

CS-1390 / PHY-1390-1: Introduction to Machine Learning
Monsoon 2024
Assignment 1

Datasets

We have provided two datasets for this assignment:

1. Online Payment Fraud

- **Type:** **Decision Tree Classification Task**
- **Description:**
 - step: represents a unit of time where 1 step equals 1 hour
 - type: type of online transaction
 - amount: the amount of the transaction
 - nameOrig: customer starting the transaction
 - oldbalanceOrig: balance before the transaction
 - newbalanceOrig: balance after the transaction
 - nameDest: recipient of the transaction
 - oldbalanceDest: initial balance of recipient before the transaction
 - newbalanceDest: the new balance of recipient after the transaction
 - isFraud: fraud transaction
- **Training Data:** *[Download link will be provided via Classroom]*
- **Target Feature:** isFraud
- **Test Data:** Will not be provided to students.

2. Fuel Consumption Dataset

- **Type:** **Regression Task**
- **Description:** Includes data on fuel consumption.
- **Training Data:** *[Download link will be provided via Classroom]*
- **Target Feature:** Fuel Consumption
- **Test Data:** Will not be provided to students.

Instructions

1. Implement Algorithms from Scratch

- **No Machine Learning Libraries:** You must code all algorithms manually. The use of machine learning libraries such as `scikit-learn`, `TensorFlow`, `Keras`, `PyTorch`, etc., is strictly prohibited.
- **Allowed Libraries:** You may use libraries for basic data handling and mathematical computations like `NumPy` and `Pandas`.

2. Project Structure and Version Control

Directory Structure

Organize your project using the following directory structure for each task:

```
FirstName_LastName_A1/
├── report.pdf
├── regression_task/
│   ├── data/
│   │   └── training_data.csv
│   ├── notebooks/
│   │   └── data_exploration.ipynb
│   ├── models/
│   │   ├── regression_model1.pkl
│   │   ├── regression_model2.pkl
│   │   ├── regression_model3.pkl
│   │   └── regression_model_final.pkl
│   ├── src/
│   │   ├── data_preprocessing.py
│   │   ├── train_model.py
│   │   └── predict.py
│   ├── results/
│   │   ├── train_metrics.txt
│   │   └── train_predictions.csv
│   ├── requirements.txt
│   └── .git/ (Git repository for regression\_task)
└── decision_tree_task/
    ├── data/
    │   └── training_data.csv
    ├── notebooks/
    │   └── data_exploration.ipynb
    ├── models/
    │   ├── decision_tree_model1.pkl
    │   ├── decision_tree_model2.pkl
    │   └── decision_tree_model_final.pkl
    ├── src/
    │   ├── data_preprocessing.py
    │   ├── train_model.py
    │   └── predict.py
    ├── results/
    │   ├── train_metrics.txt
    │   └── train_predictions.csv
    ├── requirements.txt
    └── .git/ (Git repository for decision\_tree\_task)
```

Parent Directories: The two main tasks are separated into `regression_task` and `decision_tree_task`, each with the same internal structure.

Model Saving:

- Save your trained models in the `models/` directory.
- Models should be saved using Python's `pickle` module.

Version Control with Git

- **Separate Repositories:**
 - Initialize a separate Git repository inside each task directory (`regression_task/.git` and `decision_tree_task/.git`).
- **Final Code:**
 - The final, polished code should be in the `main` branch of each repository.
- **Commit Messages:**
 - Write clear and descriptive commit messages that reflect the changes made.
- **Commit History:**
 - We will review your commit history to assess the time and effort spent on experimentation.

3. Model Training, Saving, and Evaluation

- **Data Preprocessing:**
 - Handle missing values, encode categorical variables, and perform any necessary data preprocessing.
 - Document these steps in your code and report.
- **Algorithm Implementation:**
 - **Regression Task:** Implement a suitable regression algorithm.
 - **Classification Task:** Implement a Decision Tree classifier.
- **Training:**
 - Train your models using the training data provided.
 - Save the trained models in the specified format within the `models/` directory.

