Introduction to Machine Learning (67577)

IML Hackathon 2019

Teacher: Dr. Matan Gavish

TAs: Erez Peterfreund, Gad Zalcberg, Gadi Mintz

June 5, 2019

Contents

General Instructions **2**
Evaluation 3

Before submission - Mandatory **4**

Task 1: Who Tweeted What? **5**

Task 2: Chicago crimes **6**

General Instructions

- The hackathon starts on Wednesday, 5.6, 7pm, and ends on Friday, 7.6, 11am.
- You may work in teams of two, three or four students.
- You can find the data for both tasks here
- In the next page you will find two learning tasks. Choose **one** of them, and solve it as well as you can. All the materials for the tasks are available on the course moodle.
- To enable us to run your code without errors, we require all submissions to use virtualenv and define an environment for the project. Then submit a requirements.txt file, generated using the freeze command, that summarizes the environment you use. Instructions on the virtualenv and the freeze command can be found here and here.

  * Virtual environment will allow you to install via the command `pip install <package_name>` and use packages that are not installed on the CS computers.
- Solutions must be written in Python3 (the tester will be using Python 3.6.6, so please ensure your code works with this version). To use Python 3 on CS computers, use `python3` instead of `python`.
- Submit through the course moodle a zip file named id_number.zip, where id_number is a 9-digits id of one team member. Submit your solution using his moodle account (it is not important which team member submits the solution). The file should include:
  - A folder named `task1` or `task2`, depending on the task you choose. This folder should include all your code and supplementary files. Follow those carefully, as there will be automatic tests.
  - `README.txt` - contains a file list and a brief description of each file.
  - `USERS.txt` - contains the logins and IDs of all the team members. Use one line per student, in the format login, ID.
  - `project.pdf` - a written description of your project, up to 2 pages long, as a PDF file. Explain your solution, describe your work process, and do your best to justify any design decisions you have made. Feel free to include supporting figures if you want.
  - `requirements.txt` - the required packages for your virtual environment, as described earlier.
- We have provided a basic test script to help you make sure your submission follows the requirements. Nevertheless, you should **not** rely solely on this script, as it only performs basic checks (for example, the submitted `Classifier` class for task 2 is not checked). You can find it here.
  - The script receives one command-line argument, which is a path - either to your submission directory or to your submission zip file.
  - Usage: `python test_submission.py <submission-dir-or-zip-path>`
- Make sure to follow best coding practices, including good documentation of your code.

- We advise that you start by finding a basic solution that works. Then try to improve it as much as you can, until you're out of time.

Evaluation

In both tasks, your grade will be determined by two factors (in descending order of importance):

1. The quality of your prediction on the test set, which will be future examples, i.e. future tweets and future crimes in Chicago. The loss function will be the 0-1 loss. This will be based on a ranking between the performance of all participants, where any team that will produce a classifier that has some reasonable performance ('reasonable' depends on which task you chose) will receive a grade of 80 and the top ranking teams will receive a grade of 100, and there will be some distribution between these two extremes.
2. The quality and depth of your written description of your work that you wrote in the PDF. This will be based on clarity and originality.

Good Luck!

Before submission - Mandatory

IMPORTANT: Our tests have **zero tolerance** hence you have to follow the requested formats. Otherwise, you will receive no points.
Zero tolerance cases:

- You are obligated to submit files in the defined file folder structure, see below.
- Code has to be written in Python 3. Please make sure code works on python 3.6.6 or lower as the tester will be using Python 3.6.6.
- Output data has to be in the matching format.
- Your code should only call files that are in the folder you submit, and not use their full path. For example : ”*weights.txt*” is good, and ”$C : \backslash Users\backslash Miriam\backslash IML\backslash weights.txt$” is bad.
- You are required to submit `requirements.txt`.
- Your code cannot throw any exception or error when running.
- Your code has to finish prediction (including any necessary loading or initialization) in 5 minutes on CS computers.
- After zipping your project, it should take no more than 20 MB.
- You have to submit only **one** of the tasks (see directory structure).
- Submit the zip file only from **one** user in the team
- You must use the following directory structure in your submission zip file:

File or directory	**Description**
`<student-id>.zip`	Submission zip file
`├── USERS`	
`├── README`	
`├── project.pdf`	
`├── task1`	***Only if submitting task 1***
`│   └── src`	
`│       ├── requirements.txt`	
`│       ├── classifier.py`	
`│       └── <other source files>`	
`└── task2`	***Only if submitting task 2***
`    └── src`	
`        ├── requirements.txt`	
`        ├── classifier.py`	
`        └── <other source files>`	

Task 1: Who Tweeted What?

Task: You are given a dataset of tweets posted by 10 celebrities. Your task is to learn to predict which celebrity tweeted what. You will be tested on new tweets that those celebrities will post after the hackathon is over.

