**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 guide 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()**: Load and clean the dataset

   - Load the CSV file for this experiment you are given with:**auto-mpg.csv**

   - You will need to convert column names to lowercase and strip whitespace in the dataset

   - if you see rows with missing values, write the code to Drop those values

   - You will need to return data in the form of **Pandas.DataFrame**

Expected columns in the dataset

('mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model-year')

2. **`show\_key\_stats(df)**: Display key statistics

   - using the values of the data frame (passed as a parameter), calculate mean displacement and find minimum horsepower using inbuilt methods

   - return the value of mean and min from the method as a tuple

3. **prepare\_data(df, features, target)**: Prepare data for model training

   - Extract features and target using x(features) and y (target)

   - Scale features using StandardScaler. Which should Initialize a standard scaler

to normalize the feature data (mean = 0, std = 1).

-Fit the scaler on X and transform it to scale the features.

   - Split the scaled feature set and target into:

* 80% for training (X\_train, y\_train)
* 20% for testing (X\_test, y\_test)

    -Return X\_train, X\_test, y\_train, y\_test, scaler as tuple

4. **train\_and\_save\_model(X\_train, y\_train, model\_path)**: Train and save the model

   - Create a linear regression model

   - Train the model using the training data:

* X\_train: The input features (independent variables).
* y\_train: The target (dependent variable).  
  The model learns the relationship between the features and the target variable.

   - Save the model using joblib this will create the .pkl file

   - you are requested to name the model with the name linear\_model.pkl

- Return the model

5. **evaluate\_model(model, X\_test, y\_test)**: Evaluate model performance

* Use the model.predict() method takes in the test features (X\_test) and outputs predicted values (y\_pred), which are the model's estimates of the target variable.
* Use mean\_squared\_error(y\_test, y\_pred) from sklearn.metrics to measure the average squared difference between actual (y\_test) and predicted values.
* Predict the first 10 predictions from the model (y\_pred[:10])

**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

- Loads the dataset from the file specified by the path (default: "titanic.csv") into a pandas DataFrame.

   - Fill missing (null) values in key columns with reasonable defaults:

* df['sex'] = with ('unknown')
* df['embarked'] with ('S')
* df['age'] = with .median()
* df['fare'] = with median())

   -Converts the **sex** and **embarked** columns from string labels (e.g., "male", "female", "S", "C", "Q") into numeric codes,

Then LabelEncoder will assign: sex (Column)

* 'female' → 0
* 'male' → 1
* 'unknown' → 2

embarked Column (e.g., "S", "C", "Q"):

* 'C' → 0
* 'Q' → 1
* 'S' → 2

 - return date frame.

2. **explore\_data(df)**: Perform exploratory data analysis

* Calculate maximum fare and standard deviation.
* Use the built-in function to calculate the **max function standard deviation** of the fare, rounded to 2 decimal places
* Return max\_fare with 4 decimal places and std\_fare with 2 decimal places
* 3. **sigmoid\_demo()**: Demonstrate the sigmoid function

   - Calculate sigmoid(0)

   - **Set input z to 0**:

A close-up of a sign

AI-generated content may be incorrect.

You will write a function that gets the sigmoid value using the function and returns the value up to 4 decimal places

4 **def count\_females(df):**

* Write the function to count the number of females in the dataset
* Assuming 'female' was encoded as 0
* The user needs to return the count from the dataset
* return count

5. **train\_and\_evaluate(X\_train, y\_train, X\_test, y\_test, path)**: Train and evaluate the model

* You will need to initialize the **logistic regression** model.
* Train it using the training data (X\_train for inputs, y\_train for labels).
* Create 1000 iterations to ensure that the model has enough iterations to converge.
* Saves the trained model to disk as a .pkl file (e.g., "titanic\_log\_model.pkl").
* **y\_pred**: Predicts 0 or 1 survived or not
* **y\_pred\_prob**: Predicts **probabilities** that the result, which is the chance of survival

**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

**Testing Your Implementation**

The lab includes unit tests to verify your implementation:

**Python3 -m unittest**

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.

