

System Requirements Specification Index

For Machine learning Algorithm No 1

1.0

Machine Learning Lab: Linear Regression and Logistic Regression

Overview

This lab focuses on implementing two fundamental machine learning algorithms:

1. **Linear Regression** for predicting continuous values (Auto MPG dataset)
2. **Logistic Regression** for binary classification (Titanic Survival dataset)

You will implement code with TODOs that provide guidance on how to complete each function. The goal is to understand the core concepts of data preprocessing, model training, and evaluation for regression and classification tasks.

Datasets

1. Auto MPG Dataset (`auto-mpg.csv`)

This dataset contains information about various automobiles, including their fuel consumption (MPG) and other attributes.

Features:

- `cylinders`: Number of cylinders in the engine
- `displacement`: Engine displacement (in cubic inches)
- `horsepower`: Engine horsepower
- `weight`: Vehicle weight (in pounds)
- `acceleration`: Time to accelerate from 0 to 60 mph (in seconds)
- `model-year`: Model year (modulo 100)
- `origin`: Origin of car (1: American, 2: European, 3: Japanese)
- `car-name`: Car name

Target Variable:

- `mpg`: Fuel efficiency measured in miles per gallon

2. Titanic Dataset (`titanic.csv`)

This dataset contains information about passengers aboard the Titanic, including whether they survived or not.

Features:

- `pclass`: Passenger class (1 = 1st class, 2 = 2nd class, 3 = 3rd class)
- `sex`: Gender of passenger
- `age`: Age of passenger
- `sibsp`: Number of siblings/spouses aboard
- `parch`: Number of parents/children aboard
- `fare`: Passenger fare
- `embarked`: Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

Target Variable:

- `survived`: Survival status (0 = No, 1 = Yes)

Tasks**Task 1: Linear Regression on Auto MPG Dataset**

You will implement the following functions in `Linear_regression_auto.py`:

1. **`load_and_preprocess(path)`**: Load and clean the dataset
 - Load the CSV file
 - Convert column names to lowercase and strip whitespace
 - Drop rows with missing values
 - Print confirmation message
2. **`show_key_stats(df)`**: Display key statistics
 - Calculate mean displacement
 - Find minimum horsepower
 - Print these statistics with appropriate formatting
3. **`prepare_data(df, features, target)`**: Prepare data for model training
 - Extract features and target
 - Scale features using StandardScaler
 - Split data into training and testing sets
 - Print confirmation message
4. **`train_and_save_model(X_train, y_train, model_path)`**: Train and save the model
 - Create a LinearRegression model
 - Fit the model with training data
 - Save the model using joblib
 - Print confirmation message
5. **`evaluate_model(model, X_test, y_test)`**: Evaluate model performance
 - Generate predictions
 - Calculate mean squared error
 - Print evaluation metrics and sample predictions

Task 2: Logistic Regression on Titanic Dataset

You will implement the following functions in `titanic.py`:

1. **`load_and_prepare_data(path)`**: Load, clean, and encode the dataset
 - Load the CSV file
 - Fill missing values appropriately

- Encode categorical variables
- Print confirmation message
- 2. **explore_data(df)**: Perform exploratory data analysis
 - Calculate maximum fare and standard deviation
 - Print these statistics with appropriate formatting
- 3. **sigmoid_demo()**: Demonstrate the sigmoid function
 - Calculate sigmoid(0)
 - Print the result with appropriate formatting
- 4. **cost_function(y_true, y_pred_prob)** Implement binary cross-entropy loss
 - Add epsilon to avoid log(0)
 - Clip prediction probabilities
 - Calculate binary cross-entropy
- 5. **train_and_evaluate(X_train, y_train, X_test, y_test, path)**: Train and evaluate the model
 - Create a LogisticRegression model
 - Fit the model with training data
 - Save the model using joblib
 - Generate predictions and probabilities
 - Calculate custom cost
 - Print evaluation metrics and sample predictions

Expected Outcomes

After completing the TODOs in both files, you should be able to:

1. Load and preprocess datasets for machine learning
2. Train linear regression and logistic regression models
3. Evaluate model performance using appropriate metrics
4. Understand the core concepts of regression and classification

Testing Your Implementation

The lab includes unit tests to verify your implementation:

python -m unittest discover -s test

The tests will check if your functions work correctly. Initially, all functional tests will fail, but the boundary and exceptional tests should pass. Your goal is to implement the functions so that all tests pass.

