Name: - Aditya Gavankar

Roll no: - J072

Topic: - Machine Learning **Assignment**: - 6

Sci-kit learn API – Support Vector Regression (SVR)

- **Support Vector Regression (SVR)** is a supervised learning algorithm that is used to predict discrete values.
- The basic idea behind SVR is to find the best fit line, it is the hyperplane that has the maximum number of points.
- Unlike other Regression models that try to minimize the error between the real and predicted value, the SVR tries to fit the best line within a threshold value, which is the distance between the hyperplane and boundary line.
- In SVR, the free parameters in the model are C and epsilon.
- 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.

Code:-

*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)*

Parameters:-

- **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.

- **coef0(*float), default=0.0***
  Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

- **tol(*float), default=1e-3***
  Tolerance for stopping criterion.

- **C(*float), default=1.0***
  Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

- **epsilon(*float), default=0.1***
  Epsilon in the epsilon-SVR model. It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.

- **shrinking(*bool), default=True***
  Whether to use the shrinking heuristic.
- **cache_size(*float), default=200***
  Specify the size of the kernel cache (in MB).

- **verbose(*bool), default=False***
  Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.

- **max_iter(*int), default=-1***
  Hard limit on iterations within solver, or -1 for no limit.

Attributes:-

- **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.

Methods:-

fit(*X*, *y*, *sample_weight=None*)	Fit the SVM model according to the given training data.
get_params(*deep=True*)	Get parameters for this estimator.
predict(X)	Perform regression on samples in X.
score(*X*, *y*, *sample_weight=None*)	Return the coefficient of determination R2 of the prediction.
set_params(**params)	Set the parameters of this estimator.