

Professor: Geraldo Zimbrão da Silva

Aluno: Leniel Macaferi DRE: 116005784

Mestrado em Engenharia de Dados e Conhecimento @ COPPE/UFRJ - Universidade Federal do Rio de Janeiro

Report Classification

The test was conducted using a spreadsheet named [classification.xlsx](#) containing 10793 data samples as shown in the following screenshot:

The screenshot shows a Microsoft Excel spreadsheet titled "classification.xlsx - Excel". The ribbon menu includes File, Home, Insert, Page Layout, Formulas, Data, Review, View, Add-ins, Team, and Tell me what you want to do... The Home tab is selected, displaying various toolbar icons for Cut, Copy, Paste, Format Painter, Arial font, 10pt size, bold, italic, underline, Wrap Text, Merge & Center, General, Number, Conditional Formatting, Styles, Insert, Delete, Format, Cells, and Editing.

The data is organized into several sections:

- Row 1:** A header row containing columns A through U. Columns K, L, M, N, O, P, Q, R, S, T, and U are merged into a single "Count" label. Below it, columns C, N, V, ?, and Total are also merged into a single "Count" label.
- Row 2:** Sub-headings for "Classes" (C, N, V, ?, Total) and their corresponding counts (2767, 3335, 2516, 2175, 10793).
- Row 3:** Sub-headings for "X1" through "X10" and their corresponding counts.
- Data Rows (4-39):** Each row contains 10 numerical values representing feature values for a specific observation. The first column is labeled "id".

At the bottom of the sheet, there are tabs for "Full dataset" and "Full dataset normalized". The status bar at the bottom right shows the time as 9:37 PM and the date as 11/4/2016.

Column X1 and X12 were normalized.

X1 presented values ranging from 9 to 18. The approach taken was to create new feature columns, one for each value in this range. Boolean values were assigned to each row accordingly.

X12 presented dissonant values ranging from 14.23 to -1.06 and as such they were normalized between 0 to 1. This process is important to make the data more conformant\balanced with other existing features (columns) allowing the classifiers to do a better job.

Number crunching was done in Microsoft Excel before serving as input to the classifiers' algorithms.

Development was done using [Visual Studio Code](#) as the IDE and [Python](#) programming language with [scikit-learn](#) Machine Learning library, [NumPy](#) and [Pandas](#).

Source code is available @ <https://github.com/leniel/DataMining>

Useful reading

Naive Bayes http://scikit-learn.org/stable/modules/naive_bayes.html

SVM <http://scikit-learn.org/stable/modules/svm.html>

Neural Networks http://scikit-learn.org/stable/modules/neural_networks_supervised.html

Model evaluation: http://scikit-learn.org/stable/modules/model_evaluation.html

Cross-validation: evaluating estimator performance:

