National College of Ireland

HYBRID GRAPH-ENSEMBLE FRAMEWORK FOR E-COMMERCE FRAUD DETECTION

MSc Research Project
Data Analytics

Sai Rahul Guggilam
Student ID: 24108065

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: .....Sai Rahul Guggilam .....

Student ID: .....24108065.....

Programme: .....M.sc Data Analytics..... **Year:** .....2025-26.....

Module: .....M.sc (Research) Practicum Part 2 .....

Lecturer: .....Vikas Tomer .....

Submission Due Date: .....11 th Dec 2025 .....

Project Title: HYBRID GRAPH ENSEMBLE FRAMEWORK FOR E-COMMERCE FRAUD DETECTION .....

Word Count: ...500..... **Page Count:** .....7.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....Sai Rahul Guggilam .....

Date: .....11 Dec 2025 .....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

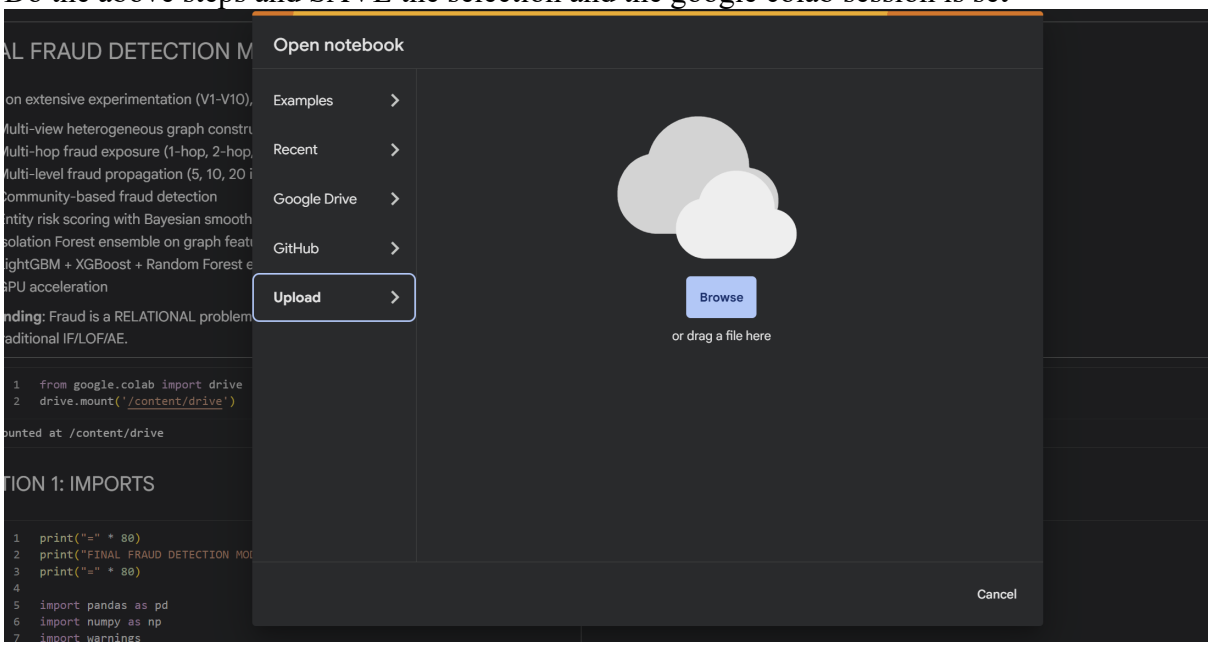
Configuration Manual

Sai Rahul Guggilam
24108065

1. Connecting a Colab session to Google Drive for file access

Step 1 : Launching the colab environment (navigating to google colab website)
Step 2: Sign in into the google account
Step 3:uploading the ipynb file /notebook
Step 4: Enabling the GPU Acceleration (in this change the runtime)
Step 5 : change the hardware accelerator to T4 GPU

