Nem-funkcionális tesztelés dokumentáció

Bevezetés

Az alábbi PDF az Integrációs és ellenőrzési technikák tárgy ellenőrzés témakörébe kapcsolódó házi feladat, nem-funkcionális jellemzők vizsgálata részfeladathoz tartozó dokumentáció.

A teszteléshez szükséges statisztikák számításához a 'pstats', 'cProfile', 'profile' Python könyvtárak voltak használva, függvény készítéséhez 'matplotlib' Python könyvtár, valamint nyelvtani helyesség ellenőrzéséhez a Language Tool eszköz.

A feladat alatt a 'Döntési fa klasszifikációhoz' tartozó algoritmuson készült Terhelés és Stressz teszt, valamint Dokumentáció teszt az egész projekten a tesztesetekkel és az új fejlesztésekhez tartozó dokumentációkkal együtt.

Terhelés teszt

Döntési fa klasszifikációhoz - (decision_tree_classification.py)

Terhelés tesztelés közben az algoritmus azon pontjait kerestem, amelyek érzékenyek a terhelésre, akadályozhatják a program futását. Mivel az algoritmus a bemenet példáiból tanul, valószínűleg a bemenet méretének növelésével lehet terhelni a rendszert.

1. esetben az adathalmaz 3 osztály 150 példáját tartalmazza, amiket 4 tulajdonság jellemez. (Iris plants dataset)
2. esetben az adathalmaz 2 osztály 569 példáját tartalmazza, amiket 30 tulajdonság jellemez. (Breast cancer wisconsin (diagnostic) dataset)

```
cProfile.run('iris = load_iris()', 'iris-load')
pstats.Stats('iris-load').strip_dirs().sort_stats(SortKey.TIME).print_stats(5)

X, y = iris.data, iris.target
```

```
Sat May  2 21:09:12 2020    iris-load

         692 function calls in 0.002 seconds

   Ordered by: internal time
   List reduced from 28 to 5 due to restriction <5>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      301    0.001    0.000    0.001    0.000 {built-in method numpy.array}
        2    0.001    0.000    0.001    0.000 {built-in method io.open}
        1    0.001    0.001    0.002    0.002 _base.py:207(load_data)
        1    0.000    0.000    0.000    0.000 {built-in method builtins.next}
      300    0.000    0.000    0.001    0.000 _asarray.py:16(asarray)
```

```
Sat May  2 21:11:04 2020    cancer-load

         2397 function calls in 0.023 seconds

   Ordered by: internal time
   List reduced from 28 to 5 due to restriction <5>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.010    0.010    0.022    0.022 _base.py:207(load_data)
     1140    0.010    0.000    0.010    0.000 {built-in method numpy.array}
        2    0.001    0.001    0.001    0.001 {built-in method io.open}
     1138    0.000    0.000    0.011    0.000 _asarray.py:16(asarray)
       17    0.000    0.000    0.000    0.000 {built-in method _codecs.charmap_decode}
```

Az adatok betöltésében nincs kimagasló változás, a függvényhívások száma és az igényelt idő közel lineárisan növekszik a bemenet nagyságával.

```
tree = DecisionTree()

cProfile.run('tree.train(X_train, y_train, max_depth=2, min_samples=1)', 'iris-learn')
pstats.Stats('iris-learn').strip_dirs().sort_stats(SortKey.TIME).print_stats(5)
```

```
Sat May  2 21:10:11 2020    iris-learn

         31135 function calls (31133 primitive calls) in 0.051 seconds

   Ordered by: internal time
   List reduced from 15 to 5 due to restriction <5>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      896    0.016    0.000    0.026    0.000 <ipython-input-4-ccef311e1f37>:44(gini_impurity)
      896    0.016    0.000    0.020    0.000 <ipython-input-4-ccef311e1f37>:10(split_dataset)
     7105    0.007    0.000    0.007    0.000 {built-in method numpy.core._multiarray_umath.implement_array_function}
     7105    0.005    0.000    0.013    0.000 <__array_function__ internals>:2(where)
        3    0.003    0.001    0.051    0.017 <ipython-input-4-ccef311e1f37>:97(find_best_split)
```

```
Sat May  2 21:11:06 2020    cancer-learn

         688659 function calls (688657 primitive calls) in 1.688 seconds

   Ordered by: internal time
   List reduced from 15 to 5 due to restriction <5>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    25560    0.892    0.000    1.036    0.000 <ipython-input-4-ccef311e1f37>:10(split_dataset)
    25560    0.302    0.000    0.523    0.000 <ipython-input-4-ccef311e1f37>:44(gini_impurity)
   152988    0.241    0.000    0.241    0.000 {built-in method numpy.core._multiarray_umath.implement_array_function}
   152988    0.091    0.000    0.352    0.000 <__array_function__ internals>:2(where)
        3    0.087    0.029    1.687    0.562 <ipython-input-4-ccef311e1f37>:97(find_best_split)
```

