Task-C: Regression outlier effect.

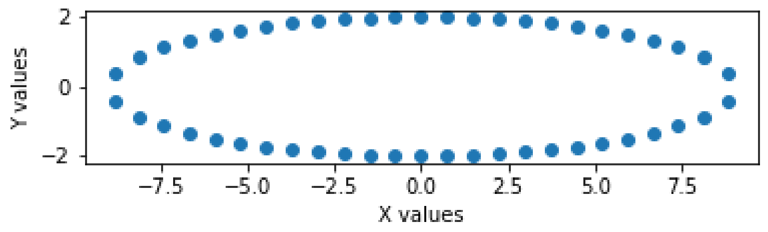
Objective:Visualization best fit linear regression line for different scenarios

In [1]:
```
# 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
```

In [76]:
```
#necessary libraries
import numpy as np
import scipy as sp
import scipy.optimize

# create the dataset
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
```

In [98]:
```
# Plot of our dataset in 2-dimensional space
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.xlabel("X values");plt.ylabel("Y values")
plt.show()
```



In [99]:
```
X = b * np.sin(phi) #-->Independant variables - has 50 values
Y = a * np.cos(phi) #--> Target values- its having 50 values
```

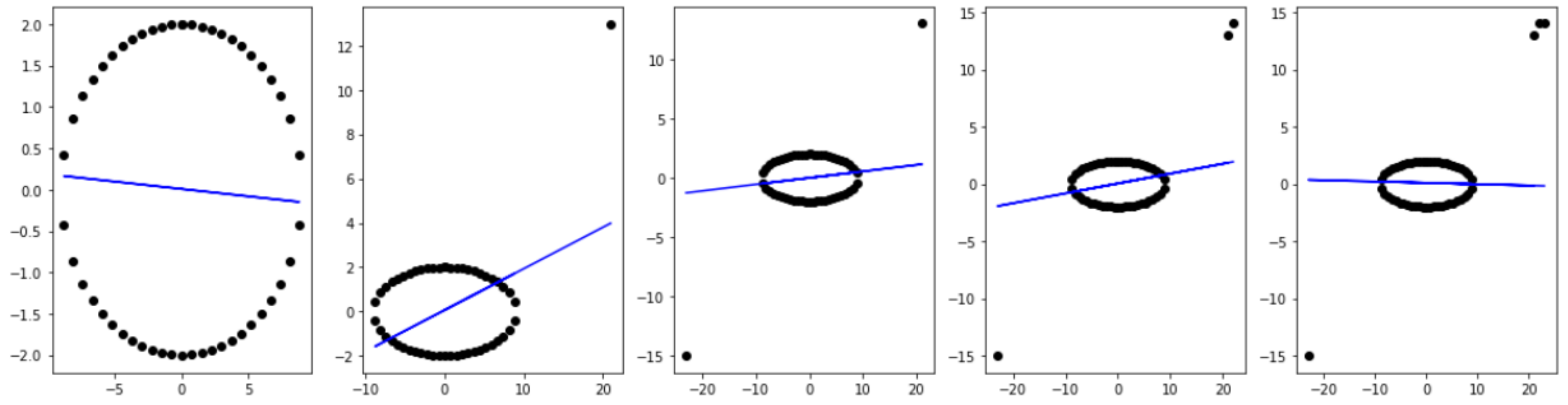
1. As a part of this assignment you will be working the regression problem and how regularization helps to get rid of outliers

2. Use the above created X, Y for this experiment.

3. to do this task you can either implement your own SGDRegression(prefered) excatly similar to "SGD assignment" with mean squared error or
you can use the SGDRegression of sklearn, for example "SGDRegressor(alpha=0.001, eta0=0.001, learning_rate='constant',random_state=0)"
note that you have to use the constant learning rate and learning rate ***eta0*** *initialized.*

4. as a part of this experiment you will train your linear regression on the data (X, Y) with different regularizations alpha=[0.0001, 1, 100] and *observe how prediction hyper plan moves with respect to the outliers*

5. This the results of one of the experiment we did (title of the plot was not metioned intentionally)



in each iteration we were adding single outlier and observed the movement of the hyper plane.

6. please consider this list of outliers: [(0,2),(21, 13), (-23, -15), (22,14), (23, 14)] in each of tuple the first elemet is the input feature(X) and the second element is the output(Y)

7. for each regularizer, you need to add these outliers one at time to data and then train your model again on the updated data.