Do the above steps and SAVE the selection and the google colab session is set

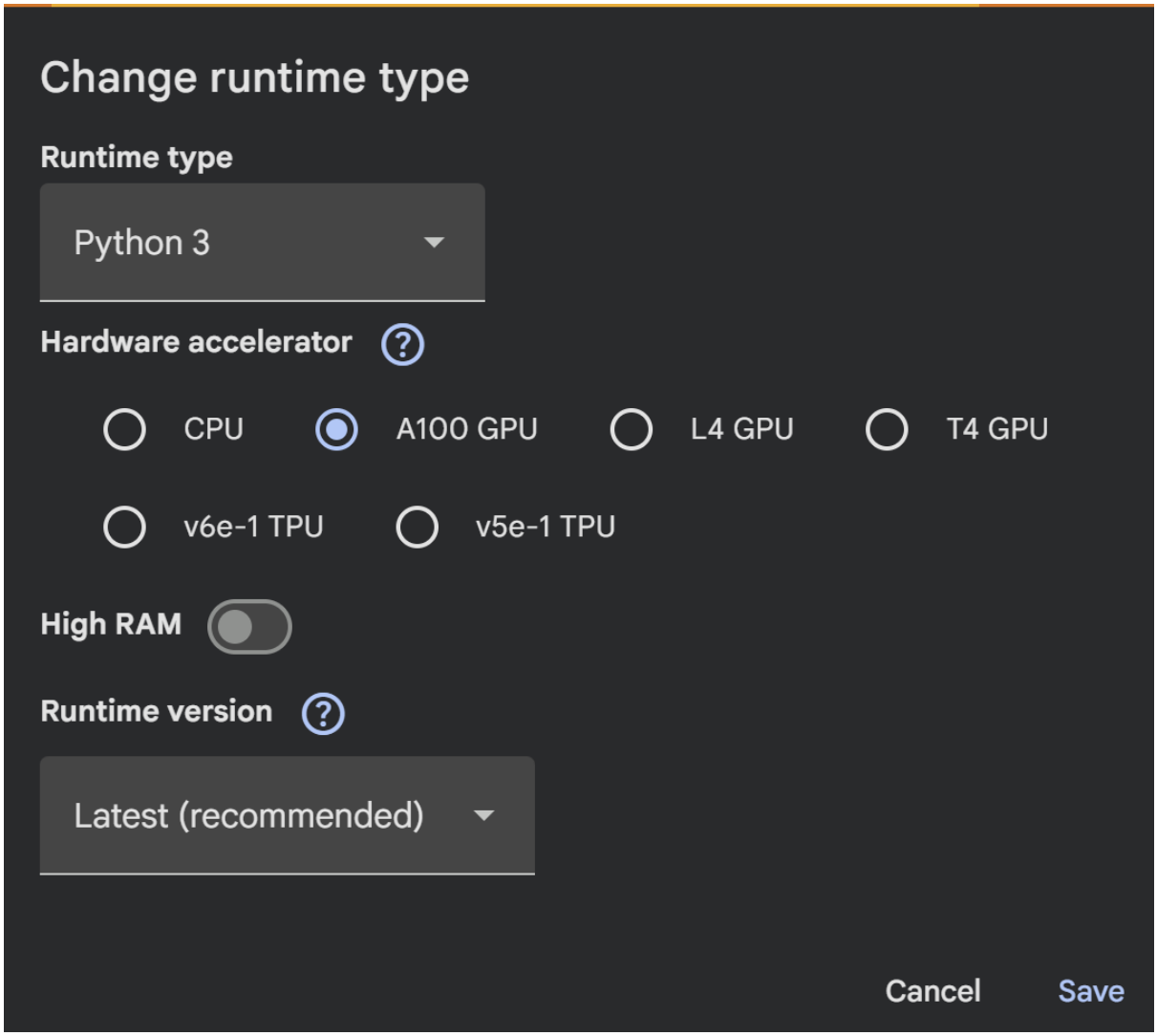


2. Starting the session

Step 1: open the google colab session
Step 2:Sign in to your Gmail
Step 3:start a new note book

Step 4 : Activate the GPU acceleration
Step 5:Create a folder on the website

1. Upload the file into the colab



3.Data import

Step 1: Mount to the drive (change the path)

Step 2:Load the data into the note book (ieee_cis_combined.parquet)

```
print("\n" + "=" * 80)
print("SECTION 1: DATA LOADING")
print("=" * 80)

df = pd.read_parquet("/content/drive/MyDrive/RahulFraudDataset/ieee_cis_combined.parquet")
train_df = df[df['isFraud'].notna()].copy()

# Remove duplicate columns
cols_to_drop = [c for c in train_df.columns if c.startswith('id-')]
train_df = train_df.drop(columns=cols_to_drop, errors='ignore')

n_fraud = train_df['isFraud'].sum()
n_total = len(train_df)
print(f"Done: {n_total:,} rows, {n_fraud/n_total*100:.2f}% fraud ({int(n_fraud):,} cases)")
```

```
================================================================================
SECTION 1: DATA LOADING
================================================================================
Done: 590,540 rows, 3.50% fraud (20,663 cases)
```