http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Data

The Training sheet contains the massaged data which is already labeled with classes C, N and V.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	id	9	10	11	12	13	14	15	16	17	18	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12 normalized	Classe			
2	10	1	0	0	0	0	0	0	0	0	0	1.001078511	1.000849217	1.000867242	1.000637997	1.000559709	1.000330534	1.001636075	1.001406654	1.001328542	1.001099191	0.125150615	C			
3	11	1	0	0	0	0	0	0	0	0	0	1.001849504	1.001681246	1.001014183	1.000846064	1.001014183	1.000846064	1.002705085	1.002536683	1.00224393	1.002075605	0.282111013	C			
4	12	1	0	0	0	0	0	0	0	0	0	1.002004658	1.001797901	1.001753493	1.001546788	1.001753493	1.00236816	1.002161328	1.001753493	1.001546788	0.112217849	C				
5	13	1	0	0	0	0	0	0	0	0	0	1.001508591	1.001248332	1.001366752	1.001106522	1.001673967	1.001413657	1.001366752	1.001106522	0.051329424	0.051329424	C				
6	14	1	0	0	0	0	0	0	0	0	0	1.001623035	1.001263606	1.001135115	1.000981566	1.000622367	1.002056405	1.00169682	1.00159576	1.00123634	0.125632581	C				
7	15	1	0	0	0	0	0	0	0	0	0	1.000841459	1.000451432	1.00109839	1.000719708	1.000649418	1.000259466	1.001263313	1.000873122	1.000802892	1.00041288	0.077596594	C			
8	16	1	0	0	0	0	0	0	0	0	0	1.001011326	1.000722527	1.00036772	1.000791106	1.000214313	1.00099925743	1.001288163	1.0009999411	1.000981348	1.000692558	0.087476906	V			
9	17	1	0	0	0	0	0	0	0	0	0	1.000428722	1.000262884	1.000773612	1.000607711	1.000160205	1.000773612	1.000607716	1.000602206	1.00045439	0.060884825	V				
10	18	1	0	0	0	0	0	0	0	0	0	1.000327545	1.000258249	1.000205476	1.0009968124	1.000989885	1.000734733	1.000665411	1.000274768	1.000205476	0.068760543	V				
11	19	1	0	0	0	0	0	0	0	0	0	0.99975457	0.999645065	0.999719646	0.999609914	0.999566415	0.999456751	0.0032954	0.999719723	0.999610042	0.072696602	C				
12	21	1	0	0	0	0	0	0	0	0	0	1.00107015	1.000957801	1.000382351	1.000270079	1.000382351	1.000270079	1.001455394	1.001343001	1.001455394	1.001343001	0.161940718	V			
13	22	1	0	0	0	0	0	0	0	0	0	1.000964873	1.000898391	1.001247102	1.00180601	1.000634062	1.000567602	1.001400362	1.00133385	1.000787322	1.000720851	0.104827697	V			
14	23	1	0	0	0	0	0	0	0	0	0	1.000473977	1.000336743	1.00047772	1.000340485	1.000171248	1.000034055	1.000784192	1.000646914	1.000171248	1.000034055	0.059121215	V			
15	24	1	0	0	0	0	0	0	0	0	0	1.000320209	1.000211945	1.00000308	0.999892151	1.000000038	0.999892151	1.000766429	1.000658118	1.00061322	1.000504924	0.055908105	V			
16	26	1	0	0	0	0	0	0	0	0	0	0.999342069	0.999269024	0.999811302	0.999738222	0.999045516	0.999897429	0.99996446	0.999891369	0.99935183	0.999278784	0.138565347	V			
17	27	1	0	0	0	0	0	0	0	0	0	0.998687634	0.998784437	0.999351834	0.999448701	0.999167124	0.999279733	0.998376495	0.999351834	0.999448701	0.998739205	0.998836012	0.118724396	C		
18	28	1	0	0	0	0	0	0	0	0	0	0.998877794	0.999077738	0.9998577441	0.999157256	0.998497869	0.998697591	0.999263823	0.999463699	0.998804251	0.999004034	0.059201542	C			
19	29	1	0	0	0	0	0	0	0	0	0	0.99870073	0.998868639	0.999096177	0.999264152	0.998483235	0.998651107	0.999096177	0.999264152	0.998942942	0.999110891	0.097035906	C			
20	30	1	0	0	0	0	0	0	0	0	0	0.999121346	0.999365736	0.99907066	0.999315014	0.998917362	0.999167124	0.99923939	0.999468356	0.99907066	0.99931504	0.020081934	C			
21	31	1	0	0	0	0	0	0	0	0	0	0.999432081	0.999690786	0.999378427	0.999637118	0.998611797	0.9987029	0.00298383	0.999057312	0.999991731	1.000250581	0.180576753	C			
22	32	1	0	0	0	0	0	0	0	0	0	1.000316781	1.000487971	1.000434881	1.000606092	1.0012814	1.000299299	1.000434881	1.000606092	1.000434881	1.000606092	0.229737333	C			
23	33	1	0	0	0	0	0	0	0	0	0	1.00070584	1.000706573	1.000557268	1.000250491	1.000251222	1.001017435	1.001018168	1.001017435	1.001018168	0.099686722	C				
24	34	1	0	0	0	0	0	0	0	0	0	1.000870938	1.000834576	1.000794066	1.000757707	1.000640687	1.000947444	1.00091108	1.000794066	1.000757707	0.033576994	C				
25	35	1	0	0	0	0	0	0	0	0	0	1.000316114	1.000151654	1.000674193	1.000509674	0.999754033	0.999589665	1.000827553	1.000663009	0.12033095	0.12033095	C				
26	37	1	0	0	0	0	0	0	0	0	0	1.000123167	0.999924156	1.000152904	0.999953887	0.999629993	0.999490468	1.000612815	1.000413707	1.00045912	1.000260433	0.066591694	C			
27	38	1	0	0	0	0	0	0	0	0	0	1.000099498	1.000043598	1.000114746	1.000058844	0.99964923	0.999599047	1.000421294	1.000365375	0.999961471	0.999905578	0.045947466	C			
28	39	1	0	0	0	0	0	0	0	0	0	0.999524372	0.999500178	1.000054484	1.000030277	0.999134894	0.999110708	1.000054484	1.00003277	0.999594689	0.999570493	0.087637561	C			
29	40	1	0	0	0	0	0	0	0	0	0	0.999385473	0.999456838	0.999467187	0.999385658	0.999160469	0.999231998	0.999620456	0.99961838	0.999160469	0.999231998	0.027793397	C			
30	41	1	0	0	0	0	0	0	0	0	0	0.999354457	0.999367526	0.999439564	0.999452631	0.998826411	0.998834969	0.999899428	0.999912501	0.999132987	0.99914602	0.126865035	C			
31	42	1	0	0	0	0	0	0	0	0	0	1.000053112	1.000225916	0.999209248	0.999381806	0.999209248	0.999381806	1.000588948	1.000761744	1.000435648	1.000608418	0.120571933	C			
32	43	1	0	0	0	0	0	0	0	0	0	1.0000991746	1.001023601	1.000585823	1.000617665	1.000279177	1.000311009	1.001352438	1.001384304	1.001199115	1.001230976	0.170375131	C			
33	44	1	0	0	0	0	0	0	0	0	0	1.00128638	1.001273822	1.001273801	1.001115945	1.000975188	1.000568234	1.001588437	1.001575875	1.001435125	0.120332585	0.11856374	N			

