Contents

1 README.txt

```
########################################################################
#                                                                      #
#                  README FILE for IML 2019 Hackaton task              #
#                                  Tweets                              #
#                                                                      #
#                  Tal Porezky,   tal.poreky, 311322499                #
#                  Ilya Merkulov, iliamark,   317559631                #
#                  Yoav Cohen,    cyoavc,     307944017                #
#                  Amit Weiss,    amit.weiss, 304873334                #
#                                06.06.2019                            #
#                                                                      #
########################################################################


classifier.py - Contains our implementation of the classifier.
requirements.txt - Contains the requirements for the virtualenv.
countries.txt - contains a list of all the countries in the universe.
README - THIS VERY FILE!
USERS.txt - we!
project.pdf - The description of our choices in design.

The following files are binary files for the classifier:
vcTags.pickle
vcQuestionMark.pickle
vcDotsMark.pickle
vc.pickle
train_avg_len.pickle
vcExclamationMark.pickle
best_classifier_ever.pickle

```

2 USERS.txt

```
tal.porezky, 311322499
iliamark, 317559631
cyoavc, 307944017
amit.weiss, 304873334
```

IML Hackaton

Cohen Yoav
Porezky Tal
Merkulov Ilya
Weiss Amit

June 7, 2019

Shortly after we started working on our classifier we reasoned that our main problem is the feature selection. Both choosing them and deciding whether they are binary or not. The choice was often against basic logic or intuition, was made to the appeasing of the test data.

- Our first step was to use the bag of words to maximum results by making use of the inherent methods of the CountVectorizer class such as stop words, both English and Portuguese.
- It is worth noting, that the first step already achieved 70% success rate and from there, we managed to pull a few minor tweaks to raise the success rate by approximately 10 additional percents. Among these ’tweaks’:
    - Length of each tweet
    - Tags - We counted the number of appearances of the symbol ’@’ representing the amount of tags a tweet is containing and the tendency of a user to tag other people in posts.
    - Countries - We examined whether a word from the file countries.txt, a text file containing every country, appear in a tweet.
    - Marks - Various marks and their combinations were examined. Among them: ’!’ ’,’ ’?’ and the regex ...* which detected a continuous use of 2 or more periods.
- Binary - Some features, much to our surprise, proved to be more useful in a binary score function. Describing whether a country was mentioned in a tweet instead of counting the number of countries, raised the success rate by 3%. However, in others e.g tag, no improvement was achieved.
- Correlation - We have also tried adding features describing correlation between length and mark or tag. These proved to be inconsequential.
- We used various classifiers of sklearn and found the best results with MultinomialNB classifier. We have tried to use the nltk library to process the effects of noun adjectives etcetra and emotions core but with no avail!

4 task1/src/classifier.py

```python
#!/usr/bin/env python
# coding: utf-8

import numpy as np
from sklearn.feature_extraction.text import TfidfTransformer, CountVectorizer
import pandas as pd
import os
from scipy.sparse import hstack
from sklearn.naive_bayes import MultinomialNB
import pickle

# The name of the folder were the tweets are held.
TWEETS_FOLDER = "tweets"
COUNTRIES_FILE = "countries.txt"


def parse_data(data_folder):
    """
    Read all the csv files in the given folder and merge them to a single
    pandas dataframe. Note that this function drops the 'user' column and
    keeps the tweets owner id number at the head of each column.
    :param data_folder:
    :return:
    """
    tweets = list()
    for filename in os.listdir(data_folder):
        # Only choose files with CSV ending and ignore the test file.
        if filename.endswith(".csv") and filename != 'tweets_test_demo.csv':
            current_tweet = pd.read_csv(TWEETS_FOLDER + '/' + filename)
            tweets.append(current_tweet)
            continue
        else:
            continue

    # Join all the tweets to a single pandas array.
    tweets = pd.concat(tweets, sort=True, axis=0)
    return tweets


def len_feature(tweets):
    feature_vec = np.zeros((len(tweets), 1))
    for i, tweet in enumerate(tweets):
        feature_vec[i] = len(tweet.split())
    return feature_vec


def country_parse():
    """
    Generates an np.array of all the countries.
    :param countries_file: The file of countries location.
    :return: An array of all countries in lower case!
    """
    f = open(COUNTRIES_FILE, "r")
    countries = list()
    for country in f:
        countries.append(country.replace('\n', '').lower())
    np.array(countries)
    return countries

