Configuration Manual

MSc Research Project

NAGA SAI BHASKAR NAVEEN YEDDLA
Student ID: X23245077

School of Computing
National College of Ireland

Supervisor: Dr.William Clifford

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	NAGA SAI BHASKAR NAVEEN YEDDLA
Student ID:	X23245077
Programme:	MSc. Data Analytics **Year:**2024-2025
Module:	MSc. Research Project
Lecturer:	Dr.William Clifford
Submission Due Date:	12-12-2024
Project Title:	Fedrated Learning and Privacy-Preserving Artificial Intelligence
Word Count:	7346 **Page Count:** 24

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:	NAGA SAI BHASKAR NAVEEN YEDDLA
Date:	11-12-2024

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

NAGA SAI BHASKAR NAVEEN YEDDLA
Student ID: x23245077

1 Introduction

This manual provides a comprehensive guide for implementing fraud detection systems using machine learning models and federated learning techniques on Google Colab with GPU acceleration. It includes environment setup, exploratory data analysis (EDA), preprocessing, model building, evaluation, and federated learning with differential privacy for secure distributed training.

2 System Configuration

2.1 Hardware Specifications

Component	**Specification**
Operating System	Google Colab Environment
GPU	NVIDIA Tesla T4 or P100
RAM	12 GB
Disk Space	100 GB (Colab Environment)

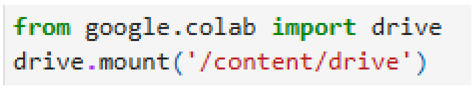
2.2 Software Specifications

Software	**Specification**
Programming Language	Python 3.9
Libraries	Pandas, Numpy, Scikit-learn, Imbalanced-learn, Matplotlib, Seaborn, PyTorch
IDE/Tools	Google Colab
Dataset Formats	CSV

3 Environment Setup

3.1 Google Colab Setup

1. **Accessing Colab:**
   - Open Google Colab in your browser.
   - Ensure you are logged in to your Google account.
2. **Enable GPU Acceleration:**
   - Navigate to **Runtime > Change Runtime Type**.
   - Select **Hardware Accelerator** as **GPU** and save.
3. **Mount Google Drive:**
   - Mount Google Drive to access datasets:


```python
from google.colab import drive
drive.mount('/content/drive')
```

4. **Install Required Libraries:**
   - Install libraries using pip

3.2 Dataset Preparation

- **Dataset 1:** `creditcard.csv` (Credit card fraud detection) (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)
- **Dataset 2:** `PS_20174392719_1491204439457_log.csv` (Synthetic Financial Datasets For Fraud Detection). (https://www.kaggle.com/datasets/ealaxi/paysim1)
- Upload these datasets to Google Drive and provide their file paths in the Colab notebook.

4 Implementation

4.1 Exploratory Data Analysis (EDA)

1. **Dataset Inspection:**
   - Load and display dataset structure:

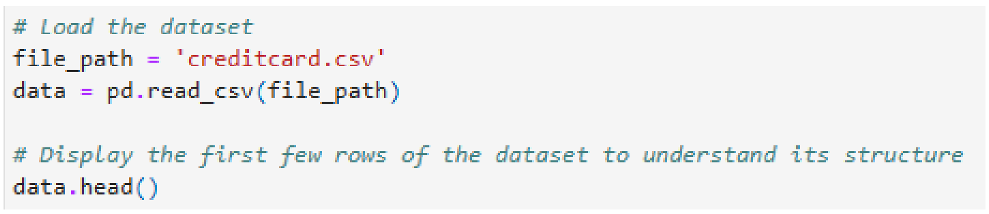

```python
# Load the dataset
file_path = 'creditcard.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset to understand its structure
data.head()
```

