# Prototyping and Development

This section explains what prototyping and development has been completed to date, giving details on building the web front-end and the creation of a basic machine-learning model.

## Version Control

## Prototyping

## Vertical Prototype

The prototyping for this project revolves around creating a vertical prototype, which shows the basic structure and functionality of both sub-sections of the project. This will then be reviewed over the December break and if found satisfactory, built upon heavily to create the final application.

### Web Front-End

For this section of the vertical prototype, the goal was to have a working web-app that could connect to the Twitter API and retrieve data from it, in this case the twenty most recent tweets from the account linked to the username chosen by the web-app user.

Below are two screenshots of the working web page which asks for the user to input a Twitter username then retrieves and outputs the data received back from the Twitter API to screen.

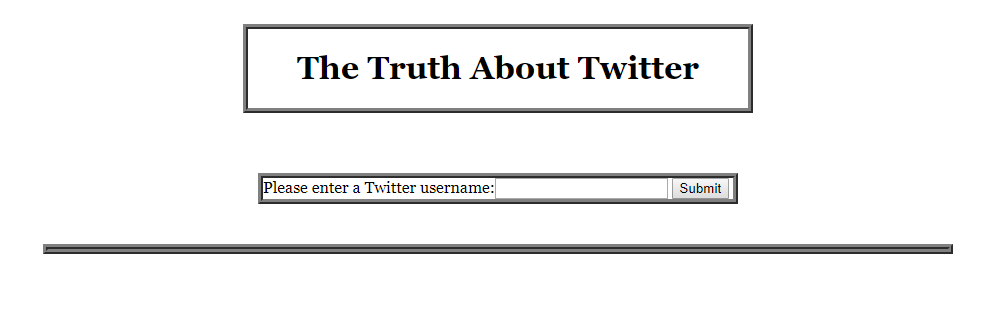


Figure 17 Web Front End

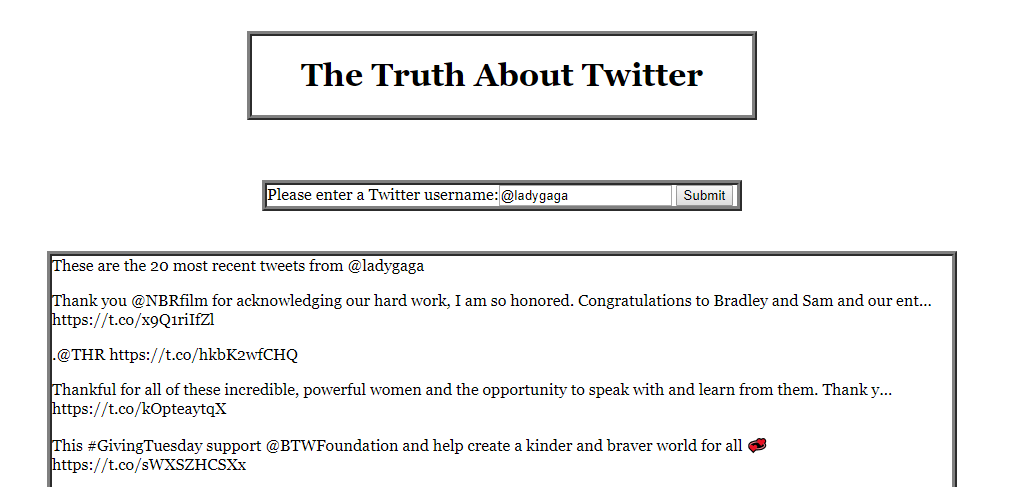


Figure 18 Web Front End with returned tweets

### Basic Machine-Learning Model

For this section of the vertical prototype, the goal was to have a basic machine-learning model that uses data from the acquired datasets and using k-fold cross validation, uses the entirety of the selected data as training and testing sets, outputting, to console, the average accuracy result across the partitions, repeated five times.

Seven features were chosen for this initial model and will be re-evaluated and changed further into development: Whether the account has the default profile picture, has a screen name, has a description, has less than 30 friends, has more than 1000 friends, has never tweeted, the account is geo located and the ratio of friends to followers is 3:1.

