Date __/__/__
Page _____

— : LAB - 06 : —

Random Forest Algorithm :—

For each tree t=1 to T,
Draw a boot strap sample D_t of size n from training data D (sam with replacement)
Train a decision tree h_i on D_{ti} at each node.
Randomly select m features from the full feature set.
Choose the best feature & split point among m features using Impurity, Entropy or MSE (for regression)
Split the node into child nodes.

Aggregate the predictions:

For classification: Use majority voting over all trees:

$$\hat{y} = \text{mode}(h_1(x), h_2(x), \ldots h_T(x))$$

For regression: Use mean prediction:

$$\hat{y} = \frac{1}{T} \sum_{t=1} h_t(x)$$

#Code

```
class import Random Forest:
    def init (self, n_trees =10, max_feature
                sqrt'):
    self.n_trees n_trees
    self.max_features = max_features
    self.trees = []
    def boost strap_sample (self. x, y):
    n_sample = x.slope[0]
```

Date __/__/__
Page _____

```
size = n_sample, replace = True
return x[indexs], y[indexs]
def _get_max_features(self, n_features):
    if self.max_features == 'sqrt':
        return int(np.sqrt(n_features))
    elif isinstance(self.max_features, int):
        return self.max_features
def fit(self, X, y):
    self.tree = []
    n_features = X.shape[1]
    max_feature = self._get_max_features
```

⭐ => K-Means Algorithm:-

01. Initialize centroids:
Randomly choose k data pointer from x as initial cluster centroids:

$$\mu_1, \mu_2, \ldots \mu_k.$$

02. Repeat until convergence.
@ Repeat each data point to the nearest centroid. For each point x_i, find the closet centroid μ_j based on distance

$$\mu_j = \frac{1}{|C_i|} \sum_{x_i \in C_i} x_i$$

03. check for convergence
if cluster assignments don't change
centroids don't move significantly, the stop

Date / /
Page

Code:-

```
import matplotlib.pyplot as plt
x, _ = make_blobs(n_sample=300,
centers=3, random_state=42
model = K.means(n_clusters=3)
  model.fit(x)
plt.scatter(x[:,0], x[:,1]
  c=model.labels_, cmap='viridis')
  plt.show()
```

Principle component Model Algorithm.

(1) Standardize the data
PCA is affected by scale of features.
∴ data should be standardized

$$x_j = \frac{x_j - \mu_j}{\sigma_j}$$

(2) Compute the covariance matrix.

$$\Sigma = \frac{1}{n-1} x^T x$$

(03) Compute the sign value and eigen value of covariance matrix.

$$\Sigma v = \lambda v$$

(4) Sort the eigen values & vectors in descending order.

Date __/__/__
Page _____

(5) select the top i.e eigen vectors

(6) project the data onto new space

(7) Output the transformed data

Code

```
data = load_iris()
x = data.data
pca = PCA(n_components = 2)
x_pca = Pca.fit_transform(x)

xPca = Pca.fit_transform(x)
plt.Scatter(x_Pca[:,0], x_pca[:],
  c = data.target)
print('Explained variance by each
    component: {3}.
```