# Week 1

The aim of this week to introduce you to the core concepts in developing and building a machine learning model, including:

* an overview of machine learning and types of machine learning
* an introduction of two main problem types: regression and classification in machine learning
* an introduction of the key machine learning libraries in python
* define machine learning workflow
* explain methods and techniques to build and train a machine learning model (such as data exploration; feature selection, feature engineering , preprocessing (outliers, normalise, missing values), model selection and evaluation)

# Week 2: Hands On exercises

**Exercise: Boston House Prices prediction**

In this exercise, we aim to predict the Boston house prices based on several environmental, economic, demographic, and societal features using the Boston House dataset. Firstly we do data exploration and data processing. We continue to build a regression model(Support Vector machines for Regression) and evaluate the performance of the model.

**Exercise: Breast Cancer classification**

In this exercise, we aim to create a classification model (mainly Support vector Classification) that predicts if the cancer diagnosis is benign or malignant based on several features.

# Week 3:

The aim of this week to explain model selection and evaluation, including:

* An overview of model selection and model evaluation
* What are regression and classification evaluation metrics for measuring the performance of this trained model and explain when to choose these metrics
* An overview of cross validation
* An overview of ensemble methods
* An overview of hyperparameter tuning (such as GridSearchCV and RandomizedSearchCV)

# Week 4:

The aim of this week is to cover feature engineering.

# Week 5:

The aim of this week is to build a deep learning model in Keras, including:

* An overview of a deep learning; MLP, CNN, SLP, VGG16, ResNet models
* An overview of Keras Model API: the sequential and the functional API
* Steps to define and train a model in Keras
* Explain how to improve a baseline model: adding hidden layers, dropout layer, changing optimizer, epochs, batch sizes, using SGD
* Types of layer; convolutional layer, pooling layer, max layer, fully connected layer

# Week6: Hands on Exercises

**Exercise: Classification FM without dimensionality**

The project consists of a main task which is to classify items in the Fashion MNIST dataset successfully. We will divide the implementation course into two main parts, one which we will be doing just for once and we won’t need to rewrite the code for it every time we run the whole code. The second part, which will contain the different models we will be using in this project mainly: Convolutional Neural Networks (CNN), Multi-Layer Perceptron (MLP), Single Layer Perceptron (SLP), VGG16, ResNet, Gaussian Mixture Model and Clustering more specifically we are going to use K-means Clustering techniques. To have more fun! We will apply these models on processed data before and after applying Principal Component Analysis (PCA).

# Week7:

The aim of this week is to decide where and when to use PCA, including;

* An overview of Principal Component Analysis
* Explain how to apply PCA; data standardisation, create a covariance matrix,eigen decomposition, feature transformation,

# Week8:

**Exercise: Classification FM with dimensionality**

The project consists of a main task which is to classify items in the Fashion MNIST dataset successfully. We will divide the implementation course into two main parts, one which we will be doing just for once and we won’t need to rewrite the code for it every time we run the whole code. The second part, which will contain the different models we will be using in this project mainly: Convolutional Neural Networks (CNN), Multi-Layer Perceptron (MLP), Single Layer Perceptron (SLP), VGG16, ResNet, Gaussian Mixture Model and Clustering more specifically we are going to use K-means Clustering techniques. To have more fun! We will apply these models on processed data before and after applying Principal Component Analysis (PCA).

# Notebook Regression - boston.py

Step 1: Import libraries

* **pandas**
* **matplotlib**
* **numpy**
* **seaborn**
* **sklearn**
  + **datasets**
    - **load\_boston**
  + **preprocessing**
  + **model selection**
    - **train\_test\_split**
    - **GridSearchCV**
  + **svm** 
    - **SVR (Support Vector Regression)**

Step 2: Data exploration

* Get the data
* Print target and features

Step 3: Plot graphs to understand data

* Plotting the heatmap of correlation between features

Step 4: Preprocess data

* Remove outliers
* Normalise data - **StandardScaler()**
* Preprocess data

Step 5: Build a baseline model

* Train a model using **SVR()** linear regression model

Step 6: Evaluate the model

* Evaluate the model using different metrics predefined in the sklearn.metrics
  + **R^2**
  + **MAE**
  + **MSE**
  + **RMSE**

