UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Bizilj

Modul za obdelavo slik v sistemu za podatkovno rudarjenje Orange

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2013

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Bizilj

Modul za obdelavo slik v sistemu za podatkovno rudarjenje Orange

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Luka Šajn

Ljubljana, 2013

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00363/2013
Datum: 01.02.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BLAŽ BIZILJ**

Naslov: **MODUL ZA OBDELAVO SLIK V SISTEMU ZA PODATKOVNO RUDARJENJE ORANGE**

MODULE FOR IMAGE PROCESSING IN THE ORANGE DATA MINING SYSTEM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V okviru diplomske naloge razvijte dodatek za rudarjenje na slikah v sistemu za podatkovno rudarjanje Orange. Z današnjo tehnologijo se danes dnevno ustvari zelo velika količina podatkov med katerimi je tudi veliko slik. Tako kot s podatkovnim rudarjenjem po besedilih ali številkah pridobimo nove informacije, lahko tudi z različnimi algoritmi iz slik pridobimo atribute, nad katerimi nato izvajamo podatkovno rudarjenje. Tako lahko tudi iz njih pridobimo nove informacije. Naredite dodatek, ki z različnimi algoritmi iz slik pridobi atribute, nato pa preko teh atributov z že razvitimi rešitvami v sistemu Orange omogočite podatkovno rudarjenje.

Mentor:

doc. dr. Luka Šajn

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Blaž Bizilj,

z vpisno številko 63060090,

sem avtor/-ica diplomskega dela z naslovom:

Modul za obdelavo slik v sistemu za podatkovno rudarjenje Orange

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  doc. dr. Luka Šajn

  in somentorstvom (naziv, ime in priimek)

  _____

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

ZAHVALA

Zahvalil bi se doc. dr. Luki Šajnu za vodenje in vse nasvete pri izdelavi diplomskega dela. Prav tako bi se zahvalil tudi članom laboratorija za bioinformatiko Fakultete za računalništvo in informatiko, ki so mi pomagali pri izdelavi dodatka za sistem Orange, še posebej asistentu Mihi Štajdoharju.

KAZALO

KAZALO SLIK

POVZETEK

V diplomskem delu smo se osredotočili na razvoj dodatka za podatkovno rudarjenje v sistemu Orange.

Sistem Orange omogoča vizualno podatkovno rudarjenje, ima komponente za pripravo, filtriranje in vrednotenje podatkov. Poleg tega vsebuje veliko gradnikov, ki omogočajo vse od najrazličnejšega klasifikacije podatkov do predstavitvenih modelov in strojnega učenja, ki omogoča napovedovanje manjkajočih podatkov.

Cilj diplomskega dela je bilo narediti dodatek za sistem Orange, ki bi omogočal, da se kot začetni podatki lahko uporabijo slike. Iz teh bi z različnimi algoritmi pridobili atribute, na katerih bi lahko s pomočjo naših rešitev in že narejenih rešitev v sistemu Orange, izvajali podatkovno rudarjenje.

Dodatek za sistem Orange smo razvili tako, da se gradniki, ki smo jih dodali v sistem, lahko povežejo z že obstoječimi gradniki. S tem smo dosegli, da ima dodatek že takoj na začetku na voljo nekaj močnih orodij za podatkovno rudarjenje.

Ključne besede:

- Podatkovno rudarjenje
- Podatkovno rudarjenje na slikah
- Orange
- Parametrizacija slik

ABSTRACT

In this thesis we focused on the development of an add-on for data mining in the Orange system.

The Orange system enables visual data mining has the components for preparation, filtration and evaluation of data. In addition, it contains many widgets that enable us to do everything from data classification, construction of demonstration models and machine learning, which allows the prediction of missing data.

The aim of the thesis was to make an add-on for the Orange system that would enable images to be used as starting data. From these images we would use various algorithms to obtain the attributes on which we can use the already developed solutions in the Orange system to start data mining.

We developed the add-on for the Orange system in such a way, that the widgets that we have added to the system can connect to the already existing widgets.

By doing so we achieved that our add-on has access to some very powerful data mining tools right from the start.

Keywords:

- Data mining
- Data mining in pictures
- Orange
- Image parameterization

1 UVOD

Podatkovno rudarjenje je iskanje informacij v veliki količini podatkov.

Po podatkih IBMa se na svetu vsak dan ustvari za 2,5 * 10^18 bajtov podatkov, kar znaša približno 2,273,736 terabajtov podatkov dnevno [1] (za primer: trenutno največji komercialno pridobljiv disk je velikosti 3 terabajte).

Pridobivanje informacij iz tako velike količine podatkov je brez računalniške pomoči nemogoče. IBM tako trdi, da smo 90 % vseh podatkov na svetu ustvarili v zadnjih dveh letih [1], kar kaže na to, da je potreba po podatkovnem rudarjenju pri veliki količini podatkov dokaj nova.

Danes je podatkovno rudarjenje uporabljeno v številnih podjetjih. Nekatera med njimi ga uporabljajo kot pripomoček pri analizi poslovanja in pripravi načrtov za prihodnost, drugi pa podatkovno rudarjenje uporabljajo neposredno pri poslovnem procesu.

Najprej si poglejmo podjetje, ki uporablja podatkovno rudarjenje kot neposreden pripomoček za poslovni proces. Eno od takih podjetij je Google. Google je za analizo spletnih strani razvil posebne robotke (Googlebot-e) [2], ki neprestano pridobivajo podatke iz vseh spletnih strani na svetu. Da Googlova glavna funkcija – iskanje podatkov po internetu – deluje kot je potrebno, se morajo ti podatki analizirati in pravilno prikazati, ko uporabnik v iskalnik vpiše iskalni niz. To delo je brez računalniške podpore nemogoče opraviti, zato s pomočjo podatkovnega rudarjenja glede na iskalni niz ti posebni robotki najdejo tisto spletno stran, ki najbolj ustreza iskalnemu nizu in prikažejo povezavo nanjo.

Pa si poglejmo še Googlov glavni način pridobivanja denarja – prodajanje oglasov. Tukaj je podatkovno rudarjenje še bolj pomembno, saj Google pošlje v podatkovno rudarjenje vsa vaša elektronska sporočila (če imate Gmail račun) in vse vaše iskalne nize. Iz njih nato najde prodajalca, ki ustreza vašemu profilu in njegova oglasna sporočila se prikažejo povsod, kjer se prikazujejo Googlovi oglasi.

Vsa podjetja pa ne uporabljajo podatkovnega rudarjenja za direkten način ustvarjanja prihodkov. Večina podjetij namreč tovrstno rudarjenje uporablja kot podporo poslovanju –

tako analizirajo preteklo poslovanje, s pridobljenimi informacijami pa lahko nato spremenijo svoj poslovni proces ali pa predvidijo, kakšno bo njihovo nadaljnje poslovanje.

To diplomsko delo se ukvarja s podatkovnim rudarjenjem po slikah. Če hočemo iskati slike po internetu, moramo vpisati niz. Ta niz se nato primerja z informacijami, ki so napisane o slikah na internetu. Pri tem se nam prikažejo slike, ki so bile opisane z našim iskalnim nizom. Kot vidimo, nastane lahko problem, ko se išče slike po dodatnih informacijah, ki so jih avtorji slike napisali in ne dejansko po samih slikah. Problem se pojavi tudi, če bi radi našli še kakšno sliko, ki je podobna naši. Namreč v iskalnik lahko vpišemo le iskalni niz, ne moremo pa dodati naše slike.

Slike so tudi dejansko podatki, saj so na podatkovnih nosilcih zapisani z 0 in 1, tako kot vsi ostali podatki. Vendar pa na slikah ne moremo uporabiti enakih algoritmov kot na besedilih in drugih podatkih, ker so slike za uporabnika veliko več kot skupek 0 in 1, kar predstavljajo računalniku. Računalniški vid ni dovolj razvit, da lahko ocenjuje lepoto slike (in mogoče nikoli ne bo, navsezadnje je lepota slike subjektivno mnenje), lahko pa ga naučimo analizirati slike. Za analizo slik obstaja kar nekaj algoritmov, ki pridobijo podatke iz slik, nad katerimi lahko nato izvajamo podatkovno rudarjenje [3]. Ti algoritmi nam ne povedo ali je slika lepa ali grda, povedo pa nam nekaj specifičnih lastnosti slike, ki jih lahko potem primerjamo s podatki pridobljenimi iz drugih slik z istim algoritmom.

Podatkovno rudarjenje po slikah je uporabno pri kar nekaj dejavnostih. Vzemimo zopet za primer iskalnik Google. Če bi imeli sliko, ki nam je všeč in bi radi dobili še kakšno, ki je zelo podobna, bi v iskalnik lahko podali našo sliko, ki bi jo z algoritmi analizirali in tako našli takšno, ki bi bila podobna naši.

Še ena izmed dejavnosti, pri katerih bi bilo uporabno podatkovno rudarjenje po slikah, je nakupovanje oblačil preko spletne trgovine. Spletna trgovina bi si zapomnila slike oblačil, ki smo jih že kupili in bi pri naslednjem obisku lahko prikazala oblačila, ki so podobna našim prejšnjim nakupom, ali pa nam pokazala nova oblačila v barvi, katero smo že nekajkrat nakupili.

Kot vidimo, je podatkovno rudarjenje po slikah lahko že takoj zelo uporabno pri nekaterih poslovnih procesih. Na Fakulteti za računalništvo in informatiko je želja, da bi se podatkovno

rudarjenje po slikah uporabilo predvsem v Katedri za umetno inteligenco, največ pa v laboratoriju za računalniški vid.

V laboratoriju za računalniški vid »učijo« računalnike, kako prepoznati vse, kar jim prikažejo pred kamero. Računalnik nato predmet, ki ga vidi, poskuša prepoznati tako, da ga primerja s slikami, ki jih ima v spominu. Problem nastane takoj, ko je določen predmet postavljen malce drugače, ali pa je celo delno zakrit kot je na sliki v spominu – lahko je pod drugačnim kotom ali pa je na neravni površini. Računalnik nato skuša predmet prestaviti v enak položaj, kot je na sliki. S povezavo računalnika s sistemom Orange bi ta problem lahko odpravili, saj bi z algoritmom lahko našli sliko, ki bi bila dovolj podobna sliki v spominu in tako bi računalnik predmet lažje prepoznal.

Problema smo se lotili tako, da smo najprej analizirali delovanje sistema Orange. Ko smo imeli dobro razumevanje sistema Orange, smo naredili nekaj načrtov, kako bi radi, da se dodatek obnaša. Po izdelavi prototipa smo nato izbrali končni načrt, ki je omogočal želeno funkcionalnost.

