

CT475 Assignment One

Michael Murphy

15300856

4BS2 – 4th Year Science (Maths & Computer Science)

1. Selected machine learning package

The open-source machine learning package that I selected is [*scikit-learn*](#), a machine learning package for Python which is built on NumPy, SciPy, and matplotlib. It's a simple python module that can be easily imported for use in scripts/at the interpreter. It has several techniques for supervised learning, unsupervised learning, and model selection & evaluation.

I felt that *scikit-learn* would be suitable due to its simplicity; the Python code required to build a classification model for the provided data was less than 30 lines, and still provided valuable information. It also required minimal pre-processing of the data to be used for training of the model. The techniques that were valuable to me were the [Nearest Neighbors](#) and [Support Vector Machines](#) classification techniques, both of which were very easy to understand, in no small part due to the extent of [documentation](#) available for *scikit-learn*. This package accepts data stored as numpy arrays, which are very powerful, and importing data into numpy arrays is also very simple, making it a good choice for storing external data sets. The fact that *scikit-learn* works seamlessly with these arrays simplifies the learning process even further.

2. *Data preparation*

For both classification techniques, *scikit-learn* can recognise numeric data stored as numpy arrays or scipy sparse matrices. A number of tools are recommended for converting data into a format that scikit-learn can use, but given the simplicity of the data set, I opted to use NumPy's [*genfromtxt*](#) method, which was easily able to generate a numpy array from the training data.

The only other step required to prepare the dataset for reading by the *genfromtxt* method was to switch the columns and rows in the text file, as *genfromtxt* expects individual cases to be stored in rows, rather than columns. I achieved this by copying the contents of the *autoimmune.txt* file into an Excel workbook, which conveniently detects the tab delimiter used in the .txt file. At this point, I moved the row describing the Autoimmune_Disease feature to the bottom row of the dataset, to simplify the separation of data and target feature. I then **transposed this data (see steps below)** and saved it into a new tab delimited .txt file, called *autoimmune_transpose.txt*

Steps for transposing:

- *Starting point*

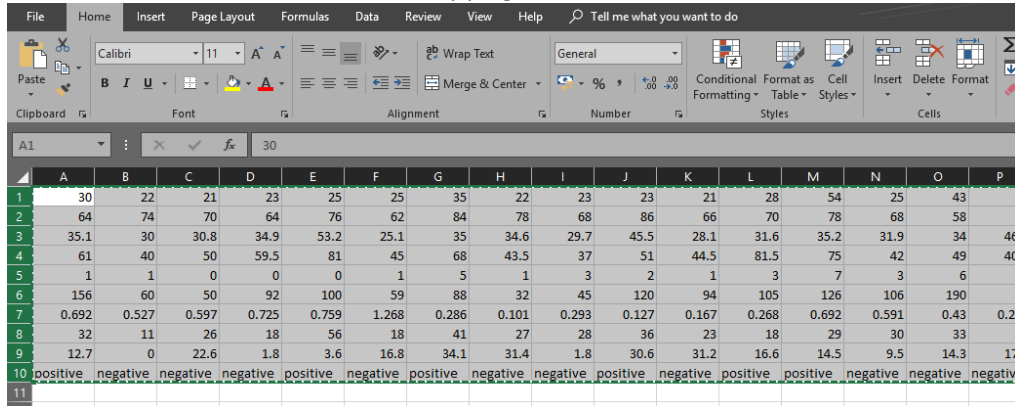
[illegible]

CT475 Assignment One

Name: Michael Murphy Student Number: 15300856 Class: 4BS2 – 4th Year Science (Maths & Computer Science)