Step 7: Improve model

* Optimise model using **GridSearchCV** by defining best parameters

# Notebook Classification - breast\_detection.py

Step 1: Import libraries

* **pandas**
* **matplotlib**
* **numpy**
* **seaborn**
* **sklearn**
  + **datasets**
    - **load\_breast\_cancer**
  + **preprocessing**
  + **model selection**
    - **train\_test\_split**
    - **GridSearchCV**
  + **svm** 
    - **SVC (Support Vector Classification)**

Step 2: Data exploration

* Get the data
* Print target and features

Step 3: Plot graphs to understand data

* Plotting the heatmap of correlation between features

Step 4: Preprocess data

* Remove outliers
* Normalise data - **StandardScaler()**
* Preprocess data

Step 5: Build a baseline model

* Train a model using **SVC()**

Step 6: Evaluate the model

* Evaluate the model using different metrics predefined in the sklearn.metrics
  + **confusion\_matrix**
  + **accuracy\_score**

Step 7: Improve model

* Optimise model using **GridSearchCV** by defining best parameters

# Notebook1 FM without dimensionality reduction

Here we will start implementing our project step by step. The project consists of a main task which is to classify items in the Fashion MNIST dataset successfully. We will divide the implementation course of two main parts, one which we will be doing just for once and we won’t need to rewrite the code for it every time we run the whole code. The second part, which will contain the different models we will be using in this project mainly: Convolutional Neural Networks (CNN), Multi-Layer Perceptron (MLP), Single Layer Perceptron (SLP), VGG16, ResNet, Gaussian Mixture Model and Clustering more specifically we are going to use K-means Clustering techniques.

## Part1: Preprocessing the Fashion MNIST dataset

Step 1: Import libraries

* **tensorflow**
* **numpy**
* **pandas**
* **keras**
* **sklearn**
* **sklearn.cluster**
* **matplotlib**
* **keras.preprocessing**
* **keras.models**
* **keras.layers**
* **keras.optimizers**

Step 2: Writing a callback function

* Define the training variables: **number of classes, class names, epochs, batch size, channels, verbose, metrics**
* Load the dataset from keras
  + Define **width, height, layer(dense, conv2d), channels, class names**
  + **Flatten** images
  + print **shape**
* **Normalize** the values between 0 and 1
* Convert labels to **categorical/dummy encoding** to use "**categorical\_crossentropy**" as loss.
* Plot images
* Train a model
  + Define inputs
* Plot loss and metrics
  + **accuracy**
  + **mae**
  + **mse**
* Plot **ROC curve** to visualise the evaluation metrics
* Load best model, evaluate and predict on unseen data
* Plot confusion matrix

## Part 2:

here we will use **GridSearchCV** to find optimal parameters for the model

Create a baseline model

Provide scoring method and various parameters

Finally, GridSearchCV function will then iterate through each parameter combination to find the best scoring parameters.

* **Single Layer Perceptron**
* **Multi-Layer Perceptron**
* **Convolutional Neural Network**
* **Kmeans (Clustering)**
* **Resnet**
* **Gaussian Mixture Model**

Step 1: **Create a single layer perceptron network**

* Build a model
  + **keras.models.Sequential()**
  + **model.add(keras.layers.Dense(nb\_classes, input\_dim=(height\* width), use\_bias=False, activation='softmax'))**
  + **model.compile(loss = 'categorical\_crossentropy', optimizer=tensorflow.keras.optimizers.Adam(lr=.0001), metrics=metrics)**
* Train a model using callback function
* Print the loss and accuracy using callback function
* Print Confusion Matrix, Classification report, ROC curve
* Predict on unseen test data

Step 2: **Create a multi-layer perceptron network**

* Build a model
  + **keras.models.Sequential()**
  + **model.add**(keras.layers.Dense(256, input\_dim=(height\* width), activation='relu'))
  + **model.compile**(loss = 'categorical\_crossentropy', optimizer='rmsprop', metrics=metrics)
* Train a model using callback function
* Print the loss and accuracy using callback function
* Print Confusion Matrix, Classification report, ROC curve
* Predict on unseen test data