Da bo naš dodatek sistemu Orange učinkovit, smo ga naredili tako, da smo sprogramirali nekaj gradnikov, ki se lahko povezujejo tako med seboj kot tudi z že obstoječimi gradniki. Dodaten problem so predstavljali algoritmi, s katerimi analiziramo slike. V različnih laboratorijih na Fakulteti za računalništvo in informatiko imajo veliko različnih algoritmov za analizo slik, a ti algoritmi ponavadi niso napisani v istih programskih jezikih. Zaradi nekonsistentnosti že narejenih algoritmov in zaradi možnega nadaljnjega razvoja teh algoritmov, smo dodatek naredili tako, da lahko vsak uporabnik sistema Orange z znanjem programiranja dodaja algoritme in ustvari svoje gradnike. S tem smo hkrati spoštovali osnovno zamisel sistema Orange in javno licenco, pod katero je registrirana, da lahko vsak uporabnik prispeva k tej programski opremi.

2 ANALIZA PODROČJA

2.1 PODATKOVNO RUDARJENJE

Sistem Orange, s katerim se ukvarjamo v diplomskem delu, ni edino orodje za podatkovno rudarjenje, zato poglejmo še nekaj konkurenčnih orodij za podatkovno rudarjenje.

2.1.1 WEKA

WEKA je odprtokodna rešitev za podatkovno rudarjenje in strojno učenje, ki so jo razvili na Univerzi Weaikato v Novi Zelandiji. WEKA je razvita v programskem jeziku Java in vsebuje orodja za obdelavo podatkov, klasifikacijo, regresivno povezovanje in vizualno predstavljanje rezultatov [4, 5].

V mnogih pogledih je WEKA največja konkurenca Orangu, saj je bila tako kot Orange razvita na univerzi. Delovanje obeh orodij je primerljivo, saj obe vsebujeta veliko podobnih algoritmov in drugih orodij. Obenem pa WEKA prav tako kot Orange še ne vsebuje dodatka za podatkovno rudarjenje po slikah.

2.1.2 KNIME (Konstanz Information Miner)

KNIME (Konstanz Information Miner) je odprtokodna rešitev za strojno učenje in podatkovno rudarjenje napisana v Javi. Osnovna verzija KNIME je brezplačna. Za plačilo se dobi verzija z več funkcijami, hitrejšimi nadgradnjami in popravili napak, dostop do izvorne kode in tehnična podpora. KNIME je zelo uporabljen v farmacevtski industriji, CRM analizah, poslovni inteligenci in finančni analizi podatkov [6, 7].

KNIME omogoča povezavo s številnimi dodatki in drugimi programi (med drugimi tudi z WEKO). Eden od dodatkov, s katerim se lahko poveže, je ImageJ. ImageJ je orodje za obdelavo in analiziranje slik, s katerim je KNIME dobil možnost podatkovnega rudarjenja na slikah.

2.1.3 RapidMiner (bivši YALE – Yet Another Learning Software)

RapidMiner je orodje za strojno učenje, rudarjenje podatkov, rudarjenje po besedilu, napovedne analize in poslovne analize. Osnovna verzija RapidMiner-ja je na voljo brezplačno za nekomercialne namene, s plačilom pa tudi za komercialne. Za plačilo omogočajo še

certificiranje, učenje, podporo uporabnikom in celo analiziranje podatkov. Tudi RapidMiner je napisan v Javi in tako omogoča povezavo z nekaterimi drugimi orodji, ki so napisani v Javi (med drugimi R in WEKA) [8].

RapidMiner prav tako še nima dodatka za podatkovno rudarjenje po slikah, kar pa ne pomeni, da ga ne bodo razvili v prihodnosti.

2.2 RAČUNALNIŠKI VID

Računalniški vid je področje računalništva, ki se ukvarja s pridobivanjem, analiziranjem in razumevanjem slik in drugih podatkov iz resničnega sveta. Računalniški vid sam sicer ne more narediti veliko, saj je vsaka kamera povezana na računalnik že osnovna verzija računalniškega vida. Če to kamero povežemo z različnimi programskimi sistemi, pa postane računalniški vid zelo močno orodje, ki ga lahko uporabljamo pri številnih panogah. Manj napredne funkcije, ki se opravljajo z računalniškim vidom, so na primer nadzor izdelkov na industrijski liniji ter sestavljanje mikrovezij s pomočjo robotske roke. Med bolj napredne funkcije, ki se opravljajo s pomočjo računalniškega vida, sodijo analiza medicinskih slik, modeliranje okolja in predmetov, štetje prometa in tudi podpora računalniško vodenim vozilom.

Med osnovnim računalniškim vidom (kamera) in sistemom, ki bo uporabil pridobljene podatke (npr. avto ali robot) pa mora obstajati programska oprema, ki bo te slike analizirala in poslala podatke naslednjemu sistemu, da bo ta na podlagi teh podatkov pravilno ukrepal. Dva izmed večjih paketov, ki se ukvarjata z računalniškim vidom sta ImageJ in OpenCV.

2.2.1 ImageJ

ImageJ je v Javi narejen program za urejanje, analizo, obdelavo in shranjevanje slik. Podpira številne formate slik (TIFF, GIF, JPEG ...), nad katerimi lahko izvaja standardne operacije (spreminjanje kontrasta, ostrenje, glajenje, odkrivanje robov ...) [9, 10].

ImageJ je bil zasnovan na odprti arhitekturi in tako omogoča tako dodajanje novih vmesnikov in funkcij, kot tudi povezovanje z drugimi programskimi paketi.

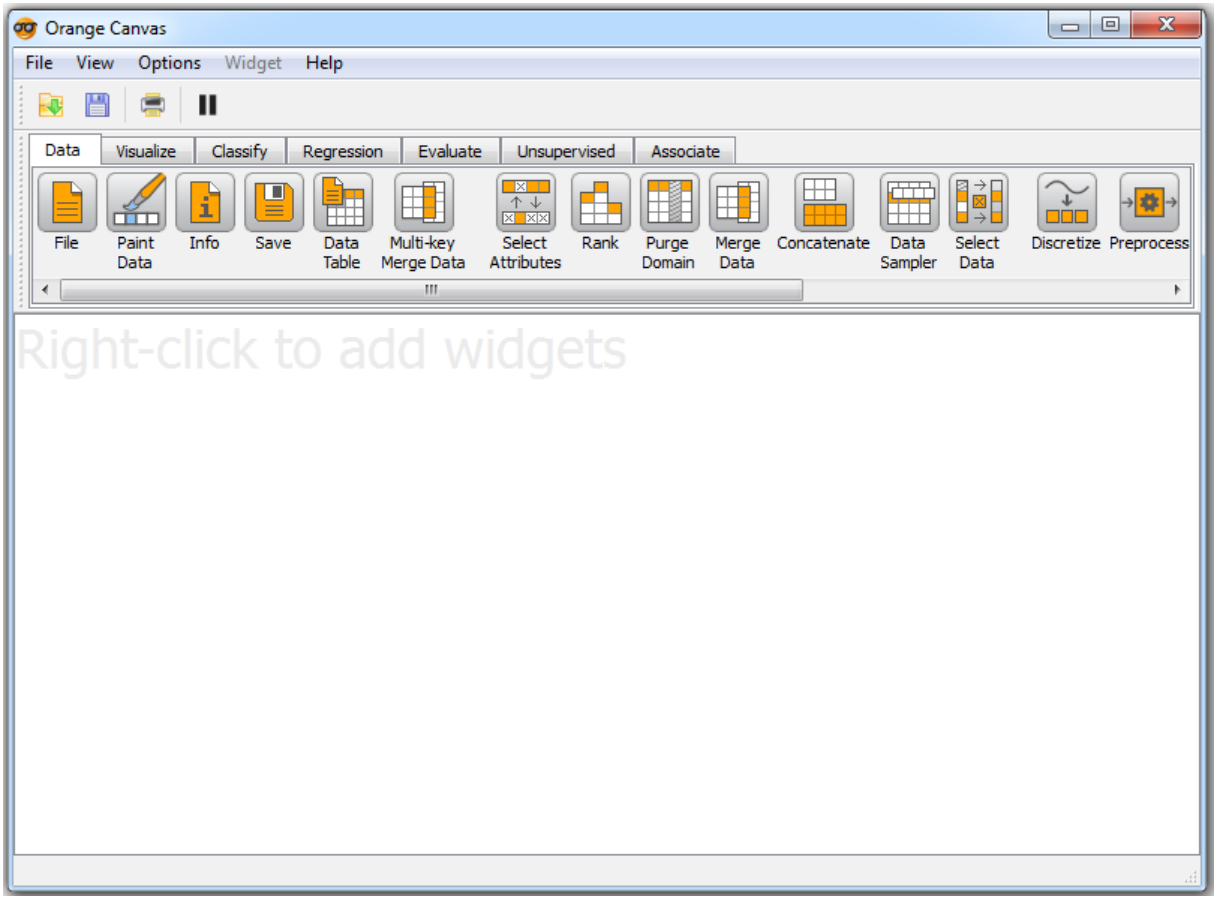
2.2.2 OpenCV

OpenCV je zelo obsežna knjižnica, ki vsebuje veliko funkcij za analiziranje računalniškega vida v realnem času. Med drugim ima funkcije za zaznavanje obraza, prepoznavanje gest, identifikacijo predmetov, sledenje gibanju in še veliko drugih. Kot podporo nekaterim funkcijam ima prav tako vgrajenih že nekaj funkcij, ki sodijo pod področje podatkovnega rudarjenja - npr.: odločitveno drevo, naivni Bayes, nevronske mreže in druge [11, 12].

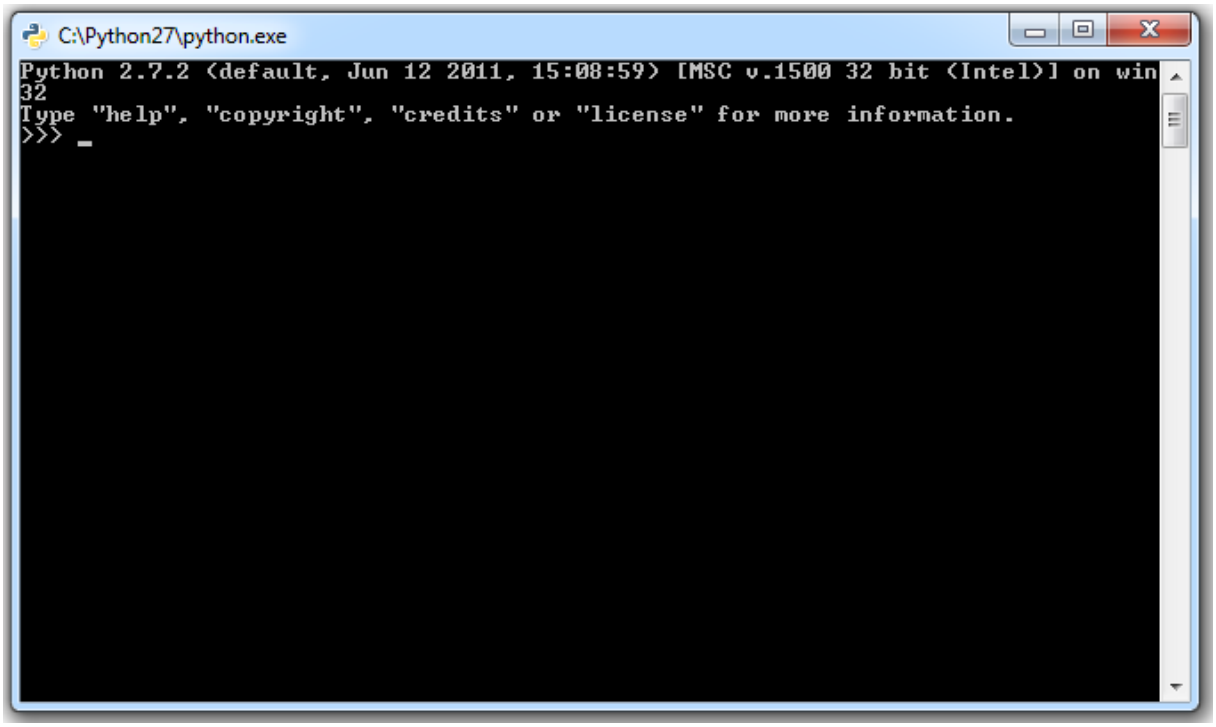
3 METODE IN ORODJA

3.1 ANALIZA SISTEMA ORANGE

Sistem Orange lahko uporabljamo na dva načina. Prvi način je preko grafičnega vmesnika kot je vidno na sliki 1, drugi način pa je preko ukazne vrstice, prikazano na sliki 2.



