

# DS203 ASSIGNMENT 7

## BASIC MACHINE LEARNING

Q1)

1. PCA
2. Clustering
3. Regression
4. Classification

Q2)

### **1. Clustering with SVM-C:**

- Target output variable - Integer (class labels)
- Parameters - Training vectors X, training labels y, sample\_weight
- Hyperparameters - Regularization parameter C (default 1.0), kernel (default 'rbf'), degree (for polynomial kernel and default value 3), gamma (default 'scale'), tol (default 0.001), verbose (default False), max\_iter (default -1), decision\_function\_shape (default 'ovr'), break\_ties (default False), random\_state (default None)
- Define: `svc = SVC()`

Training: `svc.fit(X_train, Y_train)`

Testing: `Y_pred = svc.predict(X_test)`

## **2. Regression with SVM-R:**

- Target output variable - Floating type
- Parameters - Training vectors X, training labels y, sample\_weight
- Hyperparameters - Regularization parameter C (default 1.0), kernel (default 'rbf'), degree (for polynomial kernel and default value 3), gamma (default 'scale'), tol (default 0.001), verbose (default False), max\_iter (default -1), epsilon (default 0.1)
- Define: `svr = SVR()`

Training: `svr.fit(X_train, Y_train)`

Testing: `Y_pred = svr.predict(X_test)`

## **3. NN with one hidden layer (Classification):**

- Target output variable - One-hot encoded
- Parameters - Training vectors X, training labels y
- Hyperparameters - Hidden layer size (around 100), activation function (default relu), learning rate (default 0.0001), optimization algorithm (default adam)
- Define: `clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)`

Training: `clf.fit(X_train, Y_train)`

Testing: `Y_pred = clf.predict(X_test)`

## **4. NN with one hidden layer (Regression):**

- Target output variable - Floating type
- Parameters - Training vectors X, training labels y

- Hyperparameters - Hidden layer size (around 100), activation function(default relu), learning rate (default 0.0001), optimization algorithm (default adam)

- Define: `reg = MLPRegressor(random_state=1, max_iter=500)`

Training: `reg.fit(X_train, Y_train)`

Testing: `Y_pred = reg.predict(X_test)`

## **5. Random Forest (Classification):**

- Target output variable - Integer

- Parameters - Training vectors X, training labels y, sample\_weight

- Hyperparameters - n\_estimators (default=100), criterion (default = 'gini'), max\_depth (default None), max\_features (default 'auto'), verbose (default False), random\_state (default None)

- Define: `clf = RandomForestClassifier(max_depth=2, random_state=0)`

Training: `clf.fit(X_train, Y_train)`

Testing: `clf.predict(X_test)`

## **6. Random Forest (Regression):**

- Target output variable - Floating Point

- Parameters - Training vector X, training labels y, sample\_weight

- Hyperparameters - n\_estimators (default=100), criterion (default = 'mse'), max\_depth (default None),

max\_features (default 'auto'), verbose (default False), random\_state (default None)

- Define: `reg = RandomForestRegressor(max_depth=2, random_state=0)`

Training: `reg.fit(X_train, Y_train)`

Testing: `reg.predict(X_test)`

## **7. K-means (Clustering):**

- Target output variable - Integer
- Parameters - Training vector X, sample\_weights
- Hyperparameters - eps (default 0.5), min\_samples (default 5), metric (default 'euclidean'), metric\_params (default None), algorithm (default 'auto')
- Define: `kmeans = KMeans(n_clusters=2, random_state=0)`

Training: `kmeans.fit(X_train)`

Testing: `kmeans.predict(X_test)`

## **8. DBSCAN (Clustering):**

- Target output variable - Integer
- Parameters - Training vector X, sample\_weights
- Hyperparameters - eps (default 0.5), min\_samples (default 5), metric (default 'euclidean'), metric\_params (default None), algorithm (default 'auto')
- Define: `clustering = DBSCAN(eps=3, min_samples=2)`

Training: `clustering.fit(X_train)`

Testing: `clustering.predict(X_test)`

## 9. PCA (Dimension Reduction):

- Target output variable - Input reduced to a lower dimension
- Parameters - Training vector X
- Hyperparameters - n\_components (default None), copy (default True), whiten (default False), svd\_solver (default 'auto'), tol (default 0.0), iterated\_power (default 'auto'), random\_state (default None)
- Define: `pca = PCA(n_components=2)`

Fit: `pca.fit(X)`

Transform: `X_new = pca.transform(X)`

## 10. Kernel PCA (Dimension Reduction):

- Target output variable - Input reduced to a lower dimension
- Parameters - Training vector X
- Hyperparameters - n\_components (default None), kernel (default 'linear'), gamma (default None), degree (default 3), coef0 (default 1), kernel\_params (default None), alpha (default 1.0), eigen\_solver (default 'auto'), tol (default 0.0), random\_state (default None)
- Define: `kernel_pca = KernelPCA(n_components=7, kernel='linear')`

Fit: `kernel_pca.fit(X)`

Transform: `X_new = kernel_pca.transform(X)`