Lab-5

① Random forest ensemble learning

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():
    df = pd.read_csv(filename)
    print (f"Data loaded from: {filename}")
    display (df.head())

x = df.iloc[:, :-1]
y = df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.2, random_state =42)

rf_model = RandomForestClassifier (n_estimators =100, random_state = 42)
rf_model.fit (X_train, y_train)

y_pred = rf_model.predict (X_test)

accuracy = accuracy_score (y_test, y_pred)
print (f"Accuracy of Random Forest Model : {accuracy *100 : .2f}%")

print ("Classification report :")
print (classification_report (y_test, y_pred))
```

Output :-

	precision	recall	F1-score	support
0	0.79	0.78	0.78	99
1	0.61	0.62	0.61	55

(2) Boosting algorithm

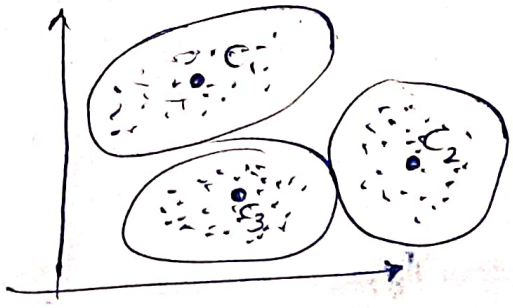
```
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score

class AdaBoost:
    def __init__(self, n_estimators=50):
        self.n_estimators = n_estimators
        self.alphas = []
        self.models = []

    def fit(self, X, y):
        n_samples, n_features = X.shape
        w = np.ones(n_samples) / n_samples

        for estimator in range(self.n_estimators):
            model = DecisionTreeClassifier(max_depth=1)
            model.fit(X, y, sample_weight=w)
            y_pred = model.predict(X)

            err = np.sum(w * (y_pred != y)) / np.sum(w)

            alpha = 0.5 * np.log((1-err)/err) if err < 1 else 0
            self.alphas.append(alpha)
            self.models.append(model)

            w = w * np.exp(-alpha * y * y_pred)
            w = w / np.sum(w)

    def predict(self, X):
        final_pred = np.zeros(X.shape[0])
        for model, alpha in zip(self.models, self.alphas):
            final_pred += alpha * model.predict(X)
        return np.sign(final_pred)

    def score(self, X, y):
        return accuracy_score(y, self.predict(X))
```

```
X, y = make_classification (n_samples = 500, n_features = 20, n_classes = 2,
                            random_state = 42)

y = 2 * y - 1
adaboost = AdaBoost (n_estimators = 50)
adaboost.fit (X, y)

accuracy = adaboost.score (X, y)
print (f "Model accuracy : {accuracy : .4f}")
```

Output: Model Accuracy : 0.9580

③ K means

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import seaborn as sns
from sklearn.cluster import KMeans

iris = datasets.load_iris ()
print ("Dataset loaded successfully")

Data = pd.DataFrame (iris.data, columns = iris.feature_names)

x = Data.iloc [:, 0:3].values

ces = []

kmeans = KMeans (n_clusters = 3, init = 'k-means++', max_iter = 100,
                 n_init = 10, random_state = 0)

y_kmeans = kmeans.fit_predict(x)
kmeans.cluster_centers_

plt.scatter (x [y_kmeans == 0, 0], x [y_kmeans == 0, 1], s = 100, c = 'red',
             label : 'Iris-Setosa')

plt.scatter (k_means.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
             s=100, c='black', label: 'Centroids')
plt.legend()
```



④ PCA

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from google.colab import files

uploaded = files.upload()
for filename in uploaded.keys():
    df = pd.read_csv(filename)
    print(f"Uploaded : {filename}")
    display(df.head())

numeric_df = df.select_dtypes(include=[np.number])
print("Numerical features found:", list(numeric_df.columns))

selected_features = numeric_df.columns
X = numeric_df[selected_features].dropna()
X_scaled = StandardScaler().fit_transform(X)

pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_scaled)

pca_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

plt.figure(figsize=(8,6))
plt.grid(True)
plt.show()
print("Explained Variance Ratio:", pca.explained_variance_ratio_)
print(f"Model accuracy : {accuracy:.4f}")
```

Sucheta B
17/5/25

O/P:-

```
Variance Ration : [0.52163044 0.98631263]
Model Accuracy: 0.9580
```