```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Alessandro Parisi \n",
      "last updated: 2019-02-19 \n",
      "\n",
      "CPython 3.5.4\n",
      "IPython 6.1.0\n",
      "\n",
      "numpy 1.16.1\n",
      "pandas 0.20.3\n",
      "matplotlib 2.0.2\n",
      "sklearn 0.20.0\n",
      "seaborn 0.8.0\n"
     ]
    }
```

```
   ],
   "source": [
    "%load_ext watermark\n",
    "%watermark -a \"Alessandro Parisi\" -u -d -v -p
numpy,pandas,matplotlib,sklearn,seaborn\n",
    "# to install watermark launch 'pip install watermark' at command line"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {
    "collapsed": true
   },
   "outputs": [],
   "source": [
    "import pandas as pd\n",
    "import numpy as np\n",
    "from sklearn import *\n",
    "from sklearn.linear_model import LogisticRegression\n",
    "from sklearn.metrics import accuracy_score\n",
    "import warnings \n",
    "warnings.simplefilter('ignore')\n"
   ]
```

```
   },
   {
    "cell_type": "code",
    "execution_count": 3,
    "metadata": {
     "collapsed": true
    },
    "outputs": [],
    "source": [
     "phishing_dataset = np.genfromtxt('../datasets/phishing_dataset.csv', delimiter=',', dtype=np.int32)\n",
     "samples = phishing_dataset[:,:-1]\n",
     "targets = phishing_dataset[:, -1]"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 4,
    "metadata": {
     "collapsed": true
    },
    "outputs": [],
    "source": [
     "from sklearn.model_selection import train_test_split\n",
```

```
      "\n",
      "training_samples, testing_samples, training_targets, testing_targets =
train_test_split(\n",
      "                samples, targets, test_size=0.2, random_state=0)"
     ]
    },
    {
     "cell_type": "code",
     "execution_count": 5,
     "metadata": {
      "collapsed": true
     },
     "outputs": [],
     "source": [
      "log_classifier = LogisticRegression()"
     ]
    },
    {
     "cell_type": "code",
     "execution_count": 6,
     "metadata": {},
     "outputs": [
      {
       "data": {
```

```
          "text/plain": [
            "LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,\n",
            "                   intercept_scaling=1, max_iter=100, multi_class='warn',\n",
            "                   n_jobs=None, penalty='l2', random_state=None,
solver='warn',\n",
            "                   tol=0.0001, verbose=0, warm_start=False)"
          ]
        },
        "execution_count": 6,
        "metadata": {},
        "output_type": "execute_result"
      }
    ],
    "source": [
      "log_classifier.fit(training_samples, training_targets)"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 7,
    "metadata": {
      "collapsed": true
    },
    "outputs": [],
```

```
  "source": [
    "predictions = log_classifier.predict(testing_samples)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 8,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "Logistic Regression accuracy: 91.72320217096338\n"
      ]
    }
  ],
  "source": [
    "accuracy = 100.0 * accuracy_score(testing_targets, predictions)\n",
    "print (\"Logistic Regression accuracy: \" + str(accuracy))"
  ]
},
{
  "cell_type": "code",
```

```
    "execution_count": null,
    "metadata": {
      "collapsed": true
    },
    "outputs": [],
    "source": []
  }
],
"metadata": {
  "kernelspec": {
    "display_name": "Py35",
    "language": "python",
    "name": "py35"
  },
  "language_info": {
    "codemirror_mode": {
      "name": "ipython",
      "version": 3
    },
    "file_extension": ".py",
    "mimetype": "text/x-python",
    "name": "python",
    "nbconvert_exporter": "python",
    "pygments_lexer": "ipython3",
```

```
      "version": "3.5.4"
    }
  },
  "nbformat": 4,
  "nbformat_minor": 2
}
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {},
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Alessandro Parisi \n",
            "last updated: 2019-02-20 \n",
            "\n",
            "CPython 3.5.4\n",
            "IPython 6.1.0\n",
            "\n",
            "numpy 1.16.1\n",
```

```
            "pandas 0.20.3\n",
            "matplotlib 2.0.2\n",
            "sklearn 0.20.0\n",
            "seaborn 0.8.0\n"
          ]
        }
      ],
      "source": [
        "%load_ext watermark\n",
        "%watermark -a \"Alessandro Parisi\" -u -d -v -p numpy,pandas,matplotlib,sklearn,seaborn\n",
