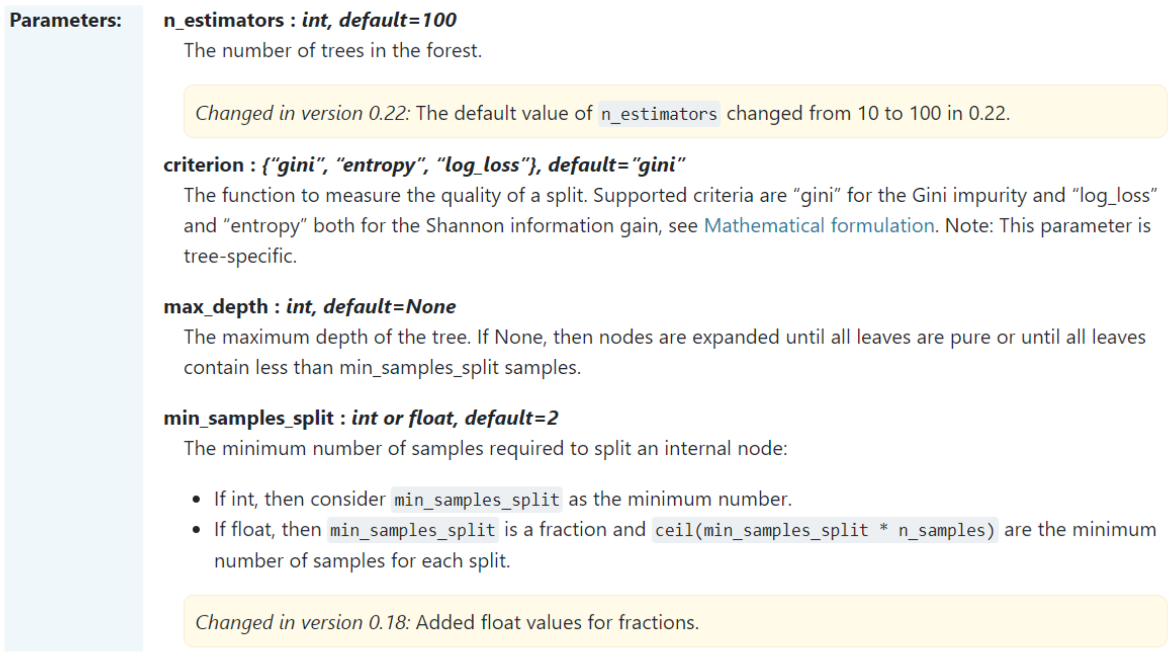
Random Forest in Sleep Stage Prediction Model

Architecture

The initial random forest used was a default random forest offered by scikit-learn.ensemble package of python. The default parameters for the random forest are as follows :-



Parameters:

n_estimators : *int, default=100*

The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation. Note: This parameter is tree-specific.

max_depth : *int, default=None*

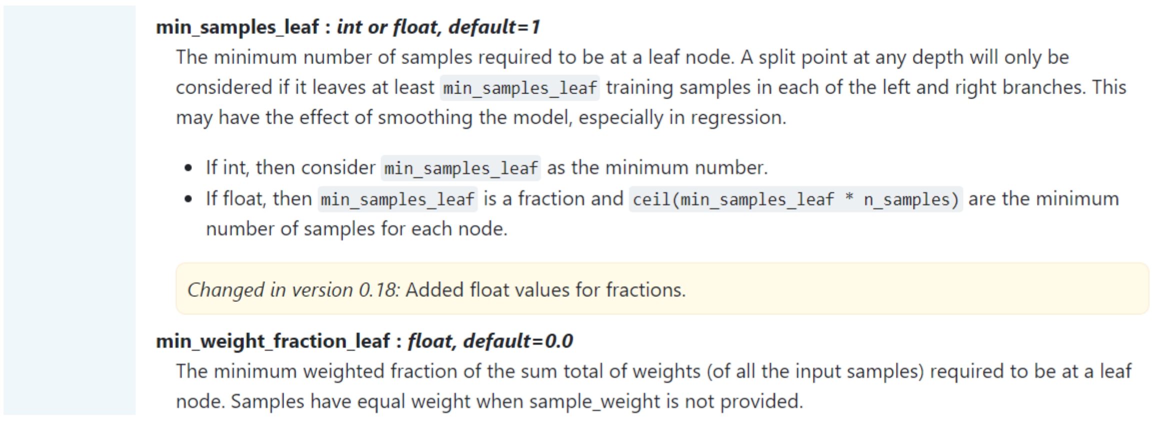
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*

The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.



min_samples_leaf : *int or float, default=1*

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

Changed in version 0.18: Added float values for fractions.

min_weight_fraction_leaf : *float, default=0.0*

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

max_features : *{"sqrt", "log2", None}, int or float, default="sqrt"*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `max(1, int(max_features * n_features_in_))` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)`.
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Changed in version 1.1: The default of `max_features` changed from `"auto"` to `"sqrt"`.

