**gender\_classification\_challenge**

##Overview

The code uses the [scikit-learn](http://scikit-learn.org/) machine learning library to train a [decision tree](https://en.wikipedia.org/wiki/Decision_tree) on a small dataset of body metrics (height, width, and shoe size) labeled male or female. Then we can predict the gender of someone given a novel set of body metrics.

##Dependencies

* Scikit-learn (<http://scikit-learn.org/stable/install.html>)
* numpy (pip install numpy)
* scipy (pip install scipy)

Install missing dependencies using [pip](https://pip.pypa.io/en/stable/installing/)

##Usage

Once you have your dependencies installed via pip, run the script in terminal via

python demo.py

##Challenge

Find 3 more classifiers from the sci-kit learn [documentation](http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html) and add them to the demo.py code. Train them on the same dataset and [compare their results](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html). You can determine accuracy by trying to predict testing you trained classifier on samples from the training data and see if it correctly classifies it. Push your code repository to [github](https://help.github.com/articles/set-up-git/).

Some Great simple sci-kit learn examples here: [https://github.com/chribsen/simple-ma...](https://www.youtube.com/redirect?event=video_description&v=T5pRlIbr6gg&q=https%3A%2F%2Fgithub.com%2Fchribsen%2Fsimple-machine-learning-examples&redir_token=KM6r6tB-MbzMOlEYQ0vjtozM1FJ8MTUyOTM3NTY0MUAxNTI5Mjg5MjQx)

and the official scikit website: [http://scikit-learn.org/](https://www.youtube.com/redirect?event=video_description&v=T5pRlIbr6gg&q=http%3A%2F%2Fscikit-learn.org%2F&redir_token=KM6r6tB-MbzMOlEYQ0vjtozM1FJ8MTUyOTM3NTY0MUAxNTI5Mjg5MjQx)

Highly recommend this online book as supplementary reading material: [https://learnpythonthehardway.org/book/](https://www.youtube.com/redirect?event=video_description&v=T5pRlIbr6gg&q=https%3A%2F%2Flearnpythonthehardway.org%2Fbook%2F&redir_token=KM6r6tB-MbzMOlEYQ0vjtozM1FJ8MTUyOTM3NTY0MUAxNTI5Mjg5MjQx)

Wondering when to use which model? This chart helps, but keep in mind deep neural nets outperform pretty much any model given enough data and computing power. so use these when you don't have access to loads of data and compute: [http://scikit-learn.org/stable/tutori...](https://www.youtube.com/redirect?event=video_description&v=T5pRlIbr6gg&q=http%3A%2F%2Fscikit-learn.org%2Fstable%2Ftutorial%2Fmachine_learning_map%2F&redir_token=KM6r6tB-MbzMOlEYQ0vjtozM1FJ8MTUyOTM3NTY0MUAxNTI5Mjg5MjQx)

**twitter\_sentiment\_challenge**

##Overview

The code uses the [tweepy](http://www.tweepy.org/) library to access the Twitter API and the [TextBlob](https://textblob.readthedocs.io/en/dev/) library to perform Sentiment Analysis on each Tweet. We'll be able to see how positive or negative each tweet is about whatever topic we choose.

##Dependencies

* tweepy (<http://www.tweepy.org/>)
* textblob (<https://textblob.readthedocs.io/en/dev/>)

Install missing dependencies using [pip](https://pip.pypa.io/en/stable/installing/)

##Usage

Once you have your dependencies installed via pip, run the script in terminal via

python demo.py

##Challenge

Instead of printing out each tweet, save each Tweet to a CSV file with an associated label. The label should be either 'Positive' or 'Negative'. You can define the sentiment polarity threshold yourself, whatever you think constitutes a tweet being positive/negative. Push your code repository to [github](https://help.github.com/articles/set-up-git/).

More on TextBlob: [https://textblob.readthedocs.io/en/dev/](https://www.youtube.com/redirect?redir_token=ypZ1MRitY3YvCXyGZiwW1FyYouR8MTUyOTM3NTQzM0AxNTI5Mjg5MDMz&q=https%3A%2F%2Ftextblob.readthedocs.io%2Fen%2Fdev%2F&event=video_description&v=o_OZdbCzHUA)

Great info on Sentiment Analysis: [https://www.quora.com/How-does-sentim...](https://www.youtube.com/redirect?redir_token=ypZ1MRitY3YvCXyGZiwW1FyYouR8MTUyOTM3NTQzM0AxNTI5Mjg5MDMz&q=https%3A%2F%2Fwww.quora.com%2FHow-does-sentiment-analysis-work&event=video_description&v=o_OZdbCzHUA)

