

# DATA SCIENCE

## CLASS 19: WEB DEVELOPMENT WITH FLASK / HEROKU

- I. WHAT IS WEB DEVELOPMENT?**
- II. WHAT IS HEROKU / FLASK?**
- III. MVC**
- IV. DEPLOYING KNN**

---

## **WHAT IS WEB DEVELOPMENT?**

---

# **I. WHAT IS WEB DEVELOPMENT?**

The work involved with  
building and maintaining  
a live website

Two types of web development:

Two types of web development:

## Front-end Development

- HTML / CSS
- Responsive design
- Makes things pretty / easy to use

## Back-end Development

- Many backend languages
- Model View Controller
- Database work
- Makes the site “work”

**Full-stack Development  
comprises of both front end  
and back end work**

Part of being a web developer is  
knowing the technologies used

Part of being a web developer is knowing the technologies used

### **Web-framework**

Consists of database work and logistics of the site

### **Deployment**

How we “serve” the website. So that other people can use it

# WHAT IS WEB DEVELOPMENT?

10



## Web-framework

Consists of database work and logistics of the site



## Deployment

How we “serve” the website. So that other people can use it

---

**HEROKU / FLASK**

---

## **II. HEROKU / FLASK**

Web Development is hard..

Which is why GA has several classes dedicated to it

We will use two very simple web development tools:  
**Heroku** and **Flask**

Did someone say Flask!?



Flask

web development,  
one drop at a time

Did someone say Flask!?



Flask

web development,  
one drop at a time

Flask is a micro-web-framework based entirely in python

**What does that mean?**

It means we can write the entire backend in Python!

Did someone say Heroku!?



Heroku is a Salesforce company that lets us deploy our websites easily

**What does that mean?**

We use heroku to rent servers to host the website

One thing that stays constant over all technologies is the idea of the

Model View Controller paradigm

---

**MODEL VIEW CONTROLLER**

---

# **III. MODEL VIEW CONTROLLER**

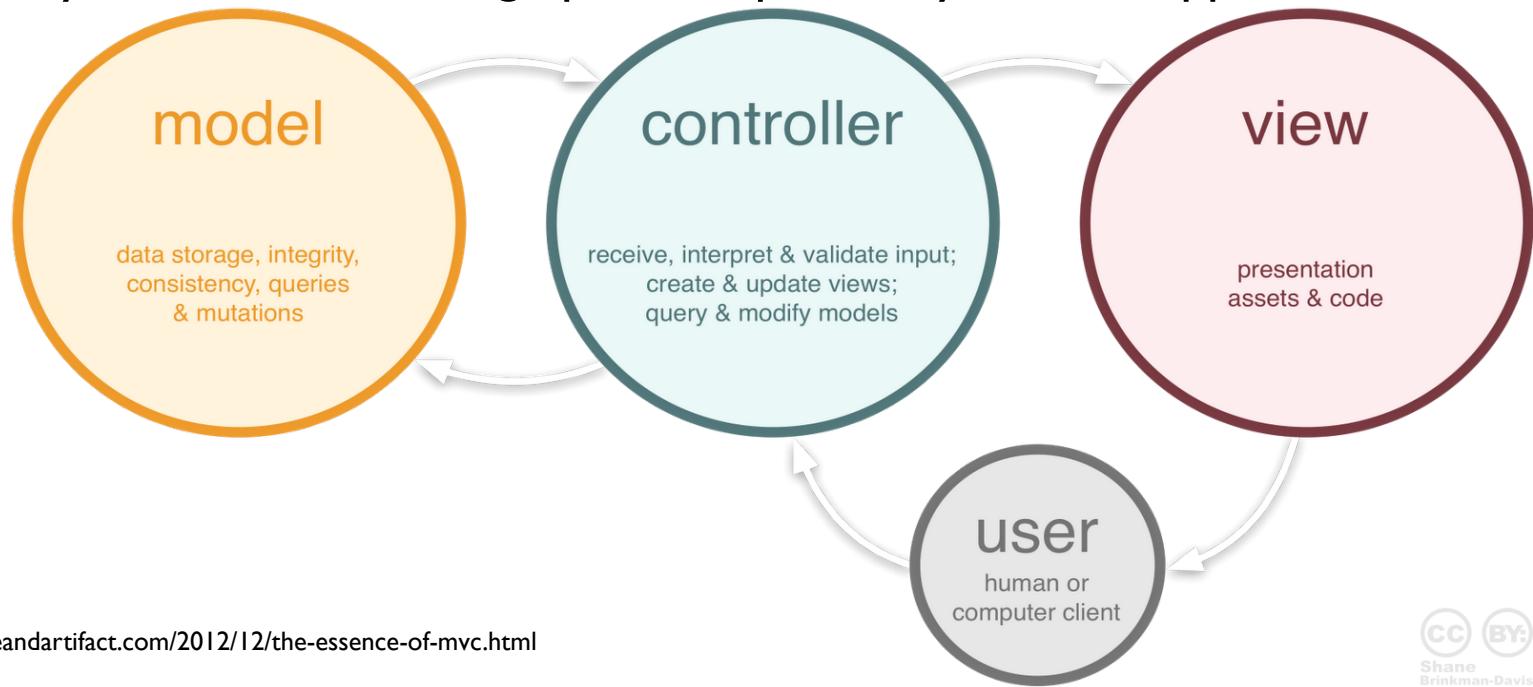
Model View Controller:

