COMM415DA Fundamentals Of Data Science

Ref/Def Assessment

This assessment comprises 100% of the overall module assessment. This is an individual exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

Question 1

[25 marks]

Aim: demonstrate competence in the use of the `pandas` library.

a) Analyze the Exasens Data Set from https://archive.ics.uci.edu/ml/datasets/Exasens.

b) Load the dataset in a `pandas` DataFrame directly from the URL of the dataset.

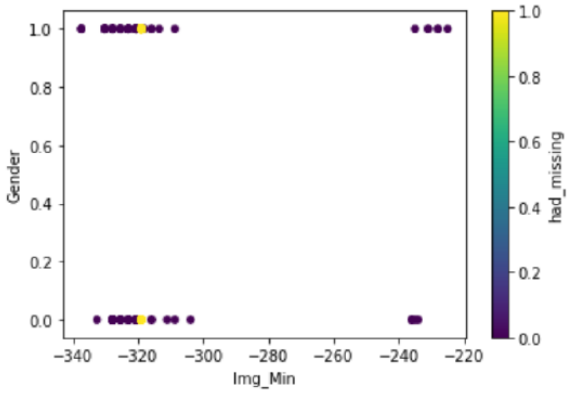
c) Ignore the following columns: `Diagnosis` and `ID`.

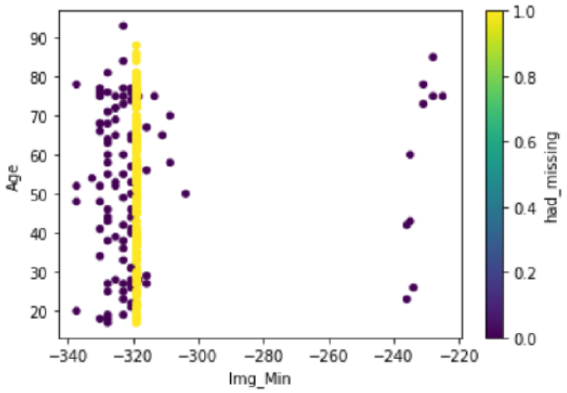
d) Replace the missing values in each column with the average between the median and the mean column value.

Example: if a column contains the values `1,2,NaN,1,1` then compute the mean (1.25) and the median (1) and substitute NaN with the average between 1.25 and 1 (1.125). By doing so the column will be `1,2,1.125,1,1`.

e) Display a scatter plot for each distinct pair of columns (e.g. for `'Immaginary_Part_Min'` and `'Real_Part_Max'`, for `'Real_Part_Min'` and `'Age'`, etc). Distinguish by color between instances that originally had at least one missing value and those that did not have any missing value.

You should obtain something like this:





Question 2

[35 marks]

Aim: demonstrate an understanding of PCA and matrix factorization using SVD.

In this exercise you will need to build a low rank approximation for a given dataset using SVD and highlight the instances that cannot be well reconstructed.

Acquire the Iris dataset using the following procedure:

```
from sklearn.datasets import load_iris
X,y = load_iris(return_X_y=True)
```

The data matrix contains 150 vectors (also called *instances*) with 4 attributes each (i.e. it is a 150 x 4 matrix) and the vector y contains the class encoded as the integers 0,1, and 2.

a) Collect all instances belonging to the same class in distinct data matrices.

b) Write your own code to perform PCA (i.e. do not use functions provided by the scikit library, such as `sklearn.decomposition.PCA` or `sklearn.decomposition.TruncatedSVD`).