Great sentiment analysis api: [http://www.alchemyapi.com/products/al...](https://www.youtube.com/redirect?redir_token=ypZ1MRitY3YvCXyGZiwW1FyYouR8MTUyOTM3NTQzM0AxNTI5Mjg5MDMz&q=http%3A%2F%2Fwww.alchemyapi.com%2Fproducts%2Falchemylanguage%2Fsentiment-analysis&event=video_description&v=o_OZdbCzHUA)

Read over these course notes if you wanna become an NLP god: [http://cs224d.stanford.edu/syllabus.html](https://www.youtube.com/redirect?redir_token=ypZ1MRitY3YvCXyGZiwW1FyYouR8MTUyOTM3NTQzM0AxNTI5Mjg5MDMz&q=http%3A%2F%2Fcs224d.stanford.edu%2Fsyllabus.html&event=video_description&v=o_OZdbCzHUA)

Best book to become a Python god: [https://learnpythonthehardway.org/](https://www.youtube.com/redirect?redir_token=ypZ1MRitY3YvCXyGZiwW1FyYouR8MTUyOTM3NTQzM0AxNTI5Mjg5MDMz&q=https%3A%2F%2Flearnpythonthehardway.org%2F&event=video_description&v=o_OZdbCzHUA)

**recommender\_system\_challenge**

##Overview

The code uses the [lightfm](https://github.com/lyst/lightfm) recommender system library to train a hybrid content-based + collaborative algorithm that uses the WARP loss function on the [movielens](http://grouplens.org/datasets/movielens/) dataset. The movielens dataset contains movies and ratings from over 1700 users. Once trained, our script prints out recommended movies for whatever users from the dataset that we choose to terminal.

##Dependencies

* numpy (<http://www.numpy.org/>)
* scipy (<https://www.scipy.org/>)
* lightfm (<https://github.com/lyst/lightfm>)

Install missing dependencies using [pip](https://pip.pypa.io/en/stable/installing/)

##Usage

Once you have your dependencies installed via pip, run the script in terminal via

python demo.py

**Note** If the lightfm dependency doesn't work for you via pip, just install it from source by running these two commands.

git clone git@github.com:lyst/lightfm.git

cd lightfm && pip install -e .

If you still have dependency version issues, use [virtualenv](http://docs.python-guide.org/en/latest/dev/virtualenvs/).

##Challenge

1. Instead of using the built-in fetch\_movielens method, create your own method to fetch and parse a recommendation dataset of your choice. You can find some good dataset options [here](https://gist.github.com/entaroadun/1653794). Make sure to look at the [fetch\_movielens](https://github.com/lyst/lightfm/blob/master/lightfm/datasets/movielens.py#L107) method to see how it works.
2. Use 3 different loss functions (so 3 different models), compare their results, and then only print the recommendations (products, songs, tv shows, etc.) for the best one. You'll find the available loss functions [here](https://github.com/lyst/lightfm/blob/master/lightfm/lightfm.py#L35).

The LightFM Python Library: [https://github.com/lyst/lightfm/tree/...](https://www.youtube.com/redirect?q=https%3A%2F%2Fgithub.com%2Flyst%2Flightfm%2Ftree%2Fmaster%2Flightfm&v=9gBC9R-msAk&event=video_description&redir_token=P5-vmOwZ62ea_qnDIOIz3ZA8Dsl8MTUyOTM3NTY5N0AxNTI5Mjg5Mjk3) Some great learning resources on recommender systems: [http://blogs.gartner.com/martin-kihn/...](https://www.youtube.com/redirect?q=http%3A%2F%2Fblogs.gartner.com%2Fmartin-kihn%2Fhow-to-build-a-recommender-system-in-python%2F&v=9gBC9R-msAk&event=video_description&redir_token=P5-vmOwZ62ea_qnDIOIz3ZA8Dsl8MTUyOTM3NTY5N0AxNTI5Mjg5Mjk3) [https://www.analyticsvidhya.com/blog/...](https://www.youtube.com/redirect?q=https%3A%2F%2Fwww.analyticsvidhya.com%2Fblog%2F2015%2F08%2Fbeginners-guide-learn-content-based-recommender-systems%2F&v=9gBC9R-msAk&event=video_description&redir_token=P5-vmOwZ62ea_qnDIOIz3ZA8Dsl8MTUyOTM3NTY5N0AxNTI5Mjg5Mjk3) [http://www.quuxlabs.com/blog/2010/09/...](https://www.youtube.com/redirect?q=http%3A%2F%2Fwww.quuxlabs.com%2Fblog%2F2010%2F09%2Fmatrix-factorization-a-simple-tutorial-and-implementation-in-python%2F&v=9gBC9R-msAk&event=video_description&redir_token=P5-vmOwZ62ea_qnDIOIz3ZA8Dsl8MTUyOTM3NTY5N0AxNTI5Mjg5Mjk3) [http://blog.manugarri.com/a-short-int...](https://www.youtube.com/redirect?q=http%3A%2F%2Fblog.manugarri.com%2Fa-short-introduction-to-recommendation-systems%2F&v=9gBC9R-msAk&event=video_description&redir_token=P5-vmOwZ62ea_qnDIOIz3ZA8Dsl8MTUyOTM3NTY5N0AxNTI5Mjg5Mjk3)

