tf.keras.wrappers.scikit_learn.KerasClassifier

Contents

Class `KerasClassifier`

Defined in `tensorflow/python/keras/_impl/keras/wrappers/scikit_learn.py`.

Implementation of the scikit-learn classifier API for Keras.

Methods

`__init__`

```
__init__(
    build_fn=None,
    **sk_params
)
```

`check_params`

```
check_params(params)
```

Checks for user typos in "params".

Arguments:

- `params`: dictionary; the parameters to be checked

Raises:

- `ValueError`: if any member of `params` is not a valid argument.

`filter_sk_params`

```
filter_sk_params(
    fn,
    override=None
)
```

Filters `sk_params` and return those in `fn`'s arguments.

Arguments:

- `fn`: arbitrary function
- `override`: dictionary, values to override sk_params

Returns:

- `res`: dictionary dictionary containing variables in both sk_params and fn's arguments.

`fit`

```
fit(
    x,
    y,
    **kwargs
)
```

Constructs a new model with `build_fn` & fit the model to `(x, y)`.

Arguments:

- `x`: array-like, shape `(n_samples, n_features)` Training samples where n_samples in the number of samples and n_features is the number of features.
- `y`: array-like, shape `(n_samples,)` or `(n_samples, n_outputs)` True labels for X.
- `**kwargs`: dictionary arguments Legal arguments are the arguments of `Sequential.fit`

Returns:

- `history`: object details about the training history at each epoch.

Raises:

- `ValueError`: In case of invalid shape for `y` argument.

`get_params`

```
get_params(**params)
```

Gets parameters for this estimator.

Arguments:

- `**params`: ignored (exists for API compatibility).

Returns:

Dictionary of parameter names mapped to their values.

`predict`

```
predict(
    x,
    **kwargs
)
```

Returns the class predictions for the given test data.

Arguments:

- `x`: array-like, shape `(n_samples, n_features)` Test samples where n_samples in the number of samples and n_features is the number of features.
- `**kwargs`: dictionary arguments Legal arguments are the arguments of `Sequential.predict_classes`.

Returns:

- `preds`: array-like, shape `(n_samples,)` Class predictions.

`predict_proba`

```
predict_proba(
    x,
    **kwargs
)
```

Returns class probability estimates for the given test data.

Arguments:

- `x`: array-like, shape `(n_samples, n_features)` Test samples where n_samples in the number of samples and n_features is the number of features.
- `**kwargs`: dictionary arguments Legal arguments are the arguments of `Sequential.predict_classes`.

Returns:

- `proba`: array-like, shape `(n_samples, n_outputs)` Class probability estimates. In the case of binary classification, tp match the scikit-learn API, will return an array of shape '(n_samples, 2)' (instead of `(n_sample, 1)` as in Keras).

`score`

```
score(
    x,
    y,
    **kwargs
)
```

Returns the mean accuracy on the given test data and labels.

Arguments:

- `x`: array-like, shape `(n_samples, n_features)` Test samples where n_samples in the number of samples and

n_features is the number of features.

- `y`: array-like, shape `(n_samples,)` or `(n_samples, n_outputs)` True labels for x.
- `**kwargs`: dictionary arguments Legal arguments are the arguments of `Sequential.evaluate`.

Returns:

- `score`: float Mean accuracy of predictions on X wrt. y.

Raises:

- `ValueError`: If the underlying model isn't configured to compute accuracy. You should pass `metrics=["accuracy"]` to the `.compile()` method of the model.

set_params

```
set_params(**params)
```

Sets the parameters of this estimator.

Arguments:

- `**params`: Dictionary of parameter names mapped to their values.

Returns:

self