```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix


#Step2: Get data
x= np.arange(10).reshape(-1,1)
y=np.array([0,1,0,0,1,1,1,1,1,1])
print(x)
print(y)
```

```
[[0]
 [1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
[0 1 0 0 1 1 1 1 1 1]
```

```
#Step 3: Create a model and train it
model= LogisticRegression(solver='liblinear', C=10.0, random_state=0)
model.fit(x,y)
```

```
LogisticRegression(C=10.0, random_state=0, solver='liblinear')
```

```
#Step 4: Evaluate the model
p_pred = model.predict_proba(x) #sigmoid function
print('p_pred',p_pred)
y_pred = model.predict(x)
print('y_pred',y_pred)
score = model.score(x,y)
print('score',score)
conf_m = confusion_matrix(y, y_pred)
print('conf_m',conf_m)
report = classification_report(y,  y_pred)
print('report',report)
```

```
print('report',report)
```

```
p_pred [[0.81999686 0.18000314]
 [0.69272057 0.30727943]
 [0.52732579 0.47267421]
 [0.35570732 0.64429268]
 [0.21458576 0.78541424]
 [0.11910229 0.88089771]
 [0.06271329 0.93728671]
 [0.03205032 0.96794968]
 [0.0161218  0.9838782 ]
 [0.00804372 0.99195628]]
y_pred [0 0 0 1 1 1 1 1 1 1]
score 0.8
conf_m [[2 1]
 [1 6]]
report               precision    recall  f1-score   support

           0       0.67      0.67      0.67         3
           1       0.86      0.86      0.86         7

    accuracy                           0.80        10
   macro avg       0.76      0.76      0.76        10
weighted avg       0.80      0.80      0.80        10
```

```
print('intercept',model.intercept_)
print('coef:',model.coef_,end='\n\n')
```

```
intercept [-1.51632619]
coef: [[0.703457]]
```

```
#KNN algo
from matplotlib import pyplot as plt #[plot graphs]
from sklearn.datasets import load_breast_cancer #data set 'breast cancer'
from sklearn.neighbors import KNeighborsClassifier #KNN algo
from sklearn.model_selection import train_test_split #split the dataset
import seaborn as sns
sns.set()


breast_cancer = load_breast_cancer()
X = pd.DataFrame(breast_cancer.data, columns = breast_cancer.feature_names)
```

```
X = X[['mean area', 'mean compactness']]
y = pd.Categorical.from_codes(breast_cancer.target, breast_cancer.target_names)
y = pd.get_dummies(y, drop_first=True)


X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=1)


knn=KNeighborsClassifier(n_neighbors=5,metric='euclidean')
knn.fit(X_train, y_train)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/neighbors/_classification.py:198: DataConversionWarning: A column-vector y was pas
  return self._fit(X, y)
KNeighborsClassifier(metric='euclidean')
```

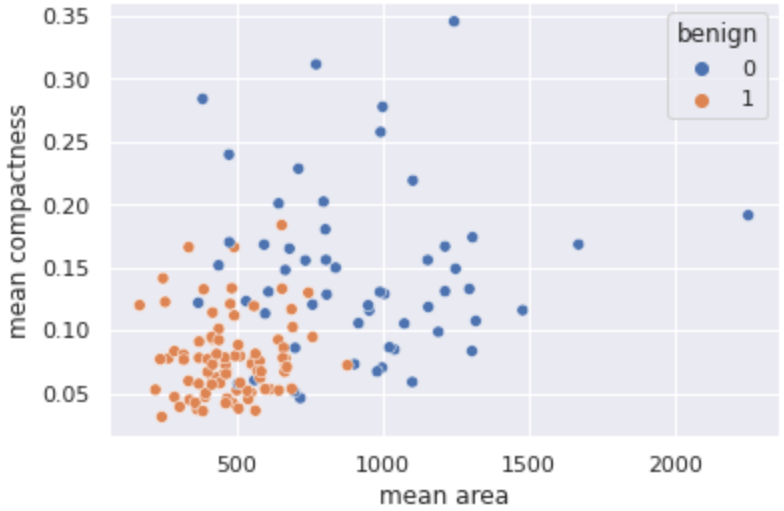
```
y_pred=knn.predict(X_test)
print(y_pred,'\n', y_test)
```

```
[1 1 1 0 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0
 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0
 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1
 1 0 1 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1 1 0 0 1]
      benign
421       1
47        0
292       1
186       0
414       0
..      ...
232       1
413       1
514       0
244       0
415       1

[143 rows x 1 columns]
```

```
sns.scatterplot(
    x='mean area',
    y='mean compactness',
    hue= 'benign',
```