#predicting\_stock\_prices ##Overview

The code uses the [scikit-learn](https://github.com/scikit-learn/scikit-learn) machine learning library to train a [support vector regression](https://en.wikipedia.org/wiki/Support_vector_machine) on a stock price dataset from [Google Finance](https://en.wikipedia.org/wiki/Support_vector_machine) to predict a future price. We use scikit-learn to build an ML model, but for the challenge you'll use the [Keras](https://keras.io) library.

There are two scripts. demo.py is the code in the repo and challenge.py is a template for the coding challenge you will complete.

##Dependencies

* numpy (<http://www.numpy.org/>)
* tweepy (<http://www.tweepy.org>)
* csv (<https://pypi.python.org/pypi/csv>)
* textblob (<https://textblob.readthedocs.io/en/dev/>)
* keras (<https://keras.io>)

Install missing dependencies using [pip](https://pip.pypa.io/en/stable/installing/)

##Demo Usage

Once you have your dependencies installed via pip, run the demo script in terminal via

python demo.py

##Challenge

You'll find the challenge template in this repo labeled challenge.py. The instructions are

1. Use the Tweepy library to retrieve tweets about a company stock from twitter
2. Use the TextBlob library to classify those tweets as either positive or negative given a threshold you define.
3. If the majority of tweets are positive, then use the Keras library to build a neural network that predicts the next stock price given a dataset of past stock prices that you pull from Google Finance. [This](http://machinelearningmastery.com/time-series-prediction-with-deep-learning-in-python-with-keras/) tutorial may be useful to you.

If you want to use your own template, that's fine too. Submit your code in the github repo.

Stock prediction with Tensorflow: [https://nicholastsmith.wordpress.com/...](https://www.youtube.com/redirect?redir_token=auyYEQyGqfce7bgpVVv9v3rjNdB8MTUyOTM3NTgwNUAxNTI5Mjg5NDA1&q=https%3A%2F%2Fnicholastsmith.wordpress.com%2F2016%2F04%2F20%2Fstock-market-prediction-using-multi-layer-perceptrons-with-tensorflow%2F&event=video_description&v=SSu00IRRraY)

Another great stock prediction tutorial: [http://eugenezhulenev.com/blog/2014/1...](https://www.youtube.com/redirect?redir_token=auyYEQyGqfce7bgpVVv9v3rjNdB8MTUyOTM3NTgwNUAxNTI5Mjg5NDA1&q=http%3A%2F%2Feugenezhulenev.com%2Fblog%2F2014%2F11%2F14%2Fstock-price-prediction-with-big-data-and-machine-learning%2F&event=video_description&v=SSu00IRRraY)

This guy made 500K doing ML stuff with stocks: [http://jspauld.com/post/35126549635/h...](https://www.youtube.com/redirect?redir_token=auyYEQyGqfce7bgpVVv9v3rjNdB8MTUyOTM3NTgwNUAxNTI5Mjg5NDA1&q=http%3A%2F%2Fjspauld.com%2Fpost%2F35126549635%2Fhow-i-made-500k-with-machine-learning-and-hft&event=video_description&v=SSu00IRRraY)

**deep\_dream\_challenge**

##Overview

The code uses the Tensorflow Machine Learning library to generate a trippy image from a given input image.

##Dependencies

* numpy (<http://www.numpy.org/>)
* functools
* PIL (<http://stackoverflow.com/questions/20060096/installing-pil-with-pip>)
* tensorflow (<https://www.tensorflow.org/versions/r0.10/get_started/os_setup.html#pip-installation>)
* matplotlib (<http://matplotlib.org/1.5.1/users/installing.html>)
* urllib (<https://pypi.python.org/pypi/urllib3>)
* os
* zipfile

Install missing dependencies using [pip](https://pip.pypa.io/en/stable/installing/)

##Demo Usage

Once you have your dependencies installed via pip, run the demo script in terminal via

python deep\_dream.py

##Challenge

The instructions are

1. Add a function to the existing deep dream python file that converts not just an image, but a video clip to deep dream.