Slika 1: Grafični vmesnik sistema Orange



Slika 2: Ukazna vrstica

V primeru, da vzamemo način uporabe preko grafičnega vmesnika, lahko vidimo, da večji del sestavlja t. i. platno (ang. canvas), na katerega z miško dodajamo gradnike. Zgornji del uporabniškega vmesnika sestavljajo standardni meniji in zavitki, ki vsebujejo različne gradnike.

Uporaba preko ukazne vrstice je za naprednejše uporabnike. Ukazna vrstica nam omogoča bolj natančno analiziranje podatkov (npr. klicanje samo one izmed funkcij gradnikov), a za tako početje je potrebno dobro poznavanje sistema Orange ter imena in delovanje njegovih funkcij.

Dodatek, ki smo ga razvili, se imenuje Orange-imaging in ob namestitvi doda nov zavitek z imenom Imaging, v katerem so vsi gradniki, ki smo jih razvili.

Za primer vzemimo preprost primer delovanja sistema Orange. Za podatkovno rudarjenje najprej potrebujemo podatke. Sistem Orange potrebuje podatke v specifični obliki, da jih lahko interpretira. Na sliki 3 vidimo standardno datoteko za sistem Orange. Datoteka je razdeljena na dva dela. Prve tri vrstice so t. i. domena (Orange.data.Domain), v kateri so zapisani sledeči podatki:

- 1. vrstica – ime stolpcev – ta vrstica nam pove ime vsakega stolpca, ki bo uporabljeno tudi pri Orangovi notranji tabeli

- 2. vrstica – tip podatka v stolpcih – je podatek, ki nam pove, kakšno vsebino lahko pričakujemo v stolpcih.

  V Orange imamo na voljo štiri različne možnosti:

  1. število (Orange.feature.Continuous),

  2. več možnosti (Orange.feature.Discrete),

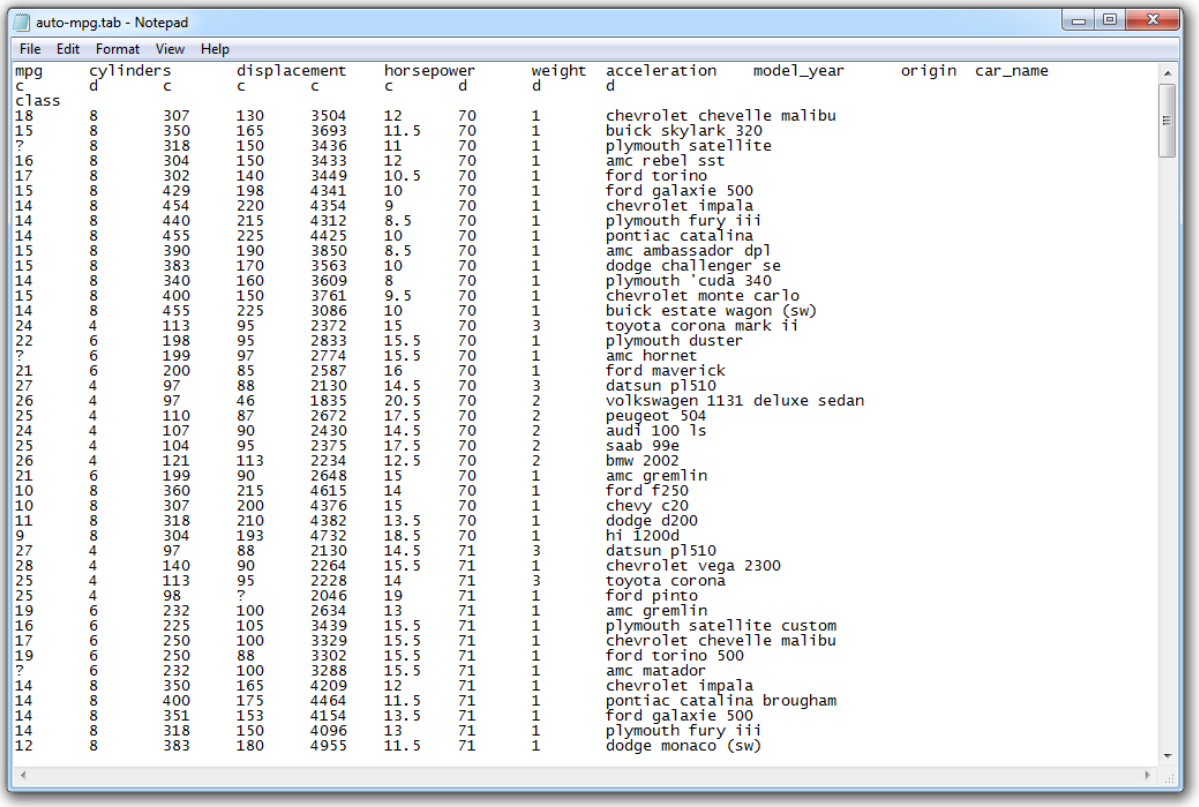
  3. črkovni niz (Orange.feature.String) in

  4. Python objekt (Orange.feature.Python)

- 3. vrstica – kateri stolpec je razred (ang. class) – ta podatek je zelo pomemben, saj nam pove, katere vrstice podatkov spadajo v isti razred. S tem podatkom lahko najdemo dodatne informacije, ki so specifične za določen razred ali pa ugotovimo, h kateremu razredu bi se uvrstila vrstica, ki ne spada v noben razred.

Vse tri vrstice so zelo pomembne, sej Orange iz njih sestavi notranjo tabelo (Orange.data.Table). Za podatkovno rudarjenje je najbolj pomembna 3. vrstica, ki nam pove, kateri stolpec je razredni, s tem pa povemo Orangu, katere vrstice spadajo skupaj.

Drugi del datoteke sestavljajo podatki. Kot lahko vidimo na sliki 3, se tam lahko pojavijo vprašaji, ki kažejo na to, da nek podatek manjka (lahko, da ga ni ali pa smo ga načrtno izpustili). S pomočjo sistema Orange bomo lahko iz teh podatkov našli kakšen vzorec, predvideli, kaj bi lahko vpisali v manjkajoče podatke ali pa našli kakšno splošno informacijo, ki je prej nismo.