Figure 1: creditcard dataset loading

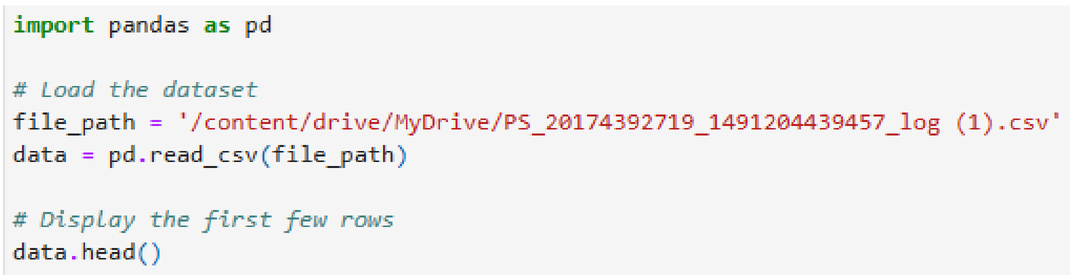

```python
import pandas as pd

# Load the dataset
file_path = '/content/drive/MyDrive/PS_20174392719_1491204439457_log (1).csv'
data = pd.read_csv(file_path)

# Display the first few rows
data.head()
```

Figure 2: synthetic dataset loading

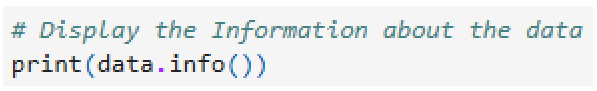
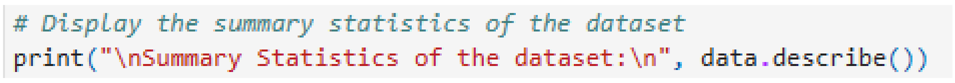

```python
# Display the Information about the data
print(data.info())
```

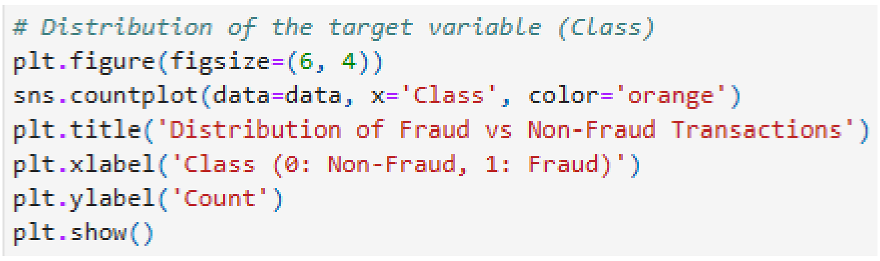
Figure 3: for displaying information



```python
# Display the summary statistics of the dataset
print("\nSummary Statistics of the dataset:\n", data.describe())
```

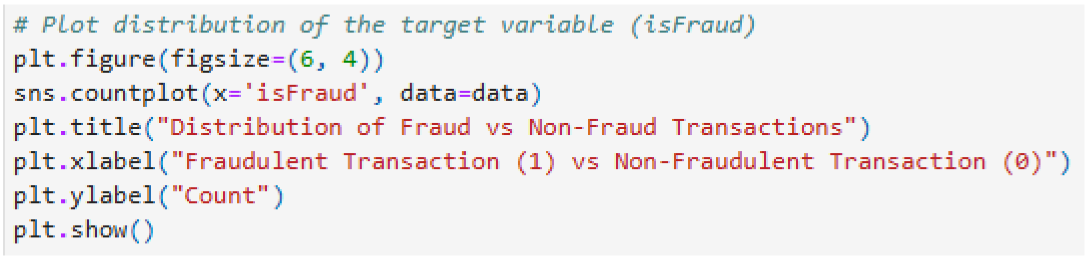
Figure 4: Display summary statistics

2. **Visualize Target Variable:**



```python
# Distribution of the target variable (Class)
plt.figure(figsize=(6, 4))
sns.countplot(data=data, x='Class', color='orange')
plt.title('Distribution of Fraud vs Non-Fraud Transactions')
plt.xlabel('Class (0: Non-Fraud, 1: Fraud)')
plt.ylabel('Count')
plt.show()
```

