Data Mining and Decision Systems Workshop 3
Advanced Pandas Indexing & Feature Transformation

Aims of the workshop

The aim of this session is to look at more advanced indexing methods for Pandas, enabling complex selection, as well as how to transform features using Pandas with concepts covered within lectures.

Please see ‘Useful Information’ below on how to lookup certain Python functionality. The concept behind this workshop is about discovery, and experimentation surrounding topics covered so far.

Feel free to discuss the work with peers, or with any member of the teaching staff.

Useful Information

Throughout this workshop you may find the following useful.

Python Documentation

https://docs.python.org/3.7/

This allows you to lookup core language features of Python 3.7 as well as tangential information about the Python Language.

Pandas Documentation

https://pandas.pydata.org/pandas-docs/stable/index.html

DataFrames:
https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

Series:
https://pandas.pydata.org/pandas-docs/stable/reference/series.html

Similarly to the above, the pandas library itself is well-documented outlining function signatures, alongside example use-cases.

Functions of note: read_csv, [ ], [[ ]], copy, unique, value_counts, isnull, sum, count, mode, fillna, index, drop, get_dummies, concat.

Useful indexing introduction:
https://www.geeksforgeeks.org/indexing-and-selecting-data-with-pandas/

Reminder

We encourage you to discuss the contents of the workshop with the delivery team, and any findings you gather from the session.

Workshops are not isolated, if you have questions from previous weeks, or lecture content, please come and talk to us.

The contents of this workshop are not intended to be 100% complete within the 2 hours; as such it's expected that some of this work be completed outside of the session. Exercises herein represent an example of what to do; feel free to expand upon this.

Running A Jupyter/IPython Notebook

A jupyter notebook is a server which runs on the local machine. It will use whichever directory it is invoked within as the root folder. It is then able to see all sub-folders within that. This is currently installed on the lab machines; however, if you are wishing to install this at home, you may need to follow pip instructions for installing jupyter.

Jupyter Notebook can be launched from the command line by invoking:

```
jupyter notebook
```

This will start a server, typically on http://localhost:8888/tree, and should automatically open a new tab in the browser.

Jupyter notebook server uses a token, generated at launch, to authenticate access. By default, the notebook server is accessible to anybody with network access to your machine over the port, with the correct token.

If you are asked for a password or token, you can always invoke:

```
jupyter notebook list
```

This will provide you a list of currently active Notebook servers (you can run multiple, on different ports, for different folders), along with their token, all within a single URL.

NOTE: You may wish to `cd` into an appropriate starting folder, such as a USB Stick, or a folder where you can store all of your Data Mining Workshop documents.

For each of the exercises below, please use a separate code cell. This allows you to compute just the given exercise and execute it, checking the output. Subsequent cells all share the same variable space - variables you assign in cell 1 will be available afterwards in other cells.

As with all workshops, after you read the original data in, create a copy to work with. my_copy = read_data.copy()

Exercise 1: Read the Titanic Dataset into a DataFrame. Get the 'Embarked' attribute values only, as a Pandas Series. Caution: Do not confuse [ ] with [[ ]]. One returns a Series, the other a DataFrame. Use type(x) to see types of variables.
Using the unique method, print the unique values of this feature.

Exercise 2: Count the number of values of each unique entry in 'Embarked'. You may find value_counts useful.

Exercise 3: As you may have noticed in Exercise 1, alongside our expected values we also have instances of nan. Find how many NaN values exist in our Embarked feature. You can use isnull to produce a series of binary True/False results. What statistical functions do you know which can produce a final number.

Additionally, given the number of NaN values, calculate the percentage of NaNs within the feature vector.

Exercise 4: Pandas enables you to have more complex selection queries. You can know to grab a column with *df['column_name']*. However, we can make boolean expressions on that column to express something we want.
E.g *df['column_name'] > 10*.

If we embed this expression as our indexing criteria, we can grab all records that satisfy the condition.
E.g df[df['column_name'] > 10].

Using this principle of boolean expressions, obtain all PassengerId > 880, and print them.

Exercise 5: Thinking back to the boolean series generated in Ex 3, use this to print all records which satisfy the condition that their ‘Embarked’ is NaN/Null.

Exercise 6: Look at the records generated from Exercise 5. Are there many of them (See ex3)? Does the record have any other NaN values for other features.
Implement both of the below mechanisms for dealing with this, which one is more appropriate, and why?

a) Repair this data record (How can we impute categorical data?)
b) Drop the record

This can be done using fillna or manually using the following to obtain the table id of the record to drop (Think workshop 2, axis 0):

```
Id_to_drop = data_copy.index[ data_copy['PassengerId'] == 123 ]
```

Exercise 7: For our repaired Embarked column, get the dummies for this categorical data, as a data frame. Print this.

Exercise 8: Using the data copy table you have been working with, concatenate the new dummy variables to it, forming a new working data frame. How can we deal with the fact that we now have the Embarked Column alongside our dummied new features?

The Extended Exercises are optional, and are offered as an advanced supplement for those who have completed the existing work and wish to expand on their knowledge and challenge themselves further.

Extended Exercise 1: Consider how you might obtain ID values from Ex 4 between some *minimum* and *maximum* value. Grab all records where PassengerID > 40 and PassengerID <= 60.

Extended Exercise 2: In Ex 8 a spurious column is introduced as a function of the concat operation. Is there a way to select all columns except 'Embarked'. Perform this, and concat that table with the dummied variables.