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	id	9	10	11	12	13	14	15	16	17	18	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	normalized	Classe		
2	20	1	0	0	0	0	0	0	0	0	0	1.000514806	1.000539084	0.999645523	0.99966978	0.999645523	0.99966978	1.000871893	1.00089618	1.000412004	1.00043628	0.115511286	?			
3	25	1	0	0	0	0	0	0	0	0	0	1.000152554	0.999932977	1.00048372	1.00026407	0.99987096	0.999651445	1.00048372	1.00026407	1.0002415	0.999804601	0.035665515	?			
4	36	1	0	0	0	0	0	0	0	0	0	1.000822541	1.000704701	1.000494501	1.0003767	1.000187836	1.000070071	1.001261163	1.001143272	1.000341168	1.000223385	0.142822717	?			
5	60	1	0	0	0	0	0	0	0	0	0	0.999853791	0.999790571	1.000197033	1.000133792	0.999585198	0.999521996	1.000349992	1.000286741	0.999891116	0.999827894	0.033496666	?			
6	61	0	1	0	0	0	0	0	0	0	0	1.00010736	1.000133319	1.000001786	1.000027742	0.999848833	0.999874786	1.00030769	1.000333664	1.00030769	1.000333664	0.032854045	?			
7	63	0	1	0	0	0	0	0	0	0	0	0.999994926	0.99992616	1.000201113	1.00132333	0.999742235	0.999673486	1.000201113	1.00132333	0.999895194	0.999826435	0.021849145	?			
8	76	0	1	0	0	0	0	0	0	0	0	1.000396591	1.00040595	1.000479889	1.000489249	1.000174073	1.000183437	1.000642159	1.000632798	1.000642159	0.04201734	?				
9	78	0	1	0	0	0	0	0	0	0	0	1.001136274	1.000993462	1.00103037	1.000940265	1.000930186	1.00138837	1.001245989	1.000930186	1.000787403	1.000787403	0.058317937	?			
10	85	0	1	0	0	0	0	0	0	0	0	1.000411014	1.000314257	1.000408321	1.00031565	1.000255634	1.000158892	1.000561009	1.000464238	1.000408321	1.000311565	0.031809784	?			
11	95	0	1	0	0	0	0	0	0	0	0	0.999394356	0.999500369	0.999622009	0.999728046	0.999163887	0.999269876	0.999774716	0.99980877	0.999163887	0.999269876	0.078319544	?			
12	101	0	1	0	0	0	0	0	0	0	0	1.000706111	1.000636105	1.000513715	1.000443722	1.000369042	1.000290959	1.000972036	1.000902011	1.000666489	1.000596485	0.092216242	?			
13	105	0	1	0	0	0	0	0	0	0	0	1.00072699	1.000475589	1.000723592	1.000472192	1.000319588	1.000876234	1.000247971	1.000723592	1.000472192	0.046831071	?				
14	112	0	1	0	0	0	0	0	0	0	0	1.000781058	1.000696721	1.0008814	1.000797054	1.000576392	1.0004942072	1.001033903	1.000949545	1.000728896	1.000644653	0.99832115	?			
15	121	0	0	1	0	0	0	0	0	0	0	0.998682556	0.998714531	1.000105675	1.000137692	0.998005202	1.000105675	1.000137692	0.999344096	0.999376088	0.475540204	?				
16	123	0	0	1	0	0	0	0	0	0	0	0.998144864	0.998309409	0.998498885	0.998663488	0.997883994	0.998053897	0.998651256	0.998815888	0.998191439	0.998358693	0.118403085	?			
17	132	0	0	1	0	0	0	0	0	0	0	1.000902062	1.000901979	1.000286226	1.000289359	1.000286226	1.000289359	1.001357718	1.000896715	1.000896985	0.130612901	?				
18	135	0	0	1	0	0	0	0	0	0	0	1.000723186	1.000491026	1.000494344	1.000262238	1.000109713	1.001257149	1.001204866	1.000952027	1.000719815	0.10971966	?				
19	137	0	0	1	0	0	0	0	0	0	0	1.001050299	1.000824711	1.000903924	1.000678369	1.00075144	1.00052592	1.001208891	1.000983268	1.00075144	1.00052592	0.106193269	?			
20	151	0	0	1	0	0	0	0	0	0	0	1.001357909	1.001244145	1.001000331	1.000986296	1.001100031	1.000986296	1.001595169	1.001100031	1.000986296	0.128524379	?				
21	163	0	0	1	0	0	0	0	0	0	0	0.998617344	0.998837066	0.998811766	0.999031531	0.998355062	0.998574726	0.998964001	0.999183799	0.998659532	0.998879263	0.156157121	?			
22	167	0	0	1	0	0	0	0	0	0	0	0.999767761	0.999919172	0.999647888	0.99979288	0.999495502	0.999646872	1.000105044	1.000256505	0.999952658	1.000104097	0.049240903	?			
23	169	0	0	1	0	0	0	0	0	0	0	0.999616807	0.999600777	0.99951639	0.99957561	0.999439216	0.999423189	0.999864845	0.999880451	0.999744062	0.99972803	0.043216323	?			
24	171	0	0	1	0	0	0	0	0	0	0	0.999846896	0.999930223	0.99985051	0.999933837	0.999698071	0.999781385	1.000029595	1.00006307306	1.000041204	1.0000451245	0.02699012	?			
25	173	0	0	1	0	0	0	0	0	0	0	1.000434659	1.000473865	1.00041204	1.000451245	1.000259584	1.000298783	1.000564945	1.000603706	1.000041204	1.0000451245	0.02699012	?			
26	178	0	0	1	0	0	0	0	0	0	0	0.999439459	0.999387602	0.99987676	0.99993579	0.999073053	0.999021215	1.000104113	1.000088219	0.999073053	0.999021215	0.175837417	?			
27	185	0	0	0	1	1	0	0	0	0	0	0.999503146	0.999287583	0.99971545	0.999499842	0.999410798	0.999195255	0.038476986	?							
28	186	0	0	0	1	1	0	0	0	0	0	0.999523977	0.999456043	0.999266445	0.999198528	0.999272357	0.999655409	0.999571053	0.999503116	0.026427826	?					
29	194	0	0	0	1	1	0	0	0	0	0	1.00009448	0.999978296	1.000000305	0.999969153	0.999655119	0.99964377	1.00030509	1.000273929	1.00030509	1.000273929	0.048116315	?			
30	202	0	0	0	1	1	0	0	0	0	0	0.999690139	0.999761076	0.999540475	0.999611402	0.999388037	0.999458952	0.999979792	1.00006875	0.999845363	0.999916301	0.061691702	?			
31	206	0	0	0	1	1	0	0	0	0	0	0.999202351	0.999268706	0.999765893	0.9998832285	0.999881746	0.999765893	0.999885015	0.999886216	0.999885015	0.9998917246	0.15157844	?			
32	229	0	0	0	1	1	0	0	0	0	0	1.000213149	1.000162375	1.000818457	1.000130684	1.000081457	1.000130684	1.000334156	1.000283376	1.000334156	1.000283376	0.004177042	?			
33	253	0	0	0	0	0	0	0	0	0	0	1.000194047	1.000162914	1.000041794	1.000055659	1.000041794	1.000055659	1.000194729	1.000208597	1.000194729	1.000208597	0.012290144	?			
34	259	0	0	0	0	0	0	0	0	0	0	1.000956479	1.000904711	1.0008871244	1.00081948	1.000718322	1.000666567	1.001177087	1.001125308	1.000718322	1.000666567	0.0440196	?			

