Imperial College London
BUSINESS SCHOOL

MSc Business Analytics 2020/21
Machine Learning

Assignment 1

1. Individual Assignment: Predicting Plant Types with *k*-Nearest Neighbours

Instructions: *This exercise should be done "by hand", that is, not using Python. All necessary calculations should be included in the submission, as well as brief explanations of what you do.*

The data below is a small subset of the famous Iris database, first used by Sir R. A. Fisher (http://archive.ics.uci.edu/ml/datasets/Iris). This is perhaps the best known database in the pattern recognition literature. Fisher's paper (R. A. Fisher (1936), "The use of multiple measurements in *taxonomic problems", Annals of Eugenics 7(2):179–188) is a classic in the field and is referenced* frequently to this day. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. There are four features: sepal length, sepal width, petal length and petal width. For the exercise below we only use petal length and petal width.

petal length	petal width	class
5.6	1.8	virginica
1.5	0.2	setosa
5.0	2.0	virginica
1.3	0.3	setosa
1.6	0.2	setosa
4.6	1.3	versicolor
6.0	2.5	versicolor
5.6	2.4	versicolor
4.5	1.6	versicolor
1.3	0.2	???
1.4	0.2	???
4.7	1.4	???
5.9	2.3	???
1.3	0.3	???
6.1	2.3	???

(a) Normalise the data using the Z-score normalisation.

Note: *For the purpose of this exercise, please use* <u>*all*</u> *data to compute the μ and σ required for the normalisation. Keep in mind, however, that in reality you have to use* *only the training data to compute μ and σ — the input for the future predictions will typically not yet be available to you!*

(b) Use the *k*-Nearest Neighbours method with $k = 3$ to predict the missing classes. Use the Euclidean norm in your distance calculations.

Note: *When making predictions, you need to transform the input data for the predictions using the same μ and σ as for the transformation of the training data in (a).*

2. Group Assignment: Predicting Wine Quality with *k*-Nearest Neighbours

Instructions: *This exercise should be done using Python. The source codes as well as all relevant outputs should be included in the submission, as well as brief explanations of the code as well as what you see.*

Use the first 800 instances in the file "winequality-white.csv".

Build a k-Nearest Neighbours classifier in Python for "winequality-white.csv" that:

1. loads the data file;
2. construct a new binary column "good wine" that indicates whether the wine is good (which we define as having a quality of 7 or higher) or not;
3. splits the data set into a training data set (first 400 samples), a validation data set (next 200 samples) and a test data set (last 200 samples) — please do *not* shuffle the data, as it is already shuffled;
4. normalises the data according to the Z-score transform;
5. loads and trains the *k*-Nearest Neighbours classifiers for $k = 1,2, \ldots,100$;
6. evaluates each classifier using the validation data set and selects the best classifier;
7. predicts the generalisation error using the test data set.
8. Try a new splitting: split the data set into a training data set (first 200 samples), a validation data set (next 200 samples), and a test data set (last 400 samples) - again, please do not shuffle the data. Then redo steps 4 to 7. What is the new generalisation error? Explain what you find.

How do you judge whether the classifier is well-suited for the data set?

Hints for Using Python

For your Python implementation, you may want to rely on the Pandas, NumPy and SciKit-Learn libraries. Part of the group assessment challenge is to explore the libraries on your own!

1. Pandas is useful to import, explore and manipulate data. A cheat sheet from the company *enthought* is attached to this assignment, to get you started.
2. SciKit-Learn is the machine learning library. The SciKit-Learn documentation is excellent and available at http://scikit-learn.org/stable/documentation.html.

Here is a non-exclusive list of commands for Pandas that you might find useful:

```
import pandas as pd
df=pd.read_csv('filename',sep='\t',names=["col1name", "col2name"])
df['newcolumn']=df.col1name.apply(lambda x: x+2) # apply the function x -> x+2
df['newcolumn']=df.col1name.apply(f) # apply the function f defined elsewhere
df.describe()
df.ix[5]
df.head(3)
df[2:4]
```

For NumPy and SciKit-Learn you may want to look at the following commands:

```
from sklearn.utils import shuffle
X,y=shuffle(X,y)
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```