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
```

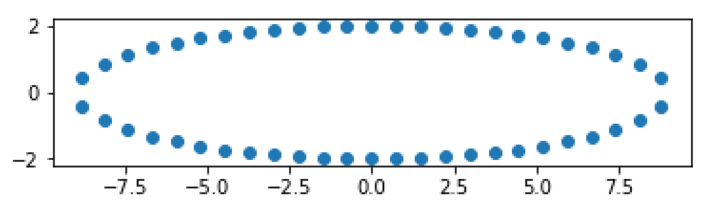
In [2]:
```
import numpy as np
import scipy as sp
import scipy.optimize

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
```

In [3]:
```
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()
```



In [4]:
```
X= b * np.sin(phi)
Y= a * np.cos(phi)
```

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 s equared error or
you can use the SGDRegression of sklearn, for example "SGDRegressor(alpha=0.001, eta0=0.001, learning_rate='constant',ran dom_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=[0.000 1, 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.

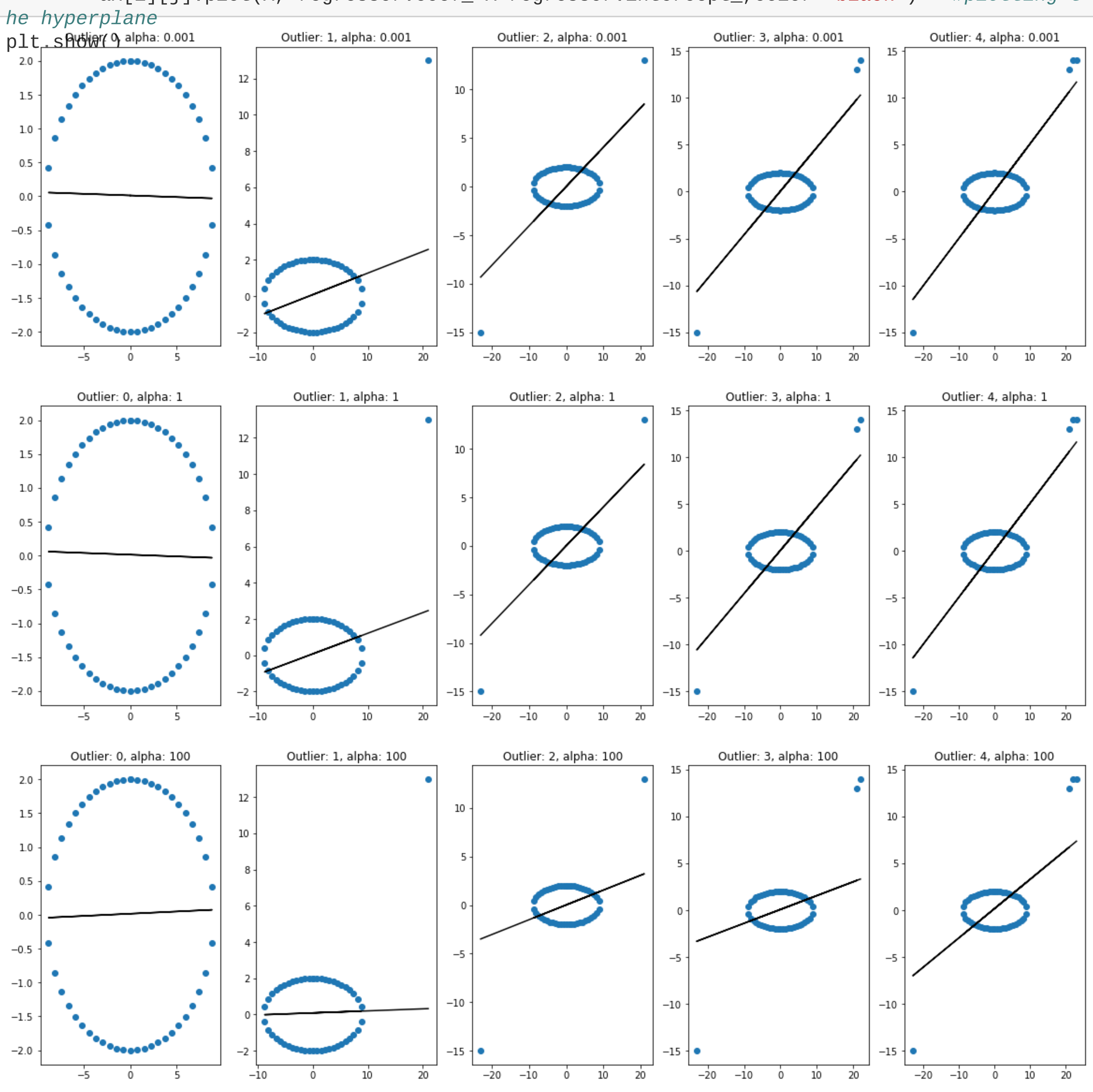
9. Algorithm:

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 SKLEAR N DOCUMENTATION

In [5]:
```
fig,ax = plt.subplots(nrows = 3, ncols = 5, figsize = (20,20))

#Looping through each hyperparameter
for i,alpha in enumerate([0.001, 1, 100]):
    X= b * np.sin(phi)
    Y= a * np.cos(phi)
    #Adding an outlier for each iteration
    for j,outlier in enumerate([(0,2),(21, 13), (-23, -15), (22,14), (23, 14)]):
        X = np.append(X,outlier[0])
        Y = np.append(Y,outlier[1])
        #Fitting the model
        regressor = SGDRegressor(alpha=alpha, eta0=0.001, learning_rate='constant',random_st
ate=0)
        regressor.fit(X.reshape(-1,1),Y.reshape(-1,1))
        ax[i][j].title.set_text(f"Outlier: {j}, alpha: {alpha}")
        ax[i][j].scatter(X,Y)  #PLOTTING SCATTER[0,0],[0,1]...[0,4] FOR EACH REGULARISATION
 PARAMETER(3)==>3*5=15PLOTS
        ax[i][j].plot(X, regressor.coef_*X+regressor.intercept_,color='black')   #plotting t
he hyperplane
plt.show()
```



Observation:

- We know that the overfitting of the model can be prevented by adding a regularization term to the loss function.As the regularization term increases it tries to negate the effect of outliers.
- Alpha is a Constant that is multiplied with the regularization term and higher the value of alpha, the stronger the regularization.
- As the number of outliers increases then the strength of the regularization term should also be increased to compensate it but very large regularization might lead to underfitting.
- As we can see from the above plots:
  - When there are no outliers,a perfect hyperplane can be observed in all 3 cases when alpha is 0.001,1 and 100
  - When 1 outlier is present in the dataset, the hyperplane is slightly aligned towards and trying to accomodate the outlier for cases when alpha is 0.001 and 1. Since the regularization term is higher when alpha is 100, the hyperplane is fairly accurate and isnt impacted the outlier.