```

```

def countries_tweets_count(tweets):
    """
    Receives an array of tweets and counts the appearances of countries
    :param tweets: an array of tweets
    :return: an array of integers
    """
    country_count = np.zeros((len(tweets), 1))
    for i, tweet in enumerate(tweets):
        tweet_country_count = 0
        for country in country_parse():
            tweet_country_count = tweet_country_count + tweet.lower().count(country)
        country_count[i] = tweet_country_count
    for i in range(len(country_count)):
        if country_count[i] != 0:
            country_count[i] = 1
    return country_count


def find_tags(tweets):
    """
    Receives an array of tweets and counts the appearances of countries
    :param tweets: an array of tweets
    :return: an array of integers
    """
    tag_count = np.zeros((len(tweets), 1))
    for i, tweet in enumerate(tweets):
        if tweet.lower().count('@') > 2:
            tag_count[i] = 1
    return tag_count


def classify(tweets):
    """
    Classifies tweets to various twitter stars.
    :param tweets:
    :return:
    """

    vc_file = open("vc.pickle", "rb")
    vc = pickle.load(vc_file)
    vc_file.close()

    vcTags_file = open("vcTags.pickle", "rb")
    vcTags = pickle.load(vcTags_file)
    vcTags_file.close()

    vcExclamationMark_file = open("vcExclamationMark.pickle", "rb")
    vcExclamationMark = pickle.load(vcExclamationMark_file)
    vcExclamationMark_file.close()

    vcQuestionMark_file = open("vcQuestionMark.pickle", "rb")
    vcQuestionMark = pickle.load(vcQuestionMark_file)
    vcQuestionMark_file.close()

    vcDotsMark_file = open("vcDotsMark.pickle", "rb")
    vcDotsMark = pickle.load(vcDotsMark_file)
    vcDotsMark_file.close()

    train_avg_len_file = open("train_avg_len.pickle", "rb")
    train_avg_len = pickle.load(train_avg_len_file)
    train_avg_len_file.close()


    print("HEY")

    tweets = tweets['tweet']

    print(tweets)
    print(tweets.shape)

    # VC transform
    hist = vc.transform(tweets)
    print("hist shape is " + str(hist.shape))

    # Lengths
    test_tweets_lengths = len_feature(tweets)
    test_tweets_lengths = test_tweets_lengths / train_avg_len

    # Find all tags
    tags = vcTags.transform(tweets)

    # Find all marks
    exclamationMarks = np.sum(vcExclamationMark.transform(tweets), axis=1)

    questionMarks = np.sum(vcQuestionMark.transform(tweets), axis=1)

    dotsMarks = np.sum(vcDotsMark.transform(tweets), axis=1)

    countries = countries_tweets_count(tweets)

    tweets = hstack([hist, tags, exclamationMarks, questionMarks, dotsMarks])
    print(test_tweets_lengths)
    print(tweets.shape)
    print(test_tweets_lengths.shape)
    tweets = hstack([tweets, test_tweets_lengths])
    tweets = hstack([tweets, countries])



    classifier_file = open("best_classifier_ever.pickle", "rb")
    classifier = pickle.load(classifier_file)
    classifier_file.close()

    print(classifier.predict(tweets))

    return classifier.predict(tweets)



def train_it():
    # generate train and test data
    data = parse_data(TWEETS_FOLDER)
    test_size = 1
    test_data = data.sample(test_size)
    train_data = data.drop(test_data.index)

    #####################
    # Generate Features #
    #####################

    # Transform all tweets to matrix of features per tweet
    vc = CountVectorizer(ngram_range=(1, 2), stop_words='english')
    hist = vc.fit_transform(train_data['tweet'])

    # Generate X length feature and join it with the previous features
    train_tweets_lengths = len_feature(train_data['tweet'])
    train_avg_len = sum(train_tweets_lengths) / len(train_tweets_lengths)
    train_tweets_lengths = train_tweets_lengths / train_avg_len

    # Generate the length feature of the train set
    test_tweets_lengths = len_feature(test_data['tweet'])
    test_tweets_lengths = test_tweets_lengths / train_avg_len

    # Find all tags
    vcTags = CountVectorizer(token_pattern=r'\b@\b')
    # tags = find_tags(train_data['tweet'])
    tags = vcTags.fit_transform(train_data['tweet'])

    # Find all marks
    vcExclamationMark = CountVectorizer(token_pattern=r'(?u)(?:[!]+)')
    exclamationMarks = np.sum(vcExclamationMark.fit_transform(train_data['tweet']), axis=1)

    vcQuestionMark = CountVectorizer(token_pattern=r'(?u)(?:[?]+)')
    questionMarks = np.sum(vcQuestionMark.fit_transform(train_data['tweet']), axis=1)

    vcDotsMark = CountVectorizer(token_pattern=r'(?u)(?:[.]{2,})')
    dotsMarks = np.sum(vcDotsMark.fit_transform(train_data['tweet']), axis=1)

    # Find Countries
    countries = countries_tweets_count(train_data['tweet'])

    features = hstack([hist, tags, exclamationMarks, questionMarks, dotsMarks])
    features = hstack([features, train_tweets_lengths])
    features = hstack([features, countries])

    # Generate Test Features
    hist_test = vc.transform(test_data['tweet'])
    # tags_test = find_tags(test_data['tweet'])
    tags_test = vcTags.transform(test_data['tweet'])
    exclamation_marks_test = np.sum(vcExclamationMark.transform(test_data['tweet']), axis=1)
    question_marks_test = np.sum(vcQuestionMark.transform(test_data['tweet']), axis=1)
    dots_marks = np.sum(vcDotsMark.transform(test_data['tweet']), axis=1)
    countries_test = countries_tweets_count(test_data['tweet'])

    test_features = hstack([hist_test, tags_test, exclamation_marks_test, question_marks_test, dots_marks])
    test_features = hstack([test_features, test_tweets_lengths])
    test_features = hstack([test_features, countries_test])
    # test_features = hstack([test_features, tags_test])


    # Transform features
    transformer = TfidfTransformer()
    freq_all = transformer.fit_transform(features)
    freq_test = transformer.fit_transform(test_features)

    clf2 = MultinomialNB().fit(freq_all, train_data['user'])
    # predicted2 = clf2.predict(freq_test)

    # Success of prediction on test_data.
    # print(len(predicted2[predicted2 == test_data['user']]) / test_size)


    save_classifier = open("best_classifier_ever.pickle", "wb")
    pickle.dump(clf2, save_classifier)
    save_classifier.close()

    save_vc = open("vc.pickle", "wb")
    pickle.dump(vc, save_vc)
    save_vc.close()

    save_vcTags = open("vcTags.pickle", "wb")
    pickle.dump(vcTags, save_vcTags)
    save_vcTags.close()

    save_vcExclamationMark = open("vcExclamationMark.pickle", "wb")
    pickle.dump(vcExclamationMark, save_vcExclamationMark)
    save_vcExclamationMark.close()

    save_vcQuestionMark = open("vcQuestionMark.pickle", "wb")
    pickle.dump(vcQuestionMark, save_vcQuestionMark)
    save_vcQuestionMark.close()

    save_vcDotsMark = open("vcDotsMark.pickle", "wb")
```

```python
    pickle.dump(vcDotsMark, save_vcDotsMark)
    save_vcDotsMark.close()

    save_train_avg = open("train_avg_len.pickle", "wb")
    pickle.dump(train_avg_len, save_train_avg)
    save_train_avg.close()


if __name__ == '__main__':


    file ="tweets_test_demo.csv"
    current_tweet = pd.read_csv(file)

    classify(current_tweet)
```

