

Software Engineering Tools Lab

Assignment No-2

(Module 2- Software Development Frameworks)

Date : 14 Feb. 2022

PRN : 2019BTECS00095

PRN : 2019BTECS00103

1. List of Frameworks/IDEs/Software

- a. Eclipse
- b. Android SDK
- c. Node.Js
- d. DotNet
- e. Ruby on Rails
- f. Anaconda
- g. Google colab

For every Frameworks/IDEs/Software's given above provide the answers for below questions.

Ans:

Node.Js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

Node.js lets developers use JavaScript to write command line tools and for server-side scripting running scripts server-side to produce dynamic web page content before the page is sent to user's web browser.

i. Original Author

The original author of Node.js is Ryan Dahl

ii. Developers

The developers of NodeJS are OpenJS Foundation

iii. Initial Release

The initial release of Node.js is 0.1.14 which was on 26 May 2009.

iv. Stable Release

The latest stable version of Node.js is 17.4.0 which was released on 18 January 2022.

v. Preview Release

The preview release of NodeJS is 17.5.0 which was released on 10 February 2022

vi. Repository (with cloud support)

The link for the Eclipse IDE Repository is
<https://github.com/nodejs/node>.

vii. Written in (Languages)

The language use is C, C++, JavaScript.

viii. Operating System Support

The Eclipse IDE is supported in z/OS, Linux, macOS, SmartOS, FreeBSD, OpenBSD, IBM AIX and Microsoft Windows.

ix. Platform, portability

Node.js is an **open-source, cross-platform, back-end JavaScript runtime environment** that runs on the V8 engine and executes JavaScript code outside a web browser.

x. Available in (Total Languages)

JavaScript is the only language that Node. js supports natively, but many compile-to-JS languages are available.

xi. List of languages supported

As a result, Node. js applications can be written in CoffeeScript, Dart, TypeScript, ClojureScript and others.

xii. Type (Programming tool, integrated development environment etc.)

NodeJS is a runtime environment.

xiii. Website

<https://nodejs.org/en>

xiv. Features

- Cross-platform compatibility
- The convenience of using one coding language
- Facilitates quick deployment and microservice development
- It is scalable
- Commendable data processing ability
- Active open-source community

xv. Size (in MB, GB etc.)

By default, Node.js (up to 11.x) uses a maximum heap size of 700MB and 1400MB on 32-bit and 64-bit platforms, respectively.

xvi. Privacy and Security

The OpenJS Foundation's core purpose is to foster an ecosystem that supports the collaborative and public development of free and open source software projects (each, a "Project"). This privacy policy ("Privacy Policy") describes our policies and procedures about the collection, use, disclosure

and sharing, or other processing of your personal information when you use OpenJS Foundation websites (e.g., openjsf.org)), or participate in or use our Project sites (collectively, the “Sites”), as well as when you interact with or participate in our events, programs, trainings and our other services and offerings (collectively, the “Services”).

xvii. Type of Software (Open Source/ License)

It is open-source, cross-platform.

xviii. Latest Version

The latest version of NodeJS is 17.5.0.

xix. Cloud Support

Node.js on Google Cloud **integrates with Cloud Monitoring, Cloud Trace, Cloud Logging, and Error Reporting**, allowing you to transparently instrument live production applications to diagnose performance bottlenecks and software bugs.

xx. Applicability

- Node.js offers an Easy Scalability.
- Node.js offers the developers the luxury of writing the server-side applications in the JavaScript. This allows the Node.js developers to write both the front-end as well as the back-end web application in JavaScript using a runtime environment.
- Node.js has been regarded as a full-stack JavaScript for serving both the client and the server-side applications.
- The speed of the code execution also enhanced by runtime environment as it supports the non-blocking I/O operations.

xxi. Drawbacks

- Reduces performance when handling heavy computing tasks
- Node.js invites a lot of code changes due to unstable API
- Node.js Asynchronous Programming Model makes it difficult to

maintain code High demand with a few Experienced Node.js Developers

Q2. Implement linear regression problem using google colab(perform preprocessing, training and testing)

Dataset 3-

<https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction>

Loading dataset



The screenshot shows the Google Colab interface. At the top, there's a toolbar with '+ Code', '+ Text', and a status message 'Cannot save changes'. On the right, there are indicators for RAM and Disk usage, along with icons for sharing, settings, and a dropdown menu. The main area displays three code cells, each with a green checkmark and a '0s' execution time. The first cell is titled 'Importing the libraries' and contains code to import numpy, matplotlib, and pandas. The second cell is titled 'Importing the dataset' and contains code to read a CSV file. The third cell contains code to split the dataset into features (X) and target (y).

```
+ Code + Text Cannot save changes RAM Disk [icons]

Importing the libraries

[1] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd

Importing the dataset

[2] dataset = pd.read_csv('energydata_complete.csv')

[25] x = dataset.iloc[:, 1:-1].values
     y = dataset.iloc[:, 2].values
```

1.preprocessing

Splitting the dataset into training set and test set

```
[26] from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 1/3 , random_state = 0)
```

Fitting Simple Linear Regression to the training set

```
[27] from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)

LinearRegression()
```

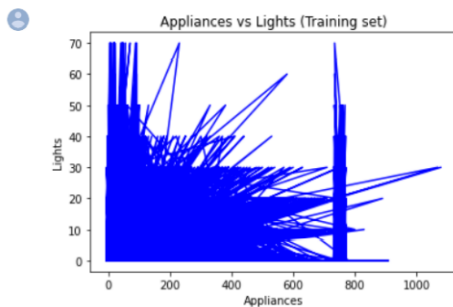
Predicting the test set results

```
[28] y_pred = regressor.predict(x_test)
```

2.training and testing

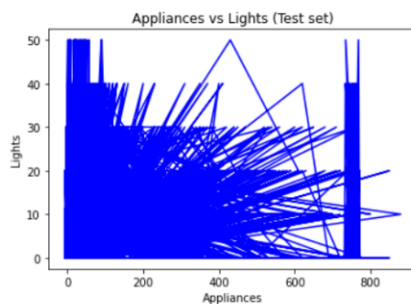
Visualising the training set results

```
[29] # plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, regressor.predict(x_train), color = 'blue')
plt.title('Appliances vs Lights (Training set)')
plt.xlabel('Appliances')
plt.ylabel('Lights')
plt.show()
```



Visualising the test set results

```
[30] plt.plot(x_test, regressor.predict(x_test), color = 'blue')
plt.title('Appliances vs Lights (Test set)')
plt.xlabel('Appliances')
plt.ylabel('Lights')
plt.show()
```



+ Code + Text Copy to Drive

✓ RAM
Disk Editing

Fitting Simple Linear Regression to the training set

```
✓ [33] from sklearn.linear_model import LinearRegression  
0s regressor1 = LinearRegression()  
regressor1.fit(x_test,y_test)  
  
LinearRegression()
```

+ Code + Text Copy to Drive

✓ RAM
Disk Editing

Visualising the test set results

```
✓ 1s ▶ plt.plot(x_test, regressor1.predict(x_test), color = 'blue')  
plt.title('Appliances vs Lights (Test set)')  
plt.xlabel('Appliances')  
plt.ylabel('Lights')  
plt.show()
```