Az fa építésekor a hívások és az idő mennyisége ugrászszerűen nőtt. Valószínűleg ez egy gyenge pont a rendszerben.

Megjegyzés:

```
tree = DecisionTree()

cProfile.run('tree.train(X_train, y_train, max_depth=4, min_samples=1)', 'cancer-learn')
pstats.Stats('cancer-learn').strip_dirs().sort_stats(SortKey.TIME).print_stats(5)
```

```
Sat May  2 21:48:36 2020    cancer-learn

         1375257 function calls (1375237 primitive calls) in 2.706 seconds

   Ordered by: internal time
   List reduced from 27 to 5 due to restriction <5>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    51120    1.348    0.000    1.583    0.000 <ipython-input-4-ccef311e1f37>:10(split_dataset)
    51120    0.544    0.000    0.900    0.000 <ipython-input-4-ccef311e1f37>:44(gini_impurity)
305445/305439    0.360    0.000    0.360    0.000 {built-in method numpy.core._multiarray_umath.implement_array_function}
   305436    0.172    0.000    0.570    0.000 <__array_function__ internals>:2(where)
       15    0.145    0.010    2.705    0.180 <ipython-input-4-ccef311e1f37>:97(find_best_split)
```

Ugyanakkor, előfordulhat az is, hogy a fa mélységének növelése is hatással van az program teljesítményére, azonban ez a tesztelés az algoritmus változtatásával nem foglalkozik.

Stressz teszt

Döntési fa klasszifikációhoz - (decision_tree_classification.py)

Stressz teszt közben az bemenet mérete lett egyre nagyobbra emelve, mert erre korábban ez rosszul hatott a futásidőre. Fontos, hogy a kód futása ne lassítson minket a folyamatos olvasás és az algoritmus követése közben, ezért ha valahol 5 percet kell várni az biztosan használhatatlanná teszi a programot.

1. esetben az Breast cancer wisconsin dataset minden példája 2-ször szerepelt a bemeneten.

```
Sun May  3 00:54:21 2020    cancer-learn

         1378471 function calls (1378469 primitive calls) in 4.714 seconds

   Ordered by: internal time
   List reduced from 15 to 5 due to restriction <5>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    51180    2.806    0.000    3.233    0.000 <ipython-input-12-ccef311e1f37>:10(split_dataset)
   306226    0.745    0.000    0.745    0.000 {built-in method numpy.core._multiarray_umath.implement_array_function}
    51180    0.650    0.000    1.213    0.000 <ipython-input-12-ccef311e1f37>:44(gini_impurity)
        3    0.182    0.061    4.713    1.571 <ipython-input-12-ccef311e1f37>:97(find_best_split)
   306226    0.179    0.000    0.964    0.000 <__array_function__ internals>:2(where)
```

A feldolgozási idő triplájára, a hívások száma duplájára nőtt.

