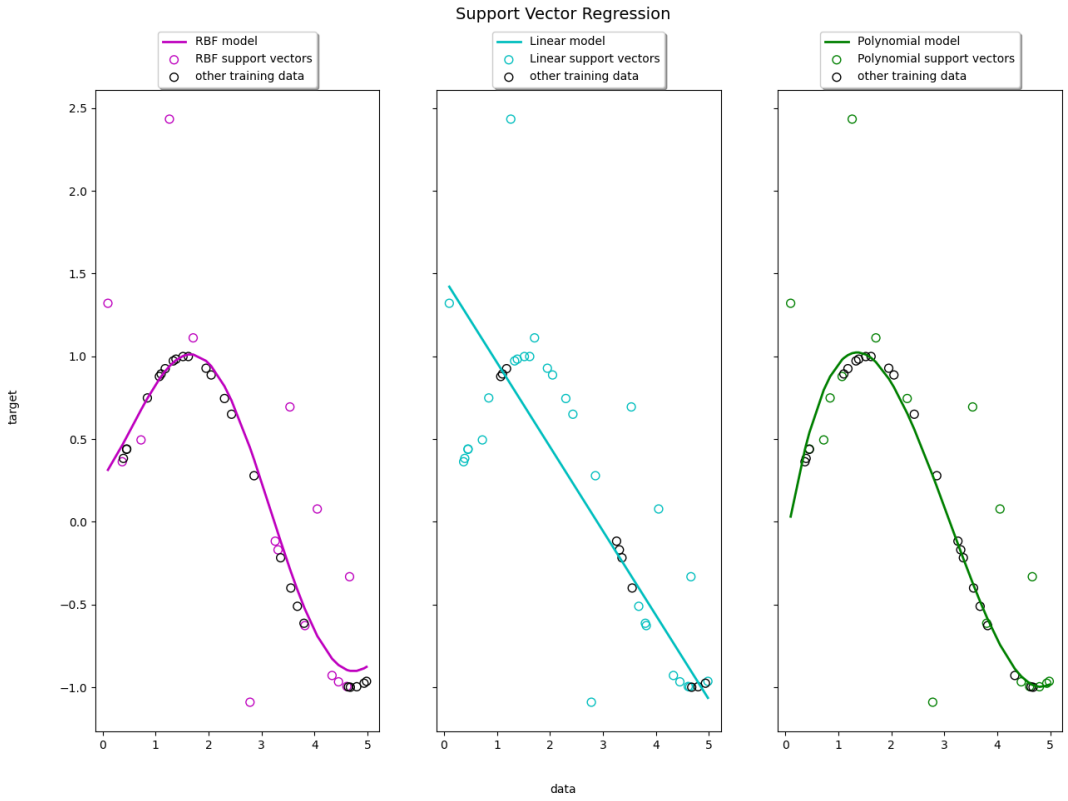
SKLEARN SVM SVR SHORT WRITE UP

Anshul Savla

J053

1)SVR



R-Support Vector Regression.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to datasets with more than a couple of 10000 samples. For large datasets consider
using `LinearSVR` or `SGDRegressor` instead, possibly after a `Nystroem` transformer.

Complete thing remain same for svr code, attribute , parameter with some minor changes.

Code :-

class `sklearn.svm.`**SVR**(**, kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False, max_iter=-1*)

Important hyperparameters are :-

kernel*{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'*
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to precompute the kernel matrix.

degree*int, default=3*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma*{'scale', 'auto'} or float, default='scale'*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.
-

Important Attributes are : -

class_weight_*ndarray of shape (n_classes,)*
Multipliers of parameter C for each class. Computed based on the `class_weight` parameter.

coef_*ndarray of shape (1, n_features)*
Weights assigned to the features (coefficients in the primal problem). This is only available in the case of a linear kernel.

`coef_` is readonly property derived from `dual_coef_` and `support_vectors_`.

dual_coef_*ndarray of shape (1, n_SV)*
Coefficients of the support vector in the decision function.

fit_status_*int*
0 if correctly fitted, 1 otherwise (will raise warning)

intercept_*ndarray of shape (1,)*
Constants in decision function.

n_support_*ndarray of shape (n_classes,), dtype=int32*
Number of support vectors for each class.

shape_fit_*tuple of int of shape (n_dimensions_of_X,)*

Array dimensions of training vector X.

support_*ndarray of shape (n_SV,)*
Indices of support vectors.

support_vectors_*ndarray of shape (n_SV, n_features)*
Support vectors.

Application :-

```
>>> from sklearn.svm import SVR
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import StandardScaler
>>> import numpy as np
>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.randn(n_samples)
>>> X = rng.randn(n_samples, n_features)
>>> regr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
>>> regr.fit(X, y)
```

Methods : -

Fit(X, y)- fit the linear model.

Predict(X)-predict using linear model.

Score(X,y)-returns the coefficient of determination R^2 of the prediction.