# CS5228-KDDM, 2024/25-2, Coursework 2

## Introduction

- This coursework comprises two parts. Part 1 involves Python programming for regression and classification, and Part 2 contains four MCQs.
- Total CA marks of this coursework is 12. Details of marks/parts are below.
- A Canvas quiz will be open for your coursework submission.
- For Python programming parts, I urge you to complete a Jupyter notebook and submit it. Cw2-template.ipynb is the template for your answer. You have to run your codes and get sure that answers are available in the notebook before submission.
- Regarding MCQs, there is one and only one correct answer for each question. So select the best option. There is no penalty for wrong answers.
- The deadline for this coursework is **30/3/2025**. Please be aware that no delayed submission is possible.
- Good luck, my friends.

## CW2, Part 1: Regression and Classification using Python (3+3=6 marks)

### CW2-1: Regression (3 marks)

Dataset: cs5228_housing.csv

Goal: Estimation of house price

Algorithm: Linear Regression

### Steps

1- The Dataset comprises information of different houses in America
2- We assume that the dataset is clean
3- The goal is to develop a linear regression model to estimate 'price' as a function of independent input variables { area, bedrooms, bathrooms, stories, main, guestroom, basement, hot water heating, air conditioning, parking, prefarea, furnishing status}
4- Ignore "**class_label**" variable. This is the class label for the next question.
5- If you need to normalize your variables, apply that only to the input/independent variables.
6- If you need to convert your categorical data to numerical, either go for one-hot or unique integer coding. In the latter case, avoid coding from zero. Having many zeros will harm the linear regression model. Usually, however, it's not the case if you use one-hot.

7- You may apply a correlation analysis to see if any input variable is discardable.
8- Go for 90% training and 10% testing splitting.
9- Check if linear regression has got any hyperparameter that should be optimized or not.
10- Metrics are the mean square error (MSE) and mean absolute error (MAE) both reportable on both training and testing subsets.
11- Keep the splitting replicable. Use random_state=110 .
12- You cant change the model. However, using preprocessing, variable removal, and hyperparameters optimization, you may try to achieve the lowest possible testing MSE/MAE.
13- Using any typical Python library is allowed.

## Deliverable

1- Use cw2-template.ipynb as your template for both cw2-1 and cw2-2 questions.
2- Report if you have discarded any input variable and the justification for that, e.g., based on correlation analysis.
3- Print a few rows of your dataset along with the description of the dataset.
4- Print the size of training and testing subsets.
5- Print the MSE and MAE of the training phase
6- Print the MSE and MAE of the testing phase
7- If you applied any normalization, report the MSE and MAE with and without normalization, both training and testing.
8- Only the last/best MAE/MSE should be printed after doing whatever you did to obtain better results.

## CW2-2: Classification using Decision Trees and Random Forest (3 marks)

Dataset: cs5228_housing.csv

Goal: Classification of houses into 4 price classes

Algorithm: Decision Tree and Random Forrest

## Steps

1- The Dataset comprises information of different houses in America
2- We assume that the dataset is clean
3- The goal is to develop a classification model to classify the houses into 4 different price classes {cheap, medium, expensive, very expensive} based on features/input variables { area, bedrooms, bathrooms, stories, main, guestroom, basement, hot water heating, air conditioning, parking, prefarea, furnishing status}
4- Ignore the "**price**" variable. This is the dependent variable employed for the regression part.
5- If you need to normalize your variables, do that.
6- If you need to convert your categorical data to numerical, either go for one-hot or unique integer coding.
7- You may apply a correlation analysis to see if any feature is discardable.
8- Go for 90% training and 10% testing splitting.

9- You will apply (train and test) decision tree and random forest models respectively. In both cases, hyperparameter optimization might be necessary.
10- Metrics are the Precision, Recall, and Accuracy (Pr/Rc/Acc), all reportable on both training and testing subsets.
11- Keep the splitting replicable. Use random_state=110 .
12- You can't change the models. However, using preprocessing, variable removal, and hyperparameters optimization, you may try to achieve the lowest possible testing Pr, Rc, and Acc.
13- Using any typical Python library is allowed.

## Deliverable

14- Use cw2-template.ipynb as your template for both cw2-1 and cw2-2 questions.
15- Report if you have discarded any input variable and the justification for that, e.g., based on correlation analysis.
16- Print a few rows of your dataset along with the description of the dataset.
17- Print the size of training and testing subsets.
18- Print Pr/Rc/Acc of the training phase
19- Print Pr/Rc/Acc of the testing phase
20- If you applied any normalization, report the Pr/Rc/Acc with and without normalization, both training and testing.
21- Only the last/best Pr/Rc/Acc should be printed after doing whatever you did to obtain better results.
22- Print the confusion matrix, too.