5 task1/src/countries.txt

```
Afghanistan
Albania
Algeria
Andorra
Angola
Antigua
Argentina
Armenia
Australia
Austria
Azerbaijan
Bahamas
Bahrain
Bangladesh
Barbados
Belarus
Belgium
Belize
Benin
Bhutan
Bolivia
Bosnia Herzegovina
Botswana
Brazil
Brunei
Bulgaria
Burkina
Burundi
Cambodia
Cameroon
Canada
Cape Verde
Central African Rep
Chad
Chile
China
Colombia
Comoros
Congo
Costa Rica
Croatia
Cuba
Cyprus
Czech Republic
Denmark
Djibouti
Dominica
Dominican Republic
East Timor
Ecuador
Egypt
El Salvador
Equatorial Guinea
Eritrea
Estonia
Ethiopia
Fiji
Finland
France
```

```
Gabon
Gambia
Georgia
Germany
Ghana
Greece
Grenada
Guatemala
Guinea
Guinea-Bissau
Guyana
Haiti
Honduras
Hungary
Iceland
India
Indonesia
Iran
Iraq
Ireland
Israel
Italy
Ivory Coast
Jamaica
Japan
Jordan
Kazakhstan
Kenya
Kiribati
Korea North
Korea South
Kosovo
Kuwait
Kyrgyzstan
Laos
Latvia
Lebanon
Lesotho
Liberia
Libya
Liechtenstein
Lithuania
Luxembourg
Macedonia
Madagascar
Malawi
Malaysia
Maldives
Mali
Malta
Marshall Islands
Mauritania
Mauritius
Mexico
Micronesia
Moldova
Monaco
Mongolia
Montenegro
Morocco
Mozambique
Myanmar
Namibia
Nauru
Nepal
Netherlands
New Zealand
Nicaragua
```