Model View Controller:

Is a way of life

Model View Controller:

But actually it's a software design pattern specifically for web apps



Model View Controller:

## Model

- Responsible for managing the data
- It's a database essentially!

## View

- Presents the data / app
- Responsible for design / user experience

## Controller

- Responds to user input and performs operations based on it
- Eg. User inputs a number of neighbors and the controller trains the model

## Model View Controller:

### Model

- Responsible for managing the data
- It's a database essentially!

#### QUESTION:

Which ones are handled by  
Back end developers?  
Front end?

### View

- Presents the data / app
- Responsible for design / user experience

### Controller

- Responds to user input and performs operations based on it
- Eg. User inputs a number of neighbors and the controller trains the model

Model View Controller:

## **Model              (Backend)**

- Responsible for managing the data
- It's a database essentially!

### **QUESTION:**

Which ones are handled by  
Back end developers?  
Front end?

## **View              (Frontend)**

- Presents the data / app
- Responsible for design / user experience

## **Controller    (Backend / Frontend)**

- Responds to user input and performs operations based on it
- Eg. User inputs a number of neighbors and the controller trains the model

---

## **DEPLOYING KNN**

---

# **IV. DEPLOYING KNN**

# Sample Flask App

[https://github.com/sinanuozdemir/iris\\_calculator](https://github.com/sinanuozdemir/iris_calculator)

# Sample Flask App

[https://github.com/sinanuozdemir/iris\\_calculator](https://github.com/sinanuozdemir/iris_calculator)

Made by yours truly and  
yours truly alone

# Sample Flask App

[https://github.com/sinanuozdemir/iris\\_calculator](https://github.com/sinanuozdemir/iris_calculator)



Just kidding thanks Liam!!

Nice hair bro...

# Sample Flask App

[https://github.com/sinanuozdemir/iris\\_calculator](https://github.com/sinanuozdemir/iris_calculator)

Notice we have:

1. Models
2. Views (called templates)
3. Controller (controller.py)

# Sample Flask App

[https://github.com/sinanuozdemir/iris\\_calculator](https://github.com/sinanuozdemir/iris_calculator)

Go ahead and clone it

NOT IN YOUR OTHER GIT  
REPOSITORY ☺

# Sample Flask App

[https://github.com/sinanuozdemir/iris\\_calculator](https://github.com/sinanuozdemir/iris_calculator)

To run locally, go to root and run

**python controller.py**

Go to <http://127.0.0.1:5000/>

# Sample Flask App

[https://github.com/sinanuozdemir/iris\\_calculator](https://github.com/sinanuozdemir/iris_calculator)

**NOTE:**

You may not have the required modules to run it right now..