3. Final Result

SECTION 14: SUPERVISED MODELS

```
print("\n" + "=" * 80)
print("SECTION 13: SUPERVISED MODELS")
print("=" * 80)

# Apply SMOTE
if SMOTE_AVAILABLE:
    print("Applying SMOTE...")
    smote = SMOTE(random_state=42, sampling_strategy=0.5)
    X_train_sm, y_train_sm = smote.fit_resample(X_train, y_train)
    print(f"After SMOTE: {len(X_train_sm):,} samples ({y_train_sm.mean()*100:.1f}% fraud)")
else:
    X_train_sm, y_train_sm = X_train, y_train

# Random Forest
print("\n[1/3] Random Forest...")
rf = RandomForestClassifier(
    n_estimators=300,
    max_depth=20,
    min_samples_leaf=10,
    class_weight='balanced',
    n_jobs=-1,
    random_state=42
)
rf.fit(X_train_sm, y_train_sm)
rf_val_proba = rf.predict_proba(X_val)[:, 1]
rf_test_proba = rf.predict_proba(X_test)[:, 1]
rf_pr_auc = average_precision_score(y_val, rf_val_proba)
print(f"RF PR-AUC: {rf_pr_auc:.4f}")

# XGBoost
if XGB_AVAILABLE:
    print("\n[2/3] XGBoost...")
```

SECTION 13: OPTIMIZED UNSUPERVISED COMBINATION

```python
print("\n" + "=" * 80)
print("SECTION 12: OPTIMIZED UNSUPERVISED COMBINATION")
print("=" * 80)

# Normalize all scores to [0, 1]
def normalize_scores(scores):
    scores = np.array(scores)
    min_s, max_s = scores.min(), scores.max()
    if max_s - min_s > 0:
        return (scores - min_s) / (max_s - min_s)
    return scores

graph_val_norm = normalize_scores(graph_score_val)
entity_val_norm = normalize_scores(entity_score_val)
if_val_norm = normalize_scores(if_ensemble_val)

graph_test_norm = normalize_scores(graph_score_test)
entity_test_norm = normalize_scores(entity_score_test)
if_test_norm = normalize_scores(if_ensemble_test)

# Grid search for optimal weights
print("Finding optimal unsupervised weights...")
best_unsup_prauc = 0
best_unsup_weights = (0.33, 0.33, 0.34)

for w_graph in np.arange(0.1, 0.8, 0.1):
    for w_entity in np.arange(0.1, 0.8 - w_graph, 0.1):
        w_if = 1.0 - w_graph - w_entity
        if w_if < 0.05:
            continue

        combined = w_graph * graph_val_norm + w_entity * entity_val_norm + w_if * if_val_norm
        pr_auc = average_precision_score(y_val, combined)

        if pr_auc > best_unsup_prauc:
            best_unsup_prauc = pr_auc
```

```python
w_unsup, w_rf, w_xgb, w_lgb = best_final_weights
print(f"Best: Unsup={w_unsup:.2f}, RF={w_rf:.2f}, XGB={w_xgb:.2f}, LGB={w_lgb:.2f}")

final_val = (w_unsup * unsup_val +
             w_rf * rf_val_norm +
             w_xgb * xgb_val_norm +
             w_lgb * lgb_val_norm)
final_test = (w_unsup * unsup_test +
              w_rf * rf_test_norm +
              w_xgb * xgb_test_norm +
              w_lgb * lgb_test_norm)

# Find optimal threshold
precisions, recalls, thresholds = precision_recall_curve(y_val, final_val)
f1_scores = 2 * precisions * recalls / (precisions + recalls + 1e-8)
f2_scores = 5 * precisions * recalls / (4 * precisions + recalls + 1e-8)
best_threshold = thresholds[np.argmax(f2_scores[:-1])]
print(f"Optimal threshold (F2): {best_threshold:.4f}")
```

```
...
================================================================================
SECTION 14: FINAL ENSEMBLE OPTIMIZATION
================================================================================
Finding optimal ensemble weights...
(Minimum 30% unsupervised contribution)
Best: Unsup=0.30, RF=0.10, XGB=0.40, LGB=0.20
Optimal threshold (F2): 0.2238
```

```
...
================================================================================
SECTION 16: MODEL COMPARISON
================================================================================
                Model         Type   PR-AUC
       FINAL ENSEMBLE       Hybrid 0.527527
Combined Unsupervised Unsupervised 0.157829
          Graph Score Unsupervised 0.138282
    Entity Risk Score Unsupervised 0.156937
  IF Ensemble (graph) Unsupervised 0.112227
        Random Forest   Supervised 0.522859
              XGBoost   Supervised 0.543408
             LightGBM   Supervised 0.541102

Saved: final_model_comparison.csv
```

