Background

The dataset is the food nutrient database from Food Standards Australia and New Zealand[1] in the file food nutrient 2011 13 AHS.csv. It contains a detailed breakdown of nutrient values for a wide range of food items. Each nutrient value is presented on a per 100 gram edible portion basis. It was used to support the 2011-13 Australian Health Survey. Table 1 below shows a high level summary description of each feature.

There is one data file provided:

• food nutrient 2011 13 AHS.csv: Nutrient information for about 5.7k foods.

Key libraries to use are Pandas, Matplotlib, NumPy, SciPy, seaborn and sklearn. You will need to write Python 3 code (Jupyter notebook).

1 http://www.foodstandards.gov.au/science/monitoringnutrients/ausnut/ausnutdatafiles/Pages/foodnutrient.aspx 1

Field Name	Description
Food ID	8 digit alpha numeric food identification code, based on FSANZ standard
Survey ID	8 digit food identification code that was specific to Australian Health Survey
Survey flag	can be ignored
Food Name	Name of the food
nutrient name (units)	Describes the full nutrient name of each of the nutrients e.g. ’Protein’ and includes the units the nutrient is presented in e.g. ’(g)’ for grams. A value is then provided for each food and nutrient.

Table 1: Summary of Features for food nutrient 2011 13 AHS.csv

Importing required Python Libraries and loading the data

Please begin your Jupyter notebook by listing all the Python libraries you will be using. Also load the dataset (.csv) in a dataframe object.

```
#import ....
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from scipy.spatial.distance import pdist, squareform
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
food =  pd.read_csv("food_nutrient_2011_13_AHS.csv", header=0,low_memory=False)
```

There are also some helpful functions like VAT (visualization of clustering tendency) and MI (mutual information) that you can use and integrate into your code. These are provided in the example Jupyter notebook file.

1. Feature standardisation

In order for a number of types of analysis techniques to be effective, it is important for features to be scaled or standardised.

a) Create a new data frame that contains only the continuous features from food nutrient 2011 13 AHS.csv. I.e. the features starting from Energy, with dietary fibre (kJ) and ranging to Total trans fatty
acids (mg). (53 features in total). This data frame is the input for part b).

b) Apply the sklearn.preprocessing.StandardScaler() function to standardise each feature separately, to have 0 mean and unit variance. Store the transformed features in ***foodscaled***.

c) Print the number of rows and columns, as well as the minimum, maximum, mean and standard deviation (computed taking into account all values) of foodscaled, using the follow- ing format:

```
***
Q1.c: foodscaled matrix details
Number of rows: #
Number of columns: #
Min: #
Max: #
Mean: #
Standard Deviation: #
***
```

where each # is the appropriate value rounded to 1 decimal place. Hint: Useful functions include Standard Scaler

2. Principal components analysis

It is difficult to visualise the data in *foodscaled*, due to the large number of features. One strategy to deal with this is to apply principal components analysis.

a) Create a feature *EnergyLevel* which has value ”1” if (unstandardized) Energy, with dietary fibre (kJ) is greater than 1000 kJ and ”0” otherwise.

b) Apply principal components analysis to your data in *foodscaled* to compute the first two principal components. Store the result in *foodreduced*.

c) Produce a scatter plot of the data in *foodreduced*. A food should have red color if it has ‘High’ energy (EnergyLevel of ”1”) and blue if it has ‘Low’ energy (EnergyLevel of ”0”). There should be an accompanying legend mapping colours to EnergyLevel.

d) Comment on your scatter plot in c). What does it show? How can it be interpreted? What are the advantages and disadvantages of using PCA for visualising this food dataset?

Hint: Useful functions are sklearn PCA

3. Clustering visualisation

a) Now, create a new feature Food category, which contains the first two digits of the feature Survey ID.

b) From foodscaled, select only foods from categories 13, 20 and 24. Store these foods in *foodscaledsample*.

c) Use the VAT() function, discussed in Workshop Week 6, to compute the ordered dissimilarity matrix for the foodscaledsample matrix.

d) Plot the heatmap for the ordered dissimilarity matrix from c). For your plot, select a colormap that is effective in revealing the cluster structure.

e) How many clusters are apparent in your heatmap? Is this expected (why/why not)? Why does use of different colormaps produce visualisations of varying usefulness? Explain what properties an optimal colormap should have for this task.

Hint: Useful items include Seaborn heatmap, Colormap reference.

4. K-means and sum of squared errors

Recall question 5 from the Week 6 workshop. This question will take you through a practical example of computing sum of squared errors (SSE), which is a quality measure that can be used for evaluating a clustering produced by k-means.

a) Using the dataset *foodscaled*, produce a plot of the SSE value of the k-means clustering of the dataset (y-axis), versus k value (x axis). k should range across 2, 3, 4, . . . , 25. This requires generation of 24 different k-means clusterings. When generating each clustering, use random state=100 and default values for all other parameters of the KMeans function, except n clusters, which will need to be varied.

b) Comment on the shape and trend of your plot from a). Where is the elbow? Is this expected (why/why not)?

Hint: Useful functions are the VAT code provided in the example Jupyter notebook and sklearn.cluster.KMeans and scipy.spatial.distance.cdist.

5. Correlation and Mutual Information

a) Visualise correlation matrix: Plot a Pearson correlation matrix for the first 10 nutri- ents (shown below) in the food data frame. The calculated 10*10 symmetric matrix should contain the correlation between every pair of attributes. For example, a value in row i and column j should contain the Pearson correlation r_{ij} between two features i and j. Plot a heatmap for your computed correlation matrix, including feature names on each axis of the heatmap.