128 Niger
129 Nigeria
130 Norway
131 Oman
132 Pakistan
133 Palau
134 Panama
135 Papua New Guinea
136 Paraguay
137 Peru
138 Philippines
139 Poland
140 Portugal
141 Qatar
142 Romania
143 Russian Federation
144 Rwanda
145 St Kitts & Nevis
146 St Lucia
147 Saint Vincent & the Grenadines
148 Samoa
149 San Marino
150 Sao Tome & Principe
151 Saudi Arabia
152 Senegal
153 Serbia
154 Seychelles
155 Sierra Leone
156 Singapore
157 Slovakia
158 Slovenia
159 Solomon Islands
160 Somalia
161 South Africa
162 South Sudan
163 Spain
164 Sri Lanka
165 Sudan
166 Suriname
167 Swaziland
168 Sweden
169 Switzerland
170 Syria
171 Taiwan
172 Tajikistan
173 Tanzania
174 Thailand
175 Togo
176 Tonga
177 Trinidad & Tobago
178 Tunisia
179 Turkey
180 Turkmenistan
181 Tuvalu
182 Uganda
183 Ukraine
184 United Arab Emirates
185 United Kingdom
186 United States
187 Uruguay
188 Uzbekistan
189 Vanuatu
190 Vatican City
191 Venezuela
192 Vietnam
193 Yemen
194 Zambia
195 Zimbabwe

