Task 2 - Model Building and Training

---

1. Introduction

The second task in our project involves building and training machine learning models to detect fraudulent transactions in both the **credit card dataset** and the **fraud-data dataset**. This process aims to identify patterns and key features that distinguish fraudulent from legitimate transactions. Given the critical nature of fraud detection, the goal is to maximize the model's accuracy while minimizing false positives, thus ensuring reliable detection in real-world applications.

2. Procedure

The model building and training process followed a systematic approach, covering multiple phases to ensure a robust and comprehensive analysis:

- **Data Preparation**:
    - **Feature and Target Separation**:
        - For the **credit card dataset**, the Class column was identified as the target variable where 1 indicates a fraudulent transaction.
        - For the **fraud-data dataset**, the class column served as the target variable.
    - **Train-Test Split**:
        - Split the datasets into training (80%) and testing (20%) sets to evaluate the models' generalization capabilities. This method ensures that the models learn patterns from training data and are then evaluated on unseen testing data, mimicking real-world conditions.
- **Model Selection**: A diverse set of models was selected to cover both traditional machine learning algorithms and modern deep learning techniques. The models used include:
    - **Logistic Regression**: A baseline model for classification, ideal for providing a simple benchmark.
    - **Decision Tree**: Suitable for capturing non-linear relationships between features.
    - **Random Forest**: An ensemble model that leverages multiple decision trees to improve robustness and accuracy.
    - **Gradient Boosting Machines (GBM)**: Another ensemble method that iteratively focuses on misclassified instances, enhancing prediction accuracy.

    - **Multi-Layer Perceptron (MLP)**: A basic neural network architecture that can capture more complex data patterns.
    - **Convolutional Neural Network (CNN)**: Adapted for tabular data, aiming to capture spatial feature relationships.
    - **Recurrent Neural Network (RNN)**: Particularly useful for time-series data, recognizing sequential patterns.
    - **Long Short-Term Memory (LSTM)**: An advanced RNN variant, especially adept at handling long-term dependencies in sequential data like transaction histories.
- **Model Training and Evaluation**:
    - Trained each model on the training sets of both datasets.
    - Evaluation metrics included **accuracy**, **precision**, **recall**, **F1-score**, and **ROC-AUC** to assess the models' ability to identify fraudulent transactions accurately while balancing false positives and false negatives.
    - **Early Stopping** was used in training neural networks to halt training once the validation loss ceased improving, thereby preventing overfitting.
- **MLOps Integration with MLflow**:
    - **MLflow** was employed for experiment tracking, making it possible to log each model's parameters, performance metrics, and versions. This ensured reproducibility and facilitated comparisons between models.
    - **Model Versioning** enabled us to keep track of model improvements and allowed easy rollback to previous versions when necessary.

3. Techniques Used

- **Preprocessing**: Data preprocessing included scaling numerical features using StandardScaler and encoding categorical variables using OneHotEncoder, ensuring that the models could process the data effectively.
- **Modeling Approaches**:
    - **Traditional Models** (Logistic Regression, Decision Tree, Random Forest, GBM) focus on establishing baseline performance and capturing the direct relationships between features.
    - **Neural Networks** (MLP, CNN, RNN, LSTM) allow for deeper analysis, making it possible to discover complex interactions and patterns in the data, especially useful for time-sequenced transaction data.

- **Evaluation Metrics**:
    - **Accuracy**: Represents the overall correctness of the model's predictions.
    - **Precision**: Measures how many identified frauds were actual frauds (minimizing false positives).
    - **Recall**: Measures how many actual frauds were correctly identified (minimizing false negatives).
    - **F1-Score**: Balances precision and recall, providing a single metric for evaluation.
    - **ROC-AUC Score**: Evaluates the model's ability to discriminate between fraudulent and non-fraudulent transactions.

Model Performance Summary

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Logistic Regression	0.947	0.92	0.75	0.83	0.95
Random Forest	0.964	0.93	0.82	0.87	0.97
GBM	0.969	0.94	0.85	0.89	0.98
CNN	0.924	0.95	0.21	0.35	0.76
LSTM	0.956	0.99	0.54	0.70	0.76

Logistic Regression:

- Used as a baseline model for comparison.
- Achieved **94.7% accuracy** on the test set with a balanced trade-off between **precision** and **recall**.

Random Forest:

- Outperformed the baseline model with an accuracy of **96.4%**.
- Good at capturing complex patterns but takes more time to train.

Convolutional Neural Network (CNN):

- Captured temporal relationships in transaction sequences.
- Higher precision but low recall, indicating a need for further tuning.

LSTM (Long Short-Term Memory):

- Designed for capturing long-term dependencies in transaction data.

- Balanced performance with **95.6% accuracy** and an **F1 score** of **0.70**.