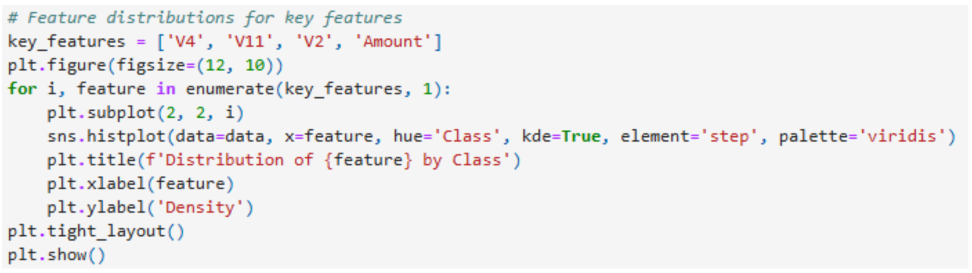
Figure 5: Distribution of Target variable of creditcard data



```python
# Plot distribution of the target variable (isFraud)
plt.figure(figsize=(6, 4))
sns.countplot(x='isFraud', data=data)
plt.title("Distribution of Fraud vs Non-Fraud Transactions")
plt.xlabel("Fraudulent Transaction (1) vs Non-Fraudulent Transaction (0)")
plt.ylabel("Count")
plt.show()
```

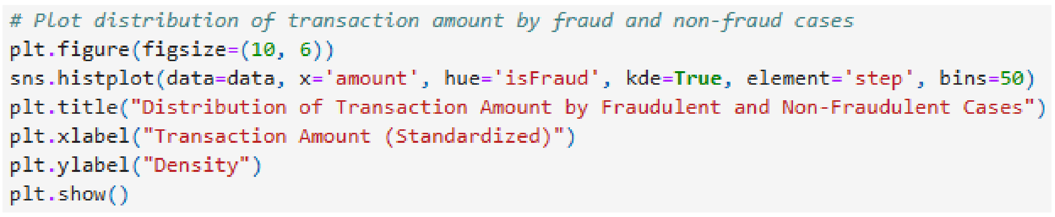
Figure 6: Distribution of Target variable of synthetic financial data

3. **Feature Analysis:**
   - Visualize key features:



```python
# Feature distributions for key features
key_features = ['V4', 'V11', 'V2', 'Amount']
plt.figure(figsize=(12, 10))
for i, feature in enumerate(key_features, 1):
    plt.subplot(2, 2, i)
    sns.histplot(data=data, x=feature, hue='Class', kde=True, element='step', palette='viridis')
    plt.title(f'Distribution of {feature} by Class')
    plt.xlabel(feature)
    plt.ylabel('Density')
plt.tight_layout()
plt.show()
```

Figure 7: Feature Distribution of creditcard data

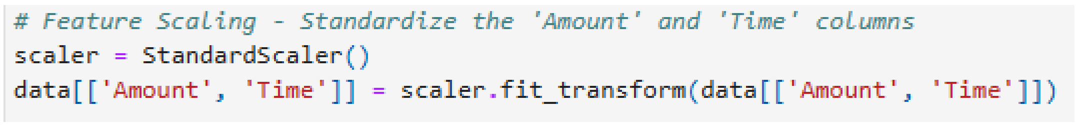


```python
# Plot distribution of transaction amount by fraud and non-fraud cases
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='amount', hue='isFraud', kde=True, element='step', bins=50)
plt.title("Distribution of Transaction Amount by Fraudulent and Non-Fraudulent Cases")
plt.xlabel("Transaction Amount (Standardized)")
plt.ylabel("Density")
plt.show()
```

Figure 8: Feature Distribution of synthetic financial data

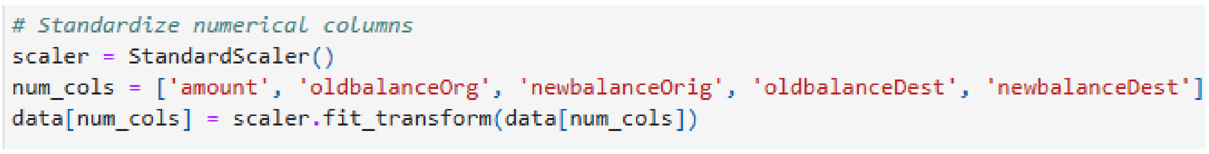
4.2 Data Preprocessing

1. **Feature Scaling:**
    - Standardize numerical features for both datasets:



```python
# Feature Scaling - Standardize the 'Amount' and 'Time' columns
scaler = StandardScaler()
data[['Amount', 'Time']] = scaler.fit_transform(data[['Amount', 'Time']])
```

