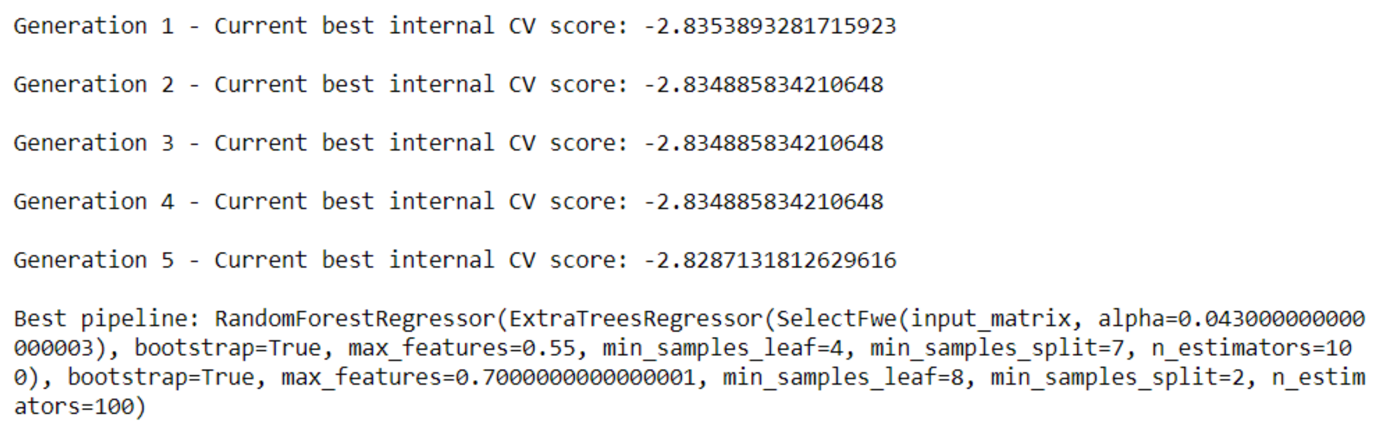
## How to use TPOT to discover top-performing models for regression tasks.

Let’s use TPOT to find a good model for the housing dataset.We will use a population size of 50 for 5 generations for the search and use all cores on the system by setting “*n_jobs*” to -1,scoring to 'neg_mean_absolute_error’ and verbosity to 2.

```
from sklearn.datasets import load_boston

from tpot import TPOTRegressor

# load dataset

X, y = load_boston(return_X_y=True)

# define search

model = TPOTRegressor(generations=5, population_size=50,
scoring='neg_mean_absolute_error', verbosity=2, random_state=1,
n_jobs=-1)

# perform the search

model.fit(X, y)

# export the best model

model.export('tpot_best_regression_model.py')
```

The accuracy of top-performing models will be reported along the way.



```
Generation 1 - Current best internal CV score: -2.8353893281715923

Generation 2 - Current best internal CV score: -2.834885834210648

Generation 3 - Current best internal CV score: -2.834885834210648

Generation 4 - Current best internal CV score: -2.834885834210648

Generation 5 - Current best internal CV score: -2.8287131812629616

Best pipeline: RandomForestRegressor(ExtraTreesRegressor(SelectFwe(input_matrix, alpha=0.043000000000
000003), bootstrap=True, max_features=0.55, min_samples_leaf=4, min_samples_split=7, n_estimators=10
0), bootstrap=True, max_features=0.700000000000001, min_samples_leaf=8, min_samples_split=2, n_estim
ators=100)
```

Running the model may take a few minutes, and you will see a progress bar.

In this case, we can see that the top-performing pipeline achieved the mean accuracy of about -2.827 (`neg_mean_absolute_error`). This is a best performing model.The top-performing model saved to a file named “tpot_best_regression_model.py“.

By opening this file, you can see that there is some generic code for loading a dataset and fitting the pipeline.

```
import numpy as np

import pandas as pd

from sklearn.ensemble import ExtraTreesRegressor,
RandomForestRegressor

from sklearn.feature_selection import SelectFwe, f_regression

from sklearn.model_selection import train_test_split

from sklearn.pipeline import make_pipeline, make_union

from tpot.builtins import StackingEstimator

from tpot.export_utils import set_param_recursive



# NOTE: Make sure that the outcome column is labeled 'target'
in the data file

tpot_data = pd.read_csv('PATH/TO/DATA/FILE',
sep='COLUMN_SEPARATOR', dtype=np.float64)

features = tpot_data.drop('target', axis=1)

training_features, testing_features, training_target,
testing_target = \

            train_test_split(features, tpot_data['target'],
random_state=1)
```

```
# Average CV score on the training set was: -2.8287131812629616

exported_pipeline =
make_pipeline(SelectFwe(score_func=f_regression,
alpha=0.043000000000000003),StackingEstimator(estimator=ExtraTr
eesRegressor(bootstrap=True, max_features=0.55,
min_samples_leaf=4, min_samples_split=7, n_estimators=100)),
RandomForestRegressor(bootstrap=True,
max_features=0.7000000000000001, min_samples_leaf=8,
min_samples_split=2, n_estimators=100))

# Fix random state for all the steps in exported pipeline

set_param_recursive(exported_pipeline.steps, 'random_state', 1)

exported_pipeline.fit(training_features, training_target)

results = exported_pipeline.predict(testing_features)
```

We can use this code to fit a final model on all available data and make a prediction for new data.