Task-C: Regression outlier effect.

Objective:Visualization best fit linear regression line for different scenarios

In [176]:

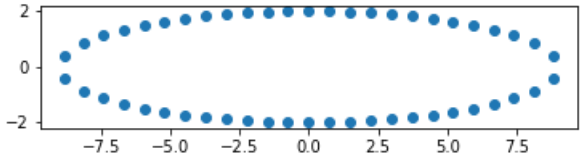
```
# you should not import any other packages
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import numpy as np
from sklearn.linear_model import SGDRegressor
```

In [177]:

```
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 [178]:

```
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 [179]:

```
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 outlie

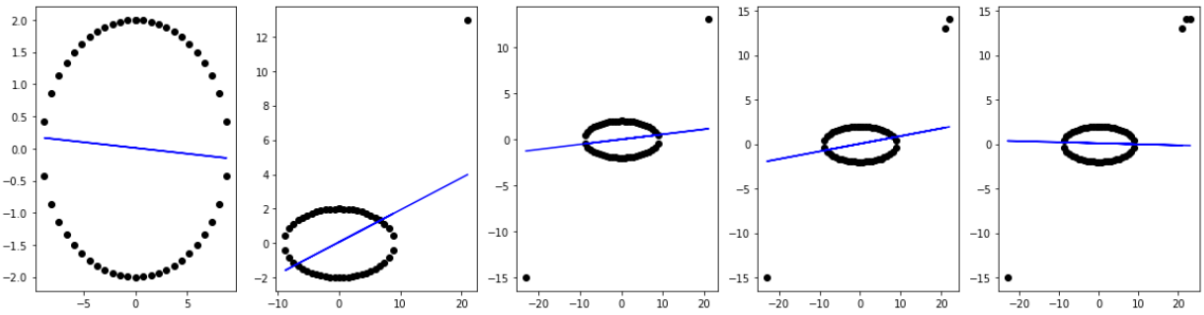
rs

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 sequared 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 ***etao*** *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

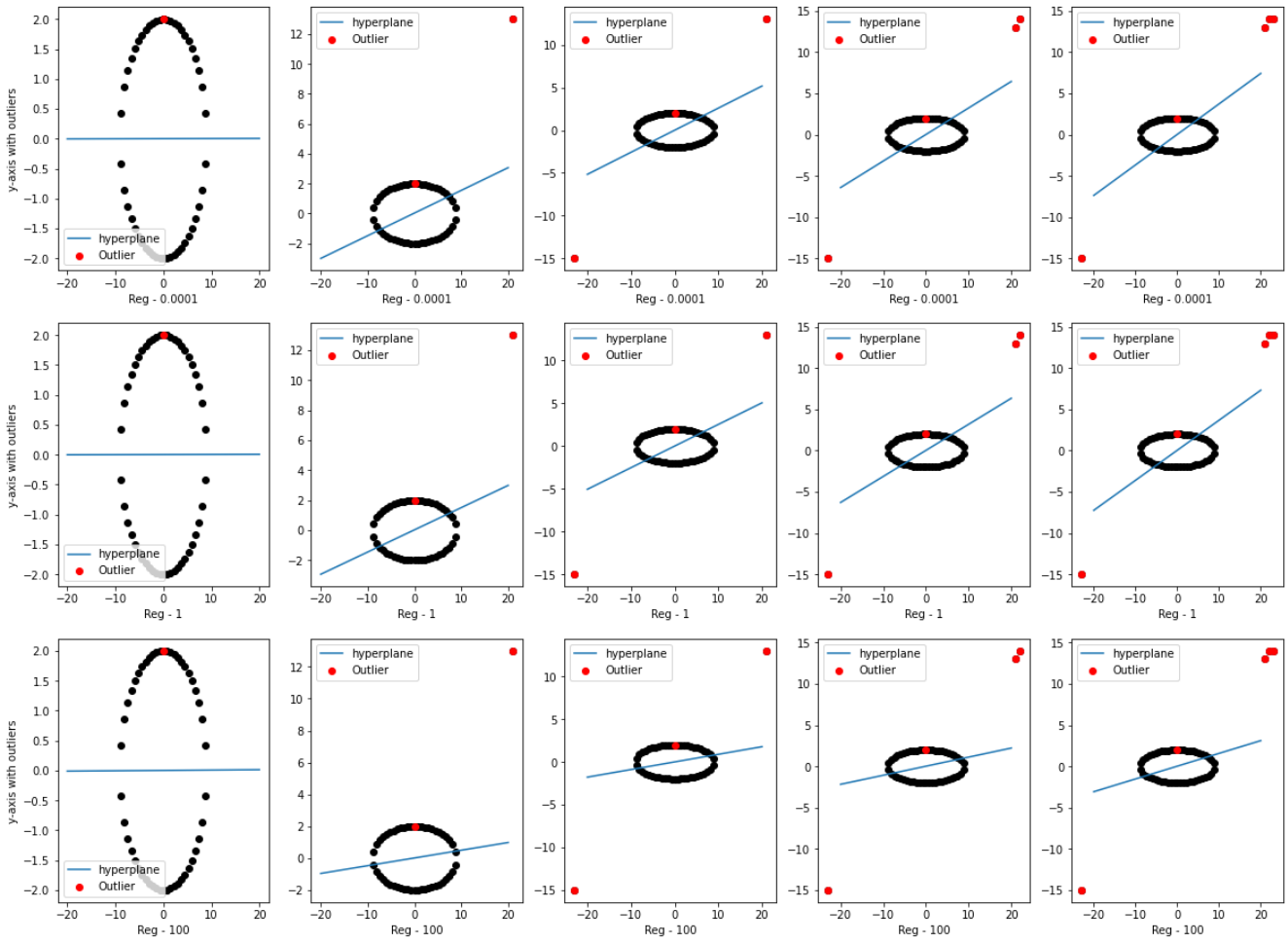
10. MAKE SURE YOU WRITE THE DETAILED OBSERVATIONS, PLEASE CHECK THE LOSS FUNCTION IN THE SKLEARN DOCUMENTATION
(please do search for it).

In [180]:

```
def draw_line(coef,intercept):
  xx=np.linspace(-20,20)
  y=coef*xx+intercept

  plt.plot(xx,y,label='hyperplane')
```

In [181]:

```
Reg = [0.0001,1,100]
Outliers = [(0,2), (21,13), (-23, -15), (22,14), (23, 14)]
#print(X.shape)
X = X.reshape(-1,1)
#print(X.shape)

plt.figure(figsize=(20,15))
plt.title("The impact of outliers and regularization on data.")
for i,j in enumerate(Reg):
  out_x=[]
  out_y=[]
  for k,l in enumerate(Outliers):
    X = np.vstack((X,np.array(l[0]).reshape(1,-1)))
    Y = np.hstack((Y,l[1]))

    out_x.append((l[0]))
    out_y.append((l[1]))

    Regress = SGDRegressor(loss = 'squared_loss',alpha= j, eta0=0.0001, learning_rate='constant',ra
ndom_state=0)
    Regress.fit(X,Y)
    w = Regress.coef_
    b = Regress.intercept_

    plt.subplot(len(Reg),len(Outliers),len(Outliers)*i+(k+1))
    plt.scatter(X,Y,color='k')
    plt.scatter(out_x,out_y,color='red',label='Outlier')

    draw_line(w,b)
    plt.legend()
    plt.xlabel('Reg - '+str(j))
    if(k==0):
      plt.ylabel('y-axis with outliers')
  X = X[:-5]
  Y = Y[:-5]


plt.show()
```



1. When Regularization term α increases, the influence of Regularization increases and model gives less importance for loss minimization.
2. When $\alpha = 0.0001$ regularization influence is less. So, the model tries to reduce the loss as much as possible and tries to fit the best possible hyperplane(line in this case).
3. Thus we can observe from the plots of row 1($\alpha = 0.0001$) hyperplane(line) slope changes when ever an outlier is added. The slope of line is changed in such a way that line is close to the outliers and making less error best fitted line.
4. we can observe from plots of cell[1][3] and cell[1][5] that as more outliers are added to the right top corner of the normal points, the slope of line has increased and resulting in less error fitted line.
5. when $\alpha = 100$ the regularization term influence is large. Thus the influence of loss minimization is less.
6. We can observe from plots of cell[3] : that the impact of outliers is not high and slope of hyperplane(line) is also low. Even though more outliers are added to the top right corner of normal points, the slope of line is not changed by a large value. This is mainly due to high value of $\alpha(regularization)$
7. From cell[1][5] and cell[3][5], we can observe for the same outliers(data) the slope of hyperplane is high when $\alpha = 0.0001$ and slope of hyperplane is low when $\alpha = 100$. This is mainly due to difference in α values and thereby difference in regularization influence.

Conclusions:

1. Regularization strength α impacts robustness of hyperplane or decision boundary due to outliers.
2. When α is low, hyperplane is easily impacted by outliers and model is outlier prone.
3. When α is high, hyperplane is robust to outliers and model is robust. But we loose the main objective of loss minimization.
4. So we have to choose moderate α value, so that we achieve the objectives of obtaining both loss minimization and robustness to outliers.