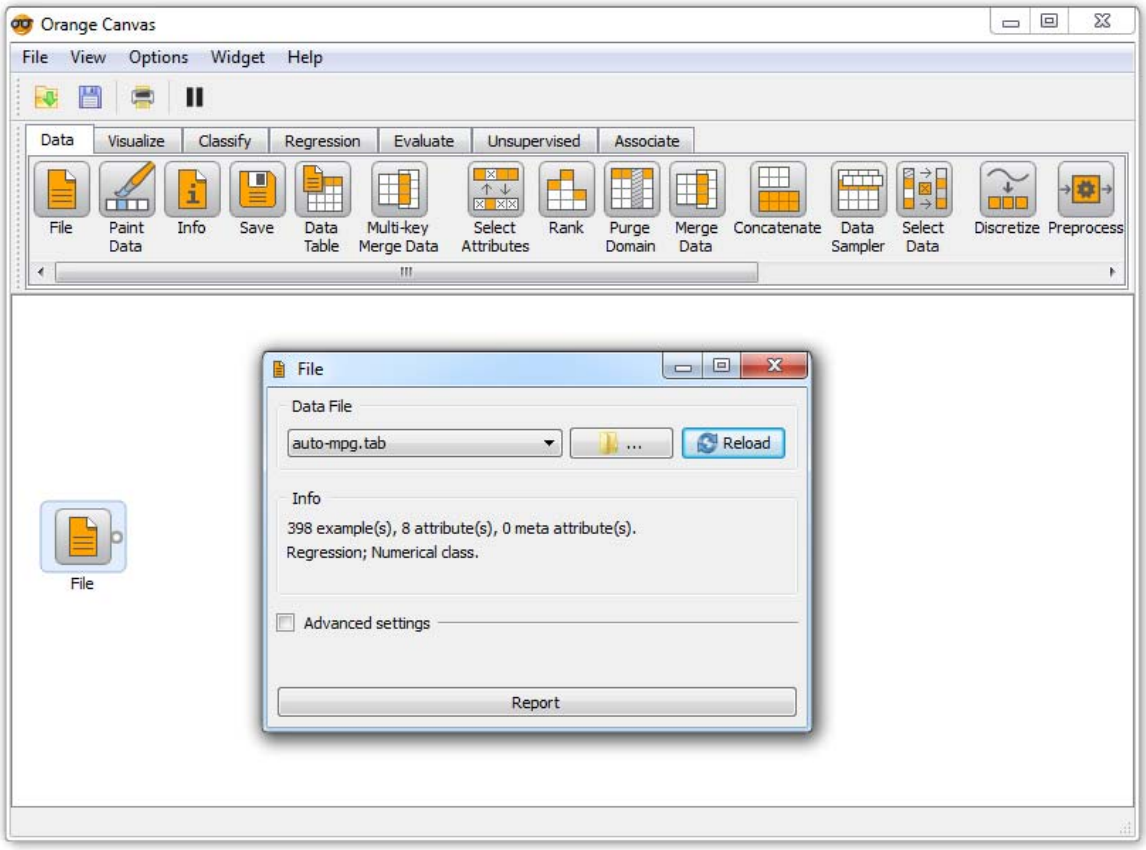
auto-mpg.tab - Notepad

File Edit Format View Help

mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
c	d	c	c	c	d	d		
class								
18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
15	8	350	165	3693	11.5	70	1	buick skylark 320
?	8	318	150	3436	11	70	1	plymouth satellite
16	8	304	150	3433	12	70	1	amc rebel sst
17	8	302	140	3449	10.5	70	1	ford torino
15	8	429	198	4341	10	70	1	ford galaxie 500
14	8	454	220	4354	9	70	1	chevrolet impala
14	8	440	215	4312	8.5	70	1	plymouth fury iii
14	8	455	225	4425	10	70	1	pontiac catalina
15	8	390	190	3850	8.5	70	1	amc ambassador dpl
15	8	383	170	3563	10	70	1	dodge challenger se
14	8	340	160	3609	8	70	1	plymouth 'cuda 340
15	8	400	150	3761	9.5	70	1	chevrolet monte carlo
14	8	455	225	3086	10	70	1	buick estate wagon (sw)
24	4	113	95	2372	15	70	3	toyota corona mark ii
22	6	198	95	2833	15.5	70	1	plymouth duster
?	6	199	97	2774	15.5	70	1	amc hornet
21	6	200	85	2587	16	70	1	ford maverick
27	4	97	88	2130	14.5	70	3	datsun pl510
26	4	97	46	1835	20.5	70	2	volkswagen 1131 deluxe sedan
25	4	110	87	2672	17.5	70	2	peugeot 504
24	4	107	90	2430	14.5	70	2	audi 100 ls
25	4	104	95	2375	17.5	70	2	saab 99e
26	4	121	113	2234	12.5	70	2	bmw 2002
21	6	199	90	2648	15	70	1	amc gremlin
10	8	360	215	4615	14	70	1	ford f250
10	8	307	200	4376	15	70	1	chevy c20
11	8	318	210	4382	13.5	70	1	dodge d200
9	8	304	193	4732	18.5	70	1	hi 1200d
27	4	97	88	2130	14.5	71	3	datsun pl510
28	4	140	90	2264	15.5	71	1	chevrolet vega 2300
25	4	113	95	2228	14	71	3	toyota corona
25	4	98	?	2046	19	71	1	ford pinto
19	6	232	100	2634	13	71	1	amc gremlin
16	6	225	105	3439	15.5	71	1	plymouth satellite custom
17	6	250	100	3329	15.5	71	1	chevrolet chevelle malibu
19	6	250	88	3302	15.5	71	1	ford torino 500
?	6	232	100	3288	15.5	71	1	amc matador
14	8	350	165	4209	12	71	1	chevrolet impala
14	8	400	175	4464	11.5	71	1	pontiac catalina brougham
14	8	351	153	4154	13.5	71	1	ford galaxie 500
14	8	318	150	4096	13	71	1	plymouth fury iii
12	8	383	180	4955	11.5	71	1	dodge monaco (sw)

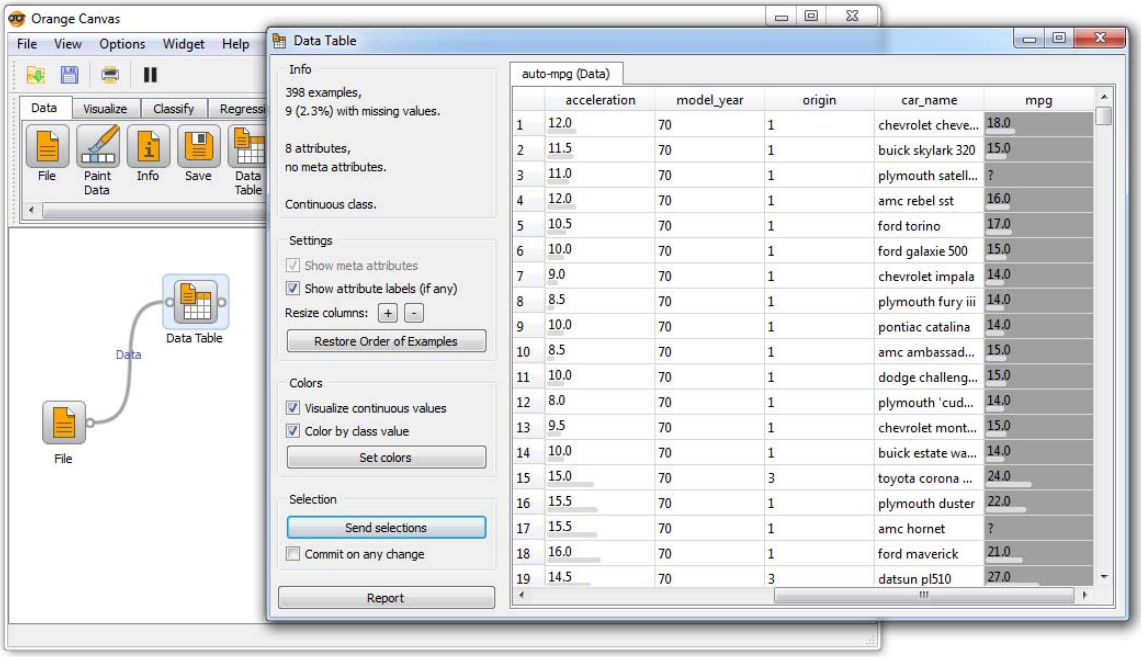
Slika 3: Vhodna datoteka za sistem Orange

Slika 4 prikazuje vnos datoteke v sistem Orange preko gradnika Datoteka (File). Ta gradnik nam takoj sporoči število podatkov, atributov in meta atributov ter kakšne vrste je razredni stolpec.



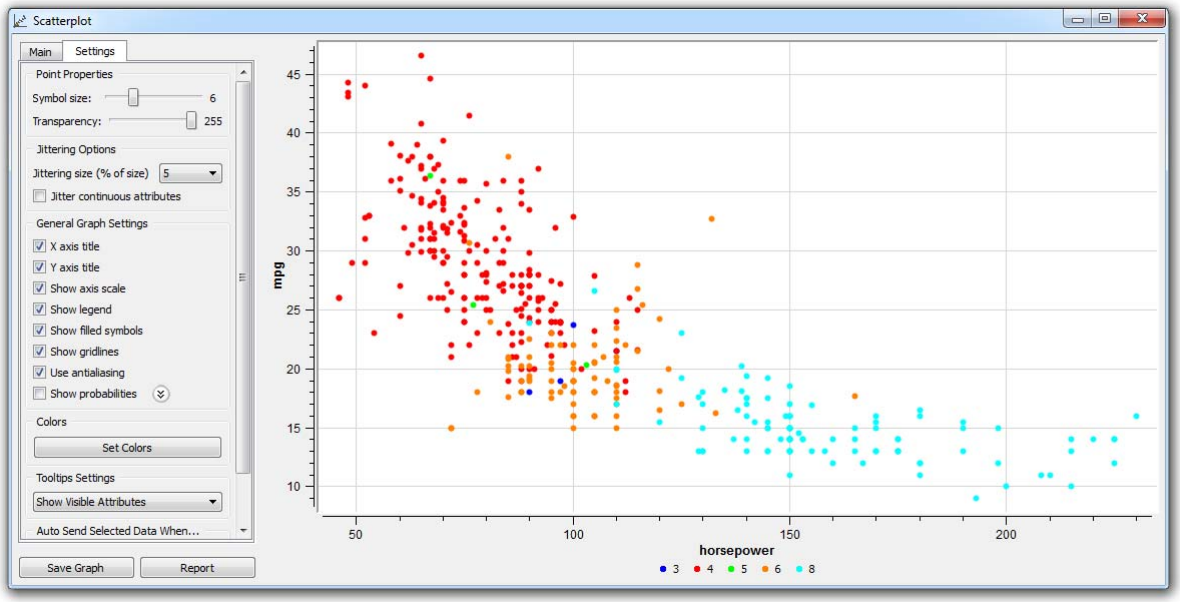
Slika 4: Vnos datoteke s podatki v sistem Orange

Na sliki 5 vidimo s pomočjo gradnika podatkovno tabelo (Data Table), iz katere je razvidno, kako je Orange iz vhodne datoteke naredil notranjo tabelo, pobarval vrstice, ki spadajo k istemu razredu in ob strani izpisal, koliko vrstic ima manjkajoče podatke (v našem primeru devet manjkajočih podatkov kar je 2, 3 % vseh podatkov).



Slika 5: Notranja tabela sistema Orange

Podatke lahko sicer že takoj prikažemo in tako najdemo kakšno zanimivo informacijo. Na sliki 6 smo prikazali samo vhodne podatke na grafu, tako da smo za os x izbrali konjsko moč vozil (horsepower), za y os smo izbrali podatek, koliko kilometrov naredi na galon (mpg – miles per gallon), za barvo pa smo izbrali število cilindrov. Že takoj lahko iz teh podatkov dobimo določene informacije – tako lahko na primer vidimo, da imajo vozila z osem cilindri največjo konjsko moč, a hkrati tudi največ porabijo. Na drugi strani pa imajo vozila s štirimi cilindri najmanjšo moč in najmanjšo porabo.



Slika 6: Prikaz grafa ki je bil izdelan na podlagi podatkov v datoteki

3.2 UPORABLJENA ORODJA

Ker je večina dodatkov in gradnikov v sistemu Orange sprogramirana v programskem jeziku Python, smo tudi naš dodatek razvili v tem jeziku. Za razvoj smo uporabili razvijalno okolje PyCharm, kjer smo z pomočjo razhroščevalnika poskrbeli za dobro delovanje, hkrati pa se je PyCharm lahko povezal s sistemom Git, ki nam je omogočal nadzor nad različnimi verzijami našega dodatka.

Ker pričakujemo, da bo naš dodatek uporabljalo več ljudi, smo ga dodali v sistem GitHub. GitHub bo omogočal vsakemu uporabniku prenos izvorne kode in dodajanje lastnih rešitev.

3.2.1 PYTHON

Python je vsenamenski, visokonivojski programski jezik. Od drugih programskih jezikov se najbolj razlikuje na dveh območjih – sintaksa in knjižnice.

Sintaksa Pythona je v primerjavi z drugimi programskimi jeziki zelo drugačna, saj namesto zavitih oklepajev ({ in }) uporablja zamike, kateri pa pri drugih programskih jezikih niso tako potrebni/zaželeni. Ker je filozofija oblikovanja Python kode poudarja berljivost kode je v

Pythonu uporabljenih veliko angleških besed namesto ločil. Ta filozofija omogoča čim večjo berljivost.

Python je tudi posebno razvit kar se tiče funkcionalnosti. Namesto da bi sledili drugim programskim jezikom in v jedro skušali vpeljati čim več funkcionalnosti, so se odločili da se bodo raje posvetili da bo jezik zalo razširljiv. Tako je jedro Pythona zelo majhno, kar omogoča da ga lahko uporabljamo na šibki strojni opremi ali pa da ga vgradimo v obstoječo aplikacijo, hkrati pa ima zelo veliko knjižnic, ki omogočajo razširitev funkcionalnosti.

Ker je Python visokonivojski programski jezik je že od začetka bil manj odvisen od operacijskega sistema in strojne opreme na kateri teče. Vse dodane knjižnice pa so bile izdelane s še posebnim namenom da se ta neodvisnost še poveča. Tako sedaj Python teče na veliko različnih operacijskih sistemih in različni strojni opremi, programska oprema razvita v Pythonu pa prav tako lahko brez velikih sprememb teče na različnih operacijskih sistemih in opremi.

3.2.2 PYCHARM

PyCharm je integrirano razvijalno okolje (IDE) za programiranje v Pythonu. PyCharm je zelo uporabno orodje pri programiranju v Pythonu, saj ima vgrajeno veliko funkcij, med drugim omogoča analizo kode, grafično razhroščevanje, vgrajeno podporo za testiranje in podporo za povezavo z ogrodjem Django, s katerim lahko razvijamo spletne aplikacije.