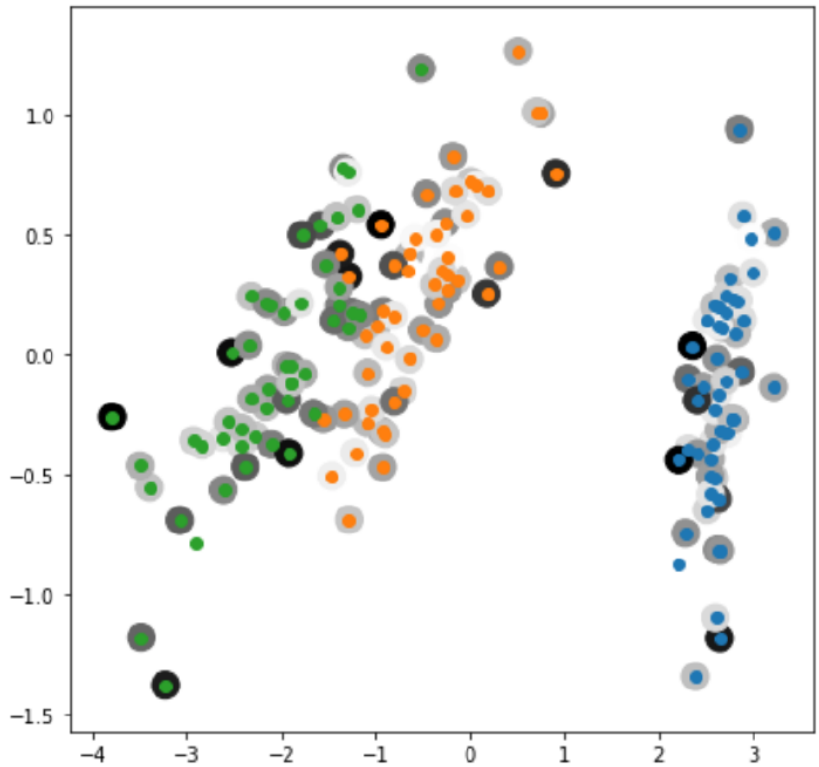
c) Display a scatter plot in 2D of the Iris dataset using your `pca` function and distinguishing instances belonging to different classes by color.

d) Write your own code to perform a low rank reconstruction of a data matrix. The function should take in input the data matrix and an integer to specify the desired rank.

e) For each instance compute the *reconstruction error* as the length of the difference vector between an original instance and the reconstructed instance. When performing the low rank approximation consider the individual class specific data matrices.

f) Display a scatter plot in 2D of the Iris dataset using your `pca` function and encoding the magnitude of the reconstruction error when `rank=3` using the gray scale colormap.

You should obtain something like this:



Question 3

[40 marks]

Aim: demonstrate an understanding of LDA.

In this exercise you will need to build a dataset using the `multivariate_normal` function provided by the numpy library. You will then fit a multiclass LDA model to your data and finally you will display the decision boundaries of the trained predictive model.

Construction of the dataset:

Given a parameter `k`, the dataset is generated using `k` multi variate normal data generators. All instances are 2 dimensional.

The multi variate normal data generators are defined by 3 parameters: a mean, a covariance matrix and the number of samples to generate.

a) The means of the data generator will lie on the vertices of a regular polygon (if `k=3` the polygon is a triangle, if `k=6` it is an hexagon, etc). Write your own code to determine the position of the vertices of a regular polygon given a radius value in input. *Hint: you can use your knowledge on linear transformations (e.g. rotations).*

b) The covariance matrices will be constructed by specifying 2 parameters: a `ratio` between the two main directions of variability (if the ratio is, say, 2:1, then the parameter is 2); and a `rotation` in degrees (i.e. 90 for a right angle) to determine the main *direction* of variability.

Example: If the ratio is 2 and the rotation parameter is 45 your covariance matrix will be equivalent to:

```
matrix([[2.5, 1.5],
        [1.5, 2.5]])
```

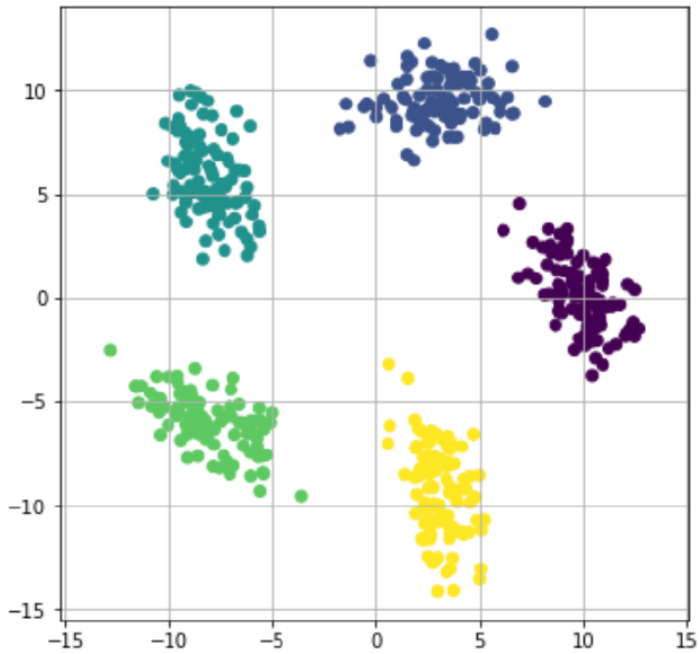
c) Using the results from the previous points, write your own function `data_matrix, targets = make_data(k, num_instances, radius, ratio)` to generate a data matrix with `num_instances` rows and 2 columns and a `targets` vector of length `num_instances` containing a class indicator for each instance (i.e. an integer between 0 and k-1). The function takes in input the number

of classes `k`, the total number of instances `num_instances`, the parameter `radius` to express the distance from the origin for the means of the multi variate normal data generators, and finally the `ratio` between the two main directions of variability to determine the covariance matrices for the multi variate normal data generators.