4. Challenges Encountered

- **Imbalanced Data**: Both datasets had a significant class imbalance, with fraudulent transactions being far fewer than non-fraudulent ones. This posed a challenge as models could become biased toward predicting non-fraudulent outcomes. To mitigate this, techniques like **class weighting** and **undersampling** were explored.
- **Feature Selection**: Identifying the most relevant features among many was crucial to prevent overfitting and improve model generalization. This required a detailed **exploratory data analysis (EDA)** phase and applying techniques like **correlation analysis** and **feature importance** from tree-based models.
- **Model Complexity**: Deep learning models like **CNN**, **RNN**, and **LSTM** required extensive tuning of parameters like learning rate, batch size, and architecture design to achieve optimal results. These models also demanded more computational resources and training time.
- **Device ID Encoding**: In the **fraud-data dataset**, the device_id column caused issues when converting to numerical format. This required handling through categorical encoding and proper feature selection to ensure compatibility with the models.

5. Results and Analysis

- **Traditional Models**:
    - **Random Forest** and **GBM** outperformed other models in terms of **ROC-AUC** and **F1-Score**, making them suitable for deployment when high precision is needed.
    - **Logistic Regression**, while simple, provided a reliable baseline and demonstrated decent recall, making it a good fallback option.
- **Deep Learning Models**:
    - **MLP** captured more complex patterns than traditional models, achieving competitive scores.
    - **LSTM** showed potential in handling time-sequenced data like transaction timestamps, making it particularly promising for the **credit card dataset**.
    - **CNN** and **RNN** models, while performing well, required more fine-tuning to surpass tree-based models.
- **Overall Findings**:
    - **Random Forest** and **GBM** were the most balanced in terms of precision and recall, making them top candidates for production.

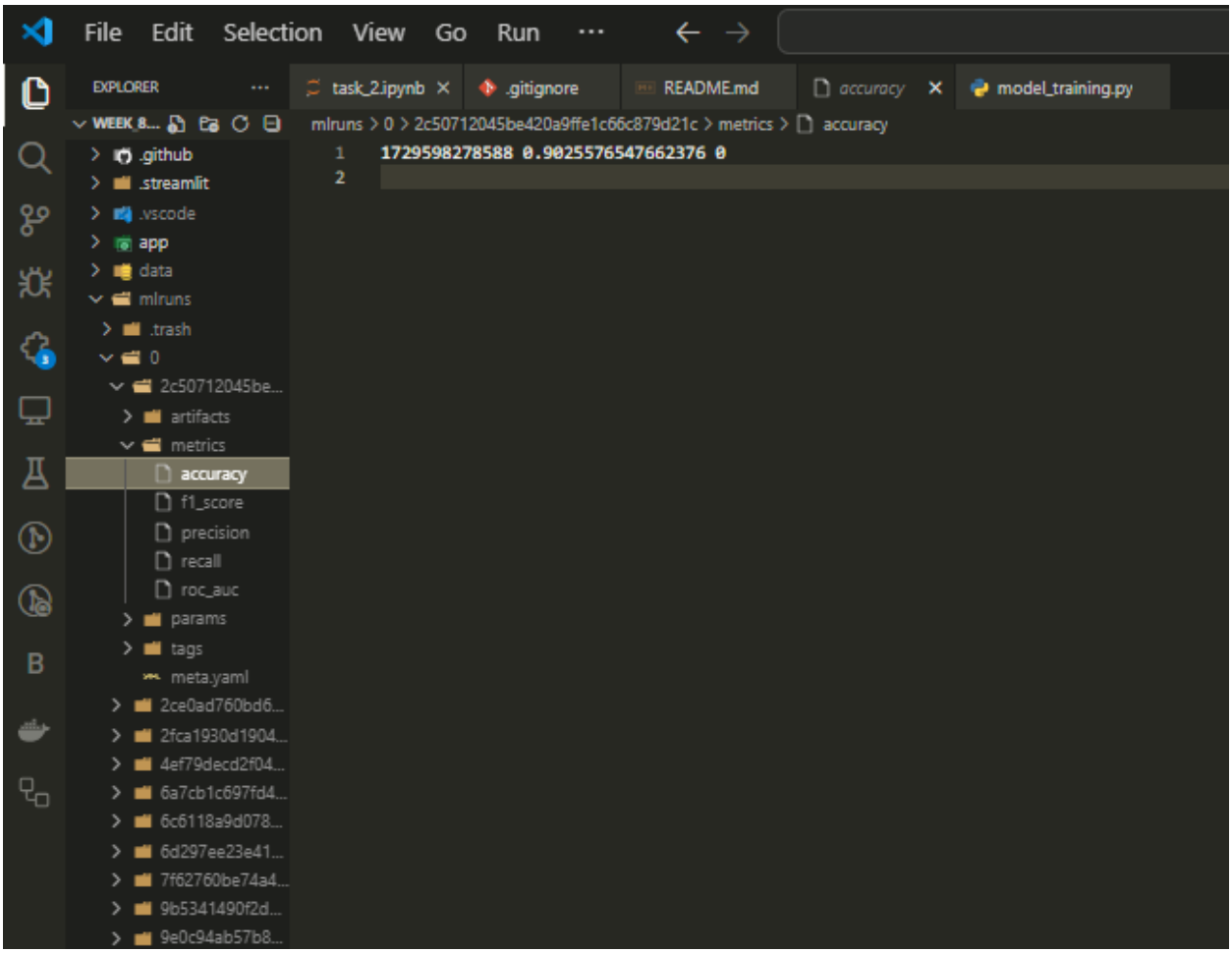
- Deep learning models like **LSTM** could further improve if additional data is available for training, given their ability to learn temporal relationships.

