Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2021-22

Experiment 1 (Regression)

NAME: Dev Patel **SAP ID:60009200016**

BATCH: K/K1 **DATE: 21/03/2022**

Aim: Implement Linear Regression on the given Dataset.

Theory:

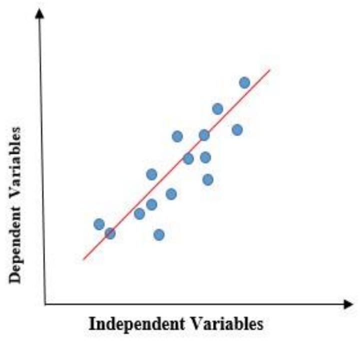
Linear regression is a **linear model**, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). When there is a single input variable (x), the method is referred to as **simple linear regression**. When there are **multiple input variables**, literature from statistics often refers to the method as multiple linear regression. Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called **Least Squares**.

The representation is a linear equation that combines a specific set of input values (x) the solution to which is the predicted output for that set of input values (y). As such, both the input values (x) and the output value are numeric. For example, in a simple regression problem (a single x and a single y), the form of the model would be:

$$y = b + m*x$$

In higher dimensions when we have more than one input (x), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients. It is common to talk about the complexity of a regression model like linear regression. This refers to the number of coefficients used in the model. When a coefficient becomes zero, it effectively removes the influence of the input variable on the model and therefore from the prediction made from the model (0 * x = 0). This becomes relevant if you look at regularization methods that change the learning algorithm to reduce the complexity of regression models by putting pressure on the absolute size of the coefficients, driving some to zero.

Linear Regression: Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), consequently called linear regression. *If there is a single input variable (x), such linear regression is called **simple linear regression**. And if there is more than one input variable, such linear regression is called **multiple linear regression**.* The linear regression model gives a sloped straight line describing the relationship within the variables.



The above graph presents the linear relationship between the dependent variable and independent variables. When the value of x (**independent variable**) increases, the value of y (**dependent variable**) is likewise increasing. The red line is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best. *To calculate best-fit line linear regression uses a traditional slope-intercept form.*

$$y = mx+b \quad \Longrightarrow \quad y = a_0+a_1x$$

y= Dependent Variable; x= Independent Variable; a0= intercept; a1 = Linear regression coefficient.

Need of a Linear regression

Let's say we want to estimate the salary of an employee based on year of experience. You have the recent company data, which indicates that the relationship between experience and salary. Here year of experience is an independent variable, and the salary of an employee is a dependent variable, as the salary of an employee is dependent on the experience of an employee. Using this insight, we can predict the future salary of the employee based on current & past information.

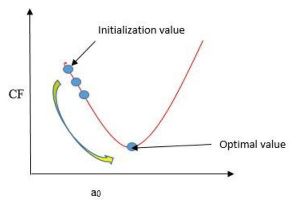
Cost function: The cost function helps to figure out the best possible values for a0 and a1, which provides the best fit line for the data points. Cost function optimizes the regression coefficients or weights and measures how a linear regression model is performing. The cost function is used to find the accuracy of the **mapping function** that maps the input variable to the output variable. This mapping function is also known as **the Hypothesis function**.

In Linear Regression, **Mean Squared Error (MSE)** cost function is used, which is the average of squared error that occurred between the predicted values and actual values. *By simple linear equation y=mx+b we can calculate MSE as: Let's y = actual values, y_i = predicted values*

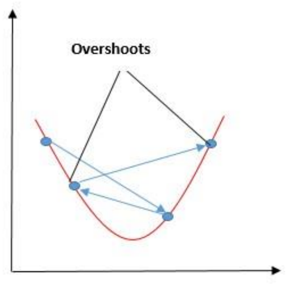
$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Using the MSE function, we will change the values of a0 and a1 such that the MSE value settles at the minima. Model parameters ***xi, b (a_0,a_1)*** can be manipulated to minimize the cost function. These parameters can be determined using the gradient descent method so that the cost function value is minimum.

