GitHub repository: https://github.com/ldabas/QCNNForecasting

Documentation

1 QuantumModelMaker

1.1 quantum-rescale Function

- **Objective**: Prepare a given dataset for quantum encoding by rescaling it to be within range of 0 and 1.
- **Input Parameter(s)**: Pandas Dataset
- **Return**: Returns x as input features and y as output label.

1.2 smooth Function

- **Objective**: Make a given dataset stationary by applying exponential smoothing to each of the columns to its training subset.
- **Input Parameter(s)**: Pandas Dataset: Input for smoothing.
  'train-split': An integer value to split the dataset into a training set that is smoothed while preserving the test set.
  'smooth-level': Alpha value for smoothing the dataset.
- **Return**: Returns the smoothed dataset

1.3 create-discrete-circuits Function

- **Objective**: Generate quantum circuits compatible with discrete-data QCNN models.
- **Input Parameter(s)**: Data: Pandas Dataset
  Qubits: `Cirq.GridQubit` Object.
  Test: Boolean present for investigating process of circuit generation. Set to False. 'smooth-level': Alpha value for smoothing the dataset.
- **Return**: Returns a tensor of quantum circuits for each input datapoint within the dataset.

1.4 encode-discrete-data Function

- **Objective**: Split dataset into Train, Validation, and Test and create return output labels and quantum circuits of input features compatible with discrete-data models.
- **Input Parameter(s)**: data: Pandas Dataset
  labels: Output Labels (Biogas Production).
  Qubits: `Cirq.GridQubit` Object.
- **Return**: Returns output labels and quantum-encoded input features for discrete-data models for each of the train, validation, and test sets.

1.5 create-time-window-circuits Function

- **Objective**: Generate quantum circuits compatible with time-window QCNN models.
- **Input Parameter(s)**: Data: Pandas Dataset
  Qubits: `Cirq.GridQubit` Object.

Test: Boolean present for investigating process of circuit generation. Set to False. 'smooth-level': Alpha value for smoothing the dataset.

- **Return**: Returns a tensor of quantum circuits for each input datapoint within the dataset.

1.6 encode-time-window-data Function

- **Objective**: Split dataset into Train, Validation, and Test and create return output labels and quantum circuits of input features compatible with time-window models.
- **Input Parameter(s)**: data: Pandas Dataset
  labels: Output Labels (Biogas Production).
  Qubits: `Cirq.GridQubit` Object.
- **Return**: Returns output labels and quantum-encoded input features for time-window models for each of the train, validation, and test sets.

1.7 create-model-circuit Function

- **Objective**: Create QCNN Architecture model
- **Input Parameter(s)**: Qubits: `Cirq.GridQubit` Object.
- **Return**: Return a circuit representation of a QCNN model

1.8 prepare-quantum-states Function

- **Objective**: Prepare quantum-circuits for inputs and model depending on the model-type passed in as an input.
- **Input Parameter(s)**: X: Input Feature Dataset
  y: Output Label Pandas Series
  feature-count: Number of features being used for model development.
  type: Type of model being built. Can be either 'time-window' or 'discrete'.
- **Return**: QCNN model circuit, qubit object, X, and y for train, test, and validation.

1.9 get-callbacks Function

- **Objective**: Create Callbacks that the model can use during training for creating model checkpoints
- **Input Parameter(s)**: Patience: Number of epochs that the model can wait to verify if the validation loss descreases. Set to 5 by default.
- **Return**: Return es-callback which is an early stopping callback, and modelckpt-callback, which is a model checkpoint callback.

1.10 build-model Function

- **Objective**: Concatenate QCNN Model with PQC layer and compile an executeable model.
- **Input Parameter(s)**:qcnn-circuit: Circuit version of QCNN
  Qubits: `Cirq.GridQubit` object.
- **Return**: a compiled executeable tefq model.

1.11 make-q-cnn-model Function

- **Objective**: A wrapper that utilises all prior functions to create an executeable model for a given model type and train it.
- **Input Parameter(s)**:type: 'discrete' or 'time-window' String object
  retrain: Boolean value to continue training if model does not train to a low enough val-loss
  model: Model Object
  history: History Object containing a model object's history.
- **Return**: Model Object and History

2 Quantum Model Validator

2.1 visualize-loss Function

- **Objective**: Visualize a model's training and validation loss
- **Input Parameter(s)**:History: 'A model's training history object
  Title: String Object.
- **Return**: None

3 ModelMaker

3.1 smooth Function

- **Objective**: Make a given dataset stationary by applying exponential smoothing to each of the columns to its training subset.
- **Input Parameter(s)**: Pandas Dataset: Input for smoothing.
  'train-split': An integer value to split the dataset into a training set that is smoothed while preserving the test set.
  'smooth-level': Alpha value for smoothing the dataset.
- **Return**: Returns the smoothed dataset

3.2 Generate-Classical-Dataset Function

- **Objective**: Create Train, Test, and Validation Datasets
- **Return**: Returns Train, Test, and Validation Datasets.

3.3 build-model Function

- **Objective**: Build an executeable tensorflow CNN model for given data-type
- **Input Parameter(s)**: mtype: Model Type, either 'discrete' or 'time-window'.
  inputs: Input object for executeable model. 'smooth-level': Alpha value for smoothing the dataset.
- **Return**: Returns an executeable tensorflow CNN model for given data-type.

3.4 get-callbacks Function

- **Objective**: Create Callbacks that the model can use during training for creating model checkpoints

- **Input Parameter(s)**: Patience: Number of epochs that the model can wait to verify if the validation loss descreases. Set to 5 by default.
- **Return**: Return es-callback which is an early stopping callback, and modelckpt-callback, which is a model checkpoint callback.

3.5 train-model Function

- **Objective**: Train the created 'time-window' or 'discrete' model
- **Input Parameter(s)**: model: model object
  dataset-train: Training Dataset
  dataset-val: Validation Dataset
  es-callback: EarlyStopping Callback
  modelckpt-callback: Model Checkpoint Callback
- **Return**: Return return a trained model and it's history.

3.6 make-discrete-data-cnn-model Function

- **Objective**: A wrapper that utilises all prior functions to create an executeable discrete-data model for a given model type and train it.
- **Input Parameter(s)**: :retrain: Boolean value to continue training if model does not train to a low enough val-loss
  model: Model Object
  history: History Object containing a model object's history.
- **Return**: Return return a trained model and it's history.

3.7 make-time-window-cnn-model Function

- **Objective**: A wrapper that utilises all prior functions to create an executeable time-window model for a given model type and train it.
- **Input Parameter(s)**: :retrain: Boolean value to continue training if model does not train to a low enough val-loss
  model: Model Object
  history: History Object containing a model object's history.
- **Return**: Return return a trained model and it's history.

3.8 Model Validator

3.9 visualize-loss Function

- **Objective**: Visualize a model's training and validation loss
- **Input Parameter(s)**:History: 'A model's training history object
  Title: String Object.
- **Return**: None

3.10 show-plot Function

- **Objective**: Visualize a model's Predictions against expected values
- **Return**: None