*8. you should plot a 3*5 grid of subplots,*
where each row corresponds to results of model with a single regularizer.

9. Algorithm:

```
for each regularizer:
    for each outlier:
        #add the outlier to the data
        #fit the linear regression to the updated data
        #get the hyper plane
        #plot the hyperplane along with the data points
```

10. MAKE SURE YOU WRITE THE DETAILED OBSERVATIONS, PLEASE CHECK THE LOSS FUNCTION IN THE SKLEARN

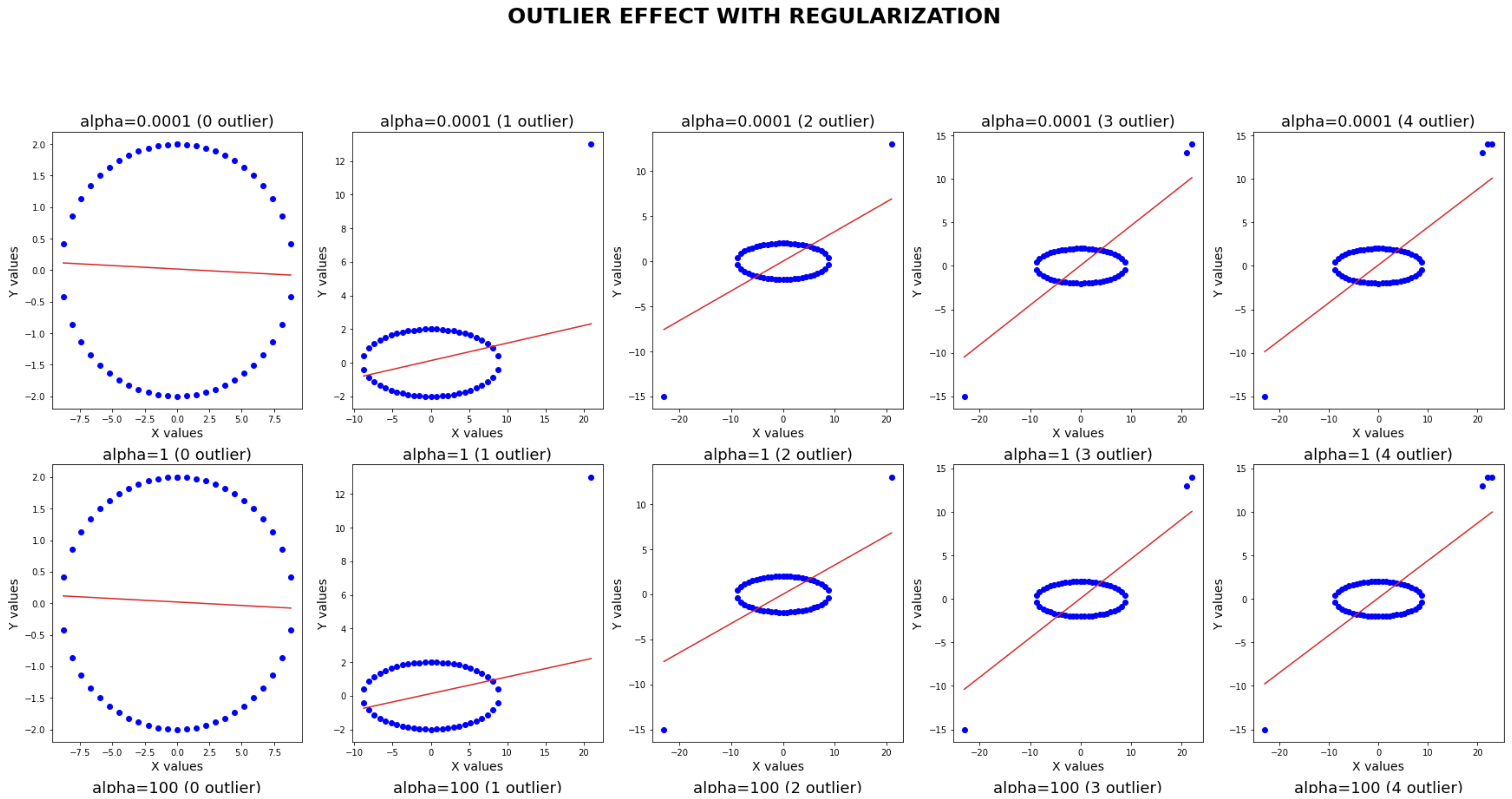
DOCUMENTATION
(please do search for it).

