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

```
!gdown 1UpLnYA48Vy_lGUMMLG-uQE1gf_Je12Lh
```

```
Downloading...
From: https://drive.google.com/uc?id=1UpLnYA48Vy_lGUMMLG-uQE1gf_Je12Lh
To: /content/cars24-car-price-clean.csv
100% 7.10M/7.10M [00:00<00:00, 28.7MB/s]
```

```
df = pd.read_csv('cars24-data.csv')
df.head()
```

	selling_price	year	km_driven	mileage	engine	max_power	age	make	model	Individual	Trustmark Dealer
0	-1.111046	-0.801317	1.195828	0.045745	-1.310754	-1.157780	0.801317	-0.433854	-1.125683	1.248892	-0.098382
1	-0.223944	0.450030	-0.737872	-0.140402	-0.537456	-0.360203	-0.450030	-0.327501	-0.333227	1.248892	-0.098382
2	-0.915058	-1.426990	0.035608	-0.582501	-0.537456	-0.404885	1.426990	-0.327501	-0.789807	1.248892	-0.098382
3	-0.892365	-0.801317	-0.409143	0.329620	-0.921213	-0.693085	0.801317	-0.433854	-0.905265	1.248892	-0.098382
4	-0.182683	0.137194	-0.544502	0.760085	0.042999	0.010435	-0.137194	-0.246579	-0.013096	-0.800710	-0.098382

```
df.shape
```

```
(19820, 18)
```

Simple Linear Regression

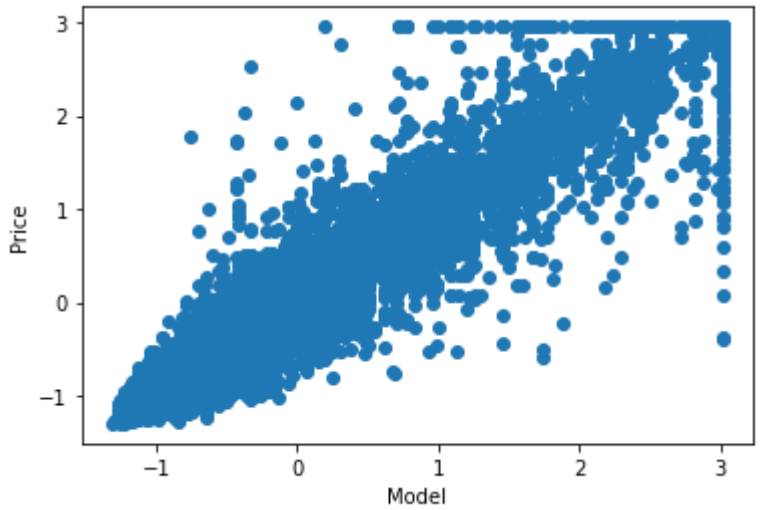
Univariate Linear Regression

```
X = df['model'].values
y = df['selling_price'].values
```

```
X.shape, y.shape
```

```
((19820,), (19820,))
```

```
plt.scatter(X, y)
plt.xlabel("Model")
plt.ylabel("Price")
plt.show()
```




```
def predict(x, weights):
```