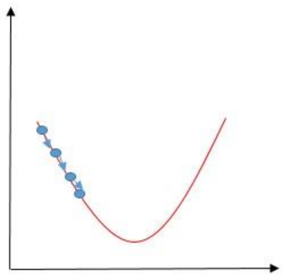
Gradient descent : Gradient descent is a method of updating a0 and a1 to minimize the cost function (MSE). A regression model uses gradient descent to update the coefficients of the line (a0, a1 => xi, b) by reducing the cost function by a random selection of coefficient values and then iteratively update the values to reach the minimum cost function.



Imagine a pit in the shape of U. You are standing at the topmost point in the pit, and your objective is to reach the bottom of the pit. There is a treasure, and you can only take a discrete number of steps to reach the bottom. If you decide to take one footstep at a time, you would eventually get to the bottom of the pit but, this would take a longer time. If you choose to take longer steps each time, you may get to sooner but, there is a chance that you could overshoot the bottom of the pit and not near the bottom. In the gradient descent algorithm, the number of steps you take is the learning rate, and this decides how fast the algorithm converges to the minima.



High learning rate



Low learning rate

To update a_0 and a_1, we take gradients from the cost function. To find these gradients, we take partial derivatives for a_0 and a_1.

$$J = \frac{1}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \cdot x_i$$

$$\frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

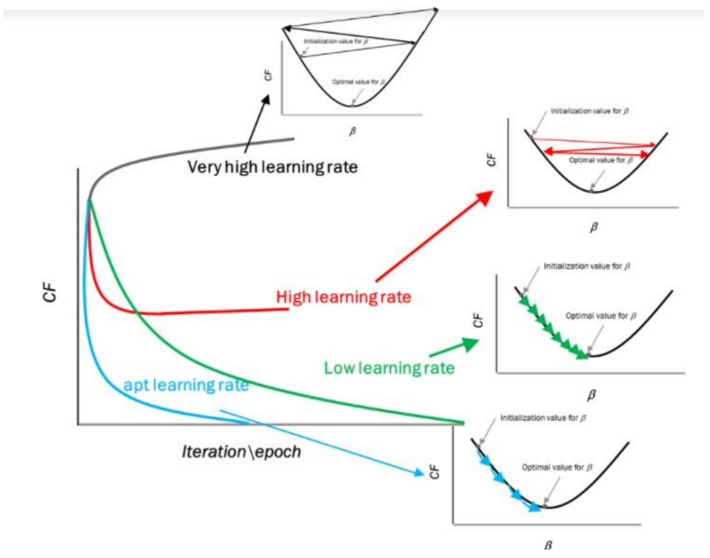
$$a_0 = a_0 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$a_1 = a_1 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

partial derivates are the gradients and they are used to update the

The partial derivates are the gradients, and they are used to update the values of a_0 and a_1. Alpha is the learning rate.

Impact of different values for learning rate



The blue line represents the optimal value of the learning rate, and the cost function value is minimized in a few iterations. The green line represents if the learning rate is lower than the optimal value, then the number of iterations required high to minimize the cost function. If the learning rate selected is very high, the cost function could continue to increase with iterations and saturate at a value higher than the minimum value, that represented by a red and black line.

When linear regression is underfitting there is no other way (given you can’t add more data) then to increase complexity of the model making it polynomial regression (cubic, quadratic,etc…) or using other complex model to capture data that linear regression cannot capture due to its simplicity.

When linear regression is overfitting, number of columns(independent variables) approach number of observations there are two ways to mitigate it

1. Add more observations
2. Regularization

Unlike already complex models like polynomial regression where we could avoid overfitting by reducing model complexity (decreasing degree of polynomial), linear model has no simpler method it could go to therefore it needs to regularize.

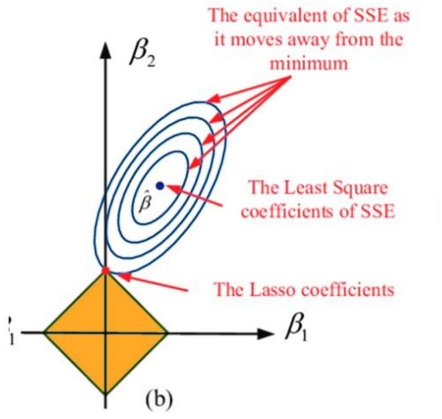
Since adding more observations is time consuming and often not provided we will use regularization technique to mitigate overfitting. There are multiple regularization techniques, all share the same concept of **adding constraints on weights** of independent variables(except theta_0) however they differ in way of constraining. We will go through three most popular regularization techniques:

- Ridge regression (L2)
- Lasso regression (L1)

Lasso Regression

The word "LASSO" denotes Least Absolute Shrinkage and Selection Operator. Lasso regression follows the regularization technique to create prediction. It is given more priority over the other regression methods because it gives an accurate prediction. Lasso regression model uses shrinkage technique. In this technique, the data values are shrunk towards a central point similar to the concept of mean. The lasso regression algorithm suggests a simple, sparse models (i.e. models with fewer parameters), which is wellsuited for models or data showing high levels of multicollinearity or when we would like to automate certain parts of model selection, like variable selection or parameter elimination using feature engineering.

Lasso Regression algorithm utilises L1 regularization technique It is taken into consideration when there are more number of features because it automatically performs feature selection.


The equivalent of SSE as it moves away from the minimum
β_2
$\hat{\beta}$
The Least Square coefficients of SSE
The Lasso coefficients
β_1
(b)

Mathematical equation of Lasso Regression Algorithm:

Residual Sum of Squares + λ * (Sum of the absolute value of the coefficients)

The equation looks like:

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

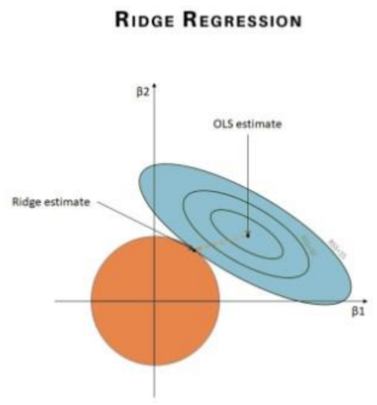
Where,

- λ = the amount of shrinkage.
- If λ = 0 it implies that all the features are considered and now it is equivalent to the linear regression in which only the residual sum of squares is used to build a predictive model.
- If λ = ∞ it implies that no feature is used i.e, as λ gets close to infinity it eliminates more and more features and feature selection is more precise.
- When the bias increases, the value of λ increases
- When the variance increases, the value of λ decreases

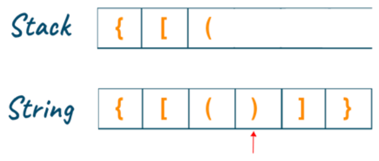
Ridge Regression

Ridge Regression is another type of regression algorithm in data science and is usually considered when there is a high correlation between the independent variables or model parameters. As the value of correlation increases the least square estimates evaluates unbiased values. But if the collinearity in the dataset is very high, there can be some bias value. Therefore, we create a bias matrix in the equation of Ridge Regression algorithm. It is a useful regression method in which the model is less susceptible to overfitting and hence the model works well even if the dataset is very small.

Department of Computer Science and Engineering (Data Science)



The cost function for ridge regression algorithm is:



Where λ is the penalty variable. λ given here is denoted by an alpha parameter in the ridge function. Hence, by changing the values of alpha, we are controlling the penalty term. Greater the values of alpha, the higher is the penalty and therefore the magnitude of the coefficients is reduced.

We can conclude that it shrinks the parameters. Therefore, it is used to prevent multicollinearity, it also reduces the model complexity by shrinking the coefficient.

Department of Computer Science and Engineering (Data Science)

Lab Assignments to complete in this session

Use the given dataset and perform the following tasks:

Dataset 1: Synthetic Data using a mathematical function. (use can use sin wave)

Dataset 2: food_truck_data.txt

1. Perform Linear Regression on Dataset 1 and Dataset 2 by computing cost function, gradient descent and inference using optimized cost function.
2. Use python libraries to perform linear regression, ridge and lasso regression on Dataset 1.

1. Perform Linear Regression on Dataset 1 and Dataset 2 by computing cost function, gradient descent and inference using optimized cost function.