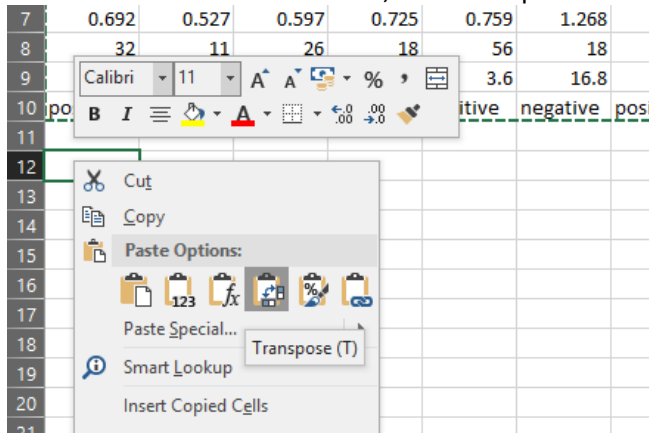
2. *Data preparation (continued)*

- Select the contents of the data for copying



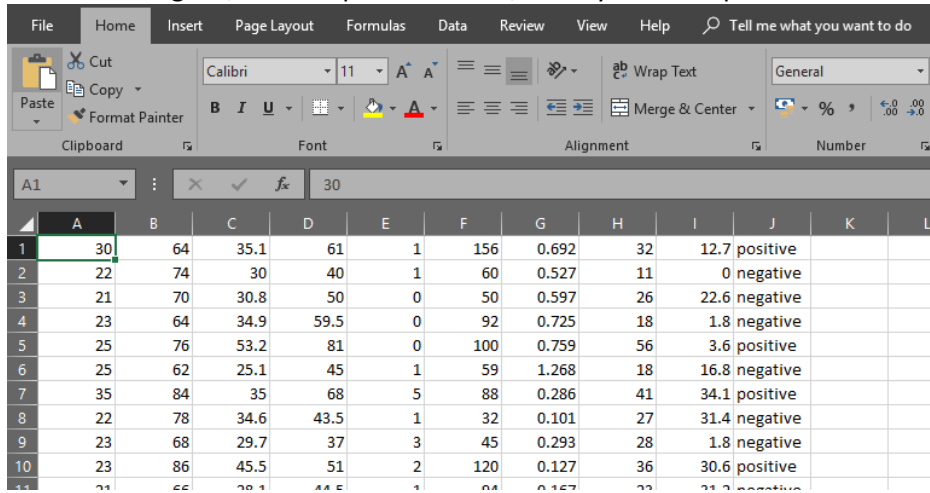
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	30	22	21	23	25	25	35	22	23	23	21	28	54	25	43	4
2	64	74	70	64	76	62	84	78	68	86	66	70	78	68	58	7
3	35.1	30	30.8	34.9	53.2	25.1	35	34.6	29.7	45.5	28.1	31.6	35.2	31.9	34	46
4	61	40	50	59.5	81	45	68	43.5	37	51	44.5	81.5	75	42	49	40
5	1	1	0	0	0	1	5	1	3	2	1	3	7	3	6	
6	156	60	50	92	100	59	88	32	45	120	94	105	126	106	190	4
7	0.692	0.527	0.597	0.725	0.759	1.268	0.286	0.101	0.293	0.127	0.167	0.268	0.692	0.591	0.43	0.26
8	32	11	26	18	56	18	41	27	28	36	23	18	29	30	33	4
9	12.7	0	22.6	1.8	3.6	16.8	34.1	31.4	1.8	30.6	31.2	16.6	14.5	9.5	14.3	17
10	positive	negative	negative	negative	positive	negative	positive	negative	negative	positive	negative	positive	positive	negative	negative	negative

- Select some cell below the data, choose to paste 'Transpose'



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
7	0.692	0.527	0.597	0.725	0.759	1.268										
8	32	11	26	18	56	18										
9	12.7	0	22.6	1.8	3.6	16.8										
10	positive	negative	negative	negative	positive	negative	positive	negative	negative	positive	negative	positive	positive	negative	negative	negative

- Delete the original, un-transposed dataset, so only the transposed dataset remains



	A	B	C	D	E	F	G	H	I	J	K	L
1	30	64	35.1	61	1	156	0.692	32	12.7	positive		
2	22	74	30	40	1	60	0.527	11	0	negative		
3	21	70	30.8	50	0	50	0.597	26	22.6	negative		
4	23	64	34.9	59.5	0	92	0.725	18	1.8	negative		
5	25	76	53.2	81	0	100	0.759	56	3.6	positive		
6	25	62	25.1	45	1	59	1.268	18	16.8	negative		
7	35	84	35	68	5	88	0.286	41	34.1	positive		
8	22	78	34.6	43.5	1	32	0.101	27	31.4	negative		
9	23	68	29.7	37	3	45	0.293	28	1.8	negative		
10	23	86	45.5	51	2	120	0.127	36	30.6	positive		