2. esetben minden példa 8-szer szerepelt

```
Sun May  3 01:32:39 2020    cancer-learn

         5519531 function calls (5519529 primitive calls) in 63.497 seconds

   Ordered by: internal time
   List reduced from 15 to 5 due to restriction <5>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   204840   47.777    0.000   52.355    0.000 <ipython-input-4-ccef311e1f37>:10(split_dataset)
  1226246    8.684    0.000    8.684    0.000 {built-in method numpy.core._multiarray_umath.implement_array_function}
   204840    4.240    0.000    9.512    0.000 <ipython-input-4-ccef311e1f37>:44(gini_impurity)
        3    1.180    0.393   63.495   21.165 <ipython-input-4-ccef311e1f37>:97(find_best_split)
  1226246    0.863    0.000    9.733    0.000 <__array_function__ internals>:2(where)
```

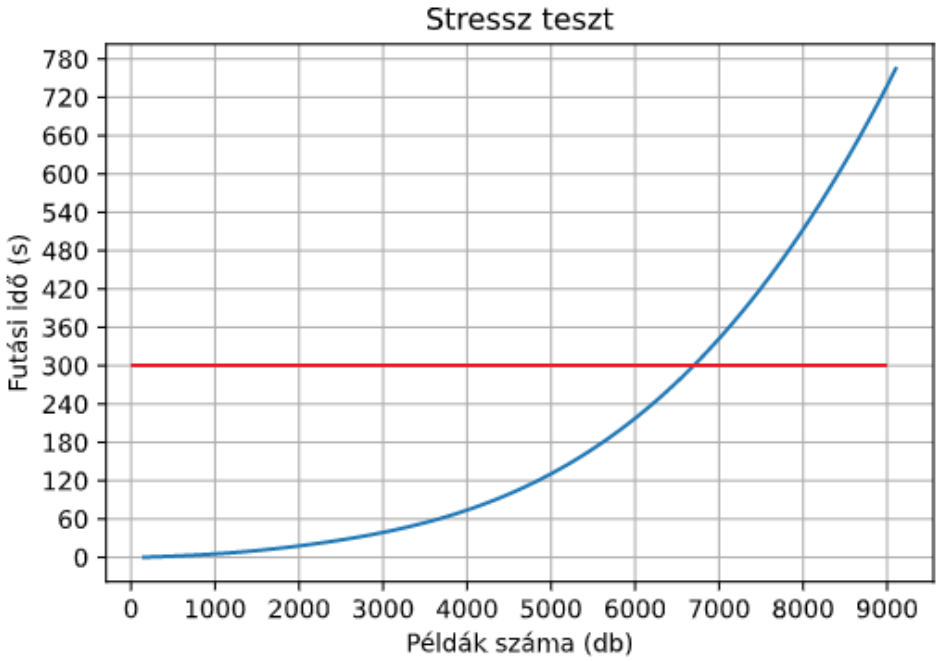
A feldolgozási idő 20-szorosára nőtt

3. esetben minden példa 16-szor szerepelt.

```
Sun May  3 01:17:41 2020    cancer-learn

         11037315 function calls (11037313 primitive calls) in 393.704 seconds

   Ordered by: internal time
   List reduced from 15 to 5 due to restriction <5>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   409680  314.822    0.001  331.044    0.001 <ipython-input-4-ccef311e1f37>:10(split_dataset)
        3   31.602   10.534  393.702  131.234 <ipython-input-4-ccef311e1f37>:97(find_best_split)
  2452062   31.226    0.000   31.226    0.000 {built-in method numpy.core._multiarray_umath.implement_array_function}
   409680   12.433    0.000   29.934    0.000 <ipython-input-4-ccef311e1f37>:44(gini_impurity)
  2452062    1.878    0.000   33.493    0.000 <__array_function__ internals>:2(where)
```

A feldolgozási idő közel 200-szorosára nőtt. A program továbbra is lefut, de nem megfelelő időn belül.

A függvény a korábbi eredményekből megkapott pontokból közelítés. A közelítés alapján arra következtethetünk, hogy a futási idő exponenciálisan függ a bemenet méretétől.
Ez alapján becsülhetjük, hogy az elvárt teljesítményhez (jelen esetben 5 percen belüli lefutás) körülbelül mekkora példahalmazt adhatunk az algoritmusnak.



Az egyenes az 5 perces határnál van berajzolva a függvénybe, leolvashatjuk, hogy az X = (6000,7000) intervallumon metszik egymást, tehát maximum 6-7000 példából álló adathalmazzal taníthatjuk a fát.