A Naïve Bayes classifier with Bernoulli distribution was chosen for this as it a good classifier to start with any data mining project and the inputs are of a binary format, 0 and 1’s with 2000 accounts chosen, 1000 random genuine accounts and 1000 random traditional bot accounts. Below is the output from this completed section the accuracy sitting at around 60% depending on which accounts are selected at the start.

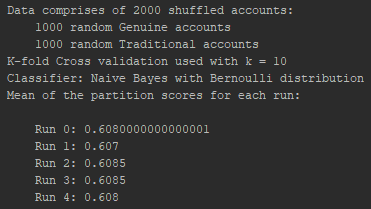


Figure 19 Results of Model being run

## Development

This section deals with all the development done to date in the creation of the vertical prototype using the Django framework and various python libraries such as Tweepy and Scikit-learn.

### Web Front-End

There were several steps to creating the web front-end shown above in the previous section and each will be explained with code snippets where needed.

A new Django project was created within PyCharm Professional, allowing a lot of the tedious groundwork for a web application to be taken care. This meant a bare-bones skeleton app was ready for use and to be built upon.

Next a Twitter developer account was created using my own Twitter account and a Twitter app created, noting its Consumer Key and Secret Key. An Access Token and Access Token Secret were created and noted next. These were outputted to a json file: twitter\_credentials.json, using twitter\_credentials.py, for use later.

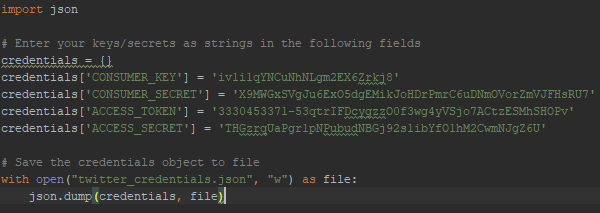


Figure 20 Saving Twitter credentials to file

Next are all changes made to the Django skeleton code:

* Creating a basic form called UsernameForm, in forms.py, to take in user input.



Figure 21 Basic form

* This form was added to views.py within the index method. If the request method was POST, the existing form instance was read in and the user input read into username for use later otherwise a new form instance was created.

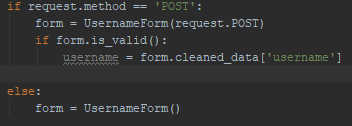


Figure 22 Creating or reading in form

* The values from twitter\_credentials.json are read in and are used in combination with the username inputted in the form to access the Twitter API to return the most recent twenty tweets from that username.[22]

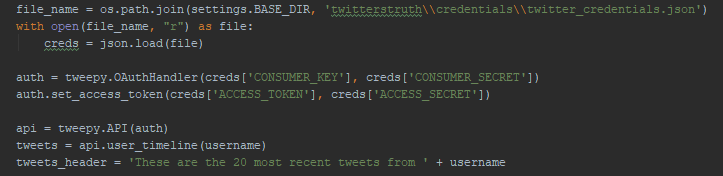


Figure 23 Twitter Authentication and tweet retrieval

* Then a response is returned, asking to render index.html with passed variables: username, tweets and tweets\_header for use in the Html file.



Figure 24 Render Index.html with variables passed

* The templates section of settings.py was altered so to know where to look for template files suck as index.html

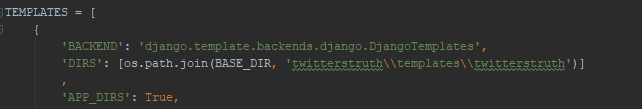


Figure 25 Template directory added

* Index.html was been altered to show the form and tweets using the variables passed to it via views.py

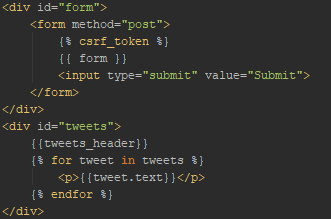


Figure 26 Index.html

* Index.py uses a CSS file web\_style.css, which is located within the twitterstruth/staticfiles/css sub-directory. [46]



Figure 27 Load web\_style.css

* To enable this CSS file, and others within the twitterstruth/staticfiles sub-directory, to be found urls.py was altered [46] :

