**System Requirements Specification Index**

**For Machine learning Algorithm No 3**

1.0

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

**House Price and Loan Default Prediction Implementation Guide**

**This document provides detailed instructions for implementing the required functions in `House.py` and `loan.py` files to pass the functional tests in `test\_functional.py`. The implementation should follow the skeleton code structure with proper functionality.**

**Dataset Information**

**Housing Dataset (`Housing.csv`)**

The housing dataset contains information about house properties with the following features:

- `rooms`: Number of rooms in the house

- `area`: Area of the house in square feet

- `bathrooms`: Number of bathrooms

- `floors`: Number of floors

- `age`: Age of the house in years

- `price`: Price of the house (target variable)

**Loan Dataset (`loan\_dataset.csv`)**

The loan dataset contains information about loans with the following features:

- `loan\_amount`: Amount of the loan

- `term`: Loan term (36 months or 60 months)

- `credit\_score`: Credit score of the borrower

- `employment\_length`: Employment length in years

- `home\_ownership`: Type of home ownership (OWN, MORTGAGE, RENT)

- `annual\_income`: Annual income of the borrower

- `defaulted`: Whether the loan defaulted (0 = no default, 1 = default) (target variable)

**House.py Implementation Requirements**

**1. `load\_and\_preprocess(path)`**

**Purpose: Load and preprocess the housing dataset.**

Requirements:

- Load the CSV file from the given path

- Clean column names (convert to lowercase and strip whitespace)

- Handle missing values (drop rows with missing values)

- Print a confirmation message: " Data loaded and cleaned."

- Return the preprocessed DataFrame

**2. `show\_key\_stats(df)`**

Purpose: Display key statistics about the housing data.

Requirements:

- Calculate the standard deviation of the price column

- Find the maximum number of rooms

- Print these statistics in a formatted way:

  - " Standard Deviation of Price: $[value]"

  - " Maximum Number of Rooms: [value]"

**3. prepare\_data(df, features, target)**

**Purpose: Prepare the data for model training.**

**Requirements:**

- Extract features (X) and target (y) from the DataFrame

- Scale the features using StandardScaler

- Split the data into training and testing sets (80% train, 20% test) with random\_state=42

- Print a confirmation message: “ Data prepared and split."

- Return X\_train, X\_test, y\_train, y\_test, and the scaler object

**4. `train\_and\_save\_model(X\_train, y\_train, model\_path="house\_price\_model.pkl")`**

Purpose: Train a linear regression model and save it.

Requirements:

- Create a LinearRegression model

- Train the model with the training data

- Save the model to the specified path using joblib.dump()

- Print a confirmation message: " Model trained and saved to '[model\_path]'"

- Return the trained model

**5. evaluate\_model(model, X\_test, y\_test)`**

**Purpose: Evaluate the model's performance.**

Requirements\*\*:

- Make predictions using the model on the test data

- Calculate the mean squared error

- Print the evaluation metrics:

  - “ Mean Squared Error: [value]"

  - " Sample Predictions: [first 10 predictions]"

**loan.py Implementation Requirements**

**1. `load\_and\_prepare\_data(path="loan\_dataset.csv")`**

**Purpose: Load and preprocess the loan dataset.**

**Requirements:**

**-** Load the CSV file from the given path

- Display loan amount statistics:

  - " Loan Amount - Mean: [value], Max: [value]"

- Encode categorical columns ('term', 'home\_ownership') using LabelEncoder

- Scale features using StandardScaler (all columns except 'defaulted')

- Print a confirmation message: "Real dataset loaded and preprocessed."

- Return the preprocessed DataFrame

**2. explore\_data(df)**

**Purpose: Explore the loan amount data.**

- This function can be left as a pass statement for now

- In the full implementation, it would display statistics about the loan amount

**3. sigmoid\_demo()**

**Purpose: Demonstrate the sigmoid activation function.**

- Calculate the sigmoid of 1.5: sigmoid(1.5) = 1 / (1 + e^(-1.5))

- Print the result: " Sigmoid(1.5) = [value]"

**4. cost\_function(y\_true, y\_pred\_prob)**

**Purpose: Implement a custom log loss cost function.**

- Add a small epsilon (e.g., 1e-15) to prevent log(0)

- Clip prediction probabilities to avoid extreme values

- Calculate binary cross-entropy: -mean(y\_true \* log(y\_pred\_prob) + (1 - y\_true) \* log(1 - y\_pred\_prob))

- Return the calculated cost

**5. `train\_and\_evaluate(X\_train, y\_train, X\_test, y\_test, path="loan\_model.pkl")`**

Purpose\*\*: Train a logistic regression model and evaluate it.