Step 3: **Create a CNN model**

* D**efine convolutional filters, kernel size, pooling, dropout**
* Create a model
  + **model = Sequential()**
  + **model.add(Conv2D(filters=nb\_filters, kernel\_size=(nb\_conv,nb\_conv), strides=(1, 1), activation='relu', input\_shape=(image\_height, image\_width, channels)))**
  + **model.add(BatchNormalization())**
  + **model.add(MaxPool2D(pool\_size=(nb\_pool,nb\_pool)))**
  + **model.add(Dropout(0.5))**
  + **model.add(Flatten())**
  + **model.add(Dense(128, activation='relu'))**
  + **model.add(Dense(nb\_classes, activation='softmax'))**
  + **model.compile(loss='categorical\_crossentropy', optimizer=optimizer, metrics=metrics)**
* Train a model using callback function
* Print the loss and accuracy using callback function
* Print Confusion Matrix, Classification report, ROC curve
* Predict on unseen test data

Step 4: Create a **KerasClassifier()** model

* Define a model
* Define optimizer (adam, rmsprop, SGD), epochs, batches
* Define parameters for GridSearch
* Define GridSearchCV
* Fit the model
* Summarise result

Step 5: Build **Kmean Clustering model**

Step 6: Build th**e Gaussian Mixture model**

# Notebook2 FM without dimensionality reduction

**Applying Transfer Learning**

Step 1: Import libraries

* **tensorflow**
* **keras**
* **keras.layers**
  + **Dense, Conv2D, BatchNormalization, Activation, MaxPooling2D, AveragePooling2D, Input, Flatten, Dropout**
* **sklearn**
* **pandas**
* **matplotlib**
* **keras.optimizers**
  + **Adam**
* **keras.models**
  + **Model**
  + **Sequential**
* **keras.regularizers**
  + **l2**

Step 2: Load dataset

Step 3: Visualise images

Step 4: Define training parameters

* epochs, batch\_size, data\_augmentation, image size, num\_classes, filters, blocks, max\_pool

Step 5: Normalise data

Step 6: Converting labels to one-hot vectors

Step 7: Create a **Resnet model**

Step 8: Print evaluation metrics

* loss and accuracy

# Notebook1 FM with dimensionality reduction

**APPLY PCA to reduce dimensionality**

Step 1: Import libraries

Step 2: Load the dataset

Step 3: Normalise the data

Step 4: Print images

Step 5: Call the PCA method

* **pca = PCA(n\_components=5)**
* pca.fit(X\_PCA)
* X\_5d = pca.transform(X\_PCA)

Step 6: Invoking the t-SNE method

* **tsne = TSNE(n\_components=2)**
* tsne\_results = tsne.fit\_transform(X\_PCA)

Step 7: **Cross validation strategy**

* Use **k-fold cross validation**
* Use the **accuracy\_score** as our metric to compare themodels

Step 8: **Random Forest (ensembling model of decision trees)**

* Apply a PCA to reduce the dimension of our images using Random Forest

Step 9: **Encode the target data to make Keras understand**

* **Use MLP, CNN, VGG16**

Step 10: Create MLP model

* Create a model and compile it for each fold
* Print performance accuracy
* Train the model
* Predict it on the test data

Step 11: Create a CNN model

* Create a model and compile it for each fold
* Print performance accuracy
* Train the model
* Predict it on the test data

Step 12: Create a VGG16 model

* Create a model and compile it for each fold
* Print performance accuracy
* Train the model
* Predict it on the test data

**Step 13: Take the best model and analyse the images**

* Use CNN
* Create confusion matrix
* **Improve CNN model using GradCAM technic**

# Notebook2 FM with dimensionality reduction

Step 1: Import different libraries

Step 2: Load the dataset

Step 3: Visualise an image

Step 4: Normalise the data

Step 5: Call PCA method

Step **6: Apply PCA using k-means clustering**