Documentation for ANN - Car Sales Price Prediction

Overview

This Jupyter Notebook demonstrates the implementation of an Artificial Neural Network (ANN) to predict car sales prices. The pipeline includes data preprocessing, model building, training, evaluation, and predictions on new data.

Detailed Structure and Workflow

1. Introduction and Imports

- Essential libraries such as `pandas`, `numpy`, and machine learning frameworks like TensorFlow/Keras are imported.
- Purpose: Enable data manipulation, ANN creation, and visualization.

2. Dataset Loading and Exploration

- Dataset: `car_purchasing.csv`.
- Key Actions:
    - Load the dataset using `pandas.read_csv`.
    - Initial exploration with `data.head()` to inspect columns and sample data.
    - Understand key features, target variable, and overall dataset structure.

3. Data Preprocessing

- **Splitting Dataset**: Separates features and target variables.
- **Handling Missing Values**: Identifies and resolves incomplete data.
- **Encoding**: Converts categorical data (if any) into numerical values using one-hot encoding or label encoding.
- **Feature Selection**: Ensures only relevant features are included for modeling.

4. Feature Scaling

- Normalizes or standardizes feature values using techniques like Z-score normalization.
- Tools: `StandardScaler` or similar modules from `sklearn`.

5. Model Initialization

- **Architecture**:

    - Input Layer: Matches the number of features.
    - Hidden Layers: Includes one or more layers with specified neuron counts and activation functions like ReLU.
    - Output Layer: Single neuron for regression tasks.
- **Initialization**:
    - Use Keras Sequential API to define layers.
    - Dropout layers (if applicable) to prevent overfitting.

6. Model Compilation and Training

- **Compilation**:
    - Loss Function: Mean Squared Error (MSE) for regression.
    - Optimizer: Adam (adaptive learning).
- **Training**:
    - Number of Epochs: Typically 50-200 based on data size.
    - Batch Size: Defines the number of samples per gradient update.
    - Monitors training and validation loss over epochs.

7. Evaluation and Results

- **Metrics**:
    - Computes MSE, MAE (Mean Absolute Error), or R-squared scores.
    - Analyzes overfitting or underfitting via loss curves.
- **Visualizations**:
    - Loss vs. Epochs plot.
    - Predicted vs. Actual values comparison.

8. Model Prediction

- Applies the trained model to unseen data.
- Outputs predictions for the target variable.
- Compares results with actual values for error analysis.

9. Conclusion and Future Scope

- **Findings**:
    - Highlights model accuracy and observed limitations.
    - Discusses the impact of preprocessing and ANN structure.
- **Improvements**:
    - Suggests hyperparameter tuning (e.g., neurons, activation functions).
    - Explores alternative architectures (e.g., deeper networks).

Key Features of the Notebook

- **Complete Workflow**: Provides an end-to-end implementation of ANN for regression.
- **Hands-On Learning**: Includes explanatory markdown cells for beginners.
- **Reproducibility**: Detailed steps allow replication and experimentation.

Additional Notes

- **Dependencies**:
  - Python libraries: `pandas`, `numpy`, `sklearn`, TensorFlow/Keras.
  - Ensure the dataset file `car_purchasing.csv` is in the same directory.
- **Hardware Considerations**:
  - For large datasets or models, GPU acceleration is recommended.

This notebook serves as a comprehensive guide for implementing and understanding ANN models in regression tasks, particularly car sales price prediction.