6 task1/src/requirements.txt

```
alabaster==0.7.8
antlr==2.7.5rc1
apsw==3.16.2.post1
apt-xapian-index==0.49
arandr==0.1.9
asn1crypto==0.24.0
astroid==1.6.5
atomicwrites==1.1.5
attrs==17.4.0
autobahn==17.10.1
Automat==0.6.0
avro==1.8.2
Axiom==0.7.5
Babel==2.4.0
backports-abc==0.5
backports.functools-lru-cache==1.5
backports.shutil-get-terminal-size==1.0.0
backports.ssl-match-hostname==3.5.0.1
basemap==1.1.0
bcrypt==3.1.4
Beaker==1.9.1
BeautifulSoup==3.2.1
beautifulsoup4==4.6.1
biom-format==2.1.6
biopython==1.72
bleach==2.1.3
blinker==1.4
bottle==0.12.13
brial==1.0.1
Brlapi==0.6.7
bx-python==0.8.1
bz2file==0.98
bzr==2.8.0.dev1
BzrTools==2.6.0
CacheControl==0.11.7
cajarename==18.1.10
cbor==1.0.0
ccsm==0.9.13.1
certifi==2018.4.16
cffi==1.11.5
Chameleon==2.24
chardet==3.0.4
Cheetah3==3.1.0
CherryPy==8.9.1
click==6.7
cloudpickle==0.5.2
Coherence==0.6.6.2
colorama==0.3.7
compizconfig-python==0.9.13.1
configobj==5.0.6
configparser==3.5.0
constantly==15.1.0
cookies==2.2.1
cryptography==2.3
cssselect==1.0.3
cssutils==1.0.2
cutadapt==1.18
cvxopt==1.1.9
cwltool==1.0.20180302231433
```

```
cycler==0.10.0
cypari2==1.1.4
Cython==0.28.4
dap==2.2.6.7
dblatex==0.3.10
decorator==4.3.0
defer==1.0.6
defusedxml==0.5.0
deluge==1.3.15
devscripts==2.11.2
dissy==9
distro==1.0.1
Django==1.11.15
django-ajax-selects==1.7.0
django-app-plugins==0.1.1
django-auth-ldap==1.4.0
django-dajax==0.9.2
django-dajaxice==0.7
django-extensions==1.8.1
django-piston==0.2.3
django-websocket==0.3.0
Djblets==0.7a0.dev0
dnspython==1.15.0
docutils==0.14
drmaa==0.5
dulwich==0.19.6
elib.intl==0.0.3.dev0
entrypoints==0.2.3.post2
enum34==1.1.6
Epsilon==0.7.1
et-xmlfile==1.0.1
fastimport==0.9.8
feedparser==5.2.1
Flask==1.0.2
Flask-AutoIndex==0.6.1
Flask-Babel==0.11.2
Flask-OldSessions==0.10
Flask-OpenID==1.2.5
Flask-Silk==0.2
foolscap==0.13.1
FormEncode==1.3.0
fpconst==0.7.2
fpylll==0.3.0.dev0
funcsigs==1.0.2
functools32==3.2.3.post2
future==0.15.2
futures==3.2.0
GDAL==2.3.1
gdata==2.0.18
gdmodule==0.59
glpk==0.4.52
gmpy==1.17
GnuPGInterface==0.3.2
gyp==0.1
h5py==2.8.0
html5-parser==0.4.4
html5lib==1.0.1
HTSeq===0.6.1p1
httplib2==0.9.2
hupper==1.2
hyperlink==17.3.1
idna==2.6
imageio==2.2.0
imagesize==1.0.0
incremental==16.10.1
ipaddr==2.2.0
ipaddress==1.0.17
IPy==0.83
```

```
ipykernel==4.8.2
ipython==5.5.0
ipython-genutils==0.2.0
ipywidgets==6.0.0
isodate==0.6.0
isort==4.3.4
itsdangerous==0.24
jdcal==1.0
jedi==0.12.0
Jinja2==2.10
joblib==0.12.1
jsonschema==2.6.0
jupyter-client==5.2.3
jupyter-core==4.4.0
keyring==13.1.0