Deprecated since version 1.1: The `"auto"` option was deprecated in 1.1 and will be removed in 1.3.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

max_leaf_nodes : *int, default=None*

Grow trees with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

min_impurity_decrease : *float, default=0.0*

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where `N` is the total number of samples, `N_t` is the number of samples at the current node, `N_t_L` is the number of samples in the left child, and `N_t_R` is the number of samples in the right child.

`N`, `N_t`, `N_t_R` and `N_t_L` all refer to the weighted sum, if `sample_weight` is passed.

New in version 0.19.

bootstrap : *bool, default=True*

Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

oob_score : *bool, default=False*

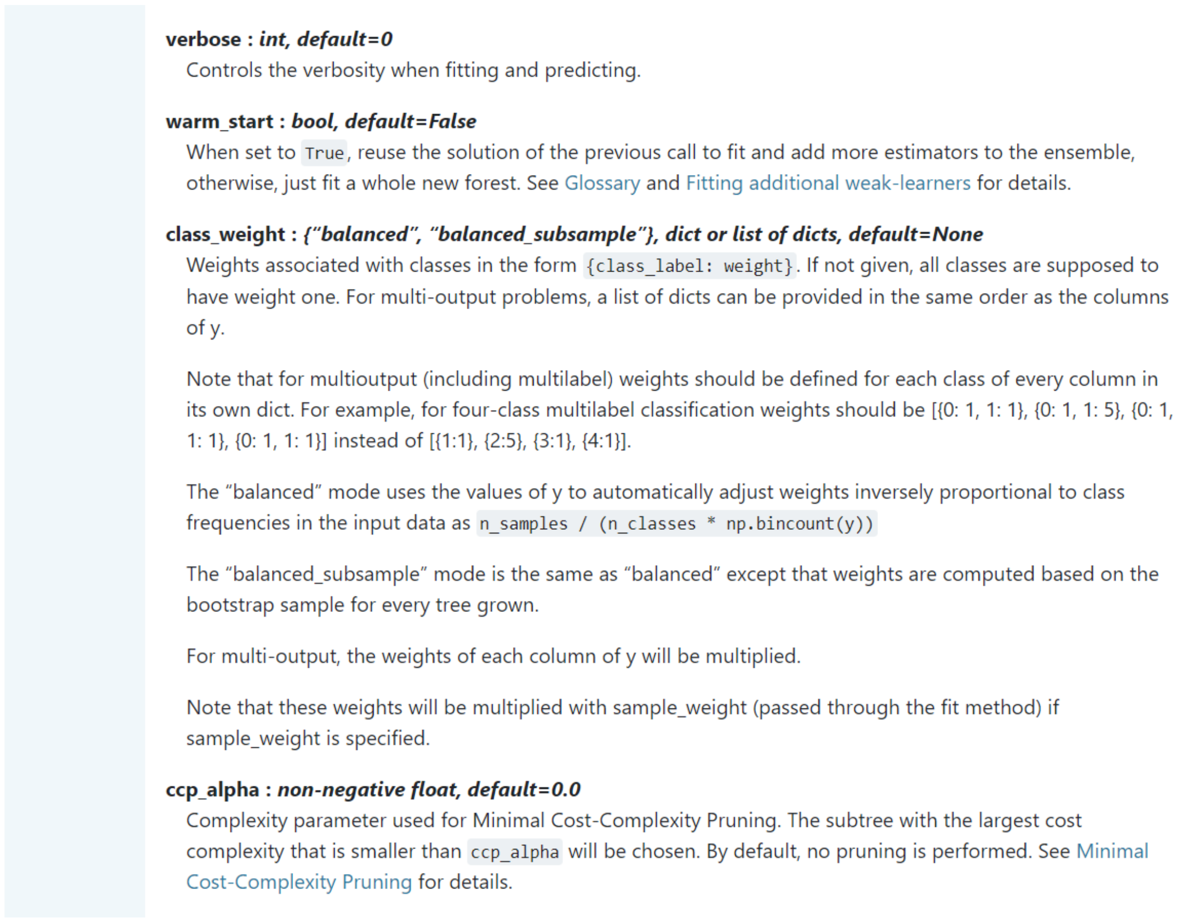
Whether to use out-of-bag samples to estimate the generalization score. Only available if bootstrap=True.

n_jobs : *int, default=None*

The number of jobs to run in parallel. `fit`, `predict`, `decision_path` and `apply` are all parallelized over the trees. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

random_state : *int, RandomState instance or None, default=None*

Controls both the randomness of the bootstrapping of the samples used when building trees (if `bootstrap=True`) and the sampling of the features to consider when looking for the best split at each node (if `max_features < n_features`). See Glossary for details.



verbose : *int, default=0*

Controls the verbosity when fitting and predicting.

warm_start : *bool, default=False*

When set to `True`, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest. See Glossary and Fitting additional weak-learners for details.

class_weight : *{"balanced", "balanced_subsample"}, dict or list of dicts, default=None*

Weights associated with classes in the form `{class_label: weight}`. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

Note that for multioutput (including multilabel) weights should be defined for each class of every column in its own dict. For example, for four-class multilabel classification weights should be [{0: 1, 1: 1}, {0: 1, 1: 5}, {0: 1, 1: 1}, {0: 1, 1: 1}] instead of [{1:1}, {2:5}, {3:1}, {4:1}].

