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

# Ridge Regression
class RidgeRegression() :

    def __init__( self, learning_rate, iterations, l2_penality ) :

        self.learning_rate = learning_rate
        self.iterations = iterations
        self.l2_penality = l2_penality

    # Function for model training
    def fit( self, X, Y ) :

        # no_of_training_examples, no_of_features
        self.m, self.n = X.shape

        # weight initialization
        self.W = np.zeros( self.n )

        self.b = 0
        self.X = X
        self.Y = Y

        # gradient descent learning

        for i in range( self.iterations ) :
            self.update_weights()
        return self

    # Helper function to update weights in gradient descent

    def update_weights( self ) :
        Y_pred = self.predict( self.X )

        # calculate gradients
        dW = ( - ( 2 * ( self.X.T ).dot( self.Y - Y_pred ) ) +
               ( 2 * self.l2_penality * self.W ) ) / self.m
        db = - 2 * np.sum( self.Y - Y_pred ) / self.m

        # update weights
        self.W = self.W - self.learning_rate * dW
        self.b = self.b - self.learning_rate * db
        return self

    # Hypothetical function  h( x )
    def predict( self, X ) :
        return X.dot( self.W ) + self.b
```

```
def main() :

    # Importing dataset
    df = pd.read_csv( "file.csv" )
    X = df.drop[‘target_column’]
    Y = df[‘target_column’]

    # Model training
    model = RidgeRegression( iterations = 1000, learning_rate = 0.01, l2_penality = 1 )
    model.fit( X, Y )

    # Prediction on test set
    Y_pred = model.predict( X)

    # Visualization on test set
    plt.scatter( X, Y, color = 'blue' )
    plt.plot( X, Y_pred, color = 'orange' )
    plt.title( 'Y vs X' )
    plt.show()

if __name__ == "__main__" :
    main()
```