keyrings.alt==3.0
kiwisolver==1.0.1
launchpadlib==1.10.6
lazr.restfulclient==0.14.0
lazr.uri==1.0.3
lazy-object-proxy==1.3.1
leveldb==0.1
libvirt-python==4.5.0
lockfile==0.12.2
logilab-common==1.4.1
Louie==1.1
louis==3.6.0
lxml==4.2.3
lz4==1.1.0
M2Crypto==0.27.0
Mako==1.0.7
Markdown==2.6.9
MarkupSafe==1.0
mate-menu==18.4.3
matplotlib==2.2.2
mccabe==0.6.1
mechanize==0.2.5
mercurial==4.7
mistune==0.8.3
mock==2.0.0
mod-python==3.3.1
moin==1.9.9
more-itertools==4.2.0
mpi4py==2.0.0
mpmath==1.0.0
msgpack==0.5.6
mysql-connector-python==8.0.11
mysql-utilities==1.6.4
mysqlclient==1.3.10
nbconvert==5.3.1
nbformat==4.4.0
netifaces==0.10.4
networkx==2.1
Nevow==0.14.2
nltk==3.3
nose==1.3.7
notebook==5.4.1
numexpr==2.6.5
numpy==1.16.4
numpydoc==0.7.0
oauth==1.0.1
oauthlib==2.0.6
olefile==0.45.1
openpyxl==2.4.9
packaging==17.1
PAM==0.4.2
pandas==0.23.3
pandocfilters==1.4.2
```

```
parameterized==0.6.1
paramiko==2.4.0
parsedatetime==2.4
parso==0.2.1
passlib==1.7.1
Paste==2.0.3
PasteDeploy==1.5.2
PasteScript==2.0.2
path.py==11.0.1
pathlib2==2.3.2
patsy==0.4.1+dev
pbr==3.1.1
pdfrw==0.4
pexpect==4.6.0
pickleshare==0.7.4
Pillow==5.2.0
pkgconfig==1.3.1
plaster==1.0
plaster-pastedeploy==0.5
pluggy==0.6.0
ply==3.11
Pmw==1.3.2
progressbar==2.3
prompt-toolkit==1.0.15
protobuf==3.0.0
psphere==0.5.2
psutil==5.4.6
psycopg2==2.7.5
py==1.5.4
py-ubjson==0.8.5
pyasn1==0.4.2
pyasn1-modules==0.2.1
PyAudio==0.2.11
pycairo==1.16.2
pycodestyle==2.3.1
pycparser==2.18
pycrypto==2.6.1
pycups==1.9.73
pycurl==7.43.0.1
pydot==1.2.4
pyenchant==2.0.0
pyExcelerator==0.6.4.1
pyflakes==1.6.0
pyfribidi==0.11.0
pygame===1.9.1release
pyglet==1.3.0
Pygments==2.2.0
pygobject==3.28.2
pygpgme==0.3
pygpu==0.6.9
pygraphviz==1.4rc1
PyGreSQL==5.0.5
PyICU==2.0.3
pyinotify==0.9.6
PyJWT==1.6.4
pyliblzma==0.5.3
pylibssh2==1.0.0
pylint==1.9.2
pymol==2.1.0
PyMySQL==0.8.1
PyNaCl==1.2.1
pyodbc==4.0.22
PyOpenGL==3.1.0
pyOpenSSL==18.0.0
pyparsing==2.2.0
pyproj==1.9.5.1
pyqi==0.3.2
pyquery==1.2.9
```

```
pyramid==1.9.1
PyRRD==0.1.0
pysam==0.14
pysaml2==4.0.2
pyserial==3.4
pyshp==1.2.12
pysqlite==2.7.0
Pyste==0.9.10
pytest==3.6.4
python-apt==1.6.2
python-dateutil==2.6.1
python-debian==0.1.32
python-debianbts==2.7.2
python-distutils-extra==2.39
python-gflags==1.5.1
python-gnutls==3.0.0
python-ldap==3.1.0
python-libtorrent==1.1.5
python-lzo==1.12
python-magic==0.4.16
python-memcached==1.53
python-openid==2.2.5
python-snappy==0.5.3
python-xlib==0.23
PyTrie==0.2
pytz==2018.5
PyWavelets==0.5.1
pyxdg==0.25
PyYAML==3.12
pyzmq==17.1.0
