SUPPLEMENTARY METHODS

EXTRACTING BULK DEFECT PARAMETERS IN SILICON WAFERS USING MACHINE LEARNING MODELS

Yoann Buratti, Quoc Le Gia, Josef Dick, Yan Zhu and Ziv Hameiri

MACHINE LEARNING HYPER-PARAMETERS - REGRESSION

```
sklearn.ensemble.RandomForestRegressor(
        bootstrap=True,
        ccp_alpha=0.0,
        criterion='mse',
        max_depth=None,
        max_features='auto',
        max_leaf_nodes=None,
        max_samples=None,
        min_impurity_decrease=0.0,
        min_impurity_split=None,
        min_samples_leaf=1,
        min_samples_split=2,
        min_weight_fraction_leaf=0.0,
        n_estimators=100, n_jobs=-1,
        oob_score=False,
        random_state=None,
        verbose=2,
        warm_start=False
)
sklearn.ensemble.AdaBoostRegressor(
        base_estimator=DecisionTreeRegressor(
                ccp_alpha=0.0,
                criterion='mse',
                max_depth=None,
                max_features=None,
                max_leaf_nodes=None,
                min_impurity_decrease=0.0,
                min_impurity_split=None,
                min_samples_leaf=1,
                min_samples_split=2,
                min_weight_fraction_leaf=0.0,
                presort='deprecated',
                andom_state=None,
                splitter='best'),
        learning_rate=1.0,
        loss='linear',
        n_estimators=100,
        random_state=None
)
sklearn.ensemble.GradientBoostingRegressor(
        alpha=0.9,
        ccp_alpha=0.0,
        criterion='friedman_mse',
        init=None,
        learning_rate=0.1,
        loss='ls',
        max_depth=10,
        max_features=None,
        max_leaf_nodes=None,
        min_impurity_decrease=0.0,
```

```
            min_impurity_split=None,
            min_samples_leaf=1,
            min_samples_split=2,
            min_weight_fraction_leaf=0.0,
            n_estimators=100,
            n_iter_no_change=None,
            presort='deprecated',
            random_state=None,
            subsample=1.0,
            tol=0.0001,
            validation_fraction=0.1,
            verbose=2,
            warm_start=False
    )
    sklearn.neural_network.MLPRegressor(
            activation='relu',
            alpha=0.001,
            batch_size='auto',
            beta_1=0.9,
            beta_2=0.999,
            early_stopping=False,
            epsilon=1e-08,
            hidden_layer_sizes=(100, 100),
            learning_rate='adaptive',
            learning_rate_init=0.001,
            max_fun=15000,
            max_iter=200,
            momentum=0.9,
            n_iter_no_change=10,
            nesterovs_momentum=True,
            power_t=0.5,
            random_state=None,
            shuffle=True,
            solver='adam',
            tol=0.0001,
            validation_fraction=0.1,
            verbose=2,
            warm_start=False
    )
    sklearn.svm.SVR(
            C=5,
            cache_size=200,
            coef0=0.0,
            degree=3,
            epsilon=0.1,
            gamma='auto',
            kernel='rbf',
            max_iter=-1,
            shrinking=True,
            tol=0.001,
            verbose=2
    )
```

MACHINE LEARNING HYPER-PARAMETERS - CLASSIFICATION

```
sklearn.ensemble.RandomForestClassifier(
        bootstrap=True,
        ccp_alpha=0.0,
        class_weight=None,
        criterion='gini',
        max_depth=None,
        max_features='auto',
        max_leaf_nodes=None,
        max_samples=None,
        min_impurity_decrease=0.0,
        min_impurity_split=None,
        min_samples_leaf=1,
        min_samples_split=2,
        min_weight_fraction_leaf=0.0,
        n_estimators=100,
        n_jobs=-1,
        oob_score=False,
        random_state=None,
        verbose=2,
        warm_start=False
)
sklearn.ensemble.AdaBoostClassifier(
        algorithm='SAMME.R',
        base_estimator=DecisionTreeClassifier(
                ccp_alpha=0.0,
                class_weight=None,
                criterion='gini',
                max_depth=None,
                max_features=None,
                max_leaf_nodes=None,
                min_impurity_decrease=0.0,
                min_impurity_split=None,
                min_samples_leaf=1,
                min_samples_split=2,
                min_weight_fraction_leaf=0.0,
                presort='deprecated',
                random_state=None,
                splitter='best'
                ),
        learning_rate=1.0,
        n_estimators=10,
        random_state=None
)
sklearn.ensemble.GradientBoostingClassifier(
        ccp_alpha=0.0,
        criterion='friedman_mse',
        init=None,
        learning_rate=0.1,
        loss='deviance',
        max_depth=3,
        max_features=None,
        max_leaf_nodes=None,
        min_impurity_decrease=0.0,
        min_impurity_split=None,
        min_samples_leaf=1,
        min_samples_split=2,
        min_weight_fraction_leaf=0.0,
        n_estimators=100,
        n_iter_no_change=None,
```

```
            presort='deprecated',
            random_state=None,
            subsample=1.0,
            tol=0.0001,
            validation_fraction=0.1,
            verbose=2,
            warm_start=False
    )
    sklearn.neural_network.MLPClassifier(
            activation='relu',
            alpha=0.001,
            batch_size='auto',
            beta_1=0.9,
            beta_2=0.999,
            early_stopping=False,
            epsilon=1e-08,
            hidden_layer_sizes=(100, 100),
            learning_rate='adaptive',
            learning_rate_init=0.001,
            max_fun=15000,
            max_iter=200,
            momentum=0.9,
            n_iter_no_change=10,
            nesterovs_momentum=True,
            power_t=0.5,
            random_state=None,
            shuffle=True,
            solver='adam',
            tol=0.0001,
            validation_fraction=0.1,
            verbose=2,
            warm_start=False
    )
    sklearn.neighbors.KNeighborsClassifier(
            algorithm='auto',
            leaf_size=30,
            metric='minkowski',
            metric_params=None,
            n_jobs=-1,
            n_neighbors=5,
            p=2,
            weights='distance'
    )
```