Important: For each one of the `k` multi variate normal data generator, sample the `rotation` parameter uniformly at random between 0 and 360.

d) Display a scatter plot in 2D of the generated dataset distinguishing instances belonging to different classes by color.

You should obtain something like this:



LDA model:

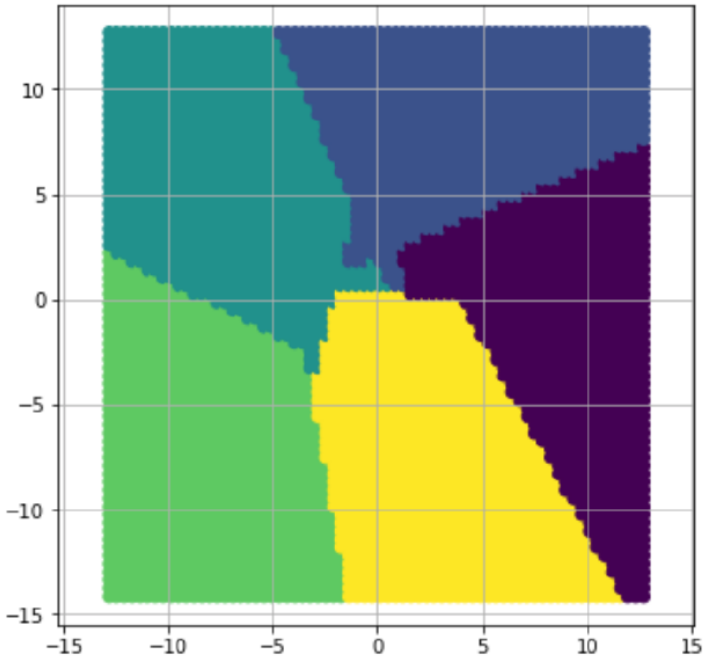
e) Write the function `params = fit_LDA(data_matrix,targets)`. The function outputs the parameters for a LDA classifier fit for the classification of the required number of classes. The number of classes will be automatically deduced from the `targets` vector.

f) Write the function `predictions = test_LDA(data_matrix, params)` to output the class predicted by the LDA model encoded in `param`.

g) Write a function `make_grid` to generate instances regularly spaced in 2D. The function should allow the user to specify the grid *density*, i.e. how many points to generate.

h) Generate 500 instances for 5 classes with radius 10 and ratio 2. Fit an LDA model. Generate a grid dataset that covers the original data set. Display a scatter plot in 2D of the grid dataset distinguishing instances belonging to different predicted classes by color.

You should obtain something like this:



Submitting your work

Please write your student ID in the first cell of the notebook.

You should submit the *Jupyter notebook* containing the code with its output for all the questions.
Make a separate cell for each point a), b), c), etc of each question.

Submit both a `PDF` copy of your notebook as a proof of execution and also the original source file with extension `.ipynb`. Make a single archive (.zip or .tar) containing the PDF **and** the notebook.

Markers will not be able to give feedback if you do not submit the PDF version of your code and marks will be deducted if you fail to do so.

Marking criteria

Work will be marked against the following criteria. Although it varies a bit from question to question they all have approximately equal weight.

- **Does your algorithm correctly solve the problem?** In most of the questions the required code has been described, but not always in complete detail and some decisions are left to you.
- **Is the code syntactically correct?** Is your program a legal Python program regardless of whether it implements the algorithm?
- **Is the code beautiful or ugly?** Is the implementation clear and efficient or is it unclear and extremely inefficient (e.g. it takes more than a few minutes to execute, or it has a quadratic complexity when a linear complexity would suffice)? Is the code well structured? Have you made good use of functions? Are you using Numpy functions on entire arrays when possible?
- **Is the code well laid out and commented?** Is there a comment describing what the code does? Have you used space to make the code clear to human readers?

There are 10% penalties for:

- Not submitting the PDF version of your programs.
- Not creating functions as instructed in the questions.