qcli==0.1.1
qrcode==6.0
QtAwesome==0.4.4
qtconsole==4.3.1
QtPy==1.3.1
rabbitvcs==0.16.0
radiotray==0.7.3
rdflib==4.2.2
rdflib-jsonld==0.4.0
recaptcha-client==1.0.6
regex==2018.6.9
reportbug==6.6.6
reportlab==3.5.2
repoze.lru==0.7
repoze.who==2.2
requests==2.18.4
responses==0.9.0
roman==2.0.0
rope==0.10.5
Routes==2.4.1
rpm==4.14.1
rpy2==2.8.6
rrdtool==0.1.10
## FIXME: could not find svn URL in dependency_links for this package:
rst2pdf===0.93.dev-r0
ruamel.yaml==0.15.34
rubber==1.4
sage==8.2
sagenb==1.0.1
sagenb-export==3.2
scandir==1.8
scgi==1.13
schema-salad==2.6.20171201034858
scikit-image==0.14.0
scikit-learn==0.20.3
scipy==1.2.1
SecretStorage==2.3.1
Send2Trash==1.4.2
```

```
service-identity==16.0.0
setproctitle==1.1.10
setuptools-scm==3.0.6
shellescape==3.4.1
simplegeneric==0.8.1
simplejson==3.15.0
singledispatch==3.4.0.3
six==1.11.0
sklearn==0.0
snowballstemmer==1.2.1
SOAPpy==0.12.22
SPARQLWrapper==1.7.6
Sphinx==1.7.6
sphinx-rtd-theme==0.4.0
spyder==3.2.6
SQLAlchemy==1.2.8
sqlparse==0.2.4
statsmodels==0.8.0
stevedore==1.28.0
subprocess32==3.5.2
suds-jurko==0.7.dev0
sympy==1.2
tables==3.4.4
tabulate==0.8.2
tagpy==2013.1
Tempita==0.5.2
termcolor==1.1.0
terminado==0.8.1
testpath==0.3.1
Theano==1.0.2
tornado==5.0.2
tqdm==4.23.4
traitlets==4.3.2
translationstring==1.3
trollius==2.0.1
Twisted==18.4.0
Twisted-Conch==11.0.0
Twisted-Core==11.0.0
Twisted-Names==11.0.0
Twisted-Web==11.0.0
txaio==2.8.1
typing==3.6.4
u-msgpack-python==2.1
urlgrabber==3.10.2
urllib3==1.22
uTidylib==0.3
vcversioner==2.16.0.0
venusian==1.1.0
virtualenv==15.1.0
virtualenv-clone==0.3.0
virtualenvwrapper==4.3.1
wadllib==1.3.3
waitress==1.1.0
wcwidth==0.1.7
webencodings==0.5
WebOb==1.7.3
WebTest==2.0.28
Werkzeug==0.14.1
wrapt==1.9.0
wsaccel==0.6.2
wstools==0.4.3
wxPython==2.8.12.1
wxPython-common==3.0.2.0
xappy==0.5
xlrd==1.1.0
XlsxWriter==0.9.6
xlwt==1.3.0
xmltodict==0.11.0
```

```
xopen==0.3.3
yum-metadata-parser==1.1.4
zope.component==4.3.0
zope.configuration==4.0.3
zope.deprecation==4.1.2
zope.event==4.2.0
zope.hookable==4.0.4
zope.i18nmessageid==4.0.3
zope.interface==4.3.2
zope.schema==4.4.2
zope.testing==4.6.2
```

Index of comments

4.1 Seems like you've done some deep research to decide which features to extract - well done!

It could be nice to see some of the work you described in a more visualized fashion, and also to extend some your explanation about the model you have decided to use.

You showed creativity and understanding of different methods you have studied in the course - good job!