# Assignment-3 Q.1

M.Vivekananda

## Task-1

```
# python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.utils import to_categorical
import numpy as np
```

**load_iris** : is used to load the iris dataset from sklearn.datsets
**train_test_split**: It splits the data into training and testing data sets.
**Standard Scaler**: Standardizes features ( zero mean and unit variance)
**to_categorical:** It converts class labels (0, 1, 2) into one-hot encoded vectors for use in classification.
**One hot encoded vectors**

- It is a binary vector used to represent categorical data where only one element is the one corresponding to the actual category
- And all the other elements are zero

Like integer class labels are represented by [0,1,2]
so for this in one hot encoding we get [ 1 , 0 , 0 ] , [ 0 , 1 , 0 ] , [ 0 , 0 , 1 ] class 0 , class 1 , class 2 respectively.
So that binaries are easily to read
**numpy:**
Used for general numerical operations

```
# python
iris = load_iris()
X = iris.data
y = iris.target
```

- **iris.data** : Like it contains 150 samples with 4 features : sepal length , sepal width , petal length , petal width
- **iris.target** : conatins class labels like the mapping 0 for setosa , 1 for versicolor , 2 for virginica

```
# python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    ↪ =0.2, random_state=42)
```

This line **splits the dataset** into:

- **Training set** → to train the model
- **Testing set** → to evaluate the model's performance on unseen data

**X**:
Features (e.g., petal length, sepal width, etc.) — shape: (150, 4)
**y**:
Target labels (e.g., 0, 1, 2 for the 3 Iris species)
**test_size = 0.2** like it means 20% of the data set will be used by testing and the remaining by the training set

In the iris data set there are 150 samples and in that Training set = 120 samples and Testing set = 30 samples

**random_state=42** like this is just a random seed and it controls the shuffling of the data before splitting and setting 42 makes sure that every time you run the code you get the same result and i t can be any number like it can be 42 or 52 but accuracy may change because the the distribution of 80-20 will be fixed but the samples may be shuffled

```
# python
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

fit_transform() learns the mean and std from training data and scales it
transform() applies the same transformation to the test data

```
# python
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

It is one hot coding like
0 → [ 1 , 0 , 0 ] ( Setosa )
1 → [ 0 , 1 , 0 ] ( Versicolor )
2 → [ 0 , 0 , 1 ] (Virginica)

# Task 2 - NN construction

```
# python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

**Sequential** is a linear stack of layers
**Dense** is a fully connected (dense) neural network layer.

```
# python
model = Sequential()
```

It initializes the model as a sequential neural network.

```
# python
model.add(Dense(8, input_shape=(4,), activation='relu'))
```

- Adds a hidden layer with:
  - 8 neurons
  - input_shape = (4,) because we have 4 input features
  - **Activation : ReLU (Rectified linear unit)** like it is defined as $f(x) = \max(0, x)$. It introduces non-linearity and helps the network learn complex patterns.

```
# python
model.add(Dense(3, activation='softmax'))
```

- Output layer with 3 neurons (for 3 classes)
- Softmax activation converts raw outputs to probabilities, ensuring the sum is 1
- Like if we observe taking an example by [0.1, 0.3, 0.6], the 0.6 is max and 0.1 and 0.3 are less probable so it will map as [0, 0, 1] which is class 2 by one hot encoded vectors.

# Task 3 - Model Compilation and training

```
# python
model.compile(optimizer='adam', loss='categorical_crossentropy',
    ↪ metrics=['accuracy'])
```

- Optimizer : adam (adaptive momentum estimation) — it is fast and used for training deep networks
- Loss : categorical_crossentropy — it is used for multiclass classification with one-hot encoded labels
- Metrics : We track accuracy during training

```
# python
model.fit(X_train, y_train, epochs=100, batch_size=5, verbose=0)
```

- Training parameters:
  - epochs=100: this means entire training data will be shown to the model 100 times
  - batch_size=5: Model updates weights every 5 samples
  - verbose=0: this means it suppresses training output; we can set verbose=1 to see progress

## Task-4 - Model Eval

```
# python
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test_Accuracy:{accuracy*100:.2f}%")
```

- model.evaluate(): Computes loss and accuracy on the test data.

## Code

- Colab Notebook Link