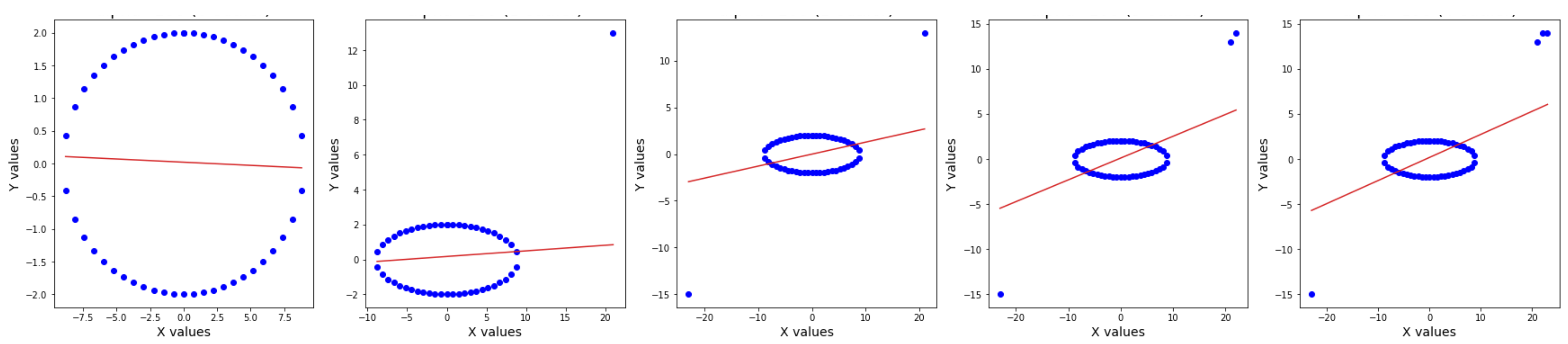
In [100…

```
# Assign Original dataset to data_X and data_Y:
data_X = X;data_Y = Y

# Define Draw line() for plotting hyperplane:
def drawline(coef,intercept,mi,ma):
    y_min = (coef * mi) + intercept
    y_max = (coef * ma) + intercept
    x = [mi,ma]; y = [y_min,y_max]
    plt.plot(x,y,c = '#d62728')
```

In [119…

```
# initialize outlier values & hyperparameters for linear regression
outliers = [(0,2),(21,13),(-23,-15),(22,14),(23,14)]
regularizer = [0.0001,1,100];index = 1;

plt.figure(figsize=(30,20)) # set figure size
plt.suptitle("OUTLIER EFFECT WITH REGULARIZATION",fontsize = 25,fontweight = 'bold') # set plots title
for r in regularizer:
    for i,j in enumerate(outliers):
        plt.subplot(3,5,index)   # subplot creation

        # dataset creation with outliers
        X1 = np.append(X,j[0]).reshape(-1,1)
        Y1 = np.append(Y,j[1]).reshape(-1,1)

        # fit & apply the model to the data
        m1 = SGDRegressor(alpha=r,eta0=0.001,learning_rate='constant',random_state=60)
        model = m1.fit(X1,Y1)

        coef = model.coef_ ;  # obtain coefficients
        intercept = model.intercept_ ; # obtain the intercepts

        # plotting the model
        plt.title("alpha={0} ({1} outlier)".format(r,i),fontsize = 18)
        plt.xlabel("X values",fontsize = 14);
        plt.ylabel("Y values",fontsize = 14)
        drawline(coef[0],intercept[0],min(X1),max(X1))
        plt.scatter(X1,Y1,c = 'b')

        X = X1; Y = Y1
        index += 1

    X = data_X  # again reassign the original input dataset to X
    Y = data_Y  # again reassign the original Target column to Y

plt.show()
```





OBSERVATION :

1)When alpha = 0.0001(regularization strength),we can infer that the model/linear hyperplane tends to approximate the data points in the 2d space and additionally as the number of outliers increases the model gives more importance to the outliers and try to reduce the error between the actual outlier & predicted outlier.

2)When alpha = 1(regularization strength),we can infer that the model/linear hyperplane tends to approximate the data points in the 2d space and additionally as the number of outliers increases the model/hyperplane is slightly way off from the outliers because of the regularization effect during the optimization of loss function.

3))When alpha = 100(regularization strength),we can infer that the model/linear hyperplane tends to approximate the data points in the 2d space and additionally as the number of outliers increases the model/hyperplane is not much impacted due to the outliers and this happens because of the regularization effect during the optimization of loss function.