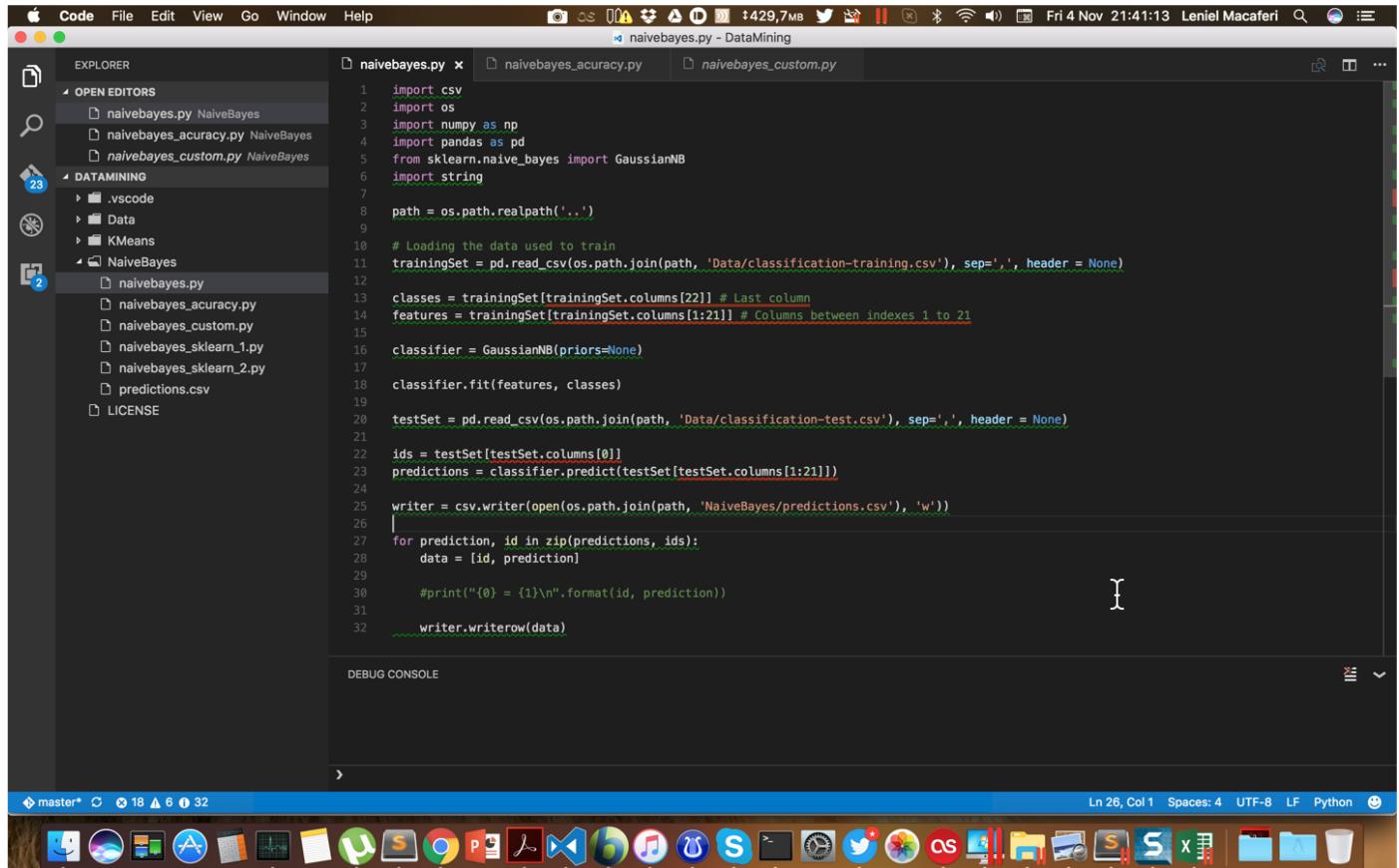
The Test sheet contains the massaged data which is unclassified ?.