**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.

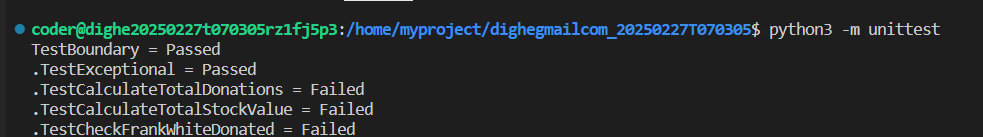
**Screen shot to run the program**

**To run the application**

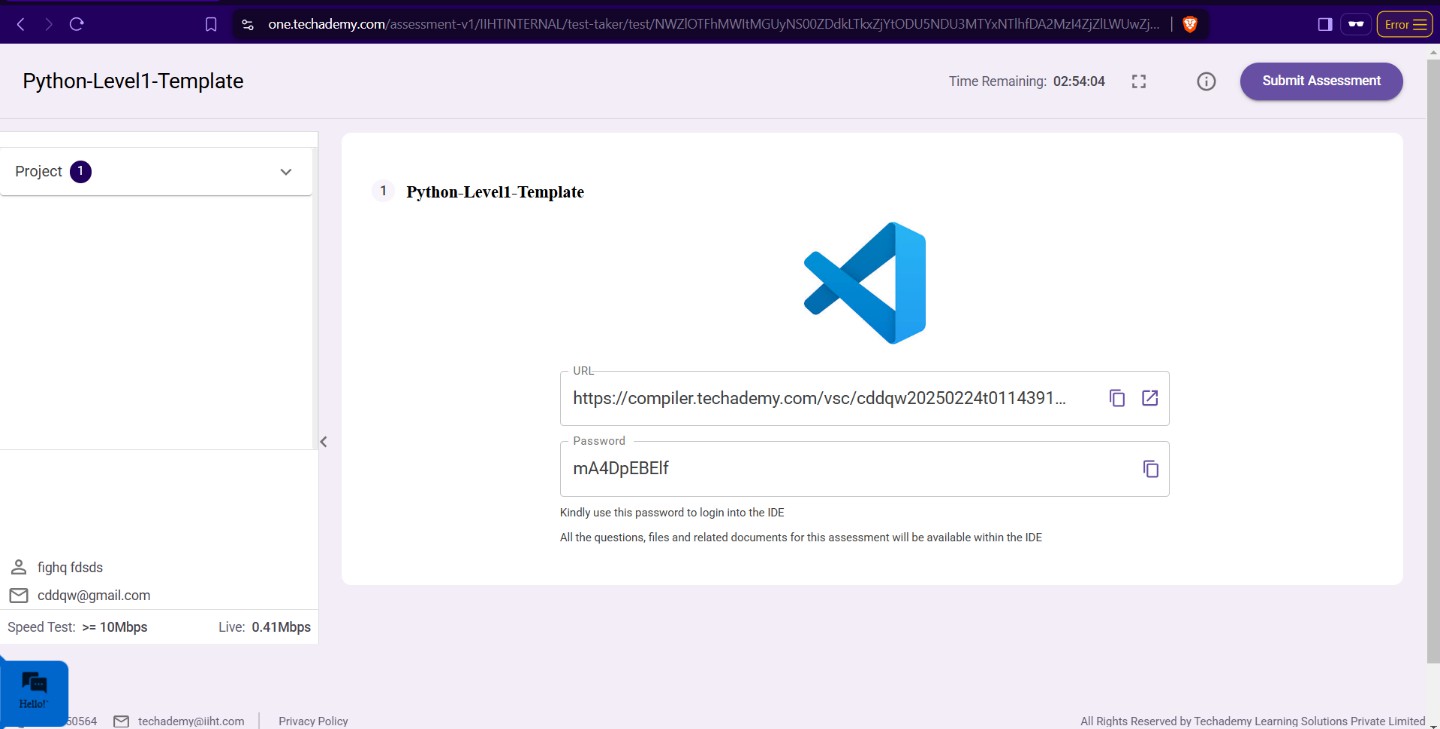


**Python3 titanic .py**

**Python3 Linear\_regression\_auto.py**



**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.**