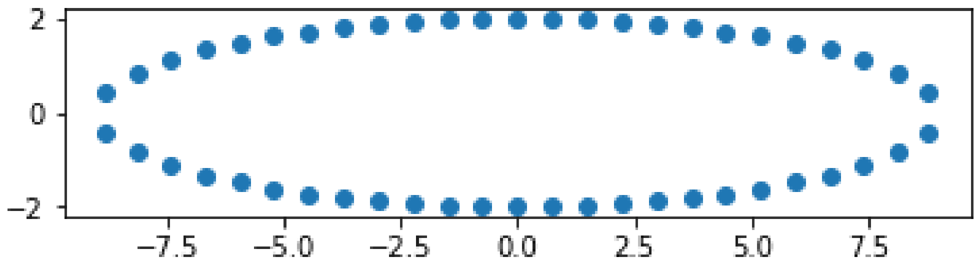
Task-C: Regression outlier effect.

Objective:Visualization best fit linear regression line for different scenarios

```
# you should not import any other packages
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import numpy as np
from sklearn.linear_model import SGDRegressor

import numpy as np
import scipy as sp
import scipy.optimize

def angles_in_ellipse(num,a,b):
    assert(num > 0)
    assert(a < b)
    angles = 2 * np.pi * np.arange(num) / num
    if a != b:
        e = (1.0 - a ** 2.0 / b ** 2.0) ** 0.5
        tot_size = sp.special.ellipeinc(2.0 * np.pi, e)
        arc_size = tot_size / num
        arcs = np.arange(num) * arc_size
        res = sp.optimize.root(
            lambda x: (sp.special.ellipeinc(x, e) - arcs), angles)
        angles = res.x
    return angles

a = 2
b = 9
n = 50

phi = angles_in_ellipse(n, a, b)
e = (1.0 - a ** 2.0 / b ** 2.0) ** 0.5
arcs = sp.special.ellipeinc(phi, e)

fig = plt.figure()
ax = fig.gca()
ax.axes.set_aspect('equal')
ax.scatter(b * np.sin(phi), a * np.cos(phi))
plt.show()
```

