In [ ]:

```
# -*- coding: utf-8 -*-

from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.initializers import glorot_normal
```

In [ ]:

```
class Classifier:

    ###############
    # Constructor #
    ###############

    def __init__(self, num_features, cls_params, seed=0):

        # fixed parameters
        self.seed = seed
        self.weight_init = glorot_normal

        # required parameters (more below)
        self.num_features = num_features
        self.cls = cls_params['cls_name']
        self.learning_rate = cls_params['learning_rate']
        self.momentum = cls_params['momentum']

        # create model
        if self.cls == 'logistic_regression':
            self.model = self.create_lg()
        elif self.cls == 'neural_network':
            self.hidden_units = cls_params['hidden_units']
            self.hidden_activation_fun = cls_params['hidden_activation_fun']
            self.model = self.create_nn()
        # self.model.summary()

        # prepare model for training
        self.model.compile(
            optimizer=SGD(lr=self.learning_rate, momentum=self.momentum),
            loss='binary_crossentropy',
            metrics=['accuracy']
        )

    ######################
    # Logistic regression #
    ######################

    def create_lg(self):
        # Input layer
        x_input = Input(shape=(self.num_features,))

        # Output layer
        y_out = Dense(
            units=1,
            activation="sigmoid",
            use_bias=True,
            kernel_initializer=self.weight_init(seed=self.seed),
            bias_initializer='zeros'
        )(x_input)

        # Model
        return Model(inputs=x_input, outputs=y_out)

    ##################
    # Neural network #
    ##################

    def create_nn(self):
```

```
        # Input layer
        x_input = Input(shape=(self.num_features,))

        #  Hidden layer
        x = Dense(
            units=self.hidden_units,
            activation=self.hidden_activation_fun,
            use_bias=True,
            kernel_initializer=self.weight_init(seed=self.seed),
            bias_initializer='zeros'
        )(x_input)

        # Output layer
        y_out = Dense(
            units=1,
            activation="sigmoid",
            use_bias=True,
            kernel_initializer=self.weight_init(seed=self.seed),
            bias_initializer='zeros'
        )(x)

        # Model
        return Model(inputs=x_input, outputs=y_out)

    ####################
    # Train classifier #
    ####################

    def train_cls(self, x, y):
        self.model.fit(
            x=x,
            y=y.astype('int'),  # cast class to integer
            epochs=1,
            batch_size=1,
            verbose=0
        )
```