'Energy, with dietary fibre (kJ)',
'Energy, without dietary fibre (kJ)',
'Moisture (g)',
'Protein (g)',
'Total fat (g)',
'Available carbohydrates, with sugar alcohols (g)',
'Available carbohydrates, without sugar alcohol (g)',
'Starch (g)',
'Total sugars (g)',
'Added sugars (g)'

b) Analysing the effect of binning

- - For the pair of features in the food dataframe: 'Protein (g)' and 'Energy, with dietary fibre (kJ)': Calculate a series of Mutual Information (MI) values, according to increasing numbers of equal-width bins. Use [2, 10, 20, 30, 40, ... , 200] numbers of bins and calculate 21 MI values, one for each number of bins.
- - Create a plot, where the x axis is number of bins and the y axis is the MI of the feature pair.
- - Explain the trend in your plot.

  c) Comparison of Pearson correlation and Mutual Information:
  - - Considering the full set of 53 continuous features from the food data frame, find the top-10 feature-pairs by Pearson correlation strength (highest to lowest)
  - - Considering the full set of 53 continuous features from the food data frame, find the top-10 feature-pairs by Mutual Information, using 20 equal-width bins
  - - Print out the top-10 feature-pairs for Pearson correlation (highest to lowest) and by MI (highest to lowest). Comment on the similarities and differences between the two lists.

    Hint: Useful functions are pandas.DataFrame.corr and the provided mutual information func- tion in the example Jupyter notebook. Note that executing the code to answer this question may be more computationally demanding, and so you may wish to use a smaller dataset during initial development and debugging.

6. Prediction models: decision trees

This question explores predicting the Food category label, using the values of the various nu- trients. I.e. Food category is the class label.

a) Randomly split the foodscaled data into 80% training and 20% testing. You should output two matrices: X train and X test, and two vectors/columns: y train and y test.

- X train contains features of training instances (i.e. 80%) and y train contains labels for the training instances.
- X test contains features of test instances (i.e. 20%) and y test contains labels for test instances.

The output of this question should print the shape (number of rows and columns) of X train, X test, y train and y test in the following format:

****
Q6.a: Train Test Split Results
X_train matrix: #
y_train labels: #
X_test matrix: #
y_test labels: #
****

where # are the calculated shape values.

b) Using the same training and testing data, you will next analyse the effect of varying the max depth parameter for the decision tree. Produce a plot showing accuracy (y-axis) versus max depth (x-axis), where max depth varies in the range 1,2,3,...,40. Each data point in the plot corresponds to the accuracy of a decision tree that is trained using the 80% training data and evaluated using the 20% test data, to predict the Food category.

c) Comment on the shape of the plot in Part c). Suggest reasons for the shape of the plot and any local peaks.

Hint: Useful functions are sklearn.model selection.train test split and sklearn.tree.DecisionTreeClassifier

7 Prediction models: K-NN (2 marks)

a) Create an instance of the KNeighborsClassifier and set k (number of neighbors) to 1. Train/fit the model using X train and y train. Evaluate the model by calculating its accuracy when applied to both the train and the test set (i.e. X train and X test).

Then print out the following:

****
Q7a: Food category prediction using k-NN (k=1)
Train accuracy: # %
Test accuracy: # %
****

Where # is the calculated k-NN classifier accuracy rounded to 1 decimal place. b) Repeat the steps of a), this time using k = 3. The output will be as follows

****
Q7.b: Food category prediction using k-NN (k=3)
Train accuracy: # %
Test accuracy: # %
****

c) Comment on the differences and similarities between your output for parts a) and b). Where they are different, suggest reasons why.

d) In parts b) and c), the reported accuracy may be over-optimistic, due to the way standardisation was applied in Question 1. Explain why this is the case and how this issue could be addressed.

Hint: Useful functions are sklearn.neighbors.KNeighborsClassifier 8 Feature generation (4 marks - harder)

In order to achieve higher prediction accuracy for k-NN, one can investigate the use of feature generation and selection. Again, Food category is the class label to be predicted.

Feature generation involves the creation of additional features, additional to the ones already present in foodscaled. Two methods are

- Interaction term pairs. Given a pair of features f_1 and f_2 in foodscaled, create a new feature $f_{pair} = f_1 \times f_2$ or by $f_{pair} = {}^{f_1}$. All possible pairs can be considered.

  f_2

- Clustering labels: Apply k-means clustering to the data in foodscaled and then use the resulting cluster labels as the values for a new feature $f_{clusterlabel}$. At test time, a label for a testing instance can be created by assigning it to its nearest cluster.

  Given a set of N features (the original features plus generated features), feature selection involves selecting a smaller set of n features (n < N). Here one computes the mutual in- formation between each feature and the class label (on the training data), sorts the features from highest to lowest mutual information value, and then retains the top n features from this ranking, to use for classification with k-NN.

  Your task in this question is to evaluate whether the above methods for feature generation and selection, can deliver a boost in prediction accuracy compared to using k-NN just on the original features in foodscaled. You should:

  - Implement the above two methods for feature generation. You may need to experiment with different parameter values, including k and different numbers of generated features.
  - Implement feature selection using mutual information. Again you may need to experi- ment with different parameter values, such as how many features to select.

    Your output for this question should include i) your implemented code that generates accuracies or plots, ii) a discussion (2-3 paragraphs) of whether feature selection+generation can deliver an accuracy boost, based on your evidence from i).

    Note that you should use a training and testing data split, to evaluate accuracies. Feature generation for training data should not require any knowledge of the testing data. The fea- ture selection step (on the training data) should not require any knowledge of the testing data.

    Hint: This is a more open ended question. It must be clear from your answer that you have successfully implemented the proposed strategy, experimented with different parameter settings, and have a clear, evidenced-based explanation of the extent to which it succeeded.