LAB-04)

7/4/25

PAGE NO :
DATE :

KNN :-

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target

np.random.seed(42)
indices = np.random.permutation(len(X))
split = int(0.8 * len(X))
train_idx, test_idx = indices[:split], indices
                      [split:]

X_train, X_test = X[train_idx], X[test_idx]
y_train, y_test = y[train_idx], y[test_idx]

mean = X_train.mean(axis=0)
std = X_train.std(axis=0)
X_train_scaled = (X_train, ~~X_train~~ mean) / std
X_test_scaled = (X_test - mean) / std

def euclidean_distance(a, b):
    return np.sqrt(np.sum((a-b)**2))

def knn_predict(X_train, y_train, X_test, k=3):
    predictions = []
    for test_point in X_test
        distances = []
        for i, train_point in enumerate
                              (X_train):
```

PAGE NO :
DATE :

```
        dist = Euclidian_distance(test_point, train_point)
        distances.append((dist, y_train[i]))

    distances.sort(key=lambda x: x[0])
    K_nearest_labels = [label for _, label in
                        distance[:K]]

    predicted_label = max(set(K_nearest_labels),
                          key=K_nearest_labels.count)
return np.array(predictions)

y_pred = knn_predict(X_train_scaled, y_train,
                     X_test_scaled, k=3)
accuracy = np.mean(y_pred == y_test)

print(f"Accuracy: {accuracy:.2f}")
print("Predictions:", y_pred)
print("True labels:", y_test)
```

Output

Accuracy: 0.97

Predictions: 1 0 2 1 0 1 2 2 0 1 2 2 0 2 0 1 2 1 1 2 2 0 1 0 1 2

true label: 1 0 1 1 0 1 2 2 0 1 2 2 0 2 0 1 2 1 1 2 2 0 1 2 0 1 2

PAGE NO :
DATE :

SVM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

np.random.seed(42)
n_samples = 1000
age = np.random.randint(18, 70, n_samples)
income = np.random.randint(30, 150, n_samples)
usage_frequency = np.random.randint(1, 11, n_samples)

Purchase_decision = (age + income / 2 + usage_freq
    * 2 > 100).astype(int)

data = pd.DataFrame({
    'Age': age,
    'Income': income,
    'usage_Frequency': usage_frequency,
    'Purchase_Decision': Purchase_decision
})

X = data[['Age', 'Income', 'usage_frequency']].values
y = data['Purchase_Decision'].values

split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

mean = X_train.mean(axis=0)
std = X_train.std(axis=0)
X_train_scaled = (X_train - mean) / std
X_test_scaled = (X_test - mean) / std
```

PAGE NO :
DATE :

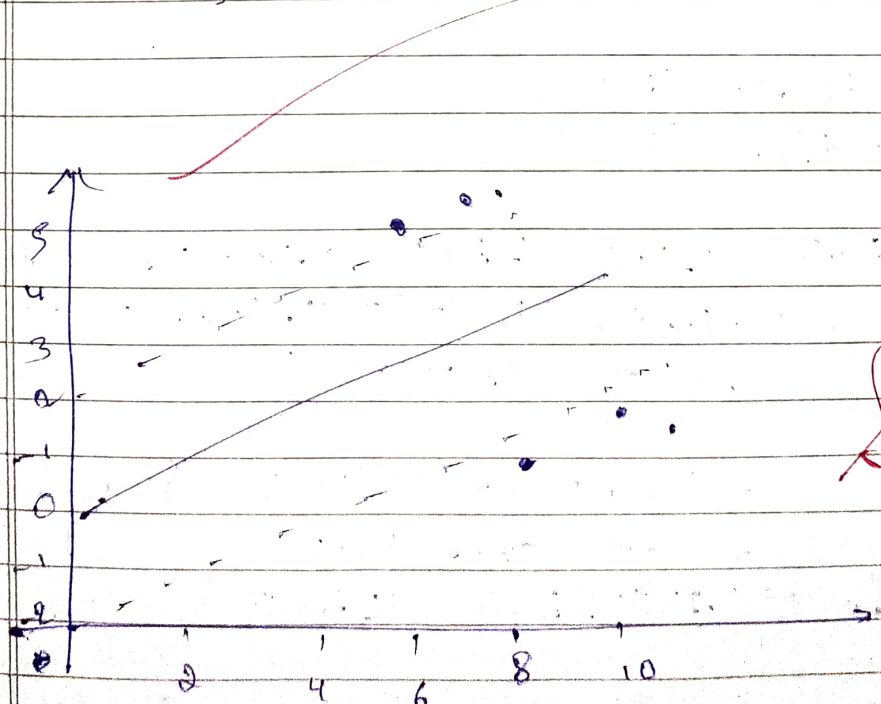
```
y_train_svm = np.where(y_train == 0, -1, 1)
y_test_svm = np.where(y_test == 0, -1, 1)

class LinearSVM:
    def __init__(self, lr=0.001, lambda_param=0.01,
                 n_iters=1000):
        self.lr = lr
        self.lambda_param = lambda_param
        self.n_iters = n_iters
        self.w = None
        self.b = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.w = np.zeros(n_features)
        self.b = 0

        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X):
                condition = y[idx] * (np.dot(x_i, self.w)
                    + self.b) >= 1
                if condition:
                    self.w -= self.lr * (2 * self.lambda_param * self.w)
                else:
                    self.w -= self.lr * (2 * self.lambda_param *
                        self.w - np.dot(x_i, y[idx]))

    def predict(self, X):
        approx = np.dot(X, self.w) + self.b
        return np.sign(approx)
```

PAGE NO :
DATE :

```
svm = LinearSVM()
svm.fit(X_train_scaled, y_train_svm)

predictions = svm.predict(X_test_scaled)
predicted_labels = np.where(predictions == -1, 0, 1)

accuracy = np.mean(predicted_labels == y_test)
print(f"Accuracy is {accuracy:.2f}")

plt.figure(figsize=(8, 6))
plt.scatter(X_test_scaled[:, 0], X_test
plt.xlabel('Age (scaled)')
plt.ylabel('Income (scaled)')
plt.title('Custom SVM class: Age vs Inco')
```

Output:

Accuracy of custom SVM: 0.52



7/4/25