CT475 Assignment One

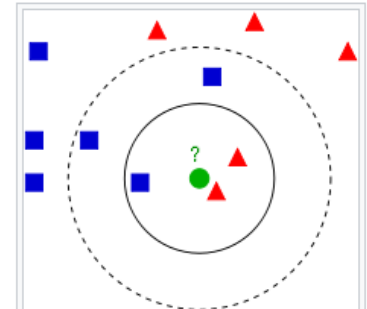
Name: Michael Murphy Student Number: 15300856 Class: 4BS2 – 4th Year Science (Maths & Computer Science)

3. Two classification algorithms applied

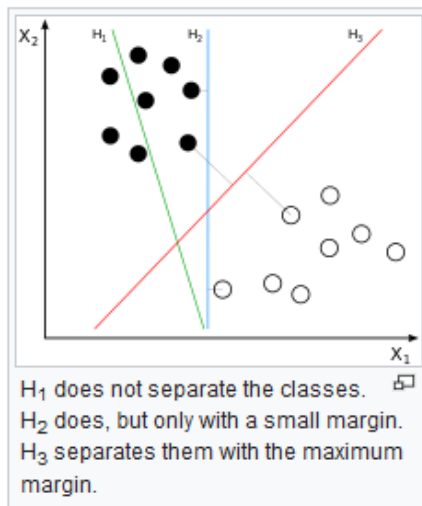
The two classification algorithms that I chose for training classification models were *k*-Nearest Neighbour (*k*NN), which we covered in lectures, and Support Vector Machines (*SVM*), which was mentioned in lectures, but not covered. As mentioned in Section 1, *scikit-learn* has classifier classes for both of these algorithms; *kNeighborsClassifier* and *SVC*, respectively. Both algorithms construct models which are represented by points in space, and each model has its own way of classifying new samples.

The k-Nearest Neighbours algorithm

As described in lectures, the *k*NN algorithm operates by considering each sample to be a point in sample space. When a new sample is being queried, the algorithm finds the samples from the training data that are closest to it; specifically, it finds the *k* closest samples, or the *k* nearest neighbours. These neighbours then vote on the value of the new sample's target feature (see diagram for example); in our case, they vote on whether the new sample should test *positive* or *negative* for Autoimmune Disease. This vote is based on a simple majority of the neighbours' values of *Autoimmune_Disease*.



Example of *k*-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If $k = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If $k = 5$ (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).



H_1 does not separate the classes.
 H_2 does, but only with a small margin.
 H_3 separates them with the maximum margin.

The Support Vector Machines algorithm

Like *k*NN, the *SVM* algorithm creates a model with each sample being a point in space. It then tries to create a *margin* in the space, which separates the samples that test *positive* for Autoimmune Disease from the samples that test *negative*. If the samples were 2-dimensional, for example, then *SVM* would essentially try to draw a line between the two groups of samples (see diagram). In our case, our samples have 9 attributes, so *SVM* aims to create an 8-dimensional hyperplane to separate the groups, called a *margin*. Then, when new samples are queried, *SVM* simply checks whether this query falls on the *positive* or *negative* side of this margin, and classes it accordingly. *SVM* strives to choose a margin with the largest possible gap between itself and any of the training data, to increase accuracy of future classifications.

4. Estimation of performance by 10-fold cross validation

sci-kit learn offers very straightforward functionality for performing cross-validation in the form of cross_val_score. We simply specify the classifier object (either a *kNeighborsClassifier* or *SVC* object in our case), and the training data and features to which that classifier is to be fitted. We may also specify the number of folds during the validation process, which should be 10 in our case. At each step, we get a validation score which scores the performance of the model. If we take the average of these scores, we get a sort-of overall performance score. The beauty of this is that scores can be compared, not only between models constructed by different algorithms, but also ones constructed by the same algorithm with different parameters.

CT475 Assignment One

Name: Michael Murphy Student Number: 15300856 Class: 4BS2 – 4th Year Science (Maths & Computer Science)

4. Estimation of performance by 10-fold cross validation (continued)

For example, in the case of the **kNN model**, I was able to compare average scores across different values of k ; specifically, I tested $k = 1, 2, \dots, 20$. The scores obtained for each value were:

0.6861, 0.6943, 0.6917, 0.7262, 0.7288, 0.7421, 0.7501, 0.7636, 0.7529, 0.7421,
0.7554, 0.7605, 0.7605, 0.7364, 0.7550, 0.7468, 0.7550, 0.7522, 0.7548

From this, I could easily see that $k=8$ generated the model with the highest cross-validation score. This would support the assertion that $k=8$ would be the optimal choice when using the kNN algorithm to classify this dataset.