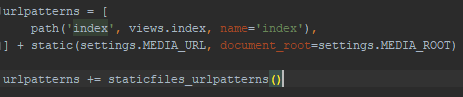


Figure 28 Enabling CSS file load part 1

* As was settings.py [46]:

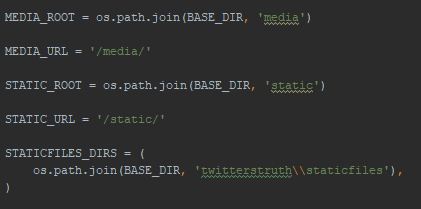


Figure 29 Enabling CSS file load part 2

### Basic Machine-Learning Model

There were several steps to creating the basic machine-learning model shown above in the previous section and each will be explained with code snippets where needed.

* As the files for this section are being run separately but still need access to certain files within the Django framework, the environmental variable, DJANGO\_SETTINGS\_MODULE, must be set and Django setup within each file:

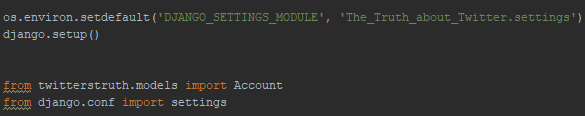


Figure 30 Enabling Django file use

* The file read\_store.py deals with all reading in and storing of the datasets.
* Each dataset is read in from their CSV files one at a time using the pandas library [18], with the tweets files being ignored until the next phase of development.

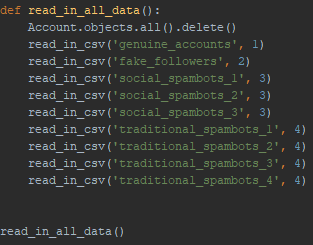


Figure 31 Read in all data

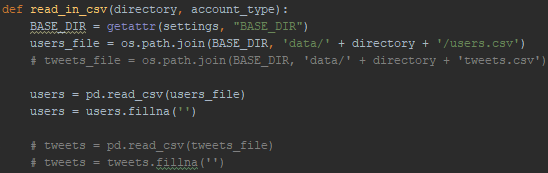


Figure 32 Read in single dataset from CSV

* Checks are done on certain columns in the dataset and binary outputs given depending on the result to form the data going into the database:

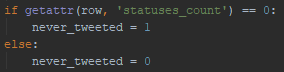


Figure 33 Data check example

* The data is then read into the database:

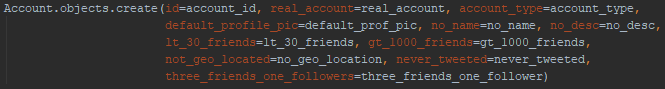


Figure 34 Adding data to database

* In machine\_learning.py, data is read out from the database and passed to through a model, giving output to the console of the model’s accuracy with each run.
* All the randomly chosen, accounts are read out from the database, being split into features and corresponding targets lists:



Figure 35 Get 2000 accounts

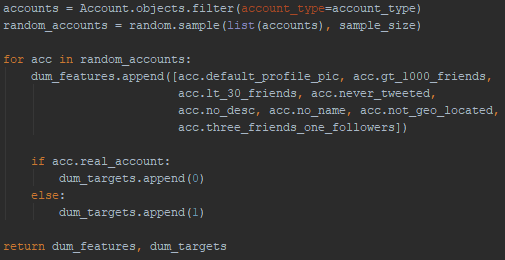


Figure 36 Return random sub-set of accounts from database

* The lists were converted into NumPy arrays [19] and the sklearn library used for k-fold cross validation and classifiers initialisation [17]:

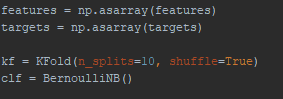


Figure 37 Convert arrays and initialise model

* This model was run five times, with the mean accuracy score across the partitions outputted each time:



Figure 38 Run model and output results to screen

# Bibliography

1. Catherine; (2013), Include CSS and Javascript in my django template, stackoverflow.com/questions/15491727/include-css-and-javascript-in-my-django-template, Date Accessed: November 2018