**System Requirements Specification Index**

**For Machine learning Algorithm No 3**

1.0

**Machine Learning Assessment: Logistic regression and Linear Regression**

**Fish Health Monitoring Dataset (fish\_data.csv)**

This dataset is designed to classify fish as either healthy or diseased based on their behavior, environment, and biological characteristics. It is used in conjunction with a **DecisionTreeClassifier**, which creates a tree-like model of decisions based on the feature values to predict the fish's health status.

**Dataset feat:**

* **Age**: Numerical feature indicating the age of the fish
* **Species**: Categorical feature representing the type of fish .
* **Water\_Temperature**: Numerical feature representing the surrounding water temperature in degrees Celsius.
* **Feeding\_Behavior**: Categorical feature describing how actively the fish is feeding **Coloration**: Categorical feature indicating the observed color of the fish, often linked to health
* **Swimming\_Behavior**: Categorical feature representing movement patterns

**Target:**

* **Disease\_Status**: Binary or categorical variable representing the health of the fish — this is the **target variable** for the DecisionTreeClassifier. It indicates whether a fish is **Healthy** or **Diseased** based on the features.

**Fish.py Implementation Requirementa**

**Requirements:**

load\_fish\_disease\_data() function:

* **Load the data** from a CSV file named "fish\_disease\_data.csv"
* **Limit the data** to the **first 1000 records** using .head(1000)
* **Returns** the loaded DataFrame

**perform\_eda\_on\_age() function:**

* **Perform exploratory data analysis (EDA)** on the 'Age' column of a fish dataset
* **Check** whether the 'Age' column exists — prints a message and exits if it's missing
* **Count** how many fish have an age **greater than 1 year**
* **Return the count**

**preprocess\_fish\_data() function:**

* **You need to Apply one-hot encoding** to categorical columns using pd.get\_dummies() with drop\_first=True
* **Validate** that the target column 'Disease\_Status\_Healthy' was created during encoding
* **Separate** the features X and the target y
* **Return** the feature matrix X, the target vector y, and the full encoded DataFrame df

**split\_fish\_data() function:**

* **Split** the feature matrix X and target vector y into training and testing sets
* Use train\_test\_split with a default **test size of 20%** and random\_state=42 for reproducibility
* **Return** the split datasets: X\_train, X\_test, y\_train, y\_test

**Create\_and\_train\_model() Function :**

* **Create** a DecisionTreeClassifier with a fixed random\_state=42
* **Train** the model on the provided X\_train and y\_train data
* **Save** the trained model to a file named 'decision\_tree\_fish\_disease\_model.pkl' using joblib
* **Return** the trained model

**To understand how balanced or imbalanced your dataset we need to calculate entropy**

**calculate\_entropy() function:**

* **Calculates the entropy** of the target variable y using the formula:

A black and white symbol

AI-generated content may be incorrect.

* Uses value\_counts(normalize=True) to get class probabilities
* **Ignores zero probabilities** to avoid log(0) errors ,helps assess the **impurity or uncertainty** in the target distribution

Return the value

**check\_new\_data\_from\_json() function:**

* **Read** new fish data from a JSON file (default: "fish\_data.json") under the "fish" key
* **Uses the trained model** to predict whether the new fish is healthy or diseased
* **Return the value prediction**

**Task 2**

**Train a Logistic Regression Model for loan prediction :**

**load\_data() function:**

* **Load a CSV file** (default: "loan\_dataset.csv") into a pandas DataFrame
* **Calculate and prints**:
  + The **mean** of the 'loan\_amount' column (rounded to 2 decimal places)
  + The **maximum** loan amount
* **Return** the full DataFrame

**explore\_home\_ownership() function:**

* **Check** whether the column 'home\_ownership' exists in the DataFrame
* **Count** how many records have 'home\_ownership' == "RENT"
* **Return** the count

**prepare\_data(df) function:**

* **Encode categorical columns**:
  + It checks if 'term' or 'home\_ownership' columns are of object type
  + If so, it applies **LabelEncoder** to convert them into numerical labels
* **Scale numerical features**:
  + Apply StandardScaler to all columns **except** the target ('defaulted')
  + Ensures all features are on the same scale (mean = 0, std = 1)
* **Return** the transformed DataFrame

sigmoid\_demo**() function:**

 **Calculates** the sigmoid of a fixed value z = 1.5 using the formula:up to 4 decimal places

 **Return** the sigmoid value

**train\_and\_evaluate() function:**

* **Train a LogisticRegression model** using X\_train and y\_train with max\_iter=1000 for stability
* **Save the trained model** to a file (default: "loan\_model.pkl") using joblib
* **Generate predictions** on X\_test:
  + y\_pred: predicted class labels
  + y\_pred\_prob: predicted probabilities for the positive class (1)
* **Print the first 10 predictions** for a quick sample output (optional)
* **Returns** a dictionary containing:
  + The trained model
  + The predicted labels
  + The predicted probabilities
* **Note The output of .predict\_proba() is a 2D array: [[prob\_class\_0, prob\_class\_1], ...]**

**[:, 1] extracts only the probability of class 1.**

* You need to return "y\_pred\_prob" – predicted probabilities for the positive class (1D array)

**Testing Your Implementation**

After implementing the functions according to these requirements, you can run the test file to verify your implementation:

**Python3 -m unittest**

The test file will check if your functions meet the requirements and provide feedback on which tests passed or failed.