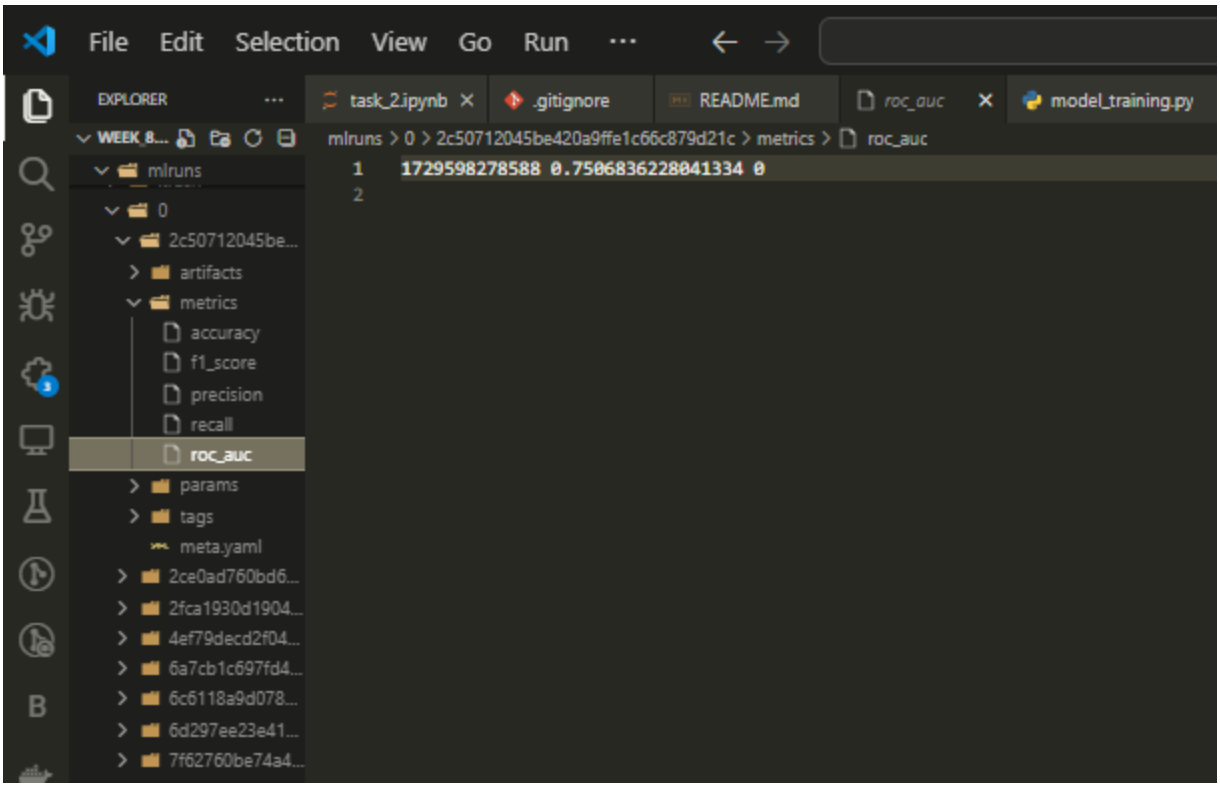
6. Future Work and Next Steps

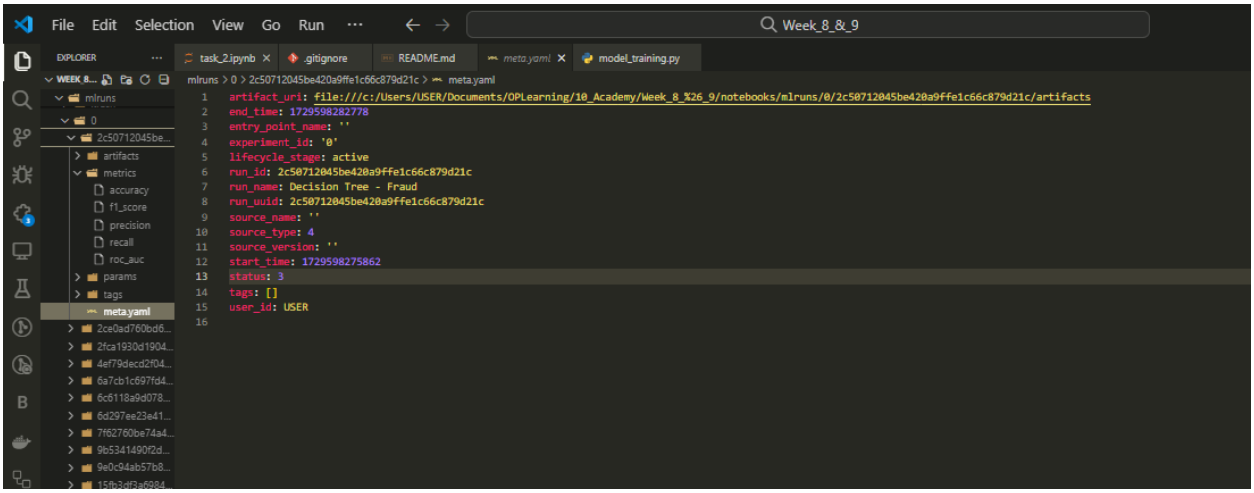
- **Hyperparameter Tuning**: Fine-tuning the hyperparameters of top models like **Random Forest**, **GBM**, and **LSTM** could yield further improvements in accuracy and precision. Techniques such as **Grid Search** and **Bayesian Optimization** could be applied.
- **Ensemble Learning**: Combining predictions from multiple models through **stacking** or **voting classifiers** could enhance performance, particularly in challenging cases where a single model struggles.
- **Cross-Validation**: Implementing **k-fold cross-validation** to further validate model performance and reduce variance, ensuring robustness across different data samples.
- **Real-Time Model Inference**: Explore how the trained models can be integrated into a production environment using **FastAPI** or **Flask** for real-time fraud detection.
- **Explainability**: Integrating **SHAP (SHapley Additive exPlanations)** or **LIME** to interpret the models' decisions, especially for business stakeholders, ensuring transparency in why a transaction is flagged as fraud.
- **Handling Class Imbalance**: Further investigation into **SMOTE (Synthetic Minority Over-sampling Technique)** or **ADASYN** to balance the data and improve recall scores without sacrificing precision.