In [30]:

```
#Import Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [31]:

```
#Read the dataset (Dataset 1)
data = pd.read_csv("/content/sample_data/food_truck_data.txt")
data.head()
```

Out[31]:

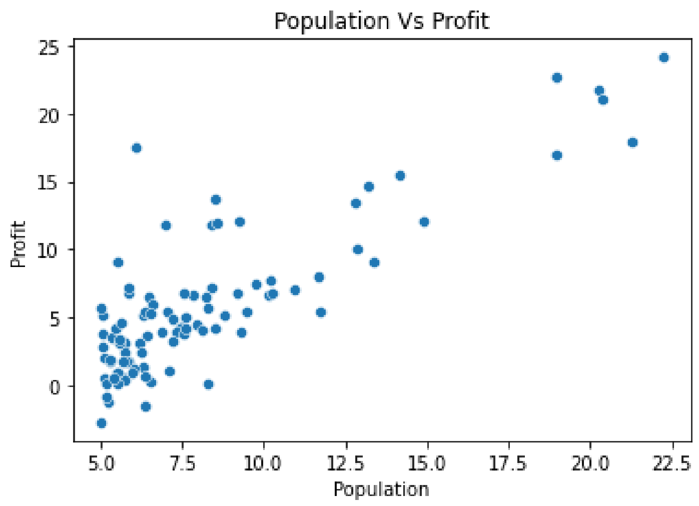
	Population	Profit
0	6.1101	17.5920
1	5.5277	9.1302
2	8.5186	13.6620
3	7.0032	11.8540
4	5.8598	6.8233

In [32]:

```
#Plot the points in the dataset on a scatterplot
ax = sns.scatterplot(x = "Population", y = "Profit", data = data)
ax.set_title("Population Vs Profit")
plt.xlabel("Population")
plt.ylabel("Profit")
```

Out[32]:

```
Text(0, 0.5, 'Profit')
```



In [33]:

```
# The predicted y value function
def y_pred(theta,x):
    return(theta[0] + theta[1]*x)
```

In [34]:

```
# Cost function J calculated using theta values [0,0]
def cost_function(theta, data):
    n = len(data)
    x = data["Population"]
    y = data["Profit"]
    error = np.sum((y - y_pred(theta, x))**2)
    return(error/(2*n))
theta = [0,0]
print(f"The Value of J at [theta0,theta1]= {theta} is {cost_function(theta, data)}")
```

```
The Value of J at [theta0,theta1]= [0, 0] is 32.072733877455676
```

In [35]:

```
# Gradient descent formula applied in the function
def gradient_descent(data, alpha = 0.01, max_iter=2000):
    n = len(data)
    x = data["Population"]
    y = data["Profit"]
    theta = np.zeros((2,)) #(0 0)
    costs = cost_function(theta, data)
    for i in range(max_iter):
        ypred = y_pred(theta, x)
        theta[0] -= alpha/n*(np.sum(ypred - y))
        theta[1] -= alpha/n*(np.sum((ypred - y)*x))
    costs = cost_function(theta, data)
    return theta, costs
```

In [36]:

```
# Finding the parameters of the best fit line of food_truck_data.txt dataset
theta, costs = gradient_descent(data, max_iter = 10000)
print(f"Parameters(Theta): {theta}\nCost: {costs}")
```

```
Parameters(Theta): [-3.89578082  1.19303364]
Cost: 4.476971375975179
```

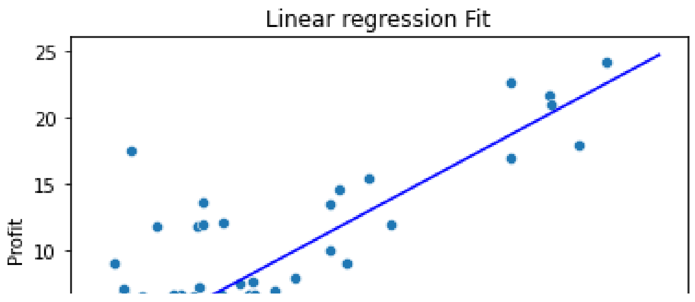
In [37]:

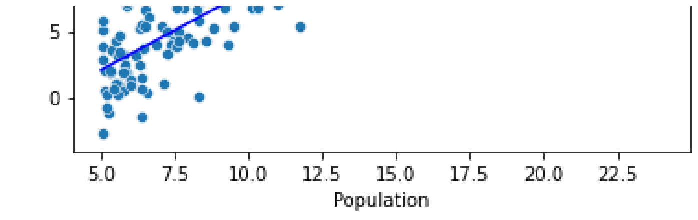
```
# Plotting the best fit line obtained with the data points
ax = sns.scatterplot(x ="Population", y = "Profit", data = data)
ax.set_title("Population Vs Profit")
x_value = [x for x in range(5,25)]
y_value = [(x * theta[1] + theta[0]) for x in x_value]
sns.lineplot(x_value, y_value, color = "blue")