The data in Excel XLSX file was saved in the CSV file format to serve as input to the Python source code.

Two files were created:

- [classification-training.csv](#) [including the sample data already classified]
- [classification-test.csv](#) [including the sample data unclassified]

Naive Bayes



```
1 import csv
2 import os
3 import numpy as np
4 import pandas as pd
5 from sklearn.naive_bayes import GaussianNB
6 import string
7
8 path = os.path.realpath('..')
9
10 # Loading the data used to train
11 trainingSet = pd.read_csv(os.path.join(path, 'Data/classification-training.csv'), sep=',', header = None)
12
13 classes = trainingSet[trainingSet.columns[22]] # Last column
14 features = trainingSet[trainingSet.columns[1:21]] # Columns between indexes 1 to 21
15
16 classifier = GaussianNB(priors=None)
17
18 classifier.fit(features, classes)
19
20 testSet = pd.read_csv(os.path.join(path, 'Data/classification-test.csv'), sep=',', header = None)
21
22 ids = testSet[testSet.columns[0]]
23 predictions = classifier.predict(testSet[testSet.columns[1:21]])
24
25 writer = csv.writer(open(os.path.join(path, 'NaiveBayes/predictions.csv'), 'w'))
26
27 for prediction, id in zip(predictions, ids):
28     data = [id, prediction]
29
30     #print "{0} = {1}\n".format(id, prediction)
31
32     writer.writerow(data)
```

The resulting predictions are saved to an external .CSV file named [nb-predictions.csv](#).

The screenshot shows the Xcode IDE interface with a Python file named `naivebayes_crossvalidation.py` open. The code implements a Naive Bayes classifier using scikit-learn and performs 5-fold cross-validation. The output window displays the accuracy results.

```
import os
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score

path = os.path.realpath('..')

# Loading the data used to train
trainingSet = pd.read_csv(os.path.join(path, 'Data/classification-training.csv'), sep=',', header = None)

classes = trainingSet[trainingSet.columns[22]] # Last column
features = trainingSet[trainingSet.columns[1:22]] # Columns between indexes 1 to 22

#pd.set_option('display.max_columns', 23)
#print(features)

classifier = GaussianNB()

scores = cross_val_score(classifier, features, classes, cv = 5)
print(scores)

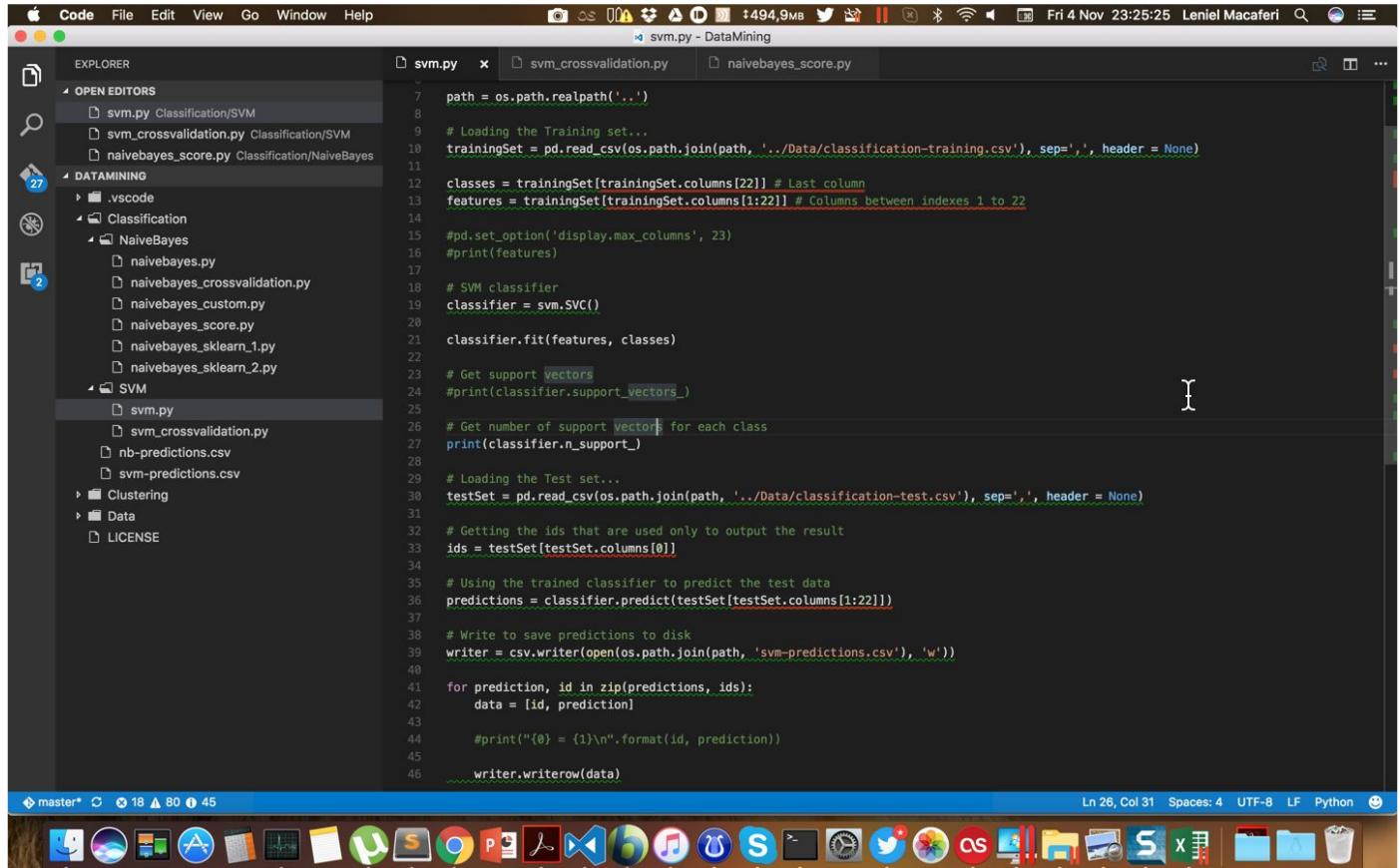
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

DEBUG CONSOLE