**Running the Tests**

To run the tests, use the following command:

Python3 -m unittest

**Submission Guidelines**

1. Complete all the required functions in `fish.py` and `Loan.py`

2. Ensure all tests pass

3. Submit your code files

**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 fish.py**

**Python3 loan.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.

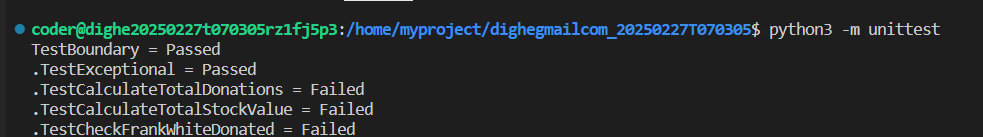
**Screen shot to run the program**

**To run the application**

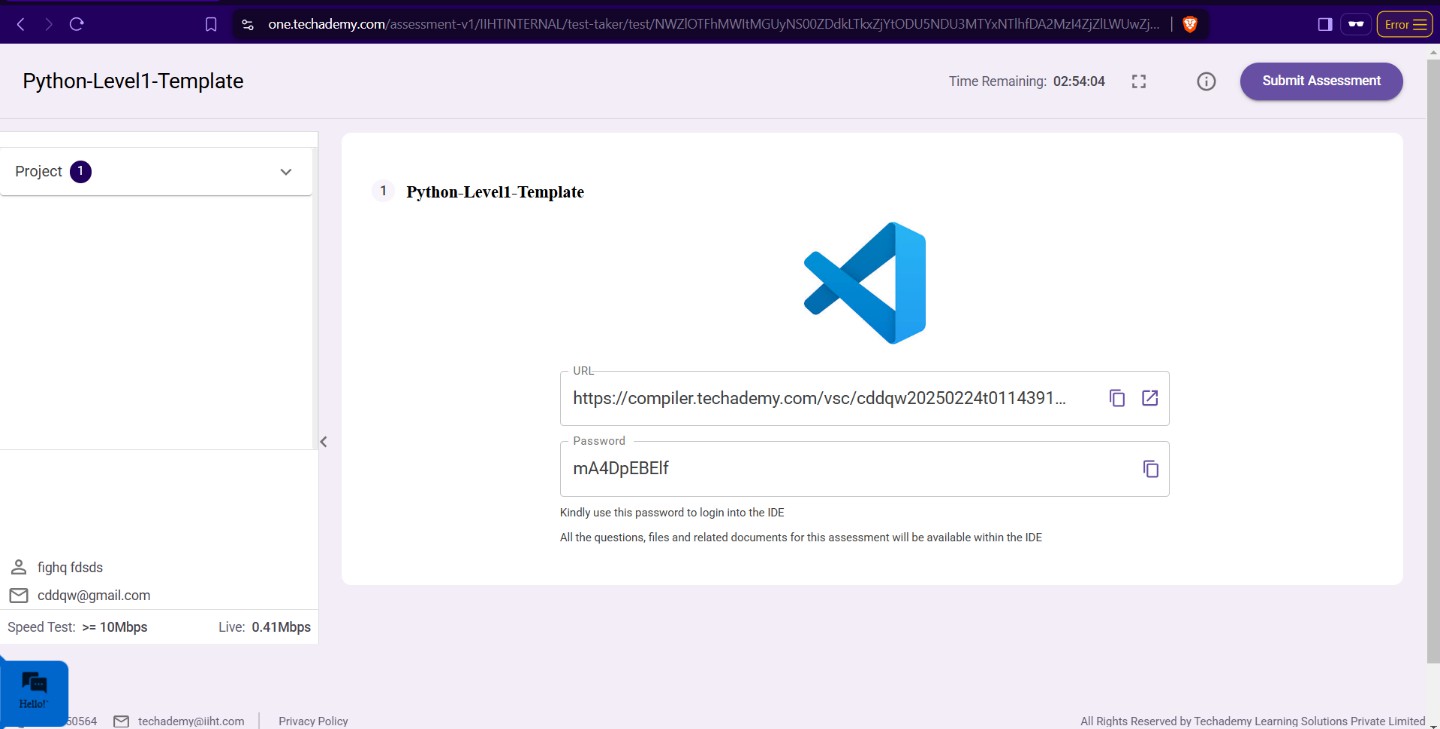


**Python3 fish .py**

**Python3 loan.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.**