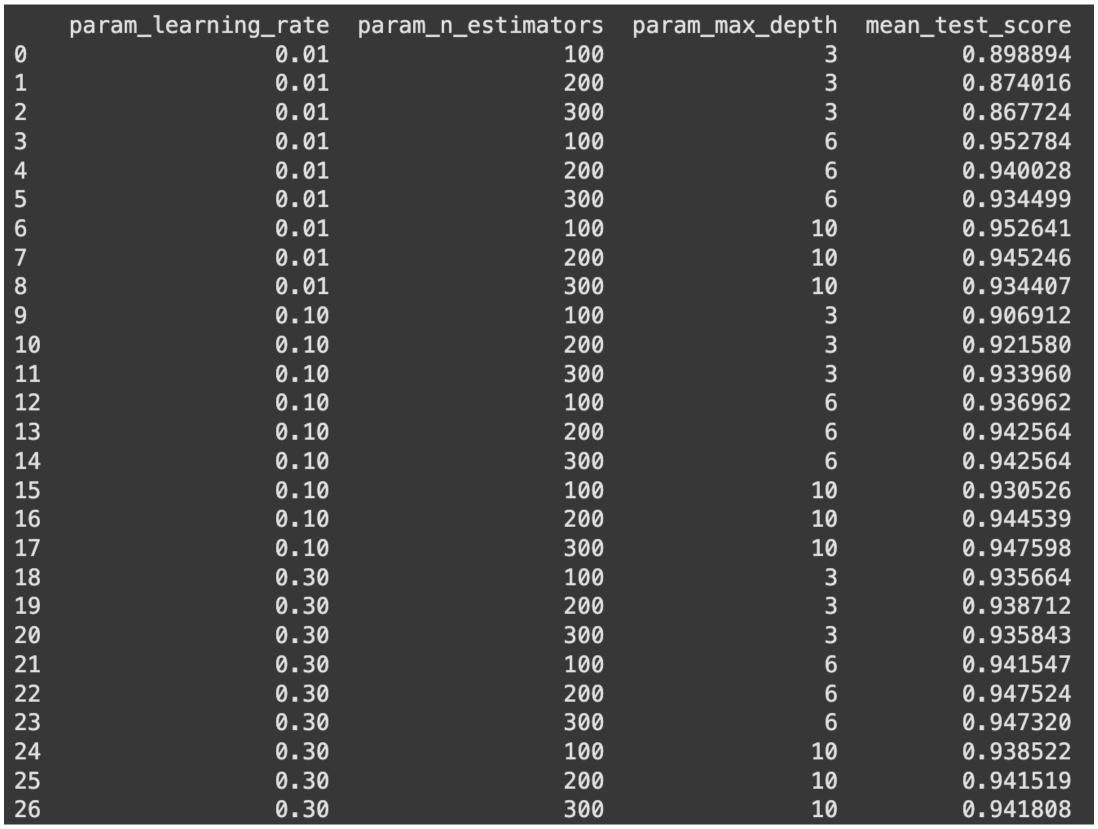
XGBoost (Extreme Gradient Boosting) is a powerful tree-based model that handles non-linear relationships and class imbalance effectively. The hyperparameters tested include learning_rate, which controls the step size for updating weights with values [0.01, 0.1, 0.3], n_estimators, which is the number of boosting rounds with values [100, 200, 300], and max_depth, which limits the tree depth to control model complexity with values [3, 6, 10]. The tuning results are visualized in a heatmap showing mean test precision (or AUPRC) for each combination of learning_rate, n_estimators, and max_depth. Higher values of max_depth and n_estimators capture complex patterns but risk overfitting, while a moderate learning_rate such as 0.1 balances convergence speed and model accuracy.

Use tables or plots to show the evaluated hyperparameter values for each model, and indicate which is best. These experiments should use only the training and validation sets. Feel free to delete the boxes, as long as it's clear.

