21/04/25

Lab 5

Random Forest

```
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.uplod()

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

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(accuracy)
print(y_test, y_pred.
```

Output

Accuracy: 72.08%

```
Classification Report:
                precision   recall   f1-score   support
0                 0.79       0.78      0.78       99
1                 0.61       0.62      0.61       55
accuracy                               0.72      154
macro avg         0.7        0.7       0.7       154
weighted avg      0.72       0.72      0.72      154
```

AdaBoost

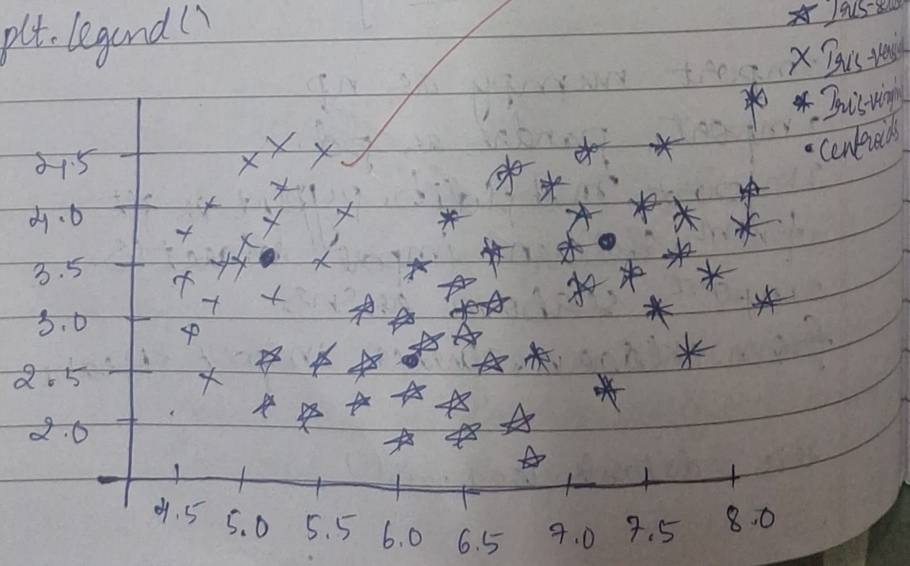
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassif
from sklearn.datasets import make_classification
from sklearn.decompission import PCA

sns.set(style="whitegrid")

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

    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
            model.fit(X, y, sample_weight=w
            y_pred = model.predict(X)

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

            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_predict)
```

Model accuracy: 0.898

K-means clustering

```
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()
```

```
Data = pd. DataFrame (iris.data, columns = iris.featu
                                         names)

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

css = [].

kmeans = KMeans (n_clusters = 3, init = 'k-means++', max_iter
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 (x[y_kmeans == 1, 0], x[y_kmeans == 1, 1]
              s = 100 , c = 'blue', label = 'Iris-versicolour

plt.scatter (x[y_kmeans == 2, 0], x [y_kmeans == 2, 1]
              s = 100, c = 'green', label = 'Iris-virginica

plt.scatter (kmeans. cluster_centers [:, 0], kmeans.cluster
              centers_[:, 1], s = 100, c = 'black', label = 'Centr

plt.legend()
```



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

uploded = files.upload()

for filename in uploded.keys():
    df = pd.read_csv(filename)

numeric_df = df.select_dtypes(include=[np.number])

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'])

print("Variance Ratio:", pca.explained_variance_ratio)

print(f"Accuracy: {accuracy:.4f}")
```

Output

```
Variance ratio: [0.5216    0.2863]
Accuracy: 0.898
```

Sneha SB 5/5/25