6. Assignment 3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Text Sans 10 Editing

Choose an appropriate dataset of your choice such that every record (example) has at least 5 features which are numeric in nature and there is at least one attribute (feature) which is binary in nature.

You can use the binary attribute as the binary target label to be predicted.

Split your dataset into a training set and a test set. You can try different splits: 70:30 (75% training, 30% testing), 80:20 or 90:10 split.

On the training set, train the following classifiers:

1. Half Space
2. Logistic Regression (using sikuli function)
3. SVM classifier (using a linear kernel)
4. SVM classifier (using a Polynomial kernel and a Gaussian kernel)
5. Logistic Regression using the SGD procedure

If your data is not linearly separable, then you will be required to use the soft SVM formulation. You can use the sikuli implementation of logistic regression and SVM in SciKit Learn.

Compare and analyze the results obtained by using the different classifiers. For the soft SVM formulation, you should compare the performance with the different values of the regularization parameter. Report the number of support vectors obtained for every dataset split. You should submit a report along with the code.

[27] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import svm
df = pd.read_csv('bill_authentication.csv')
# https://archive.ics.uci.edu/ml/datasets/Banknote+authentication

[28] df.shape

(1372, 5)

[29] df.head()

variance_of_wavelet_transformed, used as input.
skewness_of_wavelet_transformed, used as input.
curtosis_of_wavelet_transformed, used as input.
entropy_of_image, used as input.
class, used as the target. It can only have two values: 0 (false) or 1 (true).

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10945	0
3	3.45660	9.5228	-4.0112	-3.62940	0
4	0.32924	-4.4552	4.5718	-0.98880	0

[30] df.shape

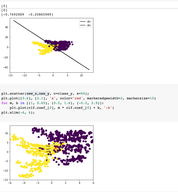
(1372, 5)

[31] df.class.value_counts()

```
0    762
1    610
Name: Class, dtype: int64
```

[32] labels = 'Positive', 'Negative'
num_pos = 610
num_neg = 762
explode = (0, 0.1)
figure, ptext,pcts = plt.pie(num_pos,
labels=labels, explode = explode,
startangle=90, shadow=True,
autopct='%1.1f%%')
plt.title('Class')
plt.show()

[33] # Import modules
%matplotlib inline
#import matplotlib
import matplotlib.pyplot as plt
# Define variable for the data to be our class
mes_pos=df[df.Class]
# Show the plot
plt.show()



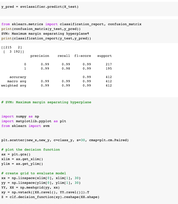
[34] # Defining features and target variable
y = df[['Class']]
X = df.drop(columns = ['Class'])
from sklearn.model_selection import train_test_split
#splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

[35] print(X_train.shape, X_test.shape)

(960, 4) (412, 4)

[36] #PAIRPLOT% Especially test
plt.rcParc
sns.set(rc={'figure'(5,4)})
plt.scatter(df['Variance'], df['Skewness'])
plt.xlabel('Variance')
plt.ylabel('Skewness')
plt.grid(True)
plt.show()

Output size 432x288 with 0 Axes



[37] # X_train, y_train
mes_p = X_train.Variance
mes_n = X_train.Skewness

points_0 = X_train[['Variance', 'Skewness']]
class_p = y_train
class_0 = class_p[class_p['Class', C == 1.0]
clf.fit(points_0,class_0)
print(clf.predict([[2.19,0.762]]))
print(clf.predict([[-1.773,-7.033]]))
# same way we can send our test set to predict all values
w = clf.coef_[0]
print(w)

a = -w[0] / w[1]

xx = points_0
yy = a * xx - clf.intercept_[0] / w[1]

h0 = plt.plot(xx, yy, 'k-', label='dis')

plt.scatter(xx,m_mes_p, c = class_p)
plt.legend()
plt.show()

[38] [1.]
[-1.773,-7.033]
[0.14292989 -0.23925393]

[39]

plt.scatter(mm_m_class_p, c=class_p, s=50)
xx = np.linspace(-5, 5, 30)
yy = np.linspace(-5, 5, 30)
plt.plot(clf.coef_[0], a * clf.coef_[0] + b, 'k')

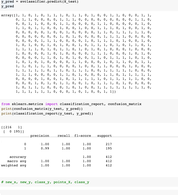
plt.show(w, b)

[40] [1.]

[41]

plt.scatter(mm_m_class_p, c=class_p, s=50)
xx = np.linspace(-5, 5, 30)
yy = np.linspace(-5, 5, 30)
plt.plot(clf.coef_[0], a * clf.coef_[0] + b, 'k')
plt.show(w, b)

[42] # Logistic Regression
%matplotlib inline
#import matplotlib
import matplotlib.pyplot as plt
# Define variable for the data to be our class
mes_pos=df[df.Class]
# Show the plot
plt.show()



[43] # Defining features and target variable
y = df[['Class']]
X = df.drop(columns = ['Class'])
from sklearn.model_selection import train_test_split
#splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

[44] print(X_train.shape, X_test.shape)

(960, 4) (412, 4)

[45] #PAIRPLOT% Especially test
plt.rcPC
sns.set(rc={'figure'(5,4)})
plt.scatter(df['Variance'], df['Skewness'])
plt.xlabel('Variance')
plt.ylabel('Skewness')
plt.grid(True)
plt.show()

Output size 432x288 with 0 Axes

[46] # X_train, y_train
mes_p = X_train.Variance
mes_n = X_train.Skewness

points_0 = X_train[['Variance', 'Skewness']]
class_p = y_train
class_0 = class_p[class_p['Class', C == 1.0]
clf.fit(points_0,class_0)
print(clf.predict([[2.19,0.762]]))
print(clf.predict([[-1.773,-7.033]]))
# same way we can send our test set to predict all values
w = clf.coef_[0]
print(w)

a = -w[0] / w[1]

xx = points_0
yy = a * xx - clf.intercept_[0] / w[1]

h0 = plt.plot(xx, yy, 'k-', label='dis')

plt.scatter(xx,m_mes_p, c = class_p)
plt.legend()
plt.show()

[47] [1.]
[-1.773,-7.033]
[0.14292989 -0.23925393]

[48]

plt.scatter(mm_m_class_p, c=class_p, s=50)
xx = np.linspace(-5, 5, 30)
yy = np.linspace(-5, 5, 30)
plt.plot(clf.coef_[0], a * clf.coef_[0] + b, 'k')

plt.show(w, b)

[49] [1.]

[50]

plt.scatter(mm_m_class_p, c=class_p, s=50)
xx = np.linspace(-5, 5, 30)
yy = np.linspace(-5, 5, 30)