Tips for Success

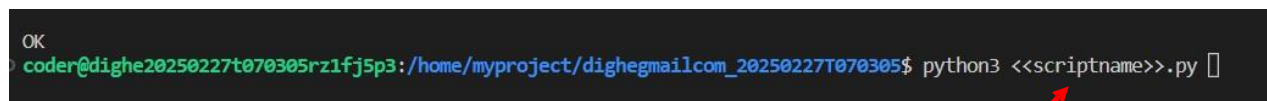
1. Read the TODOs carefully to understand what each function should do
2. Refer to the scikit-learn documentation for details on the models and preprocessing steps
3. Pay attention to the expected output formats, especially for print statements
4. Implement one function at a time and test incrementally
5. Make sure your implementation handles edge cases appropriately

Execution Steps to Follow:

- All actions like build, compile, running application, running test cases will be through Command Terminal.
- To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
- This editor Auto Saves the code
- If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use **CTRL+Shift+B** -command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
- These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
- To setup environment:
You can run the application without importing any packages
- To launch application:
Python3 titanic .py

Python3 Linear_regression_auto.py
- To run Test cases:
python3 -m unittest
- Before Final Submission also, you need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

Screen shot to run the program

A screenshot of a terminal window with a dark background. The prompt is 'coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305\$'. The command entered is 'python3 <<scriptname>>.py'. A red arrow points from the text 'To run the application' below to the '<<scriptname>>.py' part of the command.

```
OK
coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 <<scriptname>>.py
```

To run the application

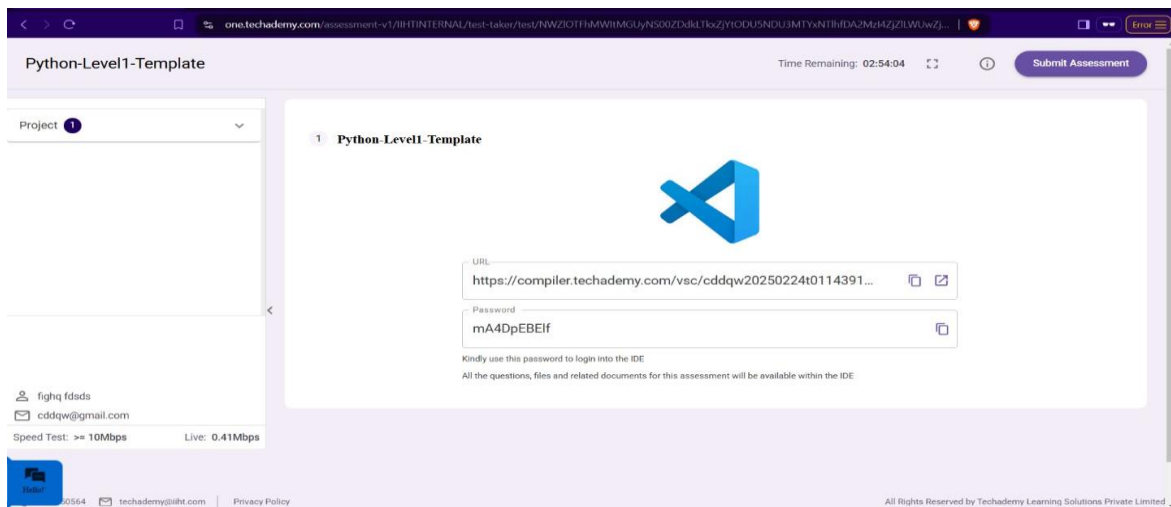
Python3 titanic .py

Python3 Linear_regression_auto.py

```
● coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 -m unittest
TestBoundary = Passed
.TestExceptional = Passed
.TestCalculateTotalDonations = Failed
.TestCalculateTotalStockValue = Failed
.TestCheckFrankWhiteDonated = Failed
```

To run the testcase python3 -

m unittest



- Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.