```
[ 0.39362319  0.41183295  0.4039466   0.369704   0.37899013]
Accuracy: 0.39 (+/- 0.03)
```

Accuracy with 5 fold validation = 0.39

SVM - Support Vector Machines



The screenshot shows a Mac OS X desktop environment. At the top, there's a menu bar with Apple, Code, File, Edit, View, Go, Window, Help, and a system status bar showing battery level (t494,9MB), signal strength, and the date/time (Fri 4 Nov 23:25:25). Below the menu bar is a dock with various application icons.

The main area features a terminal window on the left and a Visual Studio Code (VS Code) editor on the right. The terminal window has a blue header with the text "master*". It displays a list of files in the current directory:

```
master* ➜ 18 ▲ 80 ④ 45
svm.py  svm_crossvalidation.py  naivebayes_score.py
nb-predictions.csv
svm-predictions.csv
testSet.csv
```

The VS Code editor has a dark theme. The Explorer sidebar on the left shows a tree view of the project structure:

- OPEN EDITORS:
 - svm.py Classification/SVM
 - svm_crossvalidation.py Classification/SVM
 - naivebayes_score.py Classification/NaiveBayes
- DATAMINING:
 - .vscode
 - Classification
 - NaiveBayes
 - naivebayes.py
 - naivebayes_crossvalidation.py
 - naivebayes_custom.py
 - naivebayes_score.py
 - naivebayes_sklearn_1.py
 - naivebayes_sklearn_2.py
 - SVM
 - svm.py
 - svm_crossvalidation.py
- nb-predictions.csv
- svm-predictions.csv
- Clustering
- Data
- LICENSE

The main editor tab is "svm.py" with the title "svm.py - DataMining". The code in the editor is as follows:

```
7 path = os.path.realpath('..')
8
9 # Loading the Training set...
10 trainingSet = pd.read_csv(os.path.join(path, '../Data/classification-training.csv'), sep=',', header = None)
11
12 classes = trainingSet[trainingSet.columns[22]] # Last column
13 features = trainingSet[trainingSet.columns[1:22]] # Columns between indexes 1 to 22
14
15 #pd.set_option('display.max_columns', 23)
16 #print(features)
17
18 # SVM classifier
19 classifier = svm.SVC()
20
21 classifier.fit(features, classes)
22
23 # Get support vectors
24 #print(classifier.support_vectors_)
25
26 # Get number of support vectors for each class
27 print(classifier.n_support_)
28
29 # Loading the Test set...
30 testSet = pd.read_csv(os.path.join(path, '../Data/classification-test.csv'), sep=',', header = None)
31
32 # Getting the ids that are used only to output the result
33 ids = testSet[testSet.columns[0]]
34
35 # Using the trained classifier to predict the test data
36 predictions = classifier.predict(testSet[testSet.columns[1:22]])
37
38 # Write to save predictions to disk
39 writer = csv.writer(open(os.path.join(path, 'svm-predictions.csv'), 'w'))
40
41 for prediction, id in zip(predictions, ids):
42     data = [id, prediction]
43
44     #print("{0} = {1}\n".format(id, prediction))
45
46     writer.writerow(data)
```

The status bar at the bottom of the VS Code editor shows "Ln 26, Col 31" and other settings like "Spaces: 4", "UTF-8", "LF", "Python", and a smiley face icon.

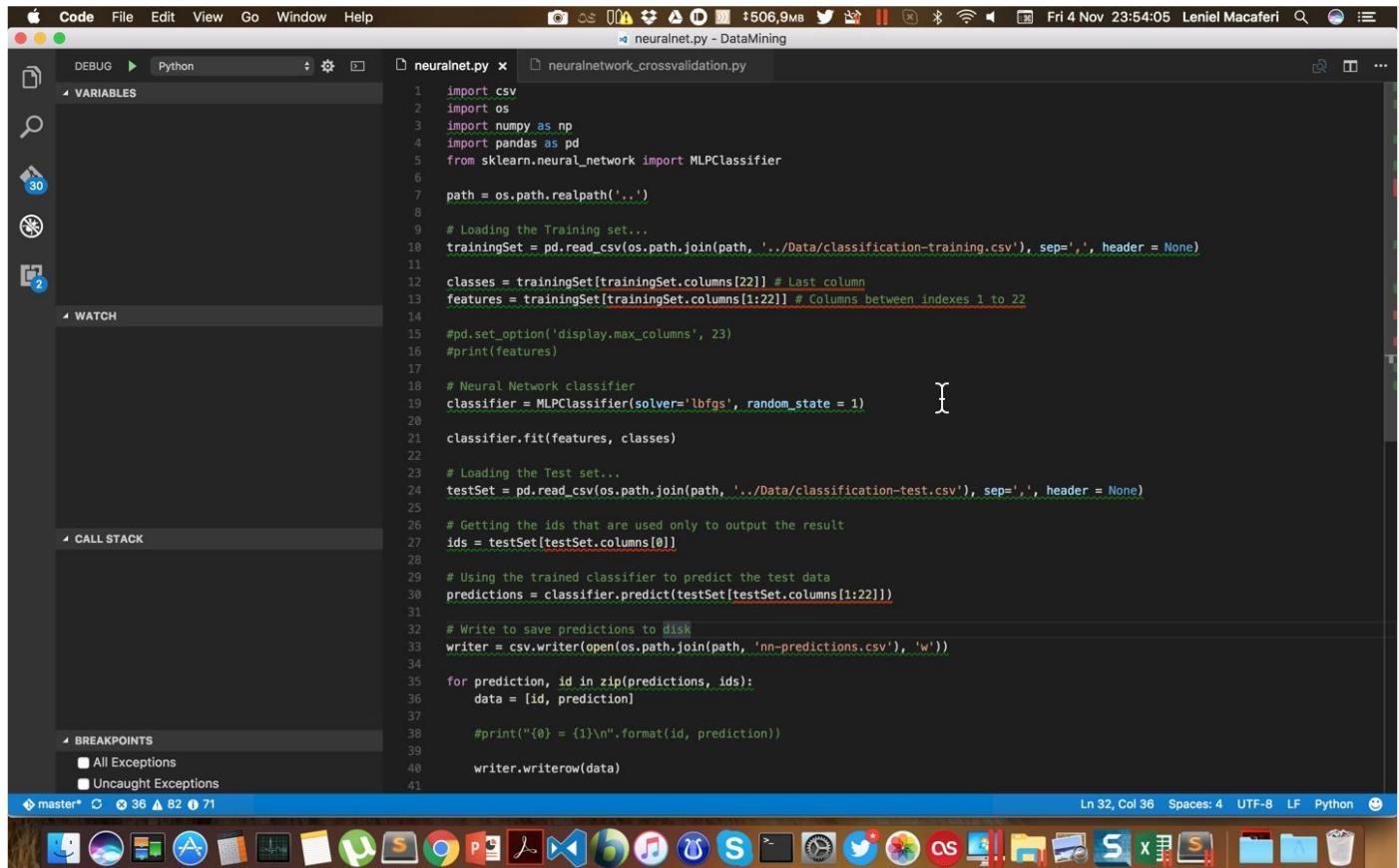
The resulting predictions are saved to an external .CSV file named [svm-predictions.csv](#).

The screenshot shows the Xcode IDE interface with the following details:

- Top Bar:** Shows the menu bar with Apple, Code, File, Edit, View, Go, Window, Help.
- Toolbar:** Includes icons for camera, file, search, and various system status indicators.
- Code Editor:** Displays a Python script named `svm_crossvalidation.py`. The code imports os, pandas, sklearn.svm, and sklearn.model_selection, then reads a CSV file, splits it into features and classes, creates an SVC classifier, performs 5-fold cross-validation, and prints the accuracy.
- Symbols View:** Shows variables, watch expressions, and breakpoints.
- Call Stack:** Shows the current call stack.
- Breakpoints:** Shows checkboxes for All Exceptions and Uncaught Exceptions.
- Debug Console:** Displays the output of the print statement: [0.50782609 0.45881671 0.42716193 0.42542078 0.46024376] Accuracy: 0.46 (+/- 0.06)
- Bottom Bar:** Shows the current branch (master), commit hash (x 18), and other system information like Ln 24, Col 73, Spaces: 4, UTF-8, LF, Python.

Accuracy with 5 fold validation = 0.46

Neural Network