4. Code Requirements

- **Modularity:**
 - Organize your code with functions and classes as appropriate.
- **Documentation:**
 - **Docstrings:** Include docstrings for all modules, classes, and functions.
 - **Comments:** Use inline comments to explain complex sections of code.

6. Evaluation Script (`predict.py`) and Standard Output Structure

Purpose:

- Loads the saved model and evaluates it on a dataset and generates predictions.
- Outputs the evaluation metrics in a standard structure in the `metrics.txt` file.

Usage:

```
python src/predict.py --model_path models/regression_model_final.pkl  
--data_path data/train_data.csv --metrics_output_path  
results/train_metrics.txt --predictions_output_path  
results/train_predictions.csv
```

Arguments:

- `--model_path`: Path to the saved model file.
- `--data_path`: Path to the data CSV file that includes features and true labels.
- `--metrics_output_path`: Path where the evaluation metrics will be saved.
- `--predictions_output_path`: Path where the predictions will be saved.

Standard Structure for `predictions.csv`

- **Single column csv file with the predictions. No heading should be provided to the csv. First row should be the first prediction and so on.**

Standard Structure for `metrics.txt`

For **Regression Task**, the `metrics.txt` file should have the following format:

```
Regression Metrics:  
Mean Squared Error (MSE): <value>  
Root Mean Squared Error (RMSE): <value>  
R-squared (R2) Score: <value>
```

For **Classification Task**, the `metrics.txt` file should have the following format:

```
Classification Metrics:  
Accuracy: <value>  
Precision: <value>  
Recall: <value>  
F1-Score: <value>  
Confusion Matrix:  
[[TN, FP],  
 [FN, TP]]
```

Formatting Guidelines:

- Begin with the task header: **Regression Metrics:** or **Classification Metrics:**.
- List the metrics in the exact order specified.
- Use the exact labels as shown (e.g., **Mean Squared Error (MSE):**).
- Values should be numerical and rounded to two decimal places.
- The confusion matrix should be presented as an array.

Report

- **Format:**
 - Submit a written report in PDF format named **report.pdf**.
- **Content:**
 - **Introduction:** Briefly describe the problem and your approach.
 - **Methodology:** Explain the algorithms and techniques you used.
 - **Experimentation:**
 - Discuss the different experimentations you conducted.
 - Explain any hyperparameter tuning or model variations.
 - **Results:**
 - Present the evaluation metrics and discuss the performance of your models.
 - Include any relevant tables, graphs, or charts.
 - **Challenges:**
 - Discuss any difficulties faced and how you overcame them.
 - **Conclusion:**
 - Summarize your findings and suggest possible improvements.
 - **References:** Include any references used.

Grading Criteria

- **Detailed Report.**
- **Code Execution:**
 - We will run your **predict.py** scripts using an unseen test dataset.
- **Result Comparison:**
 - Your model predictions in the **test_predictions.csv** file will be compared against the actual target values.
- **Commit History Review:**
 - We will review your Git commit history to assess the time and effort spent on experimentation.
- **Other:**
 - **Algorithm Implementation:** Correctness and efficiency of your algorithms.
 - **Project Structure:** Adherence to the specified directory structure.
 - **Model Saving and Loading:** Ability to save and load models correctly.

- **Evaluation Script:** Correct implementation of the `predict.py` script to generate specified metrics and predictions in the standard format.
- **Version Control Usage:**
 - Effective use of Git for different experimentations.
 - Commit frequency and messages reflecting consistent effort.
- **Code Quality:** Readability, modularity, and documentation.
- **Model Performance:** Ability of the code to generalize to new, unseen data.
- **Report Quality:** Clarity, depth, and insights provided in your report.
- **Compliance:** Following instructions, including the prohibition on machine learning libraries.

Submission Guidelines

- Submit a zip file of the entire directory structure: `FirstName_LastName_A1.zip`. **[Do not include the csv files in the data directory.]**
- Any error when we run the code will result in a 0 in the entire assignment.
- We will follow the following measures with your code:
 1. Code similarity check with your peers and the internet.
 2. AI detection tests. Use of any AI tool such as Chatgpt, Claude, etc. is strictly prohibited.
- Failure to pass these tests will result in a 0 in the assignment as well as further severe consequences under discretion of the professor.