- Name: Sarvesh Surve
- Roll No. 60

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets



class SVM:

    def __init__(self, learning_rate=0.001, lambda_param=0.01, n_iters=1000):
        self.lr = learning_rate
        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

        y_ = np.where(y <= 0, -1, 1)

        # Weights Initialization
        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]))
                    self.b -= self.lr * y_[idx]


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


if __name__ == "__main__":

    # Load the Wine dataset
    wine = datasets.load_wine()
    X = wine.data
    y = wine.target


X = X[y != 2]
y = y[y != 2]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)


 # Initialize the SVM model
clf = SVM()
clf.fit(X_train, y_train)



# Make predictions
predictions = clf.predict(X_test)


# Accuracy function
def accuracy(y_true, y_pred):
  accuracy = np.sum(y_true == y_pred) / len(y_true)
  return accuracy


print(f"SVM classification accuracy : {accuracy(y_test, predictions)}")
```

```
SVM classification accuracy : 0.5384615384615384
```