Dataset: The celebrities in this dataset are Donald Trump, Lebron James, Lady Gaga, Ellen Degeneres, Conan O’brien, Cristiano Ronaldo, Jimmy kimmel, Joe Biden, Arnold schwarzenegger and Kim Kardashian. The file `names.txt` gives each celebrity an ID number between 0 to 9. The tweets are stored in the folder here. Each `csv` file contains records of the person’s tweets and each record corresponds to a single tweet, where the first column in the file corresponds to the celebrity ID and the second column is the tweet’s text.

Code Requirements: You need to implement the module `classifier.py`. This module has a method `classify` that receives a one dimension’s numpy array of tweets strings and predicts for each one which celebrity posted it. More specifically, during test time it should return a list (or a one dimensional numpy array) of labels (ints between {0-9} for the 10 classes). Do not change the signature of this method. Except the tweets files, you are supplied with a very simple data file named `tweets_test_demo.csv`. please load it to test your api.
NOTICE, since we’re not going to train your model, you are required to load your trained model in this method. Please verify this method works as expected (with all the additional files it is required - s.a. the weights file)

In to represent text as a feature vector you will need to be creative. We suggest to start by examining the ’sklearn.feature_extraction.text’ package. One simple way of representing a document as a vector is:

Bag of Words (BoW): In this representation, each document is a vector $\mathbf{x} = \{0, 1\}^n$, where n is the size of the dictionary and $x_i = 1$ iff the word corresponding to index i appears in the document. Given a set of m documents, we can interpret the bag-of-words representation of the m documents as a joint probability over a random variable x, indicating the identity of a document (thus taking values in $[m]$), and a random variable y, indicating the identity of a word in the dictionary (thus taking values in $[n]$).

Task 2: Chicago crimes

Task: You are required to build an algorithm that, given unseen feature vectors (see below description) predicts which kind of crime (from 8 classes) matches these features.

Dataset: This dataset reflects reported incidents of crime that have occurred in the City of Chicago over the past year, minus the most recent seven days of data. Data is extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system. The data includes features such as the time (date and time of the day) and location (in several spatial resolutions) of each crime. The labels in this task are the Primary Types of each crime.
The eight crime Primary Types in the given dataset are:
'THEFT', 'BATTERY', 'NARCOTICS', 'BURGLARY', 'WEAPONS VIOLATION', 'DECEPTIVE PRACTICE', 'CRIMINAL TRESPASS', and 'PROSTITUTION'.
The file crime_types.txt gives each primary type an ID number between 0 to 7.

Code Requirements: You need to implement the module `classifier.py`. This module has a method `classify` that receives a `pandas` data frame with the feature columns (as in the training set) of crimes and predicts for each one which crime has occurred. More specifically, during test time it will return a list (or a one dimension numpy array) of labels (ints between {0-7} for the 8 classes). Do not change the signature of this method.
NOTICE, since we're not going to train your model, you are required to load your trained model in this method. Please verify this method works as expected (with all the additional files it is required - s.a. the weights file)

Label descriptions:

- **Primary Type** - The primary description of the IUCR code indicating which type of crime took place - string

Feature descriptions: feature name - description - type

- **ID** - Unique identifier for the record - int
- **Date** - Date when the incident occurred. This is sometimes a best estimate - Date and Time
- **Year** - Year the incident occurred - int
- **Updated On** - Date and time the record was last updated.
- **Block** - The partially redacted address where the incident occurred, placing it on the same block as the actual address - string
- **Location Description** - Description of the location where the incident occurred - string
- **Arrest** - Indicates whether an arrest was made - boolean

- **Domestic** - Indicates whether the incident was domestic-related as defined by the Illinois Domestic Violence Act - boolean
- **Beat** - Indicates the beat where the incident occurred. A beat is the smallest police geographic area each beat has a dedicated police beat car. Three to five beats make up a police sector, and three sectors make up a police district. The Chicago Police Department has 22 police districts - string
- **District** - Indicates the police district where the incident occurred - string
- **Ward** - The ward (City Council district) where the incident occurred - int
- **Community Area** - Indicates the community area where the incident occurred. Chicago has 77 community areas - string
- **X Coordinate** - The x coordinate of the location where the incident occurred in State Plane Illinois East NAD 1983 projection. This location is shifted from the actual location for partial redaction but falls on the same block - float
- **Y Coordinate** - The y coordinate of the location where the incident occurred in State Plane Illinois East NAD 1983 projection. This location is shifted from the actual location for partial redaction but falls on the same block - float
- **Latitude** - The latitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block - float
- **Longitude** - The longitude of the location where the incident occurred. This location is shifted from the actual location for partial redaction but falls on the same block - float