```
1 import csv
2 import os
3 import numpy as np
4 import pandas as pd
5 from sklearn.neural_network import MLPClassifier
6
7 path = os.path.realpath('..')
8
9 # Loading the Training set...
10 trainingSet = pd.read_csv(os.path.join(path, '../Data/classification-training.csv'), sep=',', header = None)
11
12 classes = trainingSet[trainingSet.columns[22]] # Last column
13 features = trainingSet[trainingSet.columns[1:22]] # Columns between indexes 1 to 22
14
15 #pd.set_option('display.max_columns', 23)
16 #print(features)
17
18 # Neural Network classifier
19 classifier = MLPClassifier(solver='lbfgs', random_state = 1)
20
21 classifier.fit(features, classes)
22
23 # Loading the Test set...
24 testSet = pd.read_csv(os.path.join(path, '../Data/classification-test.csv'), sep=',', header = None)
25
26 # Getting the ids that are used only to output the result
27 ids = testSet[testSet.columns[0]]
28
29 # Using the trained classifier to predict the test data
30 predictions = classifier.predict(testSet[testSet.columns[1:22]])
31
32 # Write to save predictions to disk
33 writer = csv.writer(open(os.path.join(path, 'nn-predictions.csv'), 'w'))
34
35 for prediction, id in zip(predictions, ids):
36     data = [id, prediction]
37
38     #print("{0} = {1}\n".format(id, prediction))
39
40     writer.writerow(data)
41
```

The resulting predictions are saved to an external .CSV file named [nn-predictions.csv](#).

The screenshot shows the Xcode IDE interface with a Python script named `neuralnetwork_crossvalidation.py` open. The code implements a neural network classifier using scikit-learn's MLPClassifier and performs 5-fold cross-validation to calculate accuracy.

```
1 import os
2 import pandas as pd
3 from sklearn.neural_network import MLPClassifier
4 from sklearn.model_selection import cross_val_score
5
6 path = os.path.realpath('..')
7
8 # Loading the data used to train
9 trainingSet = pd.read_csv(os.path.join(path, '../Data/classification-training.csv'), sep=',', header = None)
10
11 classes = trainingSet[trainingSet.columns[22]] # Last column
12 features = trainingSet[trainingSet.columns[1:22]] # Columns between indexes 1 to 22
13
14 #pd.set_option('display.max_columns', 23)
15 #print(features)
16
17 # Neural Network classifier
18 classifier = MLPClassifier(solver='lbfgs', random_state = 1)
19
20 scores = cross_val_score(classifier, features, classes, cv = 5)
21
22 print(scores)
23
24 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

The DEBUG CONSOLE window shows the output of the script execution:

```
[ 0.52463768  0.45707657  0.45676146  0.4451538   0.47533372]
Accuracy: 0.47 (+/- 0.06)
```

Accuracy with 5 fold validation = 0.47

Analysis

The spreadsheet is available @

<https://github.com/leniel/DataMining/blob/master/Data/classification.xlsx>

The Predictions sheet lists all output from the classifiers:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1				Naive Bayes		SVM		Neural Network		Count	862	% Matches	39.63%											
2				id	Class	id	Class	id	Class			Match												
3				20	C	20	C	20	C			TRUE												
4				25	C	25	C	25	C			TRUE												
5				36	C	36	C	36	C			TRUE												
6				60	C	60	C	60	C			TRUE												
7				61	V	61	V	61	V			TRUE												
8				63	V	63	V	63	V			TRUE												
9				76	C	76	V	76	V			FALSE												
10				78	C	78	V	78	V			FALSE												
11				85	C	85	V	85	V			FALSE												
12				95	V	95	V	95	V			TRUE												
13				101	C	101	V	101	V			FALSE												
14				105	C	105	V	105	V			FALSE												
15				112	C	112	V	112	V			FALSE												
16				121	V	121	C	121	V			FALSE												
17				123	V	123	C	123	C			FALSE												
18				132	C	132	C	132	C			TRUE												
19				135	C	135	C	135	C			TRUE												
20				137	C	137	C	137	C			TRUE												
21				151	C	151	C	151	C			TRUE												
22				163	V	163	C	163	V			FALSE												
23				167	V	167	C	167	C			FALSE												
24				169	V	169	C	169	C			FALSE												
25				171	V	171	C	171	C			FALSE												
26				173	C	173	C	173	C			TRUE												
27				178	V	178	C	178	V			FALSE												
28				185	V	185	V	185	N			FALSE												
29				186	V	186	V	186	N			FALSE												
30				194	V	194	V	194	N			FALSE												
31				202	V	202	V	202	V			TRUE												
32				206	V	206	V	206	V			TRUE												
33				229	V	229	V	229	N			FALSE												
				262	N	262	N	262	N			FALSE												

Out of 2175 unclassified samples, 862 matched in all classifiers experimented. That's 39.63% of samples.