*HINT* Think of a video as a collection of images (frames).

If you want to use your own template, that's fine too. Submit your code in the github repo.

More Deep Dream tutorials: [http://www.alanzucconi.com/2016/05/25...](https://www.youtube.com/redirect?v=MrBzgvUNr4w&redir_token=KFDKj4xYa4cUSJw_9mDu2vxcZzd8MTUyOTM3NTk3N0AxNTI5Mjg5NTc3&event=video_description&q=http%3A%2F%2Fwww.alanzucconi.com%2F2016%2F05%2F25%2Fgenerating-deep-dreams%2F) [https://github.com/awanninger/deepdream](https://www.youtube.com/redirect?v=MrBzgvUNr4w&redir_token=KFDKj4xYa4cUSJw_9mDu2vxcZzd8MTUyOTM3NTk3N0AxNTI5Mjg5NTc3&event=video_description&q=https%3A%2F%2Fgithub.com%2Fawanninger%2Fdeepdream) [http://ryankennedy.io/running-the-dee...](https://www.youtube.com/redirect?v=MrBzgvUNr4w&redir_token=KFDKj4xYa4cUSJw_9mDu2vxcZzd8MTUyOTM3NTk3N0AxNTI5Mjg5NTc3&event=video_description&q=http%3A%2F%2Fryankennedy.io%2Frunning-the-deep-dream%2F)

Generate Deep Dream's online: [http://deepdreamgenerator.com/generat...](https://www.youtube.com/redirect?v=MrBzgvUNr4w&redir_token=KFDKj4xYa4cUSJw_9mDu2vxcZzd8MTUyOTM3NTk3N0AxNTI5Mjg5NTc3&event=video_description&q=http%3A%2F%2Fdeepdreamgenerator.com%2Fgenerator-style)

Still my favorite intro to neuroscience class: [https://www.mcb80x.org/](https://www.youtube.com/redirect?v=MrBzgvUNr4w&redir_token=KFDKj4xYa4cUSJw_9mDu2vxcZzd8MTUyOTM3NTk3N0AxNTI5Mjg5NTc3&event=video_description&q=https%3A%2F%2Fwww.mcb80x.org%2F)

# genetic\_algorithm\_challenge

# Overview

In this demo code we use the [MAGIC Gamma Telescope dataset](https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope) to build a classifer. The classifier will train on the dataset and then be able to classify whether or not some energy is either Gamma Radiation or Hadron Radiation. Instead of guessing and checking the best ML model and hyperparameters to use, we use a genetic programming library called [tpot](https://github.com/rhiever/tpot) to do that for us by trying out a bunch of them. See [this](https://github.com/rhiever/tpot/tree/master/tutorials/MAGIC%20Gamma%20Telescope) link for an IPython notebook version of this code.

# Dependencies

* Numpy
* tpot
* scikit-learn
* pandas

Use [pip](https://pypi.python.org/pypi/pip) to install any missing dependencies

# Usage

To run the demo code, after installing the dependencies, just run the following in terminal

python3 demo.py

# Challenge

The challenge for this module is to use the TPOT library to make a discovery based on a question you pose. '

Step 1 - Download this Climate Change Dataset

Step 2 - Think of a question that [this](https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data) dataset will help you answer like "Has the temperature in India risen over the past 20 years?"

Step 3 - Clean the data and use TPOT to help you build a machine learning pipeline to answer your question

Step 4 - Post in your GitHub!

Great chapter on Genetic Algorithms: [http://natureofcode.com/book/chapter-...](https://www.youtube.com/redirect?q=http%3A%2F%2Fnatureofcode.com%2Fbook%2Fchapter-9-the-evolution-of-code%2F&event=video_description&v=dSofAXnnFrY&redir_token=7-Md9qxpEBhjtT6VnALCV6eOZjV8MTUyOTM3NjEwNEAxNTI5Mjg5NzA0)

Link to TPOT: [https://github.com/rhiever/tpot](https://www.youtube.com/redirect?q=https%3A%2F%2Fgithub.com%2Frhiever%2Ftpot&event=video_description&v=dSofAXnnFrY&redir_token=7-Md9qxpEBhjtT6VnALCV6eOZjV8MTUyOTM3NjEwNEAxNTI5Mjg5NzA0)