The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as `n_samples / (n_classes * np.bincount(y))`

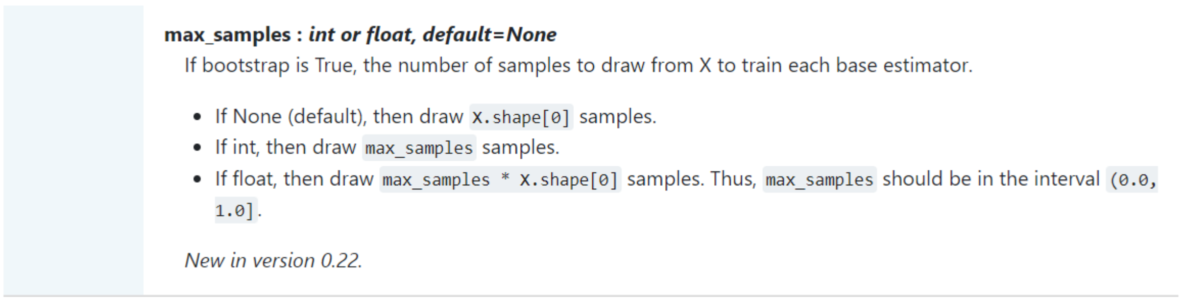
The "balanced_subsample" mode is the same as "balanced" except that weights are computed based on the bootstrap sample for every tree grown.

For multi-output, the weights of each column of y will be multiplied.

Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

ccp_alpha : *non-negative float, default=0.0*

Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than `ccp_alpha` will be chosen. By default, no pruning is performed. See Minimal Cost-Complexity Pruning for details.



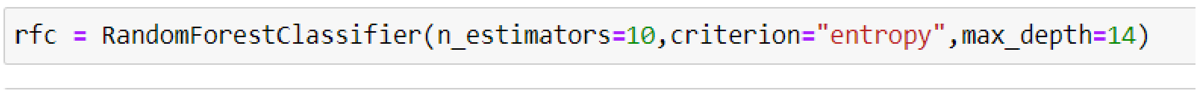
max_samples : *int or float, default=None*

If bootstrap is True, the number of samples to draw from X to train each base estimator.

- If None (default), then draw `X.shape[0]` samples.
- If int, then draw `max_samples` samples.
- If float, then draw `max_samples * X.shape[0]` samples. Thus, `max_samples` should be in the interval `(0.0, 1.0]`.

New in version 0.22.

The parameters that we have used are :-

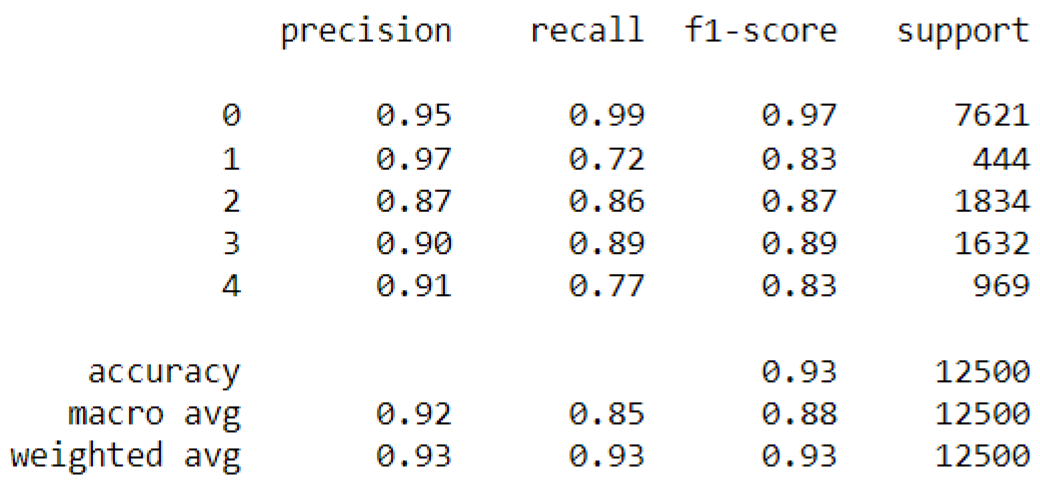


```
rfc = RandomForestClassifier(n_estimators=10,criterion="entropy",max_depth=14)
```

Keeping other parameters as default.

The {n_estimators=10} limits the number of decision trees to 10. This optimizes the size and performance of the random forest. The {criterion="entropy"} sets the criteria for splitting in each decision tree as weighted entropy. The {max_depth=14} sets the maximum depth of each decision tree as 14, thus preventing any decision tree from underfitting or overfitting.

Classification Report



```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      7621
           1       0.97      0.72      0.83       444
           2       0.87      0.86      0.87      1834
           3       0.90      0.89      0.89      1632
           4       0.91      0.77      0.83       969

    accuracy                           0.93     12500
   macro avg       0.92      0.85      0.88     12500
weighted avg       0.93      0.93      0.93     12500
```

Github Repository

https://github.com/Jibitesh-Chakraborty2811/Sleep-Stage-Random-Forest-Classifier