Dokumentáció teszt

Egész projekt

A projekt az gépi tanulási módszerek megértését segíti, ebből adódóan minden algoritmushoz van igényesen formázott és megfogalmazott, a működést matematikai képletekkel alátámasztó képes dokumentáció.

Step 1: (Only needed when training with gradient descent)

Compute a linear combination of the input features and weights. This can be done in one step for all training examples, using vectorization and broadcasting:
$\hat{\boldsymbol{y}} = \boldsymbol{X} \cdot \boldsymbol{w} + b$

where $\boldsymbol{X}$ is a matrix of shape $(m, n_{features})$ that holds all training examples, and $\cdot$ denotes the dot product.

Step 2: (Only needed when training with gradient descent)

Compute the cost (mean squared error) over the training set:

$J(\boldsymbol{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2$

Az algoritmusok működéséhez kapcsolódó függvények előtt dokumentációs kommentek szerepelnek,

```
def mean_squared_error(self, y_left, y_right, n_left, n_right):
    """
    Computes MSE of a split.

    Args:
        y_left, y_right: target values of samples in left/right subset
        n_left, n_right: number of samples in left/right subset

    Returns:
        mse_left: float, MSE of left subset
        mse_right: gloat, MSE of right subset
    """
```

A projekt megfelelően van dokumentálva, egyértelműen kiderül belőle a funkciók működése, könnyen használható és fejleszthető a program.

A projekthez egységtesztek és manuális tesztelés is készült, melyekhez tartozik egy átfogó leírás, amiből megtudhatjuk a tesztelés célját, nehézségeit, egyéb részleteit. Ezek összhangban vannak a tesztesetekkel és a tesztelés módjával.

Preconditions	Test Steps	Expected Result
Docker is installed on host machine.	1. Run docker run -p 9000:8888 schmelczera/ml-basics-notebooks on the host machine	User should see the listed notebooks.
	2. Navigate to http://localhost:9000	
-	1. Navigate to https://ml-basics-notebooks.web.app/	User should see the listed notebooks.

Manuális teszteknél a tesztelés lépései egyértelműen végrehajthatóak, követhetőek.

Egységtesztek esetében a tesztesetekhez részletes dokumentáció nincs, azonban a kód könnyen olvasható és minden esetnek beszédes neve van. Mindkettő a lehető legnagyobb mértékben a projekt működéséhez fontos funkciókat tesztel.

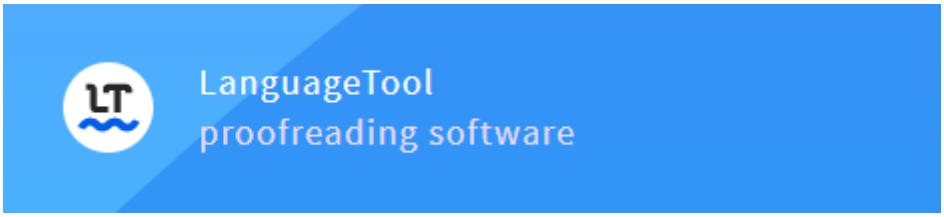
Továbbá deployment segítésével is ki lett egészítve a projekt, amihez ugyancsak tartozik egy a megvalósítást összefoglaló dokumentum, ami jól magyarázza a használt technológiákat és információkat a kivitelezés részleteiről.

Building the notebooks yourself

1. Make sure you have Python installed with a version greater than 3.6.0
2. `git clone https://github.com/BME-MIT-IET/Almafa14-iet-2020.git`
3. `cd Almafa14-iet-2020`
4. `./development/init-with-venv.sh`
5. You can find the notebooks inside of Almafa14-iet-2020/notebooks

Aműködésbe hozatal lépései az eredeti README-be lettek integrálva, ami tökéletesen ellátja a funkcióját, tehát ezt használva bárki azonnal elkezdheti tanulni mesterséges intelligenciához tartozó algoritmusokat.

A dokumentáció jellemzően nem tartalmaz nyelvtani és helyesírási hibákat. A Language Tool-t futtatva a dokumentációk nagy részén apróbb typokat talált, ezek azonnal javítva lettek.



LanguageTool - Spell and Grammer Checker