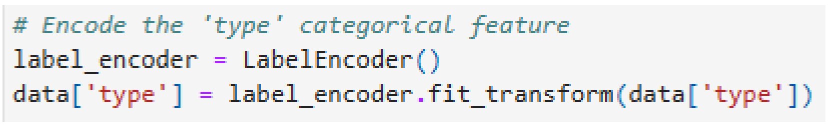
Figure 9: Feature Scaling of creditcard data



```python
# Standardize numerical columns
scaler = StandardScaler()
num_cols = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest']
data[num_cols] = scaler.fit_transform(data[num_cols])
```

Figure 10: Feature Scaling of synthetic financial data

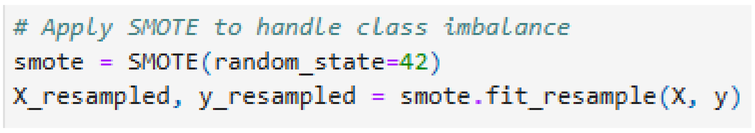
2. **Encode Categorical Features:**
    - Encode transaction types in the synthetic financial dataset:



```python
# Encode the 'type' categorical feature
label_encoder = LabelEncoder()
data['type'] = label_encoder.fit_transform(data['type'])
```

Figure 11: Encoding of categorical features of synthetic data

3. **Address Class Imbalance:**
    - Use SMOTE for oversampling in both datasets:

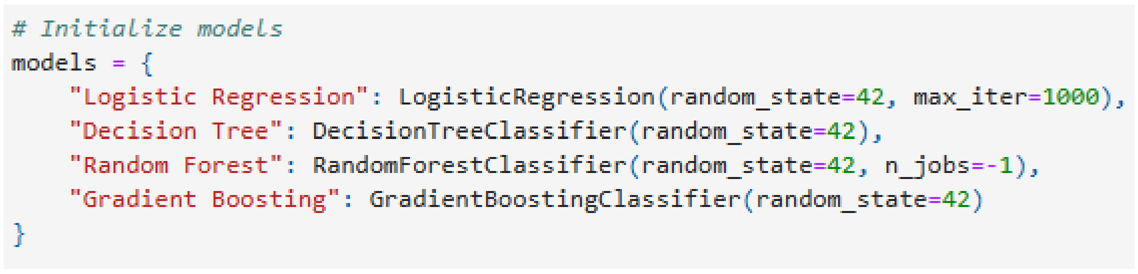


```python
# Apply SMOTE to handle class imbalance
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

Figure 12: Class imbalance for both datasets

4.3 Model Training and Evaluation

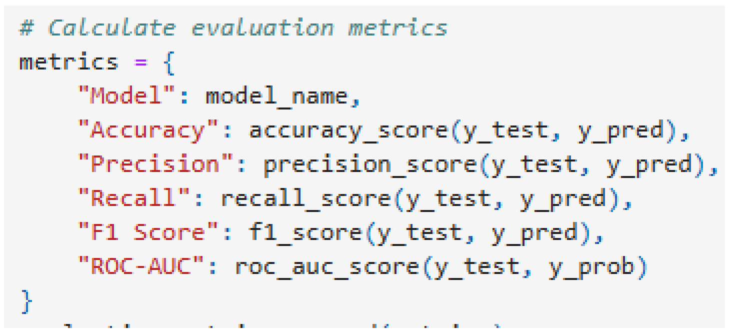
1. **Machine Learning Models:**
   - Train models on both datasets:



```python
# Initialize models
models = {
    "Logistic Regression": LogisticRegression(random_state=42, max_iter=1000),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42, n_jobs=-1),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42)
}
```

Figure 13: Models trained

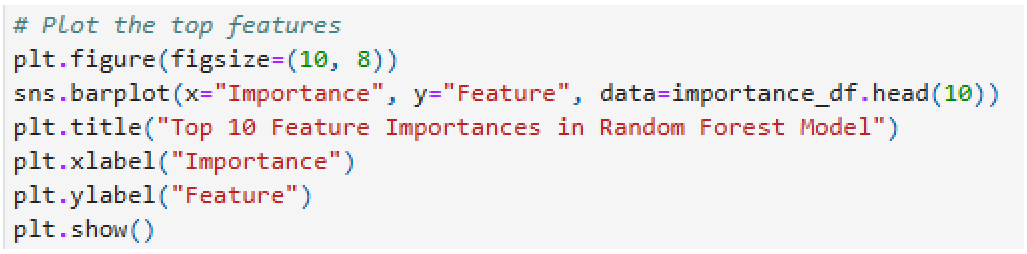
2. **Evaluation Metrics:**
   - Compute metrics (Accuracy, Precision, Recall, F1 Score, ROC-AUC):



```python
# Calculate evaluation metrics
metrics = {
    "Model": model_name,
    "Accuracy": accuracy_score(y_test, y_pred),
    "Precision": precision_score(y_test, y_pred),
    "Recall": recall_score(y_test, y_pred),
    "F1 Score": f1_score(y_test, y_pred),
    "ROC-AUC": roc_auc_score(y_test, y_prob)
}
```

Figure 14: Evaluation Metrics

3. **Feature Importance:**
   - Visualize feature importance for Random Forest:

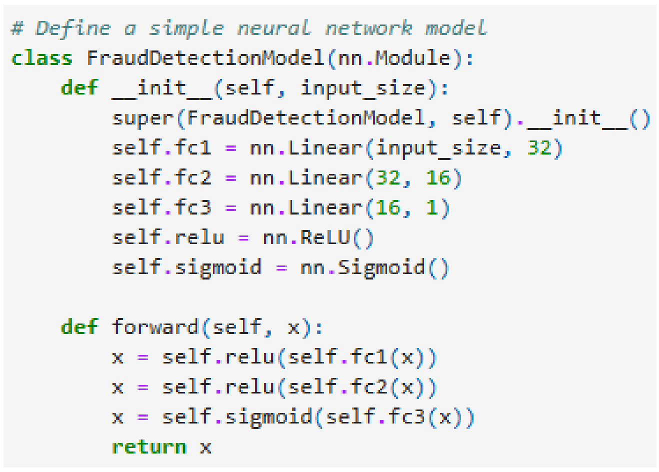


```
# Plot the top features
plt.figure(figsize=(10, 8))
sns.barplot(x="Importance", y="Feature", data=importance_df.head(10))
plt.title("Top 10 Feature Importances in Random Forest Model")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```

Figure 15: Feature Importance

4.4 Federated Learning

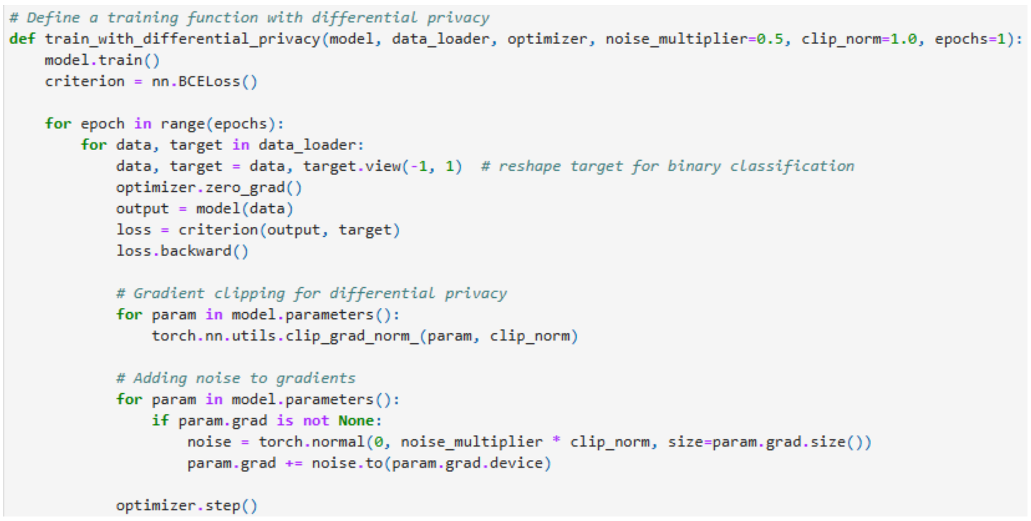
1. **Define Neural Network:**
    - Implement a simple fraud detection model:



```
# Define a simple neural network model
class FraudDetectionModel(nn.Module):
    def __init__(self, input_size):
        super(FraudDetectionModel, self).__init__()
        self.fc1 = nn.Linear(input_size, 32)
        self.fc2 = nn.Linear(32, 16)
        self.fc3 = nn.Linear(16, 1)
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.sigmoid(self.fc3(x))
        return x
```