3.2.3 GIT

Git je hiter in učinkovit sistem za nadzor različic in izvirne kode, ki je idealen pripomoček za skupinski razvoj programske opreme. Originalno ga je Linus Torvalds razvil za nadzor različic Linux jedra, sčasoma pa se je njegova uporabnost razvila na vsa področja razvoja programske opreme. Vsako Git »skladišče« omogoča preverjanje vseh prejšnjih različic kode in pregled nad spremembami.

3.2.4 GITHUB

GitHub je spletni servis za razvoj programske opreme, ki se povezuje z Git nadzorom različic. GitHub omogoča spletno gostovanje Git »skladišč«, tako da lahko vse dodatke kode vnesemo kjerkoli na svetu in drugi člani razvijalske ekipe lahko te dodatke takoj vidijo in jih tudi lahko prenesejo k sebi.

4 PROGRAMSKA REŠITEV

Kot smo lahko opazili pri analizi sistema Orange, za podatkovno rudarjenje najprej potrebujemo podatke, s katerimi nato z različnimi gradniki iščemo informacije. Ker sistem Orange deluje kot povezovanje gradnikov z različnimi funkcijami, smo našo rešitev razvili na tak način, da bo dodala novo funkcionalnost in hkrati lahko uporabljala vse funkcije, ki so že razvite v sistemu Orange. Odločili smo se, da bomo vse gradnike uvrstili v eno od treh sledečih skupin:

- 1. skupina – gradnik za vnos slike v sistem Orange
- 2. skupina – gradniki za modificiranje slik
- 3. skupina – gradniki za analizo slik

Naša ideja je ta, da je 1. skupina najpomembnejša in brez nje ne moremo začeti s podatkovnim rudarjenjem po slikah, saj gradnik z imenom Images pridobi slike nad katerimi hočemo opravljati podatkovno rudarjenje.

Gradniki 2. skupine niso nujni za podatkovno rudarjenje po slikah, a so uporabni le takrat, ko hočemo slike modificirati (spremeniti barvno paleto ali pa spremeniti velikost slike).

Gradniki 3. skupine so nujni za pridobivanje podatkov iz slik, saj vsebujejo različne algoritme, ki nam iz slik te podatke pridobijo.

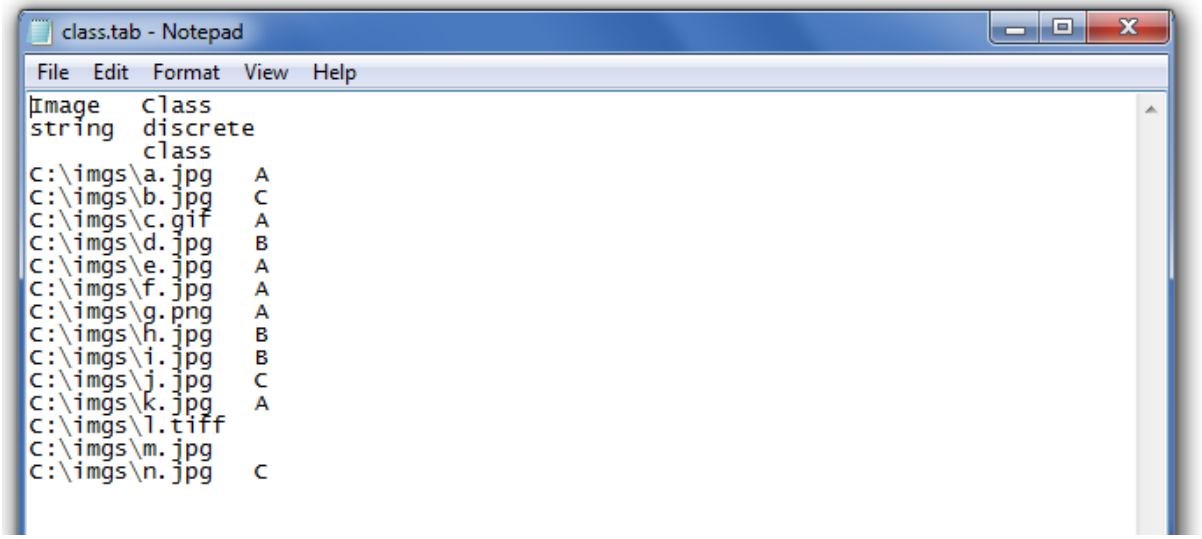
4.2 PRIPRAVA PODATKOV

Za uporabo gradnika je najprej potrebno pripraviti podatke, ki jih bomo lahko dodali v sistem Orange.

Najprej torej potrebujemo primerne podatke, kar bodo v našem primeru slike. Seveda morajo biti slike v pravilnem formatu. Knjižnica Python Image Library (PIL) omogoča ravnanje s sledečimi formati slik – BMP, BUFR, CUR, DCX, EPS, FITS, FLI, FLC, FPX, GBR, GD, GIF, GRIB, HDF5, ICO, IM, IMT, IPTC, NAA, JPEG, MCIDAS, MIC, MPEG, MSP,

PALM, PCD, PCX, PDF, PIXAR, PNG, PPM, PSD, SGI, SPIDER, TGA, TIFF, WAL, WMF, XBM, XPM in XV [13, 14].

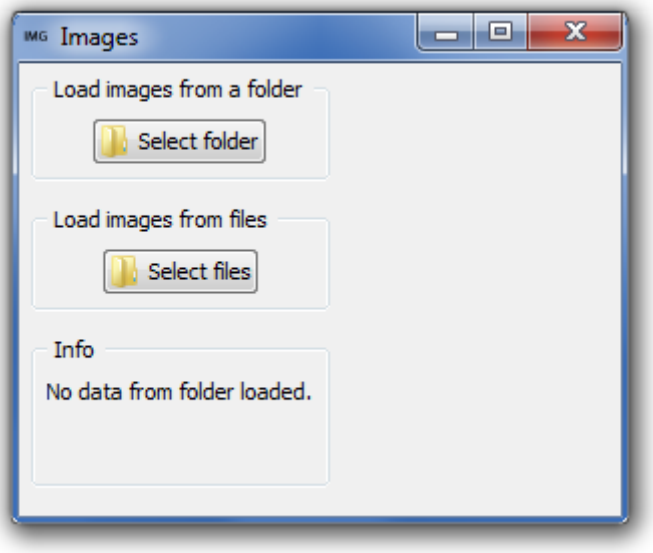
Ob tem potrebujemo datoteko, ki nam bo povedala, v kateri razred spada določena slika. Za ta namen potrebujemo nekaj podobnega datoteki, ki jo potrebuje gradnik File, a z manj podatki. Na sliki 7 je vidna datoteka, ki mora spremljati slike, ko jih preko gradnika Images dodamo v sistem Orange. Prve tri vrstice so enake za vse datoteke za sistem Orange, kot je vidno pa je podatkov precej manj. V datoteki mora biti zapisana le celotna pot do slike in h kateremu razredu spada. Celotna pot je potrebna zaradi analiziranja, zakaj je potreben razred pa smo povedali že v poglavju 3.1 Analiza sistema Orange.



Slika 7: Datoteka s podatki o slikah

4.3 OSNOVNI GRADNIK

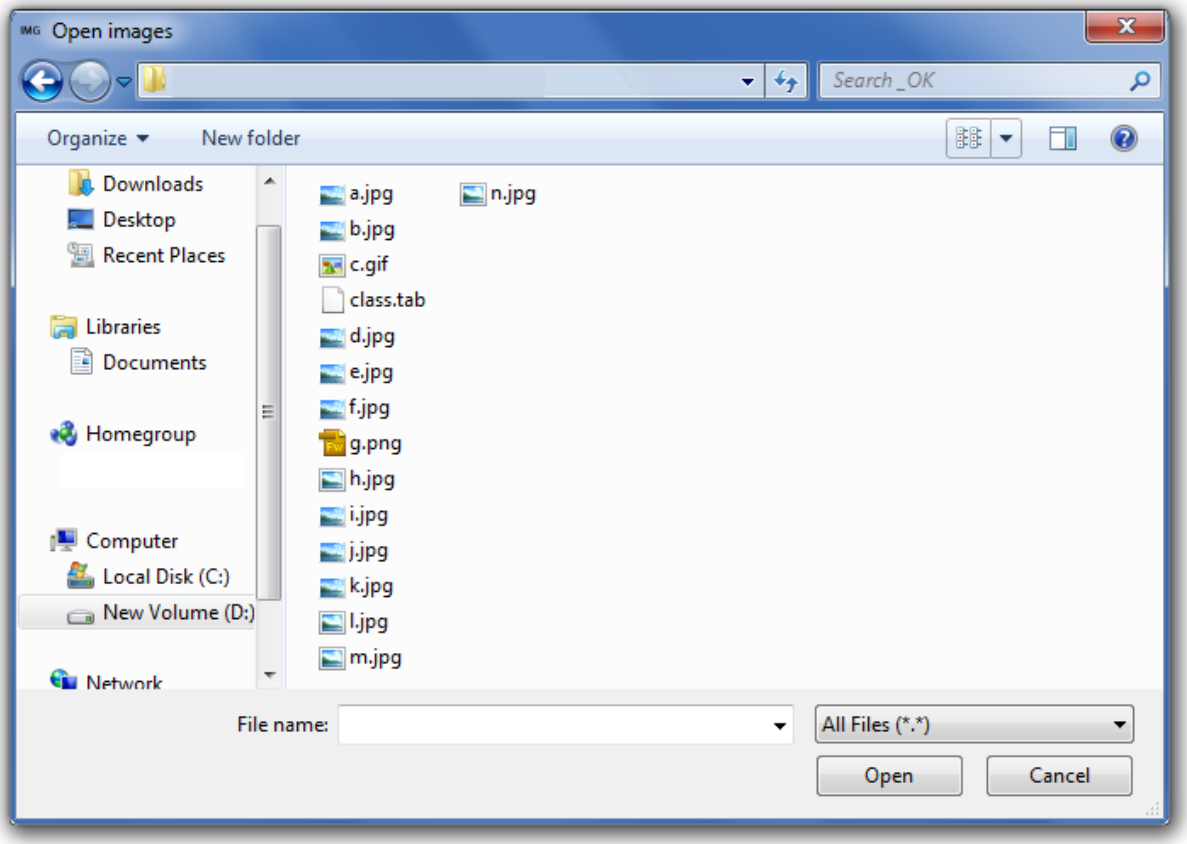
Osnovni gradnik je v našem primeru najbolj pomemben, saj preko njega vnesemo slike v sistem Orange. Kot je vidno na sliki 8, je gradnik Images vizualno zelo podoben gradniku File, ki omogoča dodajanje datotek s podatki v sistem Orange.