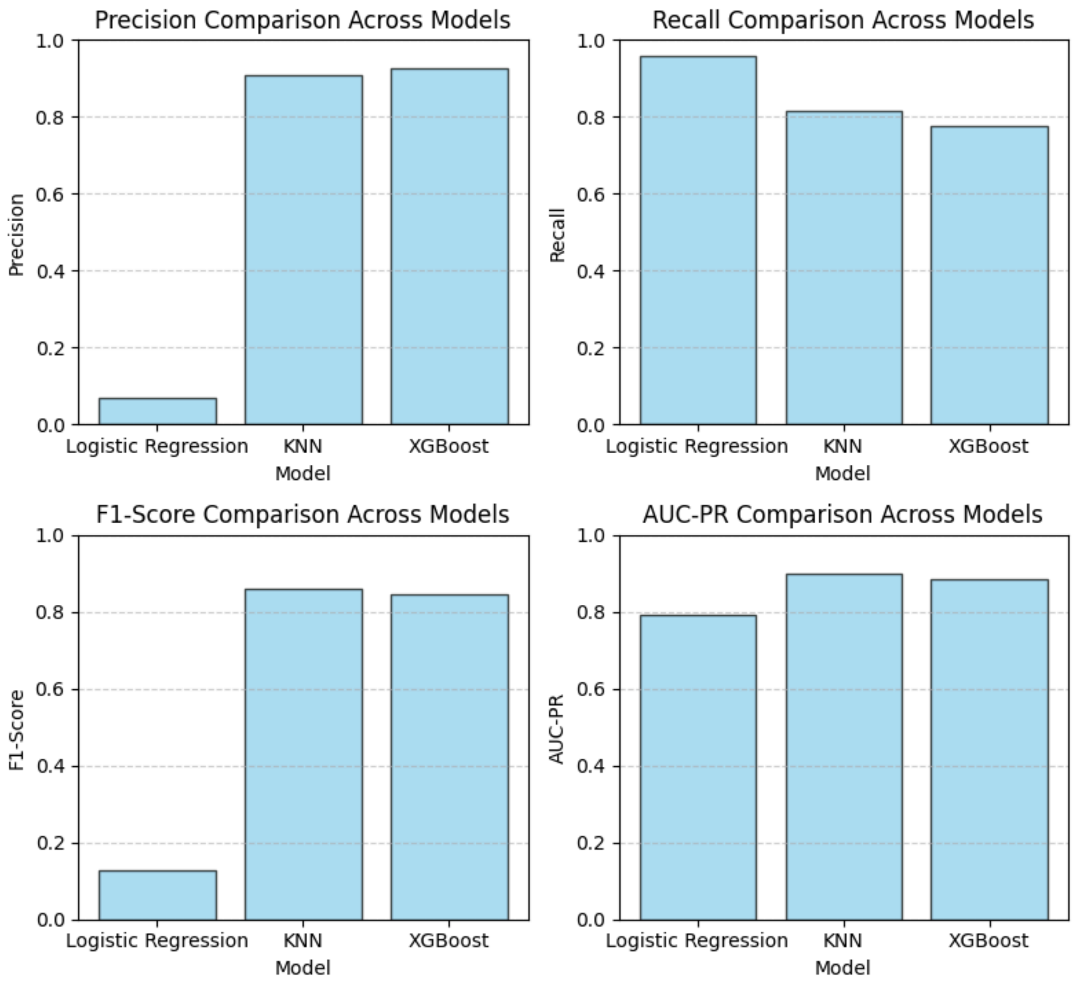


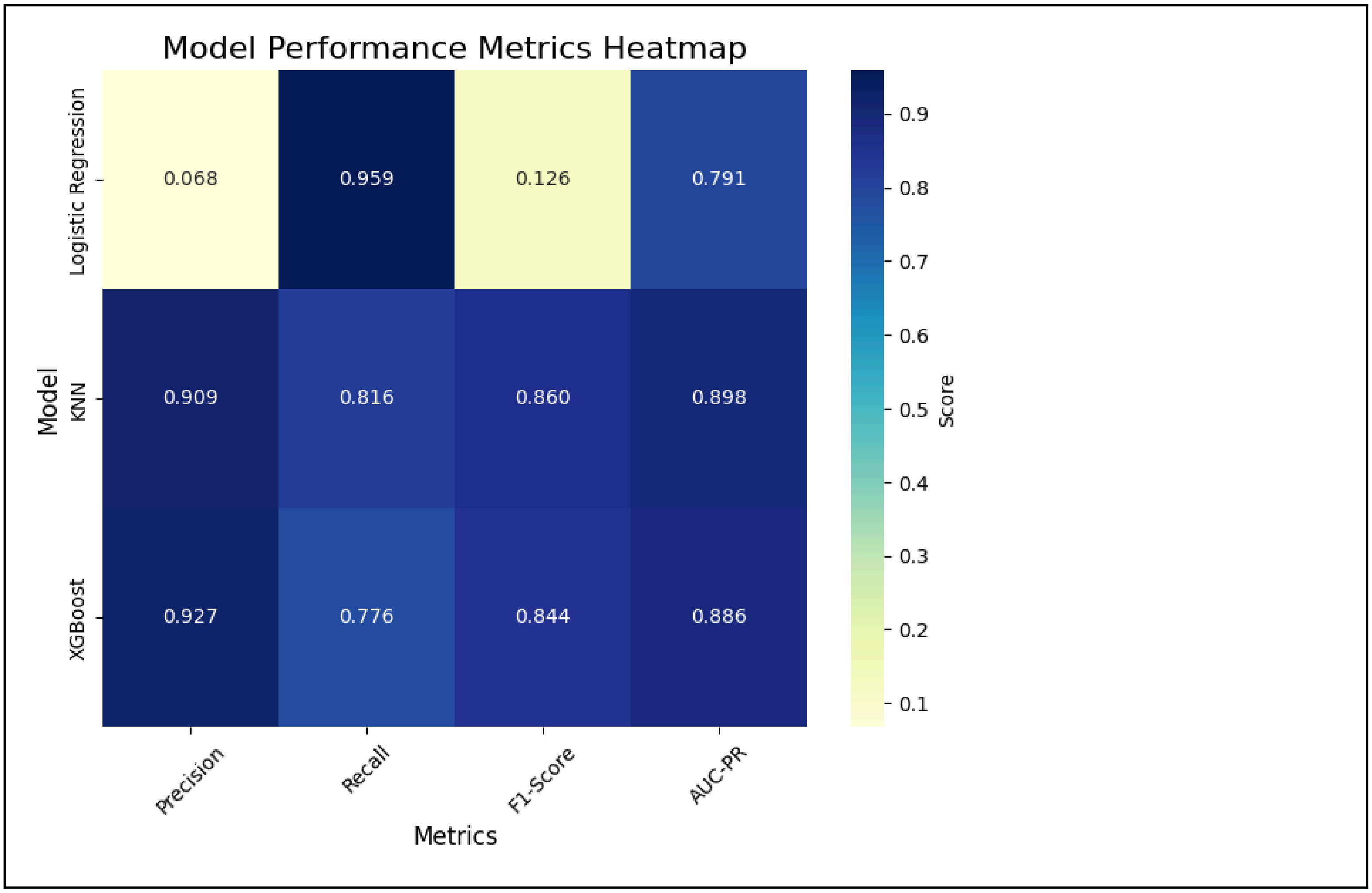

KNN Hyperparameter Tuning (AUPRC)
param_n_neighbors
3
5
7
10
param_weights
distance
uniform
0.91
0.92
0.92
0.91
0.91
0.9
0.91
0.9
AUPRC
0.920
0.915
0.910
0.905
0.900


XGBoost Hyperparameter Tuning (AUPRC)
param_n_estimators
100
200
300
param_max_depth
3
6
10
0.91
0.94
0.94
0.91
0.94
0.94
0.91
0.94
0.94
AUPRC
0.940
0.935
0.930
0.925
0.920
0.915