For the **SVM model**, although there were several optional parameters when declaring a classifier instance, trial and error showed that almost none of these affected the cross-validation score when changed. The one exception was the value of *gamma*, which could be set to either 'scale' or 'auto'. The scores obtained for each choice were 0.7424, 0.6863, respectively, leading to the obvious conclusion that 'scale' was the better choice.

5. Comparison of performance results between models

If we compare the cross-validation scores obtained by the kNN and SVM models in their optimal setups (i.e., kNN using $k=8$, SVM using *gamma*='scale'), we see that both algorithms give very similar results, with kNN outperforming SVM by only a small margin (scores were 0.7636 and 0.7424, respectively).

It is suggested that, in general, kNN is a better choice of algorithm if our dataset has a lot of points in a low dimensional space. Conversely, SVM may be the better choice for datasets with only a few points in a high dimensional space. Since the *autoimmune* dataset falls somewhere between those two categories, it stands to reason that the two algorithms should perform similarly.

Bibliography

- [1] - [Scikit-learn: Machine Learning in Python](#), Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
- [2] - [Nearest Neighbors](#), scikit-learn documentation (v0.20.0)
- [3] - [Support Vector Machines](#), scikit-learn documentation (v0.20.0)
- [4] - [documentation](#), scikit-learn documentation (v0.20.0)
- [5] - [genfromtxt](#), NumPy documentation (v1.15)
- [6] - [kNeighborsClassifier](#), scikit-learn documentation (v0.20.0)
- [7] - [SVC](#), scikit-learn documentation (v0.20.0)
- [8] - [k-nearest neighbors algorithm](#), Wikipedia
- [9] - [Support vector machine](#), Wikipedia
- [10] - [cross_val_score](#), scikit-learn documentation (v0.20.0)
- [11] - [What is better, k-nearest neighbors algorithm or Support Vector Machine classifier?](#), quora.org

CT475 Assignment One

<i>Name:</i>	<i>Student Number:</i>	<i>Class:</i>
Michael Murphy	15300856	4BS2 – 4 th Year Science (Maths & Computer Science)

Code

```
import numpy as np
import os
import matplotlib.pyplot as plt
from sklearn import neighbors, datasets, svm, ensemble
from sklearn.model_selection import cross_val_score

#change into location of dataset, specify file name
os.chdir('D:\\OneDrive - National University of Ireland, Galway\\NUIG\\2018-2019\\Semester 1\\CT475\\Assignment 1')
fname = 'autoimmune_transpose.txt'

#use np.genfromtxt to read in training data, and target feature data
autoimmune_data = np.genfromtxt(fname, delimiter='\t', encoding=None,
usecols=np.arange(0,9))
autoimmune_target = np.genfromtxt(fname, delimiter='\t', dtype=None,
encoding=None, usecols=9)

#create models with kNN, checking the the 10-fold cross validation scores for
various values of k
k_scores=[]
for i in range(1,20):
    kNN = neighbors.KNeighborsClassifier(i)
    kNN_scores = cross_val_score(kNN, autoimmune_data, autoimmune_target,
cv=10)
    k_scores.append(kNN_scores.mean())
k_max = max(k_scores)
print("The highest 10-fold cross validation scores were found for k = ",
k_scores.index(k_max)+1, ", which had a mean score of %.4f." % k_max, sep='')

#create models with SVM, checking the the 10-fold cross validation scores
when gamma is set to 'scale' or 'auto'
s_scores=[]
s_options=['scale','auto']
for i in s_options:
    svc = svm.SVC(gamma=i)
    svc_scores = cross_val_score(svc, autoimmune_data, autoimmune_target,
cv=10)
    s_scores.append(svc_scores.mean())
s_max = max(s_scores)
print("The highest 10-fold cross validation scores were found when gamma was
set to '", s_options[s_scores.index(s_max)], "', which had a mean score of
%.4f." % s_max, sep='')

if k_max >= s_max:
    print("A higher mean cross validation score was obtained by the kNN
algorithm.")
else:
    print("A higher mean cross validation score was obtained by the SVM
algorithm.")
```