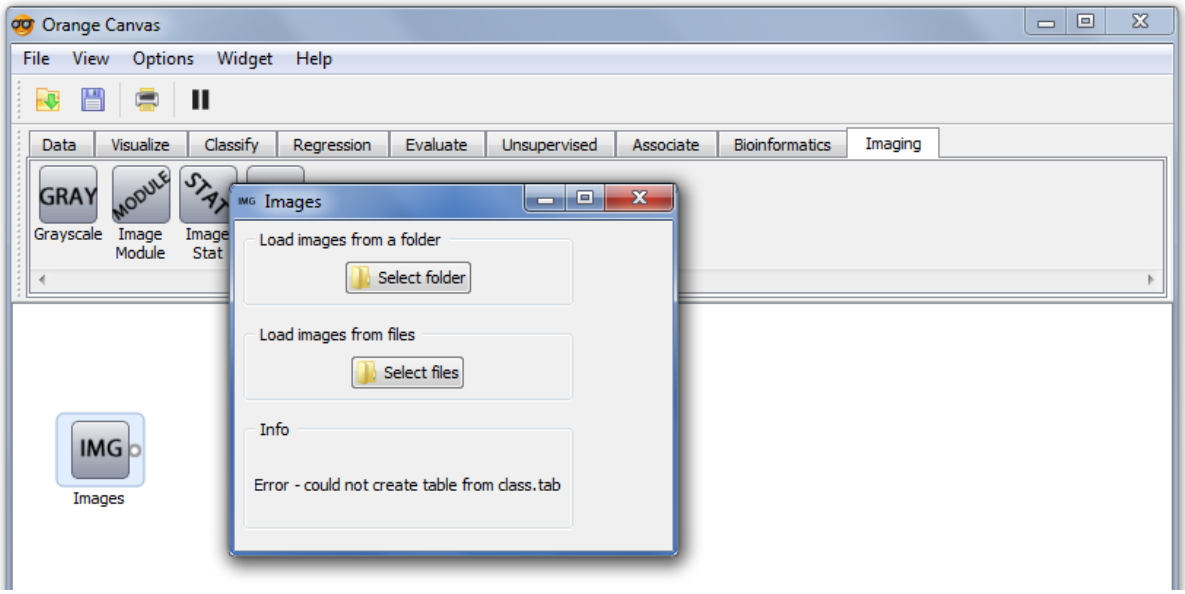
Slika 8: Gradnik Images

Gradnik Images omogoča dodajanje slik na dva načina.

Prvi način, ki je viden na sliki 9, se nam prikaže ob pritisku na gumb »Select Folder«. Odpre se nam okno, kjer izberemo mapo, ki smo jo pripravili – v njej morajo biti slike in razredna datoteka (v našem primeru class.txt), v kateri so zapisane povezave slik z razredom. Gradnik bo nato avtomatsko pregledal tako vsebino mape kot tudi vsebino razredne datoteke. Najprej bo poizkusil iz razredne datoteke ustvariti notranjo tabelo (Orange.data.Table). Ob neuspehu nas bo o tem takoj obvestil, kot je vidno na sliki 10. V primeru, da bo sistem Orange uspešno ustvaril notranjo datoteko, bo najprej pregledal, če slike, ki so zapisane v datoteki, obstajajo, nato pa še preveril, ali so slike v formatu, ki je kompatibilen s knjižnico PIL. V primeru, da sistem ne bo kompatibilen s knjižnico PIL, bo vnose odstranil iz notranje tabele. Kot zadnje bo gradnik Images pregledal celotno mapo in če bo odkril kakšno sliko, ki ni zapisana v razredni datoteki, jo bo dodal v notranjo tabelo.



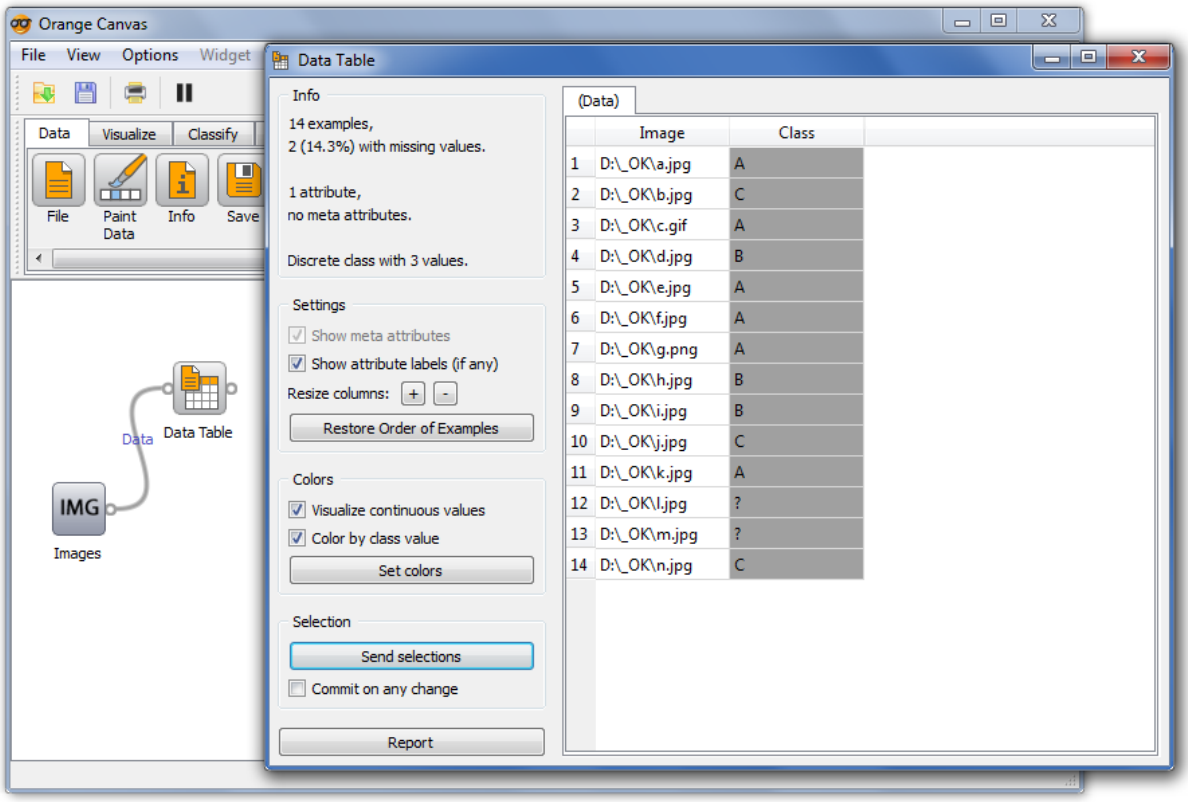
Slika 9: Okno za izbiro slik



Slika 10: Problem pri ustvarjanju notranje tabele

Drugi način za dodajanje slik je ta, da izberemo slike posamezno. Pri tem moramo tudi izbrati razredno datoteko. Ta način deluje podobno kot izbor mape, drugače je le to, da bo gradnik vedno uporabil le izbrane slike. To pomeni, da če v razredni datoteki najde še kakšne dodatne vnose, jih bo ignoriral.

V primeru, da je gradnik uspešno ustvaril notranjo datoteko, lahko z Orange gradnikom Data Table vidimo rezultat na sliki 11.



Slika 11: Uspešno ustvarjenja notranja tabela

4.4 VMESNI GRADNIKI

Vmesni gradniki kot sta »GrayScale« in »Rotate Image« nam omogočata, da slike modificiramo, preden nad njimi poženemo algoritme. Trenutna dva gradnika omogočata spremembo barvne sheme v odtenke sive, ter obračanje slike, v nadaljevanju pa bo možno

dodati gradnike za vse mogoče modifikacije, kot so spreminjanje velikosti, dodajanje različnih filtrov, odstranjevanje določenih barv in drugo.

Gradnik deluje tako, da najprej ustvari novo mapo. Nato vsako sliko glede na gradnik skozi katerega smo spustili slike, modificira in shrani modificirano sliko v novo ustvarjeni mapi. Za tem spremeni zapise v Orang-ovi notranji tabeli, da sedaj kažejo na modificirane slike in ne več na originalne.

Težava pri gradnji tega gradnika se je pojavila, ko smo skozi vmesni gradnik hoteli spraviti veliko slik, ki so bile tudi po velikosti velike. Naenkrat je namreč sistem Orange potreboval zajetne količine računalniških virov, kar je posledično upočasnilo delovanje in odzivnost celotnega sistema.

Delovanje gradnika smo nato spremenili tako, da ta ne pretvori slik in jih obdrži v spominu, temveč, da ustvari novo mapo in v to mapo shrani vsako sliko, ki je bila modificirana. Ta rešitev je izboljšala delovanje gradnika na več načinov. Najprej se je zmanjšala poraba računalnikovega delovnega spomina in procesorske moči, nadaljnje je omogočala, da se uporabijo vmesni gradniki večkrat, kar pri prejšnji rešitvi ne bi bilo mogoče, sej bi zmanjkalo spomina za vse verzije slik, kot zadnje pa nova rešitev omogoča, da lahko v prihodnje kot začetne podatke vzamemo že modificirane slike.

4.5 KONČNI GRADNIKI

Do sedaj smo vnesli slike v sistem Orange, jih po želji modificirali, sedaj pa je čas, da nad njimi poženemo algoritme, ki nam bodo pridobili podatke, nad katerimi lahko izvajamo podatkovno rudarjenje.

Razvili smo dva gradnika, ki sliko spustita skozi nekaj algoritmov, ki jih ima PIL.

Prvi gradnik se imenuje »ImageModule« in ima sledeče funkcije:

- Format – format slike (ponavadi končnica datoteke - .jpg, .tiff, .gif ...)
- Mode – tip slike (RGB, CMYK...)