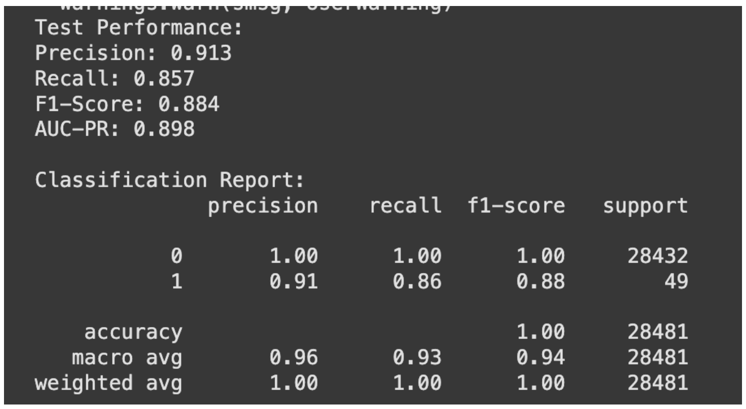
```
    param_learning_rate  param_n_estimators  param_max_depth  mean_test_score
0                  0.01                 100                3         0.898894
1                  0.01                 200                3         0.874016
2                  0.01                 300                3         0.867724
3                  0.01                 100                6         0.952784
4                  0.01                 200                6         0.940028
5                  0.01                 300                6         0.934499
6                  0.01                 100               10         0.952641
7                  0.01                 200               10         0.945246
8                  0.01                 300               10         0.934407
9                  0.10                 100                3         0.906912
10                 0.10                 200                3         0.921580
11                 0.10                 300                3         0.933960
12                 0.10                 100                6         0.936962
13                 0.10                 200                6         0.942564
14                 0.10                 300                6         0.942564
15                 0.10                 100               10         0.930526
16                 0.10                 200               10         0.944539
17                 0.10                 300               10         0.947598
18                 0.30                 100                3         0.935664
19                 0.30                 200                3         0.938712
20                 0.30                 300                3         0.935843
21                 0.30                 100                6         0.941547
22                 0.30                 200                6         0.947524
23                 0.30                 300                6         0.947320
24                 0.30                 100               10         0.938522
25                 0.30                 200               10         0.941519
26                 0.30                 300               10         0.941808
```





Which model and hyperparameters are best overall? Explain your choice. Train it using the combination of the train and validation sets and report test performance.

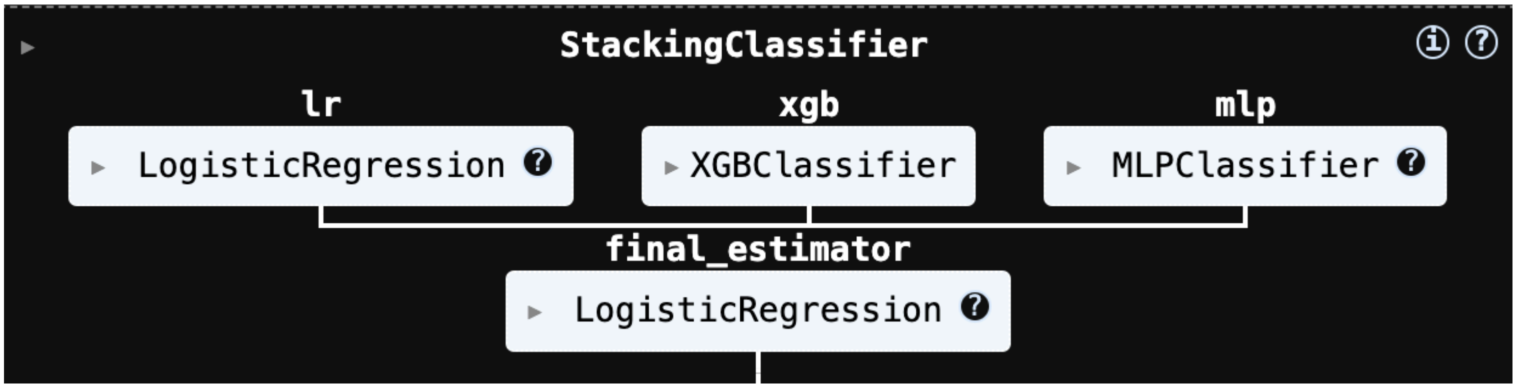
The best model is XGBoost with learning_rate=0.01, n_estimators=100, and max_depth=10, as it achieves high precision (0.952641) during validation. This configuration balances complexity and generalizability, effectively handling the dataset's non-linear relationships and class imbalance.


```
  warnings.warn(smsg, UserWarning)
Test Performance:
Precision: 0.913
Recall: 0.857
F1-Score: 0.884
AUC-PR: 0.898

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.91      0.86      0.88        49

    accuracy                           1.00     28481
   macro avg       0.96      0.93      0.94     28481
weighted avg       1.00      1.00      1.00     28481
```

