Intel® Extension for Scikit-learn Random Forest for Yolanda dataset

```
from timeit import default_timer as timer
from sklearn import metrics
from IPython.display import HTML
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
```

Download the data

```
x, y = fetch_openml(name="Yolanda", return_X_y=True)
```

Split the data into train and test sets

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=72)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((280000, 100), (120000, 100), (280000,), (120000,))
```

Patch original Scikit-learn with Intel® Extension for Scikit-learn

Intel® Extension for Scikit-learn (previously known as daal4py) contains drop-in replacement functionality for the stock Scikit-learn package. You can take advantage of the performance optimizations of Intel® Extension for Scikit-learn by adding just two lines of code before the usual Scikit-learn imports:

```
from sklearnex import patch_sklearn

patch_sklearn()
```

```
Intel(R) Extension for Scikit-learn* enabled (https://github.com/intel/scikit-learn-intelex)
```

Intel® Extension for Scikit-learn patching affects performance of specific Scikit-learn functionality. Refer to the list of supported algorithms and parameters for details. In cases when unsupported parameters are used, the package fallbacks into original Scikit-learn. If the patching does not cover your scenarios, submit an issue on GitHub.

Training Random Forest algorithm with Intel® Extension for Scikit-learn for Yolanda dataset

```
from sklearn.ensemble import RandomForestRegressor

params = {"n_estimators": 150, "random_state": 44, "n_jobs": -1}
start = timer()
rf = RandomForestRegressor(**params).fit(x_train, y_train)
train_patched = timer() - start
f"Intel® extension for Scikit-learn time: {train_patched:.2f} s"
```

```
'Intel® extension for Scikit-learn time: 42.56 s'
```

Predict and get a result of the Random Forest algorithm with Intel® Extension for Scikit-learn

```
y_pred = rf.predict(x_test)
mse_opt = metrics.mean_squared_error(y_test, y_pred)
f"Intel® extension for Scikit-learn Mean Squared Error: {mse_opt}"
```

```
'Intel® extension for Scikit-learn Mean Squared Error: 83.62232345666878'
```

Train the same algorithm with original Scikit-learn

In order to cancel optimizations, we use *unpatch_sklearn* and reimport the class RandomForestRegressor.

```
from sklearnex import unpatch_sklearn

unpatch_sklearn()
```

Training Random Forest algorithm with original Scikit-learn library for Yolanda dataset

```
from sklearn.ensemble import RandomForestRegressor

start = timer()
rf = RandomForestRegressor(**params).fit(x_train, y_train)
train_unpatched = timer() - start
f"Original Scikit-learn time: {train_unpatched:.2f} s"
```

```
'Original Scikit-learn time: 123.34 s'
```

Predict and get a result of the Random Forest algorithm with original Scikit-learn

```
y_pred = rf.predict(x_test)
mse_original = metrics.mean_squared_error(y_test, y_pred)
f"Original Scikit-learn Mean Squared Error: {mse_opt}"
```

```
'Original Scikit-learn Mean Squared Error: 83.62232345666878'
```

```
HTML(
    f"<h3>Compare MSE metric of patched Scikit-learn and original</h3>"
    f"MSE metric of patched Scikit-learn: {mse_opt} <br>"
    f"MSE metric of unpatched Scikit-learn: {mse_original} <br>"
    f"Metrics ratio: {mse_opt/mse_original} <br>"
    f"<h3>With Scikit-learn-intelex patching you can:</h3>"
    f"<ul>"
    f"<li>Use your Scikit-learn code for training and prediction with minimal changes (a couple of lines of code);</li>"
    f"<li>Fast execution training and prediction of Scikit-learn models;</li>"
    f"<li>Get the similar quality</li>"
    f"<li>Get speedup in <strong>{(train_unpatched/train_patched):.1f}</strong> times.</li>"
    f"</ul>"
)
```

Compare MSE metric of patched Scikit-learn and original

MSE metric of patched Scikit-learn: 83.62232345666878
MSE metric of unpatched Scikit-learn: 83.80131297814816
Metrics ratio: 0.9978641203208111

With Scikit-learn-intelex patching you can:

- Use your Scikit-learn code for training and prediction with minimal changes (a couple of lines of code);
- Fast execution training and prediction of Scikit-learn models;
- Get the similar quality
- Get speedup in **2.9** times.