Plot

Drawing the points

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (8,6))
plt.scatter(X[:, 0], y, color = 'b', marker = 'o', s = 30)
plt.show()
```

Plot fitted line and data

```
def draw_line (X_train, y_train, X_test, y_test, y_predicted):
    cmap = plt.get_cmap('viridis')
    fig = plt.figure(figsize = (8,6))
    m1 = plt.scatter(X_train, y_train, color = cmap(0.9), s =10)
    m2 = plt.scatter(X_test, y_test, color = cmap(0.5), s =10)
    plt.plot(X_test, y_predicted, color = 'black', linewidth = 2, label =
'prediction')
    plt.show()

draw_line(X_train, y_train, X_test, y_test, predicted)
# predicted = regressor.predict(X_test)
```

Finding corelation

```
def scatter(x,fig):
    plt.subplot(5,2,fig)
    plt.scatter(data[x],data['price'])
    plt.title(x+' vs Price')
    plt.ylabel('Price')
    plt.xlabel(x)

plt.figure(figsize=(15,30))

scatter('carlength', 1)
scatter('carwidth', 2)
scatter('carheight', 3)
```

Plot cost over time

```
import plotly.graph_objects as go
import numpy as np

y = clf_no_reg.loss_log # cost

fig = go.Figure()
fig.add_trace(go.Scatter(x=np.arange(start =1, stop = len(y)),
y=clf_no_reg.loss_log ,
                    mode='lines+markers',
                    name='No Regularization'))

fig.add_trace(go.Scatter(x=np.arange(start =1, stop = len(y)),
y=clf_reg_10.loss_log ,
                    mode='lines+markers',
                    name='Regulrization W=10'))

fig.add_trace(go.Scatter(x=np.arange(start =1, stop = len(y)),
y=clf_reg_20.loss_log ,
                    mode='lines+markers',
                    name='Regulrization W=20'))

fig.add_trace(go.Scatter(x=np.arange(start =1, stop = len(y)),
y=clf_reg_40.loss_log ,
                    mode='lines+markers',
                    name='Regulrization W=40'))


fig.update_layout(title = "Error Plot over Iterations", title_x = 0.5,
                  xaxis_title = 'Iteration',
                  yaxis_title = 'Log Loss',
                  width = 800,
                  height = 500)
fig.show()
```

Show malignent/benign count

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data,
test_size=0.20, random_state=42)

# Train and Test Data Summary
import plotly.graph_objects as go
split = ['Train','Test']
```

```
fig = go.Figure()
fig.add_trace(go.Bar(x=split, y=[np.sum(y_train), np.sum(y_test)],#
base=[-500,-600],
                     marker_color='crimson',
                     name='Malignant'))
fig.add_trace(go.Bar(x=split,
                      y=[len(y_train)- np.sum(y_train), len(y_test) -
np.sum(y_test)],
                     base=0,
                     marker_color='lightgreen',
                     name='Benign'                 ))
fig.update_layout(width = 800, height = 400)
fig.update_layout(title = 'Count of Samples in Train and Test Split',
title_x = 0.5, xaxis_title = "Category", yaxis_title = 'Sample Count')
fig.show()
```

Accuracy calculation

```
from sklearn import metrics
print(f'Accuracy {metrics.accuracy_score(y_test,
clf_no_reg.predict(X_test))*100}%')
```

Plot confusion matrix

```
import seaborn as sns
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, clf_no_reg.predict(X_test)))
sns.heatmap(confusion_matrix(y_test, clf_no_reg.predict(X_test)),
annot=True)
```

Plot image of each class

```
images = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for i in range (0,len(X)):
    images[int(y[i])] = X[i]

import matplotlib.pyplot as plt

for i in range (0, len(images)):
    plt.subplot(2, 5, i+1)
    image = np.reshape(images[i], (28, 28))
    plt. imshow(image, cmap='gray')
```

```
    plt.title(f'Label: {i}')
plt.show()
```