3. Stretch Goal: Innovation

Describe your proposed approach

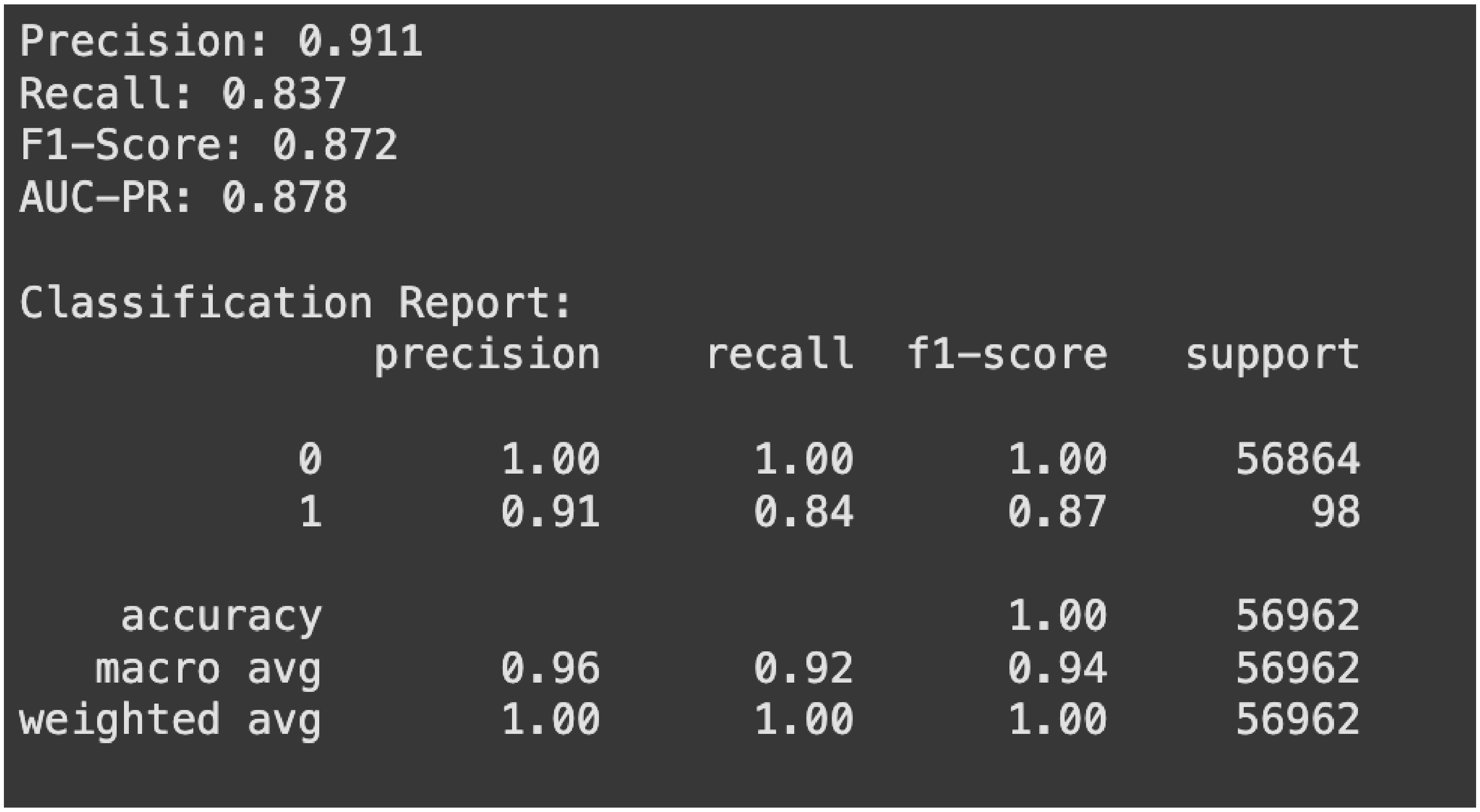
The proposed approach enhances fraud detection by integrating anomaly detection, feature engineering, and ensemble learning. Anomaly detection uses IsolationForest, which generates an anomaly_score feature and an interaction feature (amount_anomaly_interaction) to capture fraudulent patterns. An autoencoder is used for latent feature extraction, enriching the feature set with hidden representations. A stacking ensemble classifier combining Logistic Regression, XGBoost, and MLP effectively captures linear and non-linear patterns. Finally, SHAP is applied for model interpretability, providing insights into the impact of engineered and latent features on predictions. This approach addresses class imbalance, enhances model accuracy, and improves explainability.