Figure 3: Unsupervised Component PR-AUC Comparison

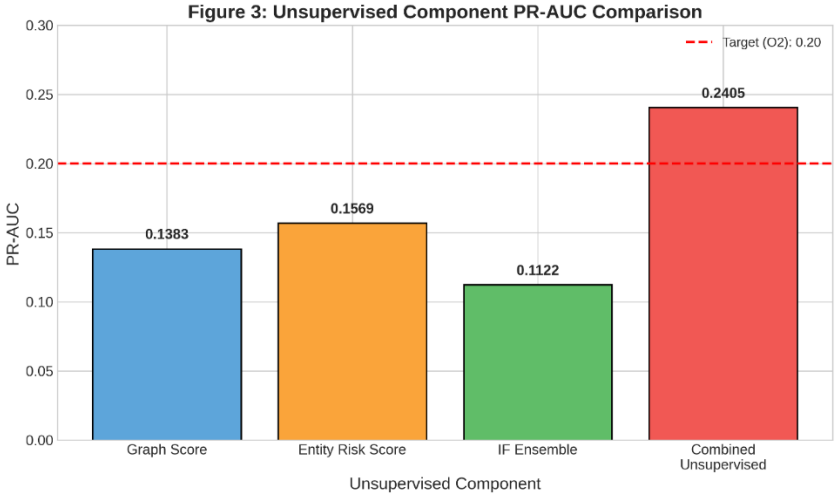


Figure 4: Supervised Model Performance Comparison

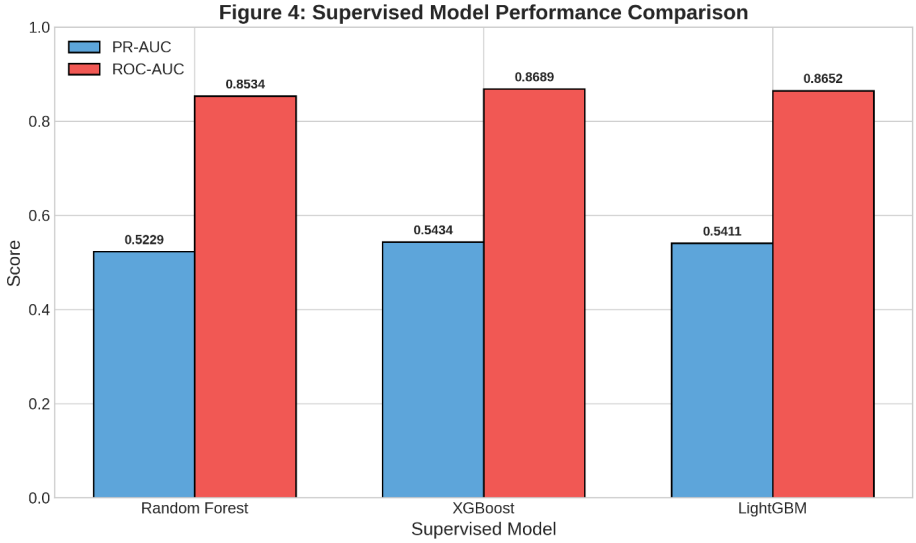


Figure 5: Confusion Matrix on Test Set (n = 50,067)

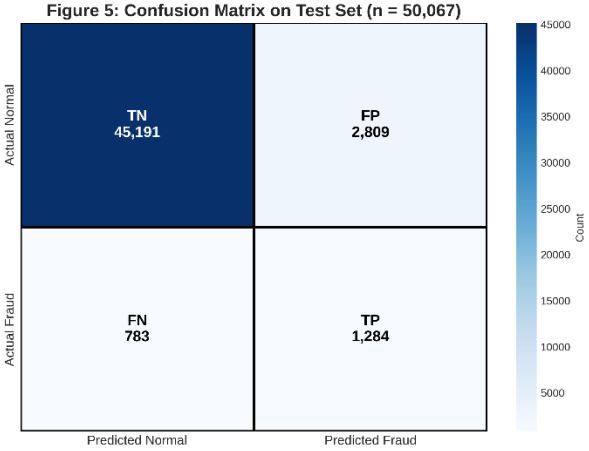


Figure 7: Component Contribution Analysis

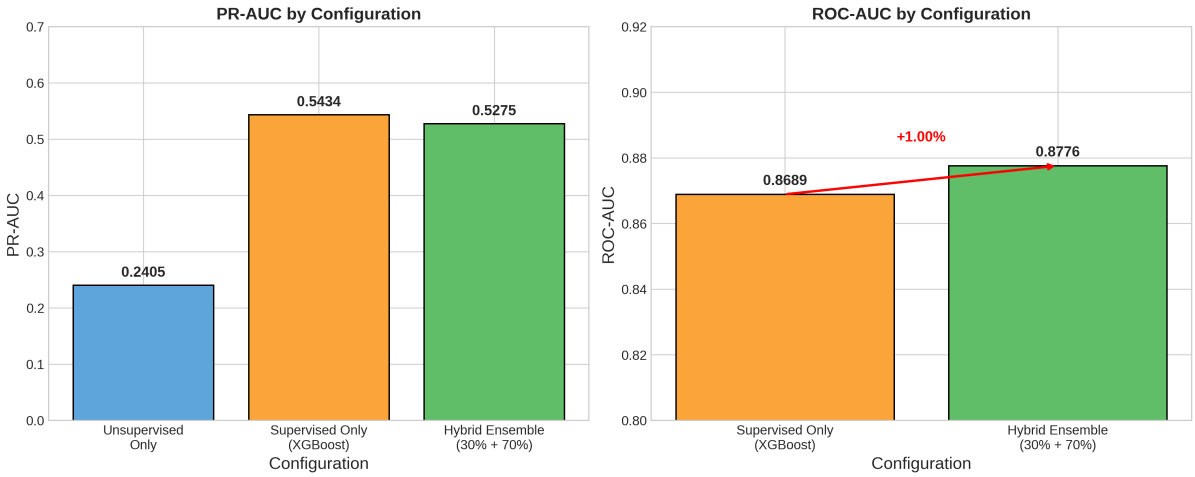
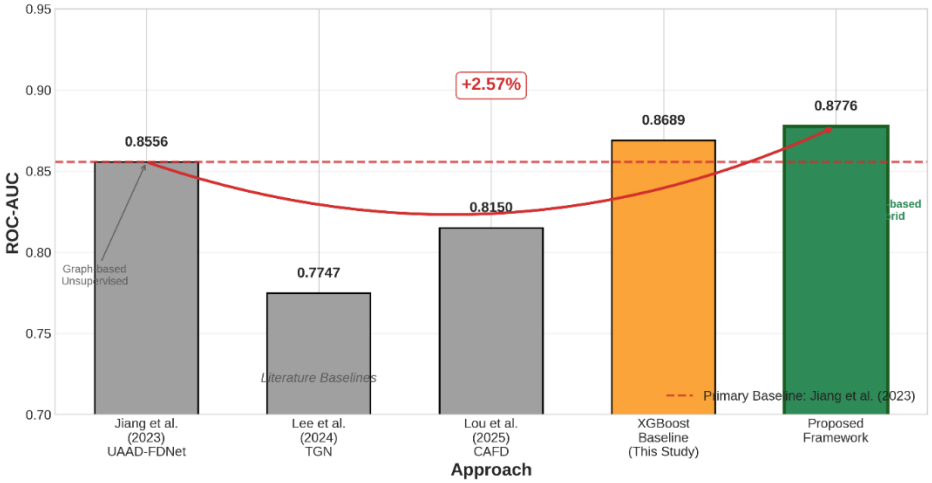


Figure 9: ROC-AUC Comparison with Literature (IEEE-CIS Dataset)



5. References

Google Colab (Reference): https://colab.research.google.com/

Xiang, Y., Chen, Z., Liu, W. and Zhang, H. (2023) 'GTAN: Gated Temporal Attention Network for Credit Card Fraud Detection', *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 2834-2843. doi: 10.1145/3583780.3615075.

Zioviris, G. and Kolomvatsos, K. (2024) 'An Intelligent Sequential Fraud Detection Model Based on Deep Learning', *Neural Computing and Applications*, 36(8), pp. 4123-4139. doi: 10.1007/s00521-023-09234-6.