If not, run

```
sudo pip install -r  
requirements_clean.txt
```

To run locally, go to root and run

**python controller.py**

Go to <http://127.0.0.1:5000/>

We have two forms

The top form **trains the model**

The bottom form **predicts incoming data**

When we submit the data....

We have two forms

The top form **trains the model**

The bottom form **predicts incoming data**

When we submit the data....

**QUESTION:**

Which part of MVC handles the input?

We have two forms

The top form **trains the model**

The bottom form **predicts incoming data**

When we submit the data the **controller** handles it!

Please fill out the form below!

Number of Neighbors:

Paramter 2 (Optional)

Paramter 3 (Optional)

Paramter 4 (Optional)

Sepal Length (cm):

Sepal Width (cm):

Petal Length (cm):

Petal Width (cm):

**Note:**

Note we included 3 optional parameters in case you need to train on more than just one!

The machine learning model lives in the **model** folder  
not to be confused with the model in MVC

It is pickled...



The machine learning model lives in the **model** folder  
not to be confused with the model in MVC

It is pickled...

You know, the standard mechanism  
for serializing an object.

Essentially we can transform a  
python object into a file!



You can pickle anything!!

1. sklearn models
2. Jsons!
3. Strings!
4. Your own models



# Comprehensive Step by Step

1. Sign up for Heroku [here](#)
2. Create a new app (make sure Heroku toolbelt is installed)
3. Clone our flask app [here](#)
  - I. Change and test at will
4. Run the command: **heroku git:remote -a <APP>**
5. Install the custom build back for scipy and numpy
  - I. heroku config:set BUILDPACK\_URL=https://github.com/thenovices/heroku-buildpack-scipy -app <APP>
  2. Run the command above in the root of your app (with the toolbelt installed)
6. Use as normal git repository:
  - I. Git add, commit, etc...
  2. Git push heroku master (instead of origin)
7. Amaze people with your live KNN

# 1. SIGN UP FOR HEROKU

Self-explanatory?

<http://heroku.com>

# 2. CREATE A NEW APP (MAKE SURE HEROKU TOOLBELT IS INSTALLED)

Not self-explanatory

<https://toolbelt.heroku.com/>

Type into your console:

**heroku login**

# 3. CLONE OUR FLASK APP HERE

Self-explanatory?

Now you can run it locally!! (remember run python controller.py)

# 4. RUN THE COMMAND: **HEROKU GIT:REMOTE -A <APP>**

At the root of the directory

This adds a new git “remote”

Essentially a new place to push ☺

Check this by running *git remote -v*

You should see **origin** and **heroku**

# 5. INSTALL THE CUSTOM BUILD BACK FOR SCIPY AND NUMPY

Not self-explanatory

<https://github.com/thenovices/heroku-buildpack-scipy>

Run:

```
heroku config:set BUILDPACK_URL=https://github.com/thenovices/heroku-buildpack-scipy --app <APP>
```

At the root of the directory

## **6. USE AS NORMAL GIT REPOSITORY**

Self-explanatory?

git add .

git commit -m "I am a genius"

Git push heroku master

# 6. USE AS NORMAL GIT REPOSITORY

Self-explanatory?

git add .

git commit -m "I am a genius"

Git push heroku master

**Note:**

It is installing a bunch of modules because of the requirements.txt file

# 7. AMAZE PEOPLE WITH YOUR LIVE KNN

Self-explanatory!!!!





**Note:**

Your unique website will have your app name instead of sinan-iris

<https://sinan-iris.herokuapp.com/>

# WHAT NOW?!?!?!!1?

Put in your own machine learning model!

But Sinan, it's hard to make it train on the website

Fine!

1. Build your model else where
2. Load it into the **model** folder manually