- Width – širina slike
- Height – višina slike

Drugi gradnik se imenuje »ImageStat« in ima sledeče funkcije:

- Max red – največja vrednost rdeče v vseh slikovnih pikah
- Min red – najmanjša vrednost rdeče v vseh slikovnih pikah
- Max green – največja vrednost zelene v vseh slikovnih pikah
- Min green – najmanjša vrednost zelene v vseh slikovnih pikah
- Max blue – največja vrednost modre v vseh slikovnih pikah
- Min blue – najmanjša vrednost modre v vseh slikovnih pikah
- Count – število vseh slikovnih pik
- Mean red – povprečje rdeče v vseh slikovnih pikah
- Mean green – povprečje zelene v vseh slikovnih pikah
- Mean blue – povprečje modre v vseh slikovnih pikah
- Median red – mediana za rdečo barvo
- Median green – median za zeleno barvo
- Median blue – mediana za modro barvo
- Var red – varianca za rdečo barvo
- Var green – varianca za zeleno barvo
- Var blue – varianca za modro barvo
- Stddev red – standardna devianca za rdečo barvo

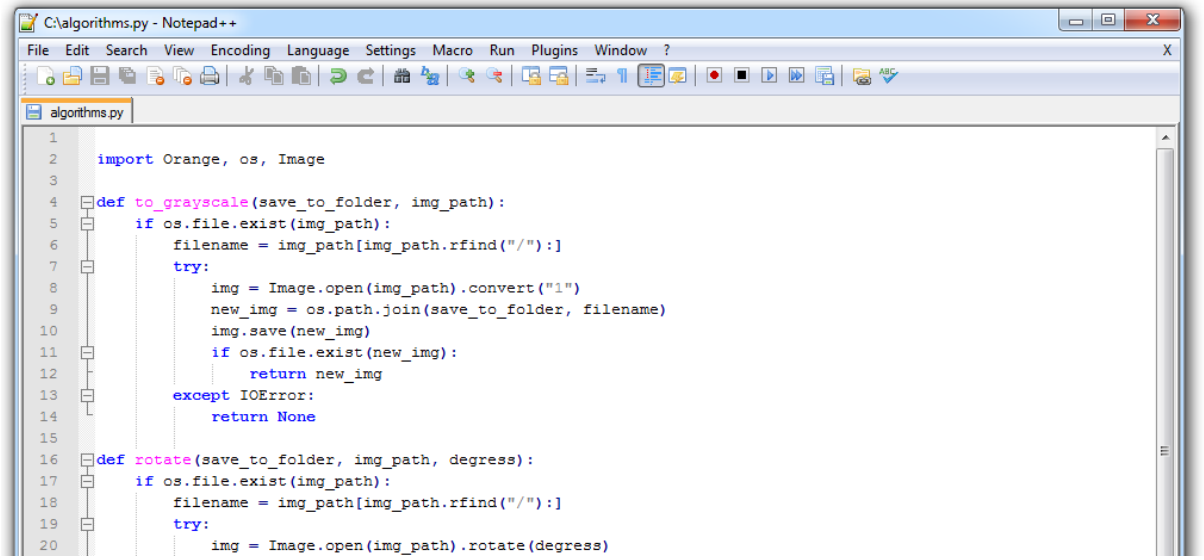
- Stddev green – standardna devianca za zeleno barvo
- Stddev blue – standardna devianca za modro barvo

4.6 DODAJANJE ALGORITMOV

Ena izmed pomembnejših funkcij, ki jih mora dodatek omogočati, je dodajanje algoritmov. Algoritmi so najpomembnejši del podatkovnega rudarjenja, saj z njimi pridobimo atribute, ki jih želimo analizirati. Za vse algoritme, ki jih bo dodatek uporabljal, smo v mapi »_imaging« ustvarili posebno datoteko z imenom »algorithms.py«. V tej datoteki se bodo nahajali vsi algoritmi.

Za primer dodajanja algoritmov bomo sedaj pogledali enega izmed algoritmov, ki je že v dodatku. Na sliki 12 je vidna vsebina datoteke »algorithms.py«, kjer lahko opazimo, da je v njej že nekaj algoritmov - prvi nam omogoča vrtenje slike, drugi pa spreminjaje barvne sheme v odtenke sive. Kot je običajno pri vsaki programski funkciji, tudi ta za samo delovanje potrebuje nekaj podatkov, prav tako pa nekaj podatkov tudi vrača. Ker naša funkcija spreminja in ustvarja nove slike za delovanje, potrebuje mapo, kjer bo slike spravila (»save_to_folder«) in pot do slike, ki jo bo spremenila (»img_path«). Obenem pa mora omogočati še vrnitev poti do nove spremenjene slike (spremenljivka »new_img«).

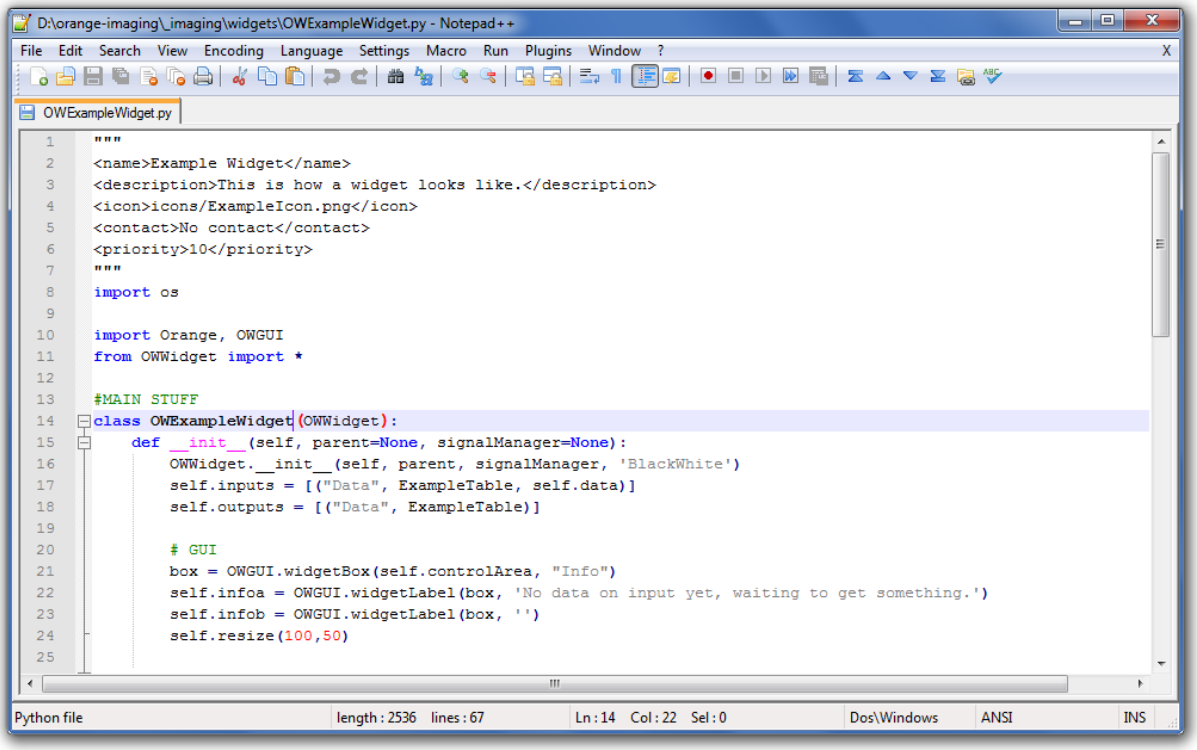
Ko želimo dodati nov algoritem, najprej funkcijo dodamo v datoteko »algorithms.py«. Ker je datoteka v imenskem prostoru imaging, našo funkcijo kličemo z ukazom Orange.imaging. ime_funkcije. Tako na primer funkcijo »to_grayscale« pokličemo z ukazom Orange.imaging.to_ grayscale.



Slika 12: Datoteka algorithms.py

4.7 DODAJANJE GRADNIKOV

V primeru, da hočemo narediti svoj gradnik, lahko to naredimo po naslednjih korakih. V mapo »widgets« dodamo datoteko z imenom »OWExampleWidget.py«, kjer je prikazano, kako izgleda enostaven gradnik. Ta gradnik iz vsake slike, ki jo dobi preko funkcije v datoteki »algorithms.py«, pridobi višino in širino slike, to pa doda v novo notranjo tabelo. Kot je vidno v prvih nekaj vrsticah na sliki 13, vsak gradnik potrebuje nekaj neprogramerskih informacij. Prvih sedem vrstic je rezerviranih za podatke o gradniku – v prvi vrstici so trije narekovaji, sledi ime gradnika, opis gradnika, pot do ikone, avtor, prioriteta, v zadnji vrsti pa so zopet trije narekovaji.



Slika 13: Datoteka s primerom gradnika

Čeprav ta gradnik ne naredi veliko pa vseeno prikaže, kako mora gradnik zgledati ter kako lahko z gradnikom kličemo algoritme iz datoteke algorithms.py. To torej pomeni, da nam ponudi dovolj informacij, da lahko prav vsak naredi nov gradnik.

5 PRIMER UPORABE

Za prikaz delovanja dodatka smo iz interneta pridobili nekaj slik (slike 14 – 23) in na njih izvedli podatkovno rudarjenje.