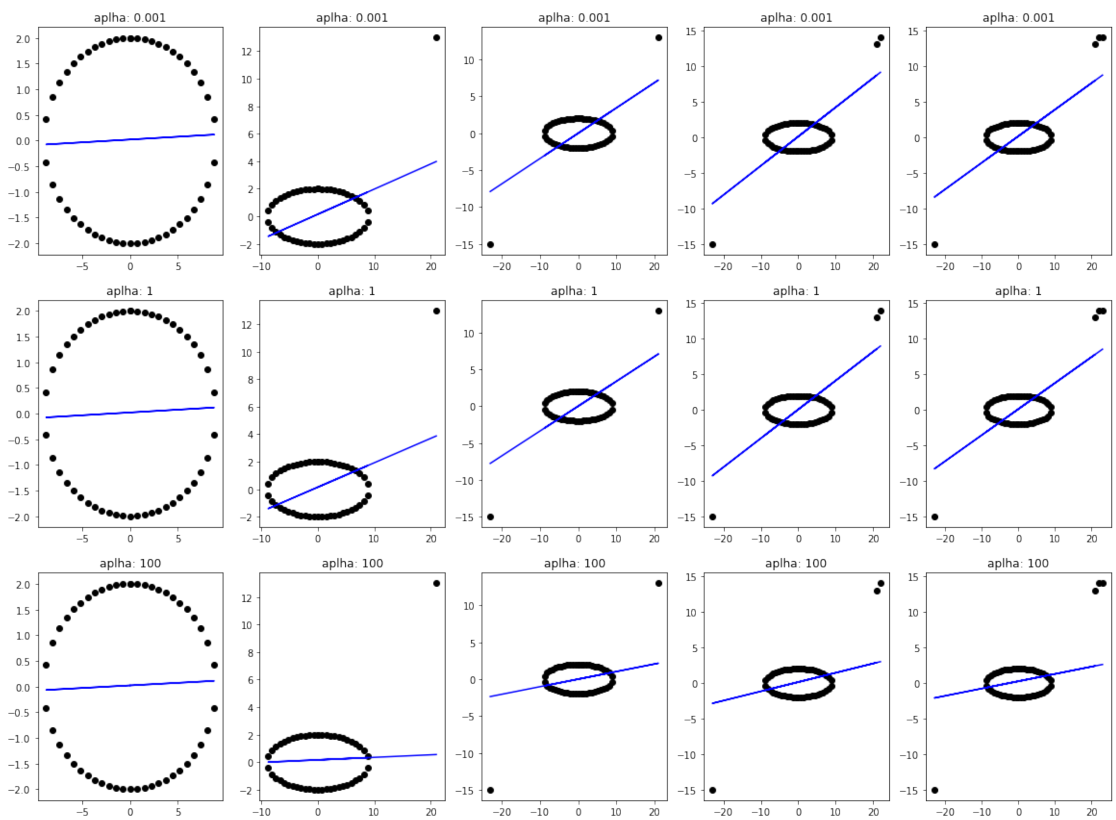



```
X= b * np.sin(phi)
Y= a * np.cos(phi)
print(X.shape)
print(Y.shape)
print(type(X))
```

```
(50,)
(50,)
<class 'numpy.ndarray'>
```

```
#Reference for this code:
https://stackoverflow.com/questions/59367939/how-to-correct-the-
position-of-hyper-plane-in-python

#Define Parameters: alpha and outlier points
alphas = [0.001,1,100]
outlier_points = [(0,2),(21, 13), (-23, -15), (22,14), (23, 14)]

plt.figure(figsize = (20,20))

for j,alpha in enumerate(alphas):

    # Re-define X and Y to remove the added outliers in the bellow
step
    X= b * np.sin(phi)
    Y= a * np.cos(phi)

    #print(range(5*j+1, 5*(j+1)+1))
    #Here range is to provide 3*5 grid subplots. we take (1,5);(6,10),
(11,15) ranges for this
    for c,k in enumerate(range(5*j+1, 5*(j+1)+1)):

        #Adding Outlier for each iteration
        X= np.append(X,outlier_points[c][0])
        Y= np.append(Y,outlier_points[c][1])


        #training the model after updating the outliers
        clf = SGDRegressor(alpha = alpha, eta0=0.001,
learning_rate='constant',random_state=10)
        clf.fit(X.reshape(-1,1), Y)
```

```
        #Get the predicted values of Y to plot prediction hyper-plane
        Y_pred =clf.predict(X.reshape(-1,1))

        #print(type(Y_pred))
        #print(X.reshape(-1,1).shape)
        #print(Y_pred.shape)
        #print("Y Actual: ",Y)
        #print("Y Pred: ", Y_pred)

        plt.subplot(4,5,k)
        plt.scatter(X,Y,color = 'black')
        plt.plot(X,Y_pred, color ='blue')
        plt.title("aplha: "+str(alpha))

plt.show()
```



Observation:

- Here alpha is a constant that multiplies the regularization term. The higher the value, the stronger the regularization.
- Therefore, High the value of alpha, more is the regularization hence less is the effect of outliers on the predictions.

- When alpha = 0.001: With one outlier point we found that the angular deviation of prediction hyper-plane is more. Therefore, the model less regularized when alpha= 0.001.
- When alpha = 1: With one outlier point we found that the angular deviation of prediction hyper-plane is more. Therefore, the model less regularized even when alpha is 1
- When alpha = 100: With one outlier point we found that the angular deviation of prediction hyper-plane is more same as the one without outliers. Therefore, the model is more regularized when alpha is 100
- When alpha = 100: With 2/3/4 outlier points we found that the angular deviation of prediction hyper-plane is slightly different as the one without outliers. But, compared to alpha =0.001 and 1, deviation is very less.Therefore, the model is more regularized when alpha is 100.