Figure 16: Neural Network

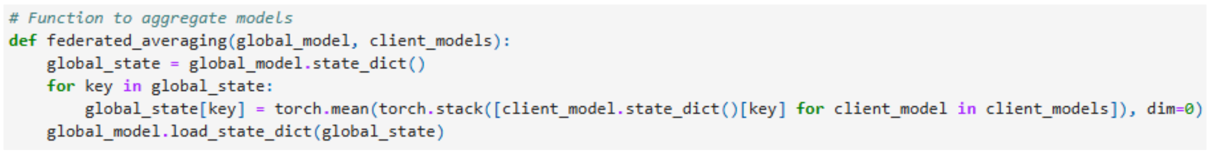
2. **Federated Training with Differential Privacy:**
    - Train client models with privacy-preserving techniques:



```python
# Define a training function with differential privacy
def train_with_differential_privacy(model, data_loader, optimizer, noise_multiplier=0.5, clip_norm=1.0, epochs=1):
    model.train()
    criterion = nn.BCELoss()

    for epoch in range(epochs):
        for data, target in data_loader:
            data, target = data, target.view(-1, 1)  # reshape target for binary classification
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, target)
            loss.backward()

            # Gradient clipping for differential privacy
            for param in model.parameters():
                torch.nn.utils.clip_grad_norm_(param, clip_norm)

            # Adding noise to gradients
            for param in model.parameters():
                if param.grad is not None:
                    noise = torch.normal(0, noise_multiplier * clip_norm, size=param.grad.size())
                    param.grad += noise.to(param.grad.device)

            optimizer.step()
```

Figure 17: Federated Training with Differential Privacy

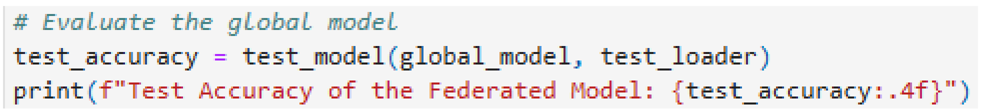
3. **Federated Averaging:**
    - Aggregate client models:



```python
# Function to aggregate models
def federated_averaging(global_model, client_models):
    global_state = global_model.state_dict()
    for key in global_state:
        global_state[key] = torch.mean(torch.stack([client_model.state_dict()[key] for client_model in client_models]), dim=0)
    global_model.load_state_dict(global_state)
```

Figure 18: Aggregrate client models

4. **Model Evaluation:**
    - Test the global model:



```python
# Evaluate the global model
test_accuracy = test_model(global_model, test_loader)
print(f"Test Accuracy of the Federated Model: {test_accuracy:.4f}")
```

Figure 19: model evaluation

References

Rahmany, M., Zin, A.M. and Sundararajan, E.A., 2020. Comparing tools provided by python and r for exploratory data analysis. IJISCS (International Journal of Information System and Computer Science), 4(3), pp.131-142.

Reina, G.A., Gruzdev, A., Foley, P., Perepelkina, O., Sharma, M., Davidyuk, I., Trushkin, I., Radionov, M., Mokrov, A., Agapov, D. and Martin, J., 2021. OpenFL: An open-source framework for Federated Learning. arXiv preprint arXiv:2105.06413.

Sahoo, K., Samal, A.K., Pramanik, J. and Pani, S.K., 2019. Exploratory data analysis using Python. International Journal of Innovative Technology and Exploring Engineering, 8(12), pp.4727-4735.