        "# to install watermark launch 'pip install watermark' at command line"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 2,
      "metadata": {
        "collapsed": true
      },
      "outputs": [],
      "source": [
        "import matplotlib.pyplot as plt\n",
        "import csv\n",
```

```
        "from textblob import TextBlob\n",
        "import pandas\n",
        "import sklearn\n",
        "import numpy as np\n",
        "\n",
        "import nltk\n",
        "# nltk.download('popular')\n",
        "# nltk.download('punkt')\n",
        "# nltk.download('averaged_perceptron_tagger')\n",
        "# nltk.download('wordnet')\n",
        "\n",
        "from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer\n",
        "from sklearn.naive_bayes import MultinomialNB\n",
        "from sklearn.metrics import classification_report, accuracy_score\n",
        "from sklearn.model_selection import train_test_split \n",
        "\n",
        "from defs import get_tokens\n",
        "from defs import get_lemmas"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 3,
```

```
    "metadata": {
     "collapsed": true
    },
    "outputs": [],
    "source": [
     "sms = pandas.read_csv('../datasets/sms_spam_no_header.csv', sep=',', names=[\"type\", \"text\"])"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 4,
    "metadata": {
     "collapsed": true
    },
    "outputs": [],
    "source": [
     "text_train, text_test, type_train, type_test = train_test_split(sms['text'], sms['type'], test_size=0.3)"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 5,
    "metadata": {
```

```
    "collapsed": true
   },
   "outputs": [],
   "source": [
    "bow = CountVectorizer(analyzer=get_lemmas).fit(text_train)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "metadata": {
    "collapsed": true
   },
   "outputs": [],
   "source": [
    "sms_bow = bow.transform(text_train)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "metadata": {
    "collapsed": true
   },
```

```
  "outputs": [],
  "source": [
    "tfidf = TfidfTransformer().fit(sms_bow)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 8,
  "metadata": {
    "collapsed": true
  },
  "outputs": [],
  "source": [
    "sms_tfidf = tfidf.transform(sms_bow)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 9,
  "metadata": {
    "collapsed": true
  },
  "outputs": [],
  "source": [
```

```
    "spam_detector = MultinomialNB().fit(sms_tfidf, type_train)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 10,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "predicted: ham\n",
      "expected: ham\n"
     ]
    }
   ],
   "source": [
    "msg = sms['text'][25]\n",
    "msg_bow = bow.transform([msg])\n",
    "msg_tfidf = tfidf.transform(msg_bow)\n",
    "\n",
    "print ('predicted:', spam_detector.predict(msg_tfidf)[0])\n",
    "print ('expected:', sms.type[25])"
```

```
  ]
},
{
  "cell_type": "code",
  "execution_count": 11,
  "metadata": {
    "collapsed": true
  },
  "outputs": [],
  "source": [
    "predictions = spam_detector.predict(sms_tfidf)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 12,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "accuracy 0.789797487823635\n"
      ]
```

```
   }
  ],
  "source": [
   "print ('accuracy', accuracy_score(sms['type'][:len(predictions)], predictions))"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 13,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "              precision    recall  f1-score   support\n",
     "\n",
     "         ham       0.87      0.90      0.88      3382\n",
     "        spam       0.13      0.10      0.11       519\n",
     "\n",
     "   micro avg       0.79      0.79      0.79      3901\n",
     "   macro avg       0.50      0.50      0.50      3901\n",
     "weighted avg       0.77      0.79      0.78      3901\n",
     "\n"
```

```
        ]
       }
     ],
     "source": [
       "print (classification_report(sms['type'][:len(predictions)], predictions))"
     ]
   }
 ],
 "metadata": {
   "kernelspec": {
     "display_name": "Py35",
     "language": "python",
     "name": "py35"
   },
   "language_info": {
     "codemirror_mode": {
       "name": "ipython",
       "version": 3
     },
     "file_extension": ".py",
     "mimetype": "text/x-python",
     "name": "python",
     "nbconvert_exporter": "python",
     "pygments_lexer": "ipython3",
```

"version": "3.5.4"

}

},

"nbformat": 4,

"nbformat_minor": 2

}