 https://doi.org/10.57017/jorit.v4.1(7).07

Part A

This code presents a simple example of implementing a RL model using TypeScript.

```
type State = {
  position: number;
  velocity: number;
};

type Action = 'accelerate' | 'decelerate' | 'maintain';

interface RLModel {
  state: State;
  reward: number;
  performAction(action: Action): void;
  updateState(newState: State): void;
  calculateReward(): number;
}

class SimpleRLModel implements RLModel {
  state: State;
  reward: number;

  constructor(initialState: State) {
    this.state = initialState;
    this.reward = 0;
  }

  performAction(action: Action): void {
    switch (action) {
      case 'accelerate':
        this.state.velocity += 1;
        break;
      case 'decelerate':
        this.state.velocity -= 1;
        break;
      case 'maintain':
        break;
    }
    this.updateState(this.state);
    this.reward = this.calculateReward();
  }

  updateState(newState: State): void {
    this.state = newState;
  }

  calculateReward(): number {
    return this.state.velocity > 0 ? 10 : -10;
  }
}

const initialState: State = { position: 0, velocity: 0 };
const model = new SimpleRLModel(initialState);

model.performAction('accelerate');

console.log(`Current State: Position=${model.state.position}, Velocity=${model.state.velocity}, Reward:
${model.reward}`);
```

Part B

In the following code example, the model adapts to new data by updating the regression coefficients based on the results:

```
class EnergyPredictionModel {
  constructor() {
    this.slope = 0;
    this.intercept = 0;
  }

  predict(temperature) {
    return this.slope * temperature + this.intercept;
  }

  updateModel(temperature, actualEnergyUsage) {
    const learningRate = 0.01;
    const predictedEnergy = this.predict(temperature);
    const error = actualEnergyUsage - predictedEnergy;

    this.slope += learningRate * error * temperature;
    this.intercept += learningRate * error;
  }
}

const model = new EnergyPredictionModel();

const temperatures = [15, 20, 25, 30];
const actualEnergyUsages = [100, 150, 200, 250];

for (let i = 0; i < temperatures.length; i++) {
  model.updateModel(temperatures[i], actualEnergyUsages[i]);
}

const newTemperature = 22;
const predictedEnergy = model.predict(newTemperature);

console.log(`Predicted Energy Usage for ${newTemperature}°C: ${predictedEnergy}`);
```