Python Script Documentation: Tensorflow DNN on Tom Brady's Superbowl Win Prediction

1 Introduction

This document provides detailed documentation for a Python script that trains and evaluates a TensorFlow deep neural network (DNN) model to predict Superbowl wins using NFL data. The main steps include loading the data, processing features, training the model, and evaluating its performance using several metrics. This documentation covers:

- A breakdown of the code.
- A layered explanation of loops and nested operations.
- A flowchart visualizing the structure of the script.
- Libraries used and error handling.

2 Libraries Used

The script uses the following libraries:

- `os`: For handling file paths and directories.
- `pandas`: For loading and manipulating the dataset.
- `sklearn.preprocessing.StandardScaler`: To standardize feature data.
- `sklearn.metrics`: For calculating evaluation metrics like accuracy, precision, recall, F1 score, and generating confusion matrices.

- `matplotlib.pyplot`: To create visualizations, particularly the confusion matrix.
- `tensorflow`: For building, training, and evaluating the deep neural network.

3 Main Functionality

The script's main tasks include:

- Loading NFL data from a CSV file.
- Preprocessing the data (removing missing values and standardizing features).
- Defining and training a deep neural network using TensorFlow.
- Evaluating the trained model on test data.
- Saving confusion matrix plots and evaluation metrics to a specified directory.

4 Code Breakdown

4.1 Function: `dnn`

The `dnn` function trains and evaluates the deep neural network for a specific range of years.

- **Input Parameters:**
  - `train_year_start`: Start year of the training period.
  - `train_year_end`: End year of the training period.
  - `features`: DataFrame containing feature variables.
  - `target`: Series representing the target variable (Superbowl Win).
  - `save_directory`: Path to save model outputs (metrics and plots).
- **Steps:**

  - Split the data into training and testing sets based on the year range.
  - Standardize the feature data using `StandardScaler`.
  - Build a Sequential TensorFlow model with three hidden layers and an output layer.
  - Compile the model with the Adam optimizer and binary cross-entropy loss.
  - Train the model for 100 epochs.
  - Evaluate the model on the test data and calculate metrics (accuracy, precision, recall, F1 score).
  - Generate and save a confusion matrix plot and metrics file.

4.2 Main Function

The `main` function orchestrates the workflow of data loading, processing, and model training.

- Define the paths for data loading and result saving.
- Load NFL data from a CSV file into a pandas DataFrame.
- Set the DataFrame index to `Season` for year-based operations.
- Separate features from the target variable (Superbowl Win).
- Remove columns with missing values from the features.
- Call the `dnn` function for two different year ranges (2002–2009 and 2002–2019).

5 Explanation of Loops and Nested Operations

The `dnn` function contains a loop during the model training phase:

- **Training Loop**:

- `model.fit(features_train, target_train, epochs=100, batch_size=32)`: This function trains the model for 100 epochs with a batch size of 32. Inside each epoch:
  - The model performs forward propagation through the layers.
  - The loss is calculated based on the predictions and true values.
  - The Adam optimizer adjusts the weights to minimize the loss.

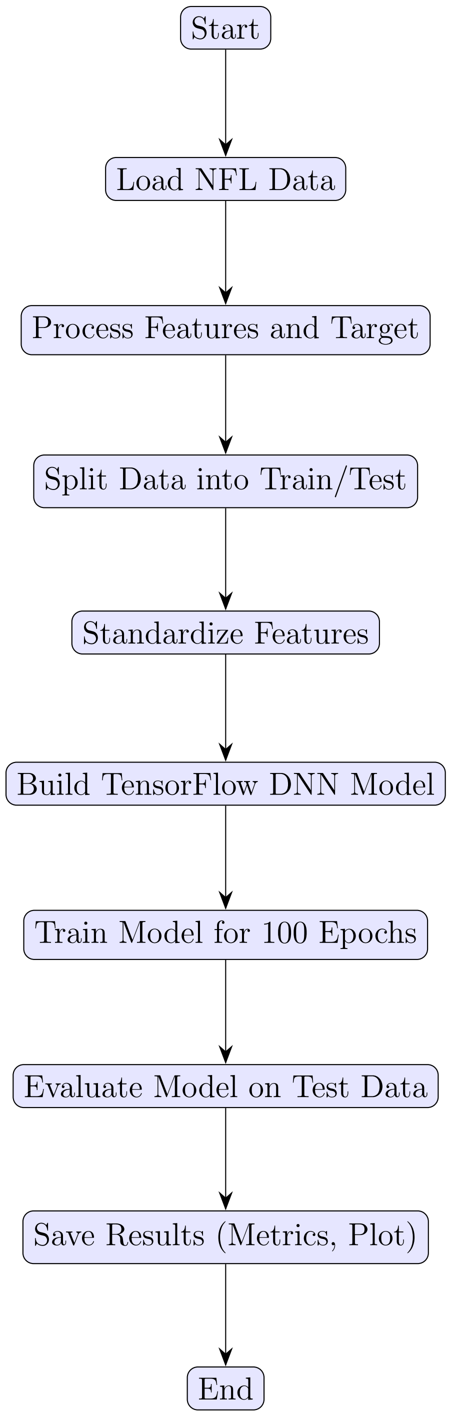
6 Error Handling

Although the script does not explicitly handle errors, potential improvements include:

- Checking for the existence of the data file before attempting to load it.
- Verifying that directories for saving results exist and creating them if necessary.
- Adding try-except blocks to catch potential errors during model training, evaluation, and file writing.

7 Flowchart

The following flowchart visualizes the structure of the Python script, created using the TikZ package:


Start
Load NFL Data
Process Features and Target
Split Data into Train/Test
Standardize Features
Build TensorFlow DNN Model
Train Model for 100 Epochs
Evaluate Model on Test Data
Save Results (Metrics, Plot)
End

8 Conclusion

This Python script demonstrates how to build and train a deep neural network (DNN) model using TensorFlow to predict Superbowl wins based on NFL data. It preprocesses the data, trains the model for specified year ranges, and evaluates the model's performance using accuracy, precision, recall, and F1 score. The script also visualizes the results using confusion matrices and saves the metrics for further analysis.