Requirements:

- Create a LogisticRegression model with max\_iter=1000

- Train the model with the training data

- Save the model to the specified path using joblib.dump()

- Print a confirmation message: " Model trained and saved to '[path]'"

- Make predictions and calculate probabilities

- Calculate the custom cost using the cost\_function

- Print evaluation metrics:

  - " Log Loss (Custom Cost): [value]"

  - " Sample Predictions: [first 10 predictions]"

**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 `House.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 house .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.

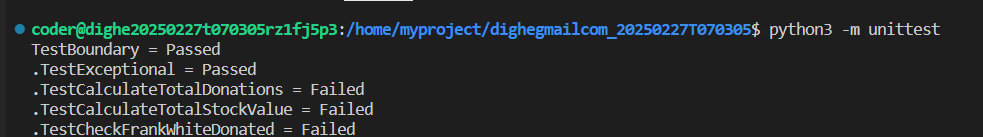
**Screen shot to run the program**

**To run the application**

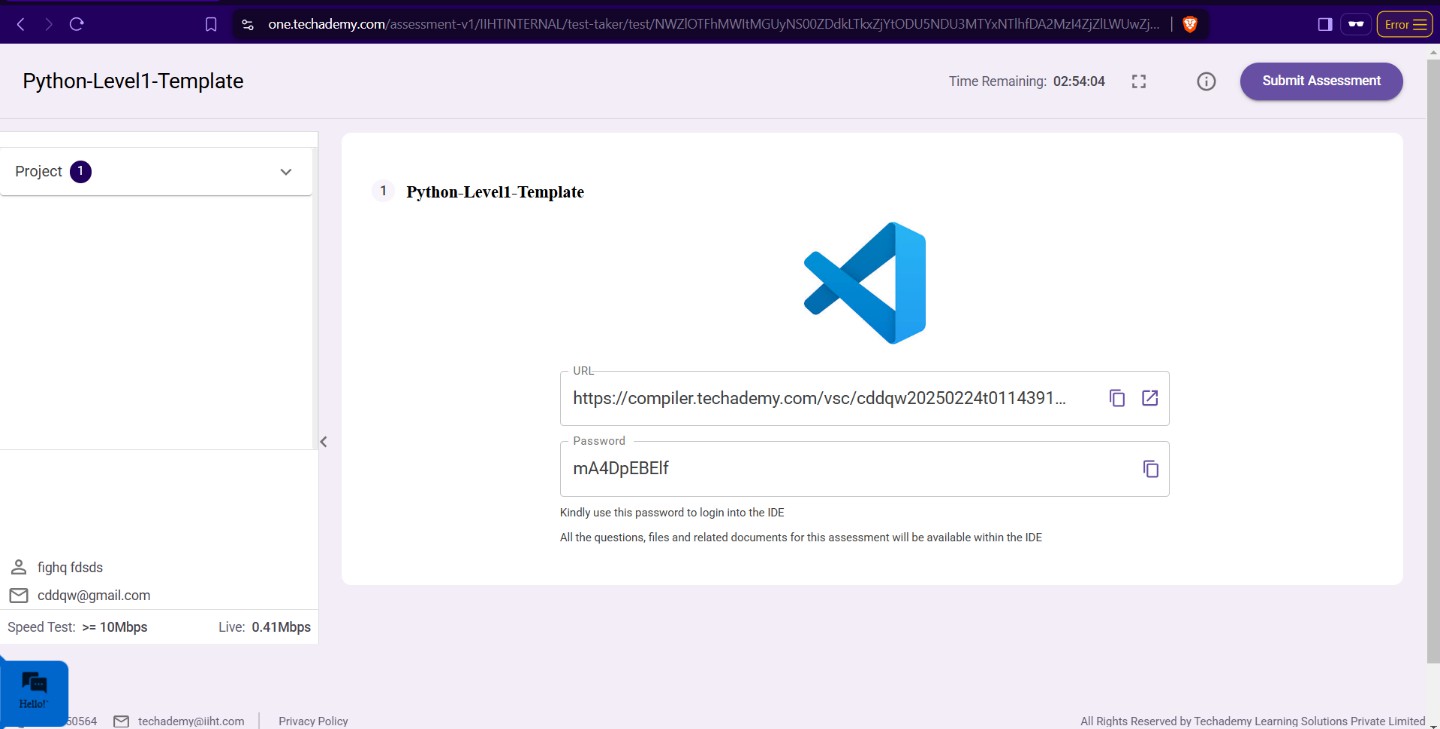


**Python3 house .py**

**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.**