plt.xlabel("Population")
plt.ylabel("Profit")
plt.title("Linear regression Fit")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resu
lt in an error or misinterpretation.
  FutureWarning
```

Out[37]:

```
Text(0.5, 1.0, 'Linear regression Fit')
```



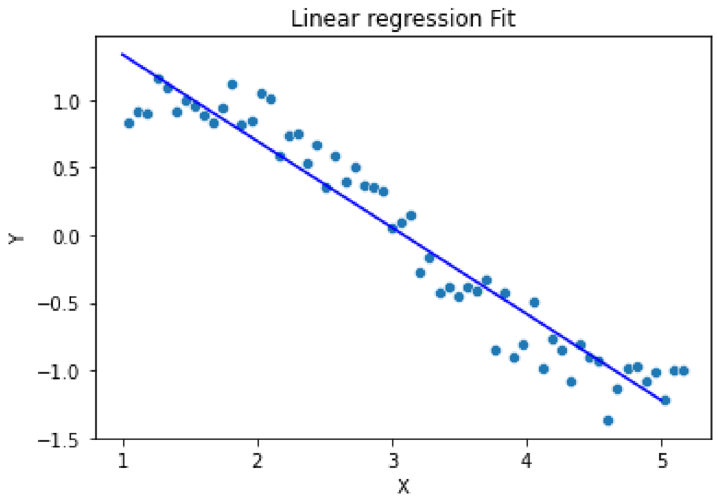


In [38]:

```
#Data Set 2: Gradient descent on synthetic dataset (sin wave)

x = np.array([i*np.pi/180 for i in range(60,300,4)])
y = np.sin(x) + np.random.normal(0,0.15,len(x))

data = pd.DataFrame(np.column_stack([x,y]),columns=['Population','Profit'])

theta, costs = gradient_descent(data, max_iter = 10000)

ax = sns.scatterplot(x = "Population", y = "Profit", data = data)
ax.set_title("X Vs Y")
x_value = [x for x in range(1, 6)]
y_value = [(x * theta[1] + theta[0]) for x in x_value]
sns.lineplot(x_value, y_value,color = "blue")

plt.xlabel("X")
plt.ylabel("Y")
plt.title("Linear regression Fit")
print(f"Parameters(Theta): {theta}\nCost: {costs}")
```

```
Parameters(Theta): [ 1.97483859 -0.63900119]
Cost: 0.021440204674711298
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resu
lt in an error or misinterpretation.
  FutureWarning
```



2. Use python libraries to perform linear regression, ridge and lasso regression on synthetic data.

In [39]:

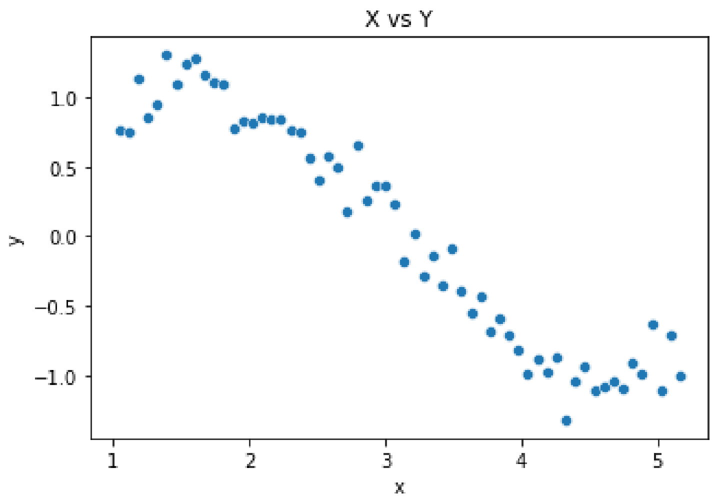
```
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
import math
```

In [40]:

```
# Generate random data

x = np.array([i*np.pi/180 for i in range(60,300,4)])
y = np.sin(x) + np.random.normal(0,0.15,len(x))

data = pd.DataFrame(np.column_stack([x,y]),columns=['x','y'])
for i in range(2,4):  #power of 1 is already there
    colname = 'x_%d'%i      #new var will be x_power
    data[colname] = data['x']**i


ax = sns.scatterplot(x = "x", y = "y", data = data)
ax.set_title("X vs Y")
```

Out[40]:

```
Text(0.5, 1.0, 'X vs Y')
```



In [41]:

```
def rmse(y_hat , y):
  err=0
  for i in range(0,len(y)):
    err+=np.sqrt(((y_hat[i]-y[i])**2)/len(y))
  return err
```

In [42]:

```
comparison_df={'Regression Type' : ['LR' , 'Ridge(l2)' , 'Lasso(l1)'] , 'RMSE': []}
```

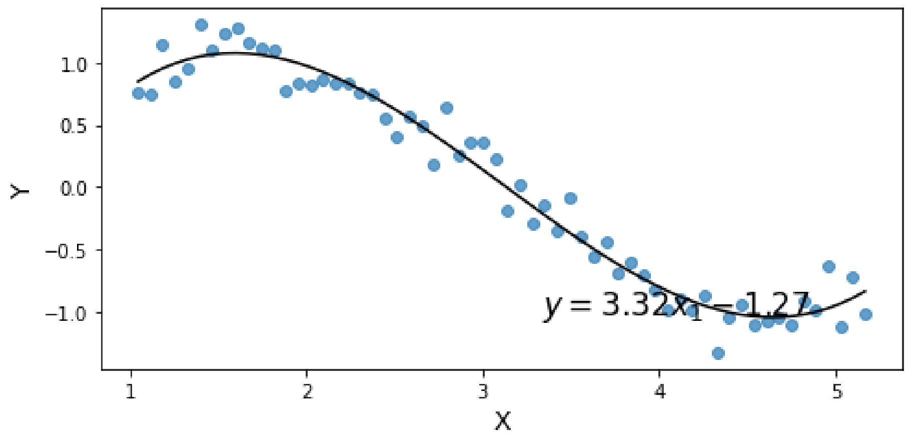
In [43]:

```
#Linear Regression
lr = LinearRegression()
model=lr.fit(data.drop('y' , axis=1),data['y'])

pred_train_lr= lr.predict(data.drop('y' , axis=1))
comparison_df['RMSE'].append(rmse(pred_train_lr , y))
print(rmse(pred_train_lr , y))
```

```
0.8902297395884522
```

In [44]:

```
fig, ax = plt.subplots(figsize=(7, 3.5))

ax.plot(x, pred_train_lr, color='k', label='Regression model')
ax.scatter(x, y,  alpha=0.7, label='Sample data')
ax.set_ylabel('Y', fontsize=14)
ax.set_xlabel('X', fontsize=14)
ax.text(0.55, 0.15, '$y = %.2f x_1 - %.2f $' % (model.coef_[0], abs(model.intercept_)),
fontsize=17, transform=ax.transAxes)

fig.tight_layout()
```



In [45]:

```
#Ridge Regression
X=data.drop('y' , axis=1)
Y=data['y']
rr = Ridge(alpha=0.01)
rr.fit(X, Y)
pred_train_rr= rr.predict(X)
comparison_df['RMSE'].append(rmse(pred_train_rr , Y))
print(rmse(pred_train_rr , Y))
```

```
0.8996231517134742
```

In [46]:

```
#Lasso Regression

ls = Lasso(alpha=0.01)
ls.fit(X, Y)
pred_train_ls= ls.predict(X)
comparison_df['RMSE'].append(rmse(pred_train_ls , Y))
print(rmse(pred_train_ls , Y))
```

```
1.4026209629622222
```

In [47]:

```
df = pd.DataFrame(comparison_df)
df.min()
```

Out[47]:

```
Regression Type          LR
RMSE                0.89023
dtype: object
```

Inference

We infer that linear regression has the least RMSE, because it is overfitting to the model as compared to the lasso and ridge regression.