Provide and explain the experimental analysis of your proposed approach

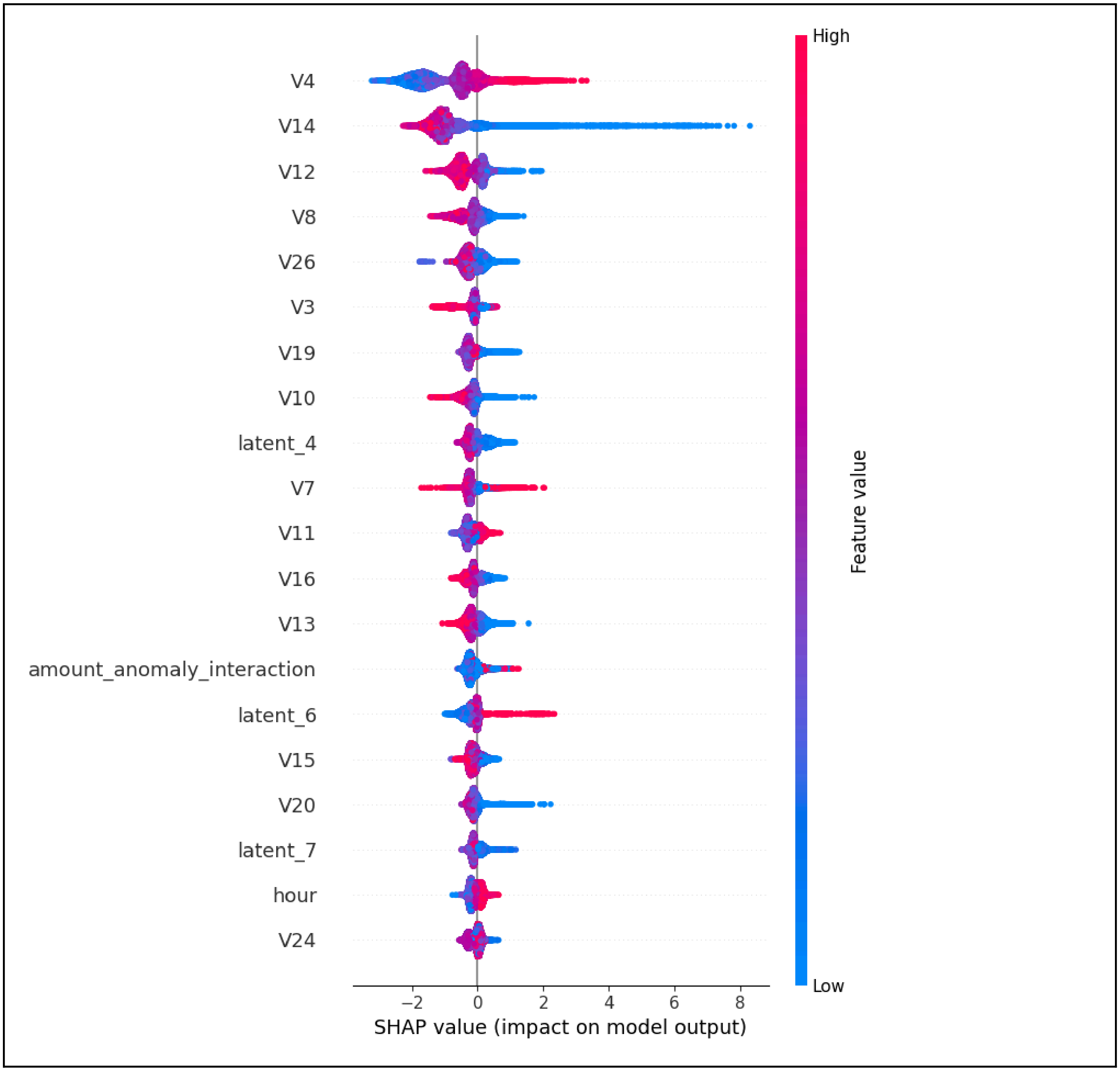
The experimental analysis of the proposed approach demonstrates its effectiveness through robust metrics and feature interpretability. The model achieved a precision of 0.911, recall of 0.837, F1-score of 0.872, and an AUC-PR of 0.878. These metrics indicate a strong balance between precision and recall, crucial for fraud detection. The high AUC-PR highlights the model's ability to distinguish between fraudulent and legitimate transactions in an imbalanced dataset.

