8C_LR_SVM-1

November 18, 2020

0.1 Task-C: Regression outlier effect.

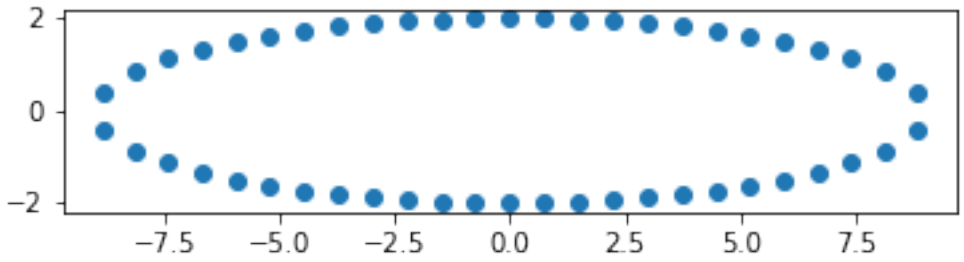
Objective:Visualization best fit linear regression line for different scenarios

[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
```

[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
```

[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()
```

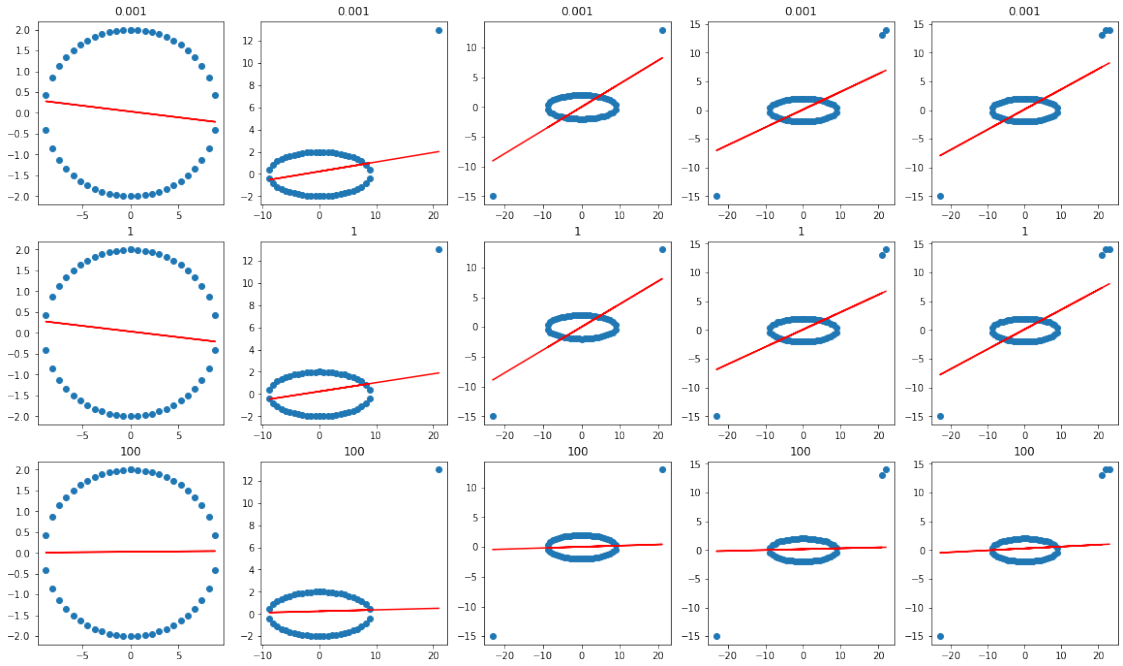


[4]:
```
X= b * np.sin(phi)
Y= a * np.cos(phi)
```

[5]:
```
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(X.reshape(-1,1))

X = X.tolist()

X = [i[0] for i in X]
Y = Y.tolist()
```

[6]:
```
hypers = [0.001, 1, 100]
plt.figure(figsize=(20,16))

for j, lr in enumerate(hypers):
    outlier_points= [(0,2),(21, 13), (-23, -15), (22,14), (23, 14)]
    X= b * np.sin(phi)
    Y= a * np.cos(phi)

    for c, k in enumerate(range(5*j+1, 5*(j+1)+1)):
        X = np.append(X, outlier_points[c][0])
        Y = np.append(Y, outlier_points[c][1])

        clf= SGDRegressor(alpha=lr, random_state=12).fit(X.reshape(-1, 1), Y)

        Y_pred= clf.predict(X.reshape(-1, 1))
        plt.subplot(4, 5, k)
```

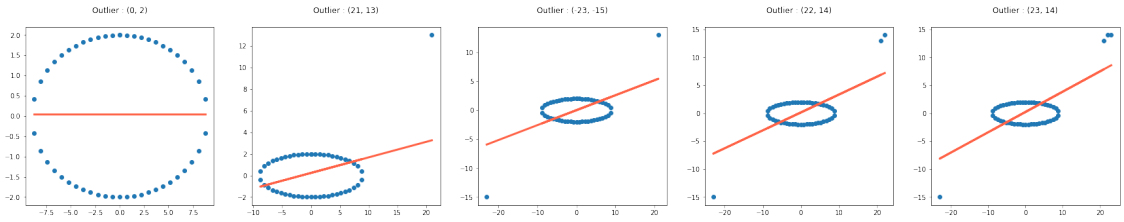
```
        plt.scatter(X, Y)
        plt.plot(X, Y_pred, color='red')
        plt.title("c-loss : ", str(lr))
plt.show()
```



[9]:
```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression

alphas = [0.001, 1, 100]
outliers=[(0,2),(21, 13), (-23, -15), (22,14), (23, 14)]
count=1

fig = plt.figure(figsize=(30,5))

print('#' * 50 + "\n" + '#' * 15 + "   LINEAR - REGRESSOR  " + "#"*15 + "\n" +␣
 ↪"#"*50)

data = b * np.sin(phi)
target = a * np.cos(phi)

for index, j in enumerate(outliers):
    x,y = j
    x = np.array(x).reshape(1,-1)
    y = np.array(y).reshape(1,-1)
```

```
    data = np.append(data, x).reshape(-1,1)
    target = np.append(target, y)

    reg  = LinearRegression()

    reg.fit(data, target)
    pred = reg.predict(data)

    intercept = reg.intercept_
    coeff = reg.coef_

    fig.add_subplot(1, 5, count)
    plt.scatter(data, target, color="tab:blue")
    plt.plot(data, pred, color="tomato", linewidth=3)
    plt.title("Outlier : " + str(j) + '\n')
    count+=1
fig.show()
```

```
#####################################################
###############   LINEAR - REGRESSOR  ###############
#####################################################
```



0.1.1 OBSERVATION :

1. If a datapoint is outlier and the alpha is less or normal, the model get impacted by outlier very large.

2. If a datapoint is outlier, and the alpha is very large, the model get impacted by outlier not much.

3. Linear Regressor is strongly affected by outliers.