User Manual for AI_FINAL_CODE

This document provides a guide to the uploaded Jupyter Notebook, detailing its structure, purpose, and instructions for use.

Table of Contents

---

1. Overview

The notebook implements a machine learning pipeline for predictive modeling, combining classical algorithms and advanced neural networks. It is suited for tasks such as classification, anomaly detection, or similar predictive applications.

---

2. Prerequisites

Ensure the following tools and libraries are installed:
- Python 3.x
- Jupyter Notebook/Google Colab
- Libraries: `numpy`, `pandas`, `scikit-learn`, `matplotlib`, `seaborn`, `tensorflow`, `catboost`, `imblearn`

To install missing libraries, run:

“! pip install numpy pandas scikit-learn matplotlib seaborn tensorflow catboost imbalanced-learn"

Why these libraries are installed:

- numpy: For efficient numerical computations.
- pandas: For data manipulation and analysis.
- scikit-learn: For classical machine learning algorithms and preprocessing utilities.
- matplotlib and seaborn: For creating visualizations to analyze data and model performance.
- tensorflow: For building and training deep learning models such as ANN, CNN, and LSTM.
- catboost: For leveraging gradient boosting with categorical features.
- imbalanced-learn (`imblearn`): For handling imbalanced datasets using techniques like SMOTE.

---

3. Workflow Phases

Phase 1: Data Loading and Exploration

Purpose: Upload and inspect the dataset.

- Code: Use `google.colab` for file uploads (specific to Google Colab) or `os` for local file management.
- Steps:
  1. Upload the dataset using the `files.upload()` function.
  2. Use `os.listdir()` to confirm successful upload.
  3. Explore the dataset structure using `pandas` methods (`head()`, `info()`, `describe()`).

Phase 2: Data Preprocessing and Feature Engineering

Purpose: Clean data and prepare features for modeling.

- Key Tasks:
  - Handle missing values (`pandas.fillna()` or `dropna()`).
  - Encode categorical variables using `LabelEncoder` or `OneHotEncoder`.
  - Normalize features using `StandardScaler`.
  - Address class imbalances with `SMOTE`.

Phase 3: Feature Selection and Dimensionality Reduction

Purpose: Improve model performance by reducing irrelevant or redundant features.

- Techniques Used:
  - Random Forest for feature importance.
  - Recursive Feature Elimination (RFE).
  - Dimensionality reduction (e.g., PCA).
- Outputs: A refined dataset (`X_selected`, `y_final`) for modeling.

Phase 4: Model Development

Purpose: Train various models, from classical to deep learning.

Classical Models

- Logistic Regression, Decision Trees, Random Forests, etc.
- Use scikit-learn functions for training and validation.

Deep Learning Models

- Artificial Neural Network (ANN): Constructed using `tensorflow.keras.Sequential` with dense layers and dropout.
- Convolutional Neural Network (CNN): Adapted for tabular data by reshaping input.
- Long Short-Term Memory (LSTM): Suitable for time-series or sequential data.

Transformer-Based Models

- Implements state-of-the-art architectures for real-time tasks.
- Includes additional preprocessing for transformer compatibility.

Phase 5: Model Evaluation

Purpose: Compare model performance using various metrics.

- Metrics Evaluated:
  - ROC-AUC and precision-recall curves.
  - Confusion matrices.
  - Accuracy, precision, recall, and F1-score.
- Visualization: Generate comparative plots using `matplotlib` and `seaborn`.

Phase 6: Model Summary and Deployment

Purpose: Summarize findings and prepare the best model for deployment.

- Steps:
  1. Consolidate results into a dataframe for comparison.
  2. Highlight the best-performing model based on metrics.
  3. Save the model for deployment using `joblib` or `pickle`.

---

4. Additional Notes

- Environment: The code includes Google Colab-specific commands. For local execution, replace these with equivalent local file handling functions.
- Customization: Modify preprocessing and model parameters to suit your dataset and objectives.
- Troubleshooting:
  - Check library versions if you encounter compatibility issues.
  - Ensure sufficient computational resources for deep learning models.