The classification report shows near-perfect performance for the majority class (0), with precision and recall of 1.00, and strong precision (0.91) and recall (0.84) for the minority class (1). This demonstrates the model's ability to address the challenge of class imbalance effectively.



```
Precision: 0.911
Recall: 0.837
F1-Score: 0.872
AUC-PR: 0.878

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.91      0.84      0.87        98

    accuracy                           1.00     56962
   macro avg       0.96      0.92      0.94     56962
weighted avg       1.00      1.00      1.00     56962
```

The stacking ensemble combines three models (Logistic Regression, XGBoost, and MLP) and uses Logistic Regression as the meta-classifier. This structure captures linear and non-linear relationships in the data, contributing to its strong performance. SHAP analysis further enhances the interpretability of the model, with SHAP values identifying key features such as V4, V14, and the engineered feature `amount_anomaly_interaction` as major contributors to predictions. Latent features extracted through the autoencoder, such as `latent_4` and `latent_6`, also play a significant role, demonstrating the value of enriching the dataset with learned representations.



What is innovative about your approach?

The innovative approach integrates anomaly detection, autoencoder-based feature extraction, and ensemble learning to enhance fraud detection. `IsolationForest` generates interaction features like `amount_anomaly_interaction`, capturing anomalies specific to fraudulent transactions. The autoencoder enriches the dataset with latent features that reveal hidden patterns in the data, improving model representation. Finally, the stacking ensemble combines Logistic Regression, XGBoost, and MLP, leveraging their strengths to handle linear and non-linear relationships while addressing class imbalance. SHAP analysis adds interpretability, making the model suitable for real-world applications requiring accuracy and transparency.

Is your approach potentially publishable? If not (the usual case), skip this part. If so, explain the research contributions and how it fits in with related work. What venue would you consider for publication?

The approach is potentially publishable due to its novel integration of anomaly detection, feature engineering, and ensemble learning for imbalanced fraud detection. The key research contributions include the use of IsolationForest to create interaction features tailored to fraud patterns, the application of an autoencoder for latent feature extraction to enhance data representation, and the combination of diverse models (Logistic Regression, XGBoost, MLP) in a stacking ensemble for improved performance. The inclusion of SHAP analysis provides interpretability, addressing a critical gap in fraud detection systems by making model predictions more transparent. However, the approach has some limitations, including higher computational complexity due to the ensemble model and autoencoder training, which may impact scalability to larger datasets.

This work aligns with related research on handling class imbalance, advanced feature engineering, and interpretable machine learning, and it demonstrates significant improvements in fraud detection metrics compared to baseline approaches. A suitable venue for publication would be a conference such as IEEE BigData, KDD (Knowledge Discovery and Data Mining), or a specialized journal on machine learning applications, such as the Journal of Machine Learning Research (JMLR) or Pattern Recognition Letters. These venues emphasize innovative methods in data science and their practical applications, making them an ideal fit for this contribution.

4. Acknowledgments / Attribution

Link to your code (required!!)

https://github.com/kasprovav/CS-441-Final-Project

Link to your data (can be a github page or repo)

https://github.com/kasprovav/CS-441-Final-Project/blob/main/data/creditcard.zip

External citations or resources

Dataset: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Group member contributions

I was the only person working on the project, which is all my work.

Did group members contribute roughly equally or unequally? If unequally, explain and specify whether one member went above and beyond, or someone contributed less than agreed or expected.

Equally

It was only me working on the project.