Slika 14: Testna slika 1

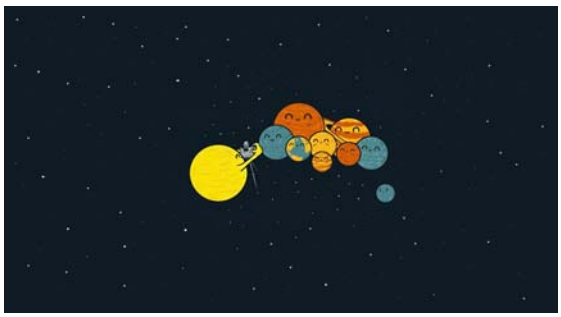
Slika 15: Testna slika 2

Slika 16: Testna slika 3

Slika 17: Testna slika 4

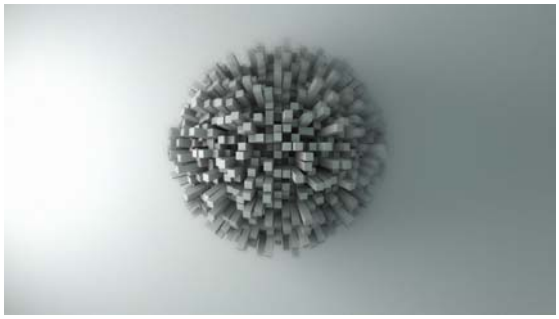
Slika 18: Testna slika 5

Slika 19: Testna slika 6



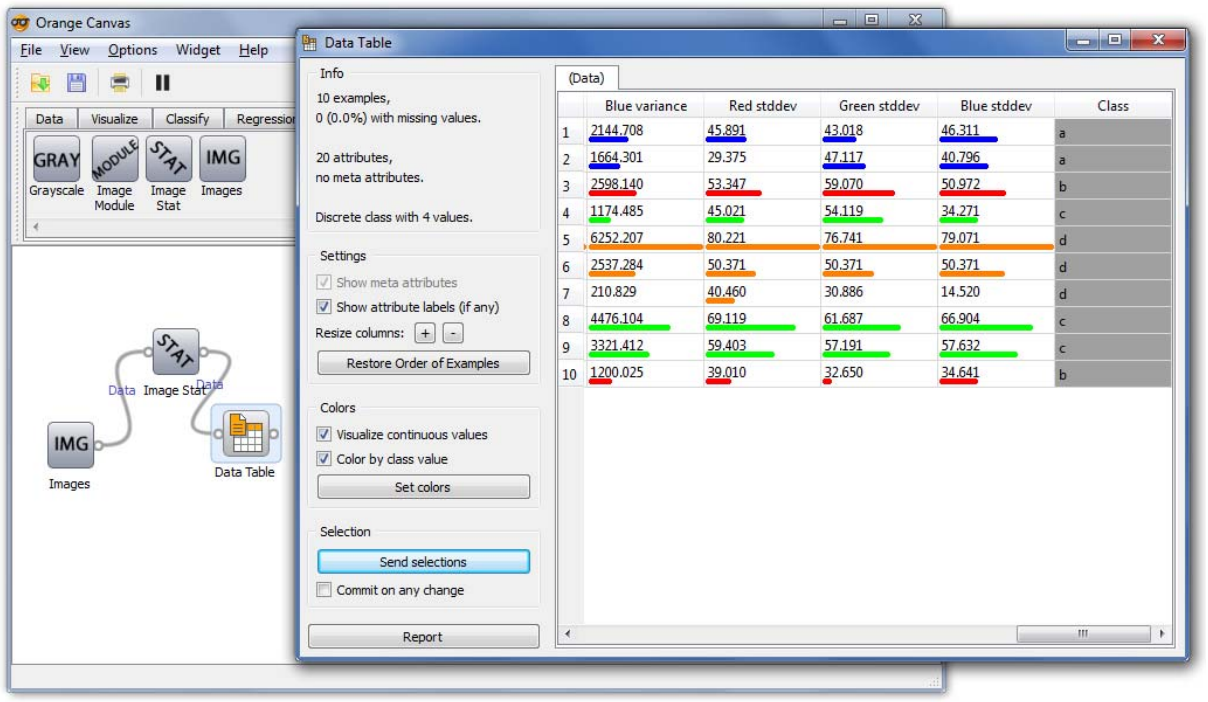
Slika 20: Testna slika 7

Slika 21: Testna slika 8

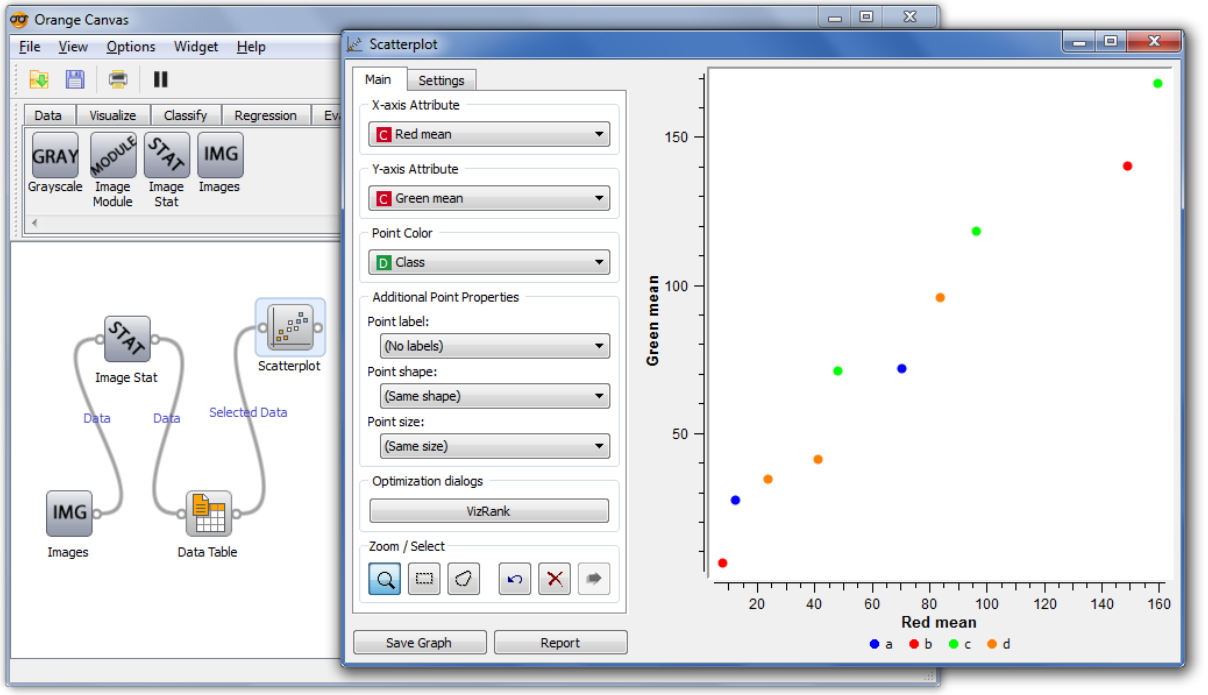


Slika 22: Testna slika 9

Slika 23: Testna slika 10



Slika 24: Tabela Orange po analizi slik z gradnikom Image Stat



Slika 25: Prikaz grafa na podlagi podatkov pridobljenih iz testnih slik

6 SKLEPNE UGOTOVITVE

Cilj diplomskega dela je bil razviti dodatek za sistem Orange, ki bi razširil funkcionalnost sistema in omogočal, da lahko tudi uporabniki sistema Orange sami naprej dodajajo algoritme in gradnike, ki bi še dodatno razširili ne le funkcionalnost našega gradnika, ampak tudi funkcionalnost celotnega sistema.

Ta funkcionalnost je bila dodana z namenom, da bi lahko sistem Orange izvajal podatkovno rudarjenje po slikah in s tem razširil območje delovanja.

Eno izmed področij, kjer bo potrebno še dodatno delo, je dodajanje algoritmov za analizo slik. Čeprav smo dodali kar nekaj algoritmov, se je pojavila težava pri tem, da nekaj bolj znanih algoritmov še ni napisanih v programskem jeziku Python, nekateri pa so bili del zelo velike knjižnice, ki bi jo morali za uporabo prenesti na svoj računalnik.

Trenutno se dodatek sicer še ne uporablja, saj je potrebno še nekaj testiranja, a so se že pojavili interesi za uporabo in pa tudi potencialne nove diplomske naloge, ki bi ta dodatek potrebovale.

PRILOGE

Priloga 1: Izvorna koda

V tem delu je izvorna koda nekaterih gradnikov našega dodatka.

Izvorna koda gradnika Image Module.

```
"""
<name>Image Module</name>
<description>Modify the images to black and white only.</description>
<icon>icons/ImageModule.png</icon>
<contact>bizilj@gmail.com</contact>
<priority>10</priority>
"""

import os
import Orange, OWGUI, Image
from OWWidget import *

class OWImageModule(OWWidget):
    def __init__(self, parent=None, signalManager=None):
        OWWidget.__init__(self, parent, signalManager, 'BlackWhite')
        self.inputs = [("Data", ExampleTable, self.data)]
        self.outputs = [("Data", ExampleTable)]

        # GUI
        box = OWGUI.widgetBox(self.controlArea, "Info")
        self.infoa = OWGUI.widgetLabel(box, 'No data on input yet, waiting to get something.')
        self.infob = OWGUI.widgetLabel(box, '')
        self.resize(100,50)

    def data(self, dataset):
        if dataset:
            self.infoa.setText('%d instances in input data set' % len(dataset))
            sample = self.get_module_data(dataset)
            self.infob.setText('%d sampled instances' % len(sample))
            self.send("Data", sample)
        else:
            self.infoa.setText('No data on input yet, waiting to get something.')
            self.infob.setText('')
            self.send("Data", None)

    def get_module_data(self, dataset):
        not_converted = 0
        new_values = []
        for row in dataset:
            file_values = Orange.imaging.get_image_module(str(row[0]), str(row[1]))
            if file_values is not None:
                new_values.append(file_values)
            else:
                not_converted +=1
        dom_filepath = Orange.feature.String("Path")
        dom_format = Orange.feature.String("Format")
        dom_mode = Orange.feature.String("Mode")
        dom_width = Orange.feature.Continuous("Width")
        dom_height = Orange.feature.Continuous("Height")
```

```
        domain = Orange.data.Domain([dom_filepath, dom_format, dom_mode, dom_width,
dom_height], dataset.domain.class_var)
        new_table = Orange.data.Table(domain, new_values)
        return new_table
```

Izvorna koda gradnika Modify Grayscale.

```
"""
<name>Grayscale</name>
<description>Modify the images to black and white only.</description>
<icon>icons/ModifyBlackWhite.png</icon>
<contact>bizilj@gmail.com</contact>
<priority>10</priority>
"""

import os
import Orange, OWGUI, Image
from OWWidget import *

class OWModifyGrayscale(OWWidget):
    def __init__(self, parent=None, signalManager=None):
        OWWidget.__init__(self, parent, signalManager, 'BlackWhite')
        self.inputs = [("Data", ExampleTable, self.data)]
        self.outputs = [("Data", ExampleTable)]

        # GUI
        box = OWGUI.widgetBox(self.controlArea, "Info")
        self.infoa = OWGUI.widgetLabel(box, 'No data on input yet, waiting to get something.')
        self.infob = OWGUI.widgetLabel(box, '')
        self.resize(100,50)

    def data(self, dataset):
        if dataset:
            self.infoa.setText('%d instances in input data set' % len(dataset))
            sample = self.convert_images(dataset)
            self.infob.setText('%d sampled instances' % len(sample))
            self.send("Data", sample)
        else:
            self.infoa.setText('No data on input yet, waiting to get something.')
            self.infob.setText('')
            self.send("Data", None)

    def convert_images(self, dataset):
        not_converted = 0
        new_values = []
        current_folder = str(dataset[0][0])[:str(dataset[0][0]).rfind("\\")]
        new_folder = Orange.imaging.create_folder(current_folder)
        for row in dataset:
            fileclass = str(row[1])
            converted_image = Orange.imaging.img_to_grayscale(new_folder, str(row[0]))
            if converted_image is not None:
```

```
            new_values.append([converted_image, fileclass])
        else:
            not_converted += 1
    new_table = Orange.data.Table(dataset.domain, new_values)
    return new_table
```

Priloga 2: Pridobitev dodatka

Dodatek bo dostopen na uradni spletni strani za dodatke sistema Orange (http://orange.biolab.si/addons/), kjer bo tudi vsa dokumentacija.

LITERATURA

[1] IBM – ova spletna stran o velikih podatkih. Dostopno na:

http://www-01.ibm.com/software/data/bigdata/

[2] Googlova uradna spletna stran o njihovih robotkih (Googlebot-ih). Dostopno na:

http://support.google.com/webmasters/bin/answer.py?hl=en&answer=182072

[3] Jan Erik Solem, *Programming Computer Vision ith Python*, United States of America: O'Reilly, 2012, poglavja 2 - 9

[4] Uradna spletna stran sistema WEKA. Dostopno na:

http://www.cs.waikato.ac.nz/ml/weka/

[5] Ian H. Witten, Eibe Frank, Mark A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, *Third Edition*, Združene države Amerike, 2011, poglavje 10

[6] Uradna spletna stran sistema KNIME. Dostopno na:

http://www.knime.org/

[7] Micheal R.Berthold, Christian Bergelt, Frank Höppner, Frank Klawonn, *Guide to Intelligent Data Analysis*, London, 2012, dodatek C

[8] Uradna spletna stran sistema RapidMiner. Dostopno na:

http://rapid-i.com/content/view/181/196/

[9] Uradna spletna stran sistema ImageJ. Dostopno na:

http://rsbweb.nih.gov/ij/

[10] Wilhelm Burger, Mark J. Burge, *Digital Image Processing An Algorithmic Introduction Using Java*, Združene države Amerike, 2007, poglavje 3

[11] Uradna spletna stran sistema OpenCV. Dostopno na:

http://opencv.willowgarage.com/wiki/

[12] Gary Bradski, Adrian Kaehler, Learning *OpenCV Computer Vision with the OpenCV Library*, Združene države Amerike, 2008, poglavje 1

[13] Uradna spletna stran sistema Python Image Library. Dostopno na:

http://www.pythonware.com/products/pil/

[14] Jan Erik Solem, *Programming Computer Vision with Python*, Združene države Amerike, 2012, poglavje 1