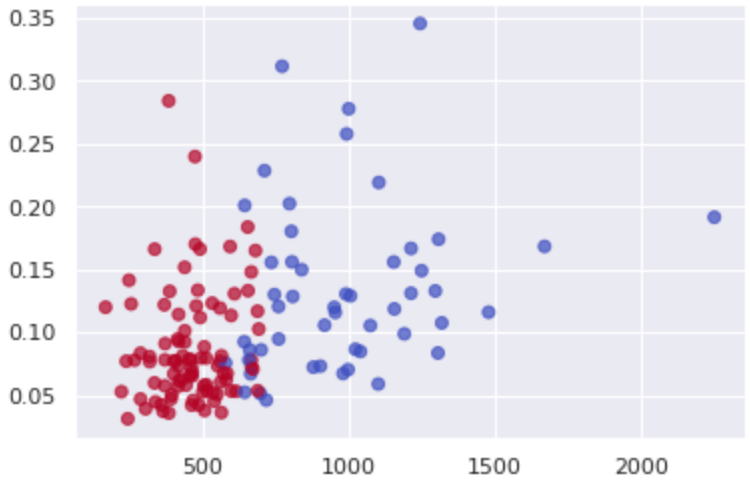
```
    data=X_test.join(y_test, how='outer')
)
```

```
<AxesSubplot:xlabel='mean area', ylabel='mean compactness'>
```

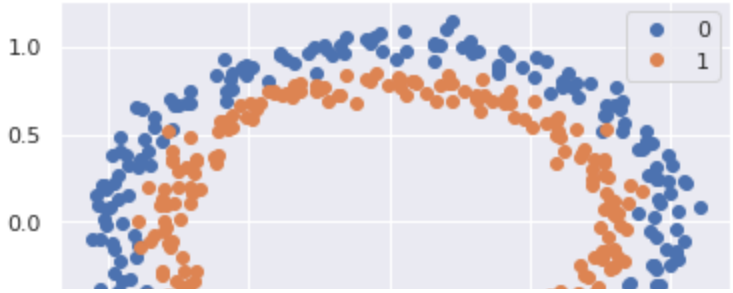



```
y_pred = knn.predict(X_test)
plt.scatter(
    X_test['mean area'],
    X_test['mean compactness'],
    c=y_pred,
    cmap='coolwarm',
    alpha=0.7
)
```

```
<matplotlib.collections.PathCollection at 0x7efc57c30fd0>
```



```
confusion_matrix(y_test,y_pred)

     array([[42, 13],
            [ 9, 79]])


#Support Vector Machines
from sklearn.datasets import make_circles
from sklearn import svm,metrics
def make_meshgrid(x,y,h=0.2):
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    return xx, yy
def plot_contours(ax, clf, xx, yy, **params):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out


samples = 500
train_prop = 0.8

#Make Data
x, y = make_circles(n_samples = samples, noise = 0.05, random_state=123)

#Plot
df = pd.DataFrame(dict(x=x[:,0], y=x[:,1], label = y))
groups = df.groupby('label')
fig,ax = plt.subplots()
ax.margins(0.05) #adds 5% padding to the autoscaling
for name, group in groups:
    ax.plot(group.x, group.y, marker='o', linestyle = '', ms=6, label=name)
ax.legend()
plt.show()
```




```
#Minmax scale
x= (x-x.min())/(x.max()-x.min())

#Linear
C = 1.0 #SVM regularization parameter
models = svm.SVC(kernel = 'linear', C=C)
models.fit(x,y)

#Title for the plots
titles = ('SVC with Linear kernel')
```