# CW2, Part 2: MCQs (4x1.5= 6 marks)

2- In a classification task, we have 4 classes, namely {'cheap' , 'medium' , 'expensive' , 'very expensive'}. The model provides classification performance mentioned in the confusion matrix below. What are the accuracy, macro average Precision, and weighted average Precision respectively?

| | | Predicted | | | | |
|---|---|---|---|---|---|---|
| | | **cheap** | **expensive** | **medium** | **very expensive** | **∑** |
| Actual | **cheap** | 9 | 1 | 10 | 0 | **20** |
| | **expensive** | 6 | 97 | 37 | 12 | **152** |
| | **medium** | 36 | 44 | 231 | 5 | **316** |
| | **very expensive** | 0 | 23 | 10 | 24 | **57** |
| | **∑** | **51** | **165** | **288** | **41** | **545** |

a- 0.721 , 0.69 , 0.73
b- 0.804 , 0.604 , 0.731
c- 0.662 , 0.537 , 0.696
d- 0.651 , 0.553 , 0.712

3- Considering the transaction database below. We are going to apply an association rules mining on that transaction database. If we set minimum_support= 0.35 and minimum_confidence= 0.7, will we come across any association rule with confidence= 1 ?

| Trans. ID | Items |
|---|---|
| 1 | ('bread', 'yogurt') |
| 2 | ('bread', 'milk', 'cereal', 'eggs') |
| 3 | ('yogurt', 'milk', 'cereal', 'cheese') |
| 4 | ('bread', 'yogurt', 'milk', 'cereal') |
| 5 | ('bread', 'yogurt', 'milk', 'cheese') |
| 6 | ('bread', 'yogurt') |
| 7 | ('bread', 'milk', 'eggs', 'coke can') |
| 8 | ('bread') |

a- No
b- Yes, {cereal} → {milk}
c- Yes, {milk} → {bread}
d- Yes, {yogurt} → {bread}

4- In classification models, e.g. KNN classifiers, we usually measure the distance between an unknown/unclassified sample and training samples based on the Euclidean distance between their feature vectors. However, we may use other distance functions such as Cosine similarity, i.e. computing the inner product of feature vectors, and Cosine between those feature vectors. The question is,

- When cosine similarity is better than the Euclidean distance?
- We have 2 classes, C1 and C2. C1 is represented with F1 feature vector and C2 with F2. Using cosine similarity, which class would be assigned to the unclassified sample X with its feature vector FX?
    - F1 = [ 735, 60, 3, 1, 2, 1000, 1, 0.0 ]
    - F2 = [ 381, 31, 2, 2, 0, 0, 1, 2.0 ]
    - FX = [ 350, 35, 3, 1, 1, 1000, 1, 0.0 ]

a- Generally, using cosine similarity instead of Euclidean distance for classification is better in scenarios where the magnitude of feature vectors is not important, but the direction (or angle) matters. More specifically when feature vectors are large and sparse cosine similarity is usable. Also, FX will be classified in C1.
b- Generally, using cosine similarity never has any advantages compared to Euclidean distance, unless we can prove that by testing both functions. FX belongs to C2.
c- If the dataset contains features with significantly different magnitudes, cosine similarity is better because it ignores scale differences. FX would be classified in C2.
d- Generally, using cosine similarity instead of Euclidean distance for classification is better in scenarios where the magnitude of feature vectors is very decisive, but the direction (or angle) doesn't matter. More specifically when feature vectors are large and sparse cosine similarity is not usable. Also, FX will be classified in C1.

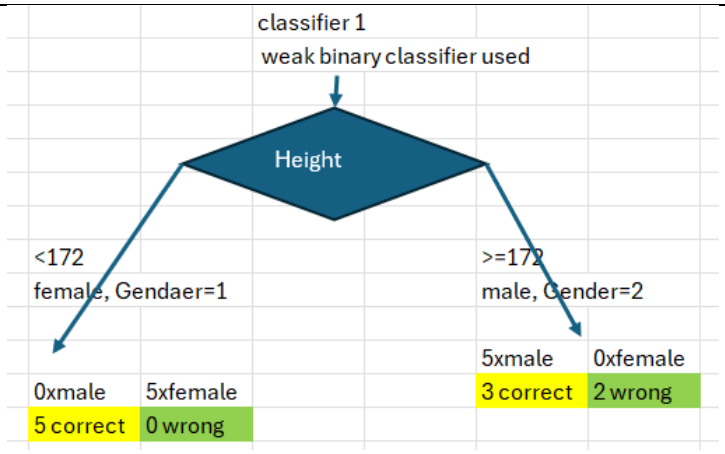
5- We are going to classify the labeled dataset below employing an Ada-Boost algorithm. We have 2 classes, Gender= Female, or 1, and Gender=Male or 2. Two features are extracted, Height and Weight. At the beginning of Ada-Boost algorithm, we have 2 weak binary classifiers, classifier 1 and 2. Their performance on the dataset and splitting threshold can be seen below, as well. Now please answer:

- What is the amount of say of each classifier?
- Rationally, which classifier will be used?
- What is the initial sample-weight array and the updated normalized sample-weight array after employing the classifier? $w_i$ indicates the sample-weight of $i_{th}$ data sample.

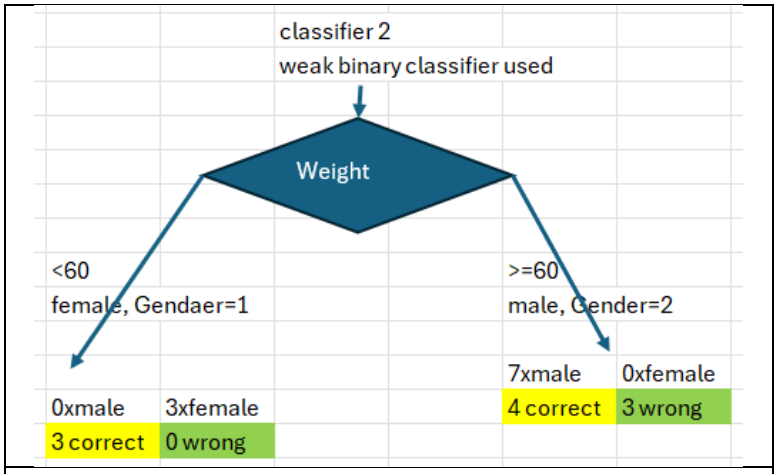
| Celebrity | Height | Weight | Gender |
|---|---|---|---|
| Kylie Minogue | 152 | 42 | 1 |
| Jason Donovan | 183 | 78 | 2 |
| Elvis Presley | 182 | 77 | 2 |
| Kathy Parry | 170 | 61 | 1 |
| Sean Connery | 188 | 75 | 2 |
| Daniel Craig | 178 | 78 | 2 |
| Catherine Z-Jones | 170 | 52 | 1 |
| Sophia Loren | 172.7 | 63.5 | 1 |
| Ingrid Bergman | 175.3 | 61.2 | 1 |
| Madonna | 161 | 45 | 1 |

1=female, 2=male

Dataset




Classifier 1




Classifier 2

a- Amount of say: classifier 1=0.5 , classifier 2=0.75, classifier 2 is selected. Sample-weights:

| $w_i$ | $w_1$ | $W_2$ | $W_3$ | $W_4$ | $W_5$ | $W_6$ | $W_7$ | $W_8$ | $W_9$ | $W_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Initial sample-weight | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 |
| Updated sample-weight | 0.11 | 0.11 | 0.25 | 0.5 | 0.11 | 0.11 | 0.11 | 0.25 | 0.11 | 0.11 |

b- Amount of say: classifier 1=0.69 , classifier 2=0.42, classifier 1 is selected. Sample-weights:

| $w_i$ | $w_1$ | $W_2$ | $W_3$ | $W_4$ | $W_5$ | $W_6$ | $W_7$ | $W_8$ | $W_9$ | $W_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Initial sample-weight | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Updated sample-weight | 0.063 | 0.063 | 0.063 | 0.063 | 0.063 | 0.063 | 0.063 | 0.25 | 0.25 | 0.063 |

c- Amount of say: classifier 1=0.69 , classifier 2=0.42, classifier 1 is selected. Sample-weights:

| $w_i$ | $w_1$ | $W_2$ | $W_3$ | $W_4$ | $W_5$ | $W_6$ | $W_7$ | $W_8$ | $W_9$ | $W_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Initial sample-weight | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Updated sample-weight | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.05 | 0.05 | 0.1 | 0.1 | 0.2 |

d- Amount of say: classifier 1=0.5 , classifier 2=0.75, classifier 2 is selected. Sample-weights:

| $w_i$ | $w_1$ | $W_2$ | $W_3$ | $W_4$ | $W_5$ | $W_6$ | $W_7$ | $W_8$ | $W_9$ | $W_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Initial sample-weight | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Updated sample-weight | 0.25 | 0.063 | 0.063 | 0.063 | 0.063 | 0.063 | 0.063 | 0.063 | 0.063 | 0.25 |

*********