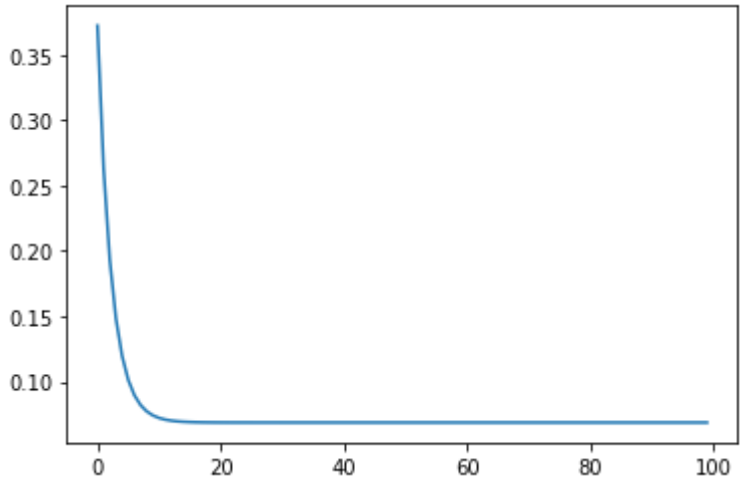
```
  # y^ = w1.x + w0
  y_hat = weights[1]*x + weights[0]
  return y_hat


def error(X, Y, weights):
  '''Implementation of MSE'''
  n = X.shape[0] # 19820

  total_err = 0.0

  for i in range(n):
    y_hat = predict(X[i], weights)
    total_err += ( Y[i] - y_hat )**2

  return total_err/n


def gradient(X, Y, weights):
  n = X.shape[0] # 19820

  grad = np.zeros((2,))

  for i in range(n):
    y_hat = predict(X[i], weights)
    grad[0] += (y_hat - Y[i])
    grad[1] += (y_hat - Y[i])*X[i]


  return 2*grad/n


def gradient_descent(X, Y, n_itr = 100, eta = 0.1):
  weights =  np.random.randn(2,)
  error_list = []

  for i in range(n_itr):
    e = error(X,Y, weights)
    error_list.append(e)
```

```
    error_list.append(e)
    grad = gradient(X, Y, weights)
    weights[0] = weights[0] - eta*grad[0]
    weights[1] = weights[1] - eta*grad[1]

  return weights.round(2), error_list


opt_weights, error_list = gradient_descent(X, y)


opt_weights
```

```
array([-0.  ,  0.97])
```

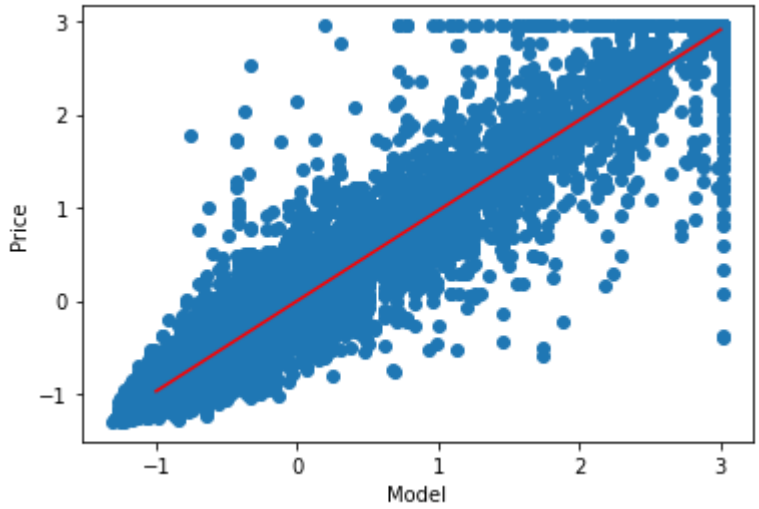
```
plt.plot(error_list)
```

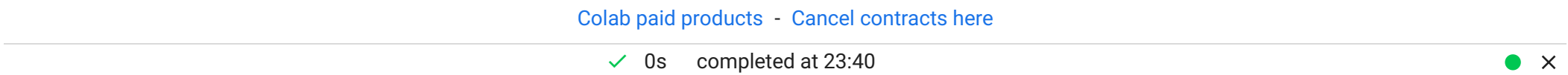
```
[<matplotlib.lines.Line2D at 0x7f850b7cd090>]
```




```
x_new = np.array([-1, 3])

predict(-1, opt_weights)
predict(3, opt_weights)
```

```
     3.2700000000000005

# (-1, -0.97), (3, 2.91)



plt.scatter(X, y)
plt.xlabel("Model")
plt.ylabel("Price")

plt.plot([-1, 3], [-0.97, 2.91], c ='red')

plt.show()
```




Colab paid products - Cancel contracts here
0s completed at 23:40