Configuration Manual

MSc Research Project
MSc Data Analytics

Sricharan Patlori
Student ID: 23231769

School of Computing
National College of Ireland

Supervisor: Prof. Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sricharan Patlori		
Student ID:	23231769		
Programme:	Msc Data Analytics	**Year:**	2024
Module:	Msc Research Project		
Lecturer:	Prof. Jorge Basilio		
Submission Due Date:	29/01/2025		
Project Title:	Privacy-Preserving Predictive Analytics in Healthcare Using Federated Learning and Deep Learning Models		
Word Count:	647	**Page Count:**	5

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: Sricharan Patlori

Date: 29/01/2025

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

Sricharan Patlori
23231769

1 Overview

The provided script implements a **Federated Learning system** to predict a patient's **Length of Stay (LoS)** in a hospital using **TensorFlow Federated (TFF)**. Federated Learning enables collaborative training of machine learning models across multiple clients without sharing their raw data. The target variable, LengthOfStay, is categorized into bins of three days, making it suitable for regression-based predictive modelling. This project handles the preprocessing, simulation of federated data distribution, federated training using TensorFlow Federated, and evaluation of the trained model.

2 System Requirements

Hardware Requirements

- Minimum 8 GB of RAM
- A CPU supporting AVX instructions (TensorFlow requirement)
- GPU optional

Software Requirements

- Python 3.9 or higher
- OS: Windows, macOS, Linux
- TensorFlow (CPU or GPU version) and TensorFlow Federated

3 Installations required

- Install Python and pip if not already installed.
- Install required Python libraries:

  pip install tensorflow tensorflow-federated scikit-learn pandas numpy openpyxl

- **tensorflow**: Machine learning library for neural networks
- **tensorflow-federated**: Framework for federated learning
- **scikit-learn**: Data preprocessing and utilities
- **pandas and numpy**: Data manipulation
- **openpyxl**: Read .xlsx files

4 Steps to execute files

- **Prepare the Dataset**
  - Place dataset (patient_data.xlsx) in the same directory as the script.
  - Execute the Python script:

    python codefed.py

- **Logging**
  - All outputs, including preprocessing steps, metrics during training, and evaluation results, are logged to federated_learning.log for analysis.
- **Evaluate Model**
  - After training, the script evaluates the model's performance on a test dataset and logs key metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE)

5 Main parts of code

1) Load and Preprocess Data

```
data = pd.read_excel('patient_data.xlsx')
logging.info(f"Columns in the dataset: {list(data.columns)}")

# Fill missing values
imputer = SimpleImputer(strategy='most_frequent')
data = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)

# Encode categorical variables
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le
```

2) Train-Test Split

```
# Split features and target
X = data.drop('LengthOfStay', axis=1)
y = data['LengthOfStay']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logging.info("Data split into training and testing sets.")
```

3) Simulating Federated Data Distribution

```
num_clients = 5
client_data = []
for i in range(num_clients):
    client_X = X_train.sample(frac=1/num_clients, random_state=i)
    client_y = y_train[client_X.index]
    client_data.append((client_X.values, client_y.values))
```

4) Neural Network Model Definition

```
# Step 3: Define Neural Network Model
def create_model():
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(16, activation='relu'),
        tf.keras.layers.Dense(1)
    ])
    return model
```

5) Federated Learning Setup with TensorFlow Federated (TFF)

```
def model_fn():
    keras_model = create_model()
    return tff.learning.from_keras_model(
        keras_model,
        input_spec=(tf.TensorSpec(shape=(None, X_train.shape[1]), dtype=tf.float32),
                    tf.TensorSpec(shape=(None,), dtype=tf.float32)),
        loss=tf.keras.losses.MeanSquaredError(),
        metrics=[tf.keras.metrics.MeanAbsoluteError()]
    )

iterative_process = tff.learning.build_federated_averaging_process(
    model_fn,
    client_optimizer_fn=lambda: tf.keras.optimizers.Adam(learning_rate=0.01),
    server_optimizer_fn=lambda: tf.keras.optimizers.SGD(learning_rate=1.0)
)
```

6) Federated Model Training

```
NUM_ROUNDS = 20
for round_num in range(NUM_ROUNDS):
    state, metrics = iterative_process.next(state, federated_data)
    logging.info(f"Round {round_num + 1}: {metrics}")
```

7) Model Evaluation

```
global_model.set_weights([
    tf.convert_to_tensor(w, dtype=tf.float32) for w in state.model.trainable
])

global_model.compile(optimizer='adam',
                     loss=tf.keras.losses.MeanSquaredError(),
                     metrics=[tf.keras.metrics.MeanAbsoluteError()])

test_loss, test_mae = global_model.evaluate(X_test, y_test, verbose=0)
test_rmse = np.sqrt(test_loss)

logging.info(f"Test MAE: {test_mae:.4f}")
logging.info(f"Test MSE: {test_loss:.4f}")
logging.info(f"Test RMSE: {test_rmse:.4f}")
```

References

Gosselin, Rémi, Loïc Vieu, Faiza Loukil, and Alexandre Benoit. "Privacy and security in federated learning: A survey." *Applied Sciences* 12, no. 19 (2022): 9901.

Hu, Kai, Sheng Gong, Qi Zhang, Chaowen Seng, Min Xia, and Shanshan Jiang. "An overview of implementing security and privacy in federated learning." *Artificial Intelligence Review* 57, no. 8 (2024): 204.

Thota, Shashi, Vinay Kumar Reddy Vangoor, Amit Kumar Reddy, and Chetan Sasidhar Ravi. "Federated Learning: Privacy-Preserving Collaborative Machine Learning." *Distributed Learning and Broad Applications in Scientific Research* 5 (2019): 168-190.

https://www.tensorflow.org/federated