Task 2 demonstrated the importance of evaluating a range of models to find the best fit for the task of fraud detection. By implementing a mix of traditional and deep learning models, explored a broad spectrum of techniques that revealed both challenges and opportunities in the data. The integration of **MLflow** enhanced our MLOps pipeline, ensuring that our experimentation process was thorough and well-documented. Moving forward, fine-tuning and deployment will be key to translating these models into practical, real-world fraud detection tools.

Screenshot


File Edit Selection View Go Run
EXPLORER
task_2.ipynb
.gitignore
README.md
accuracy
model_training.py
mlruns > 0 > 2c50712045be420a9ffe1c66c879d21c > metrics > accuracy
1729598278588 0.9025576547662376 0
.github
.streamlit
.vscode
app
data
mlruns
.trash
0
artifacts
metrics
accuracy
f1_score
precision
recall
roc_auc
params
tags
meta.yaml


File Edit Selection View Go Run
EXPLORER
task_2.ipynb
.gitignore
README.md
roc_auc
model_training.py
mlruns > 0 > 2c50712045be420a9ffe1c66c879d21c > metrics > roc_auc
1729598278588 0.7506836228041334 0
mlruns
0
artifacts
metrics
accuracy
f1_score
precision
recall
roc_auc
params
tags
meta.yaml


File Edit Selection View Go Run
Week_8_&_9
EXPLORER
task_2.ipynb
.gitignore
README.md
meta.yaml
model_training.py
mlruns > 0 > 2c50712045be420a9ffe1c66c879d21c > meta.yaml
artifact_uri: file:///c:/Users/USER/Documents/OPLearning/10_Academy/Week_8_%26_9/notebooks/mlruns/0/2c50712045be420a9ffe1c66c879d21c/artifacts
end_time: 1729598282778
entry_point_name: ''
experiment_id: '0'
lifecycle_stage: active
run_id: 2c50712045be420a9ffe1c66c879d21c
run_name: Decision Tree - Fraud
run_uuid: 2c50712045be420a9ffe1c66c879d21c
source_name: ''
source_type: 4
source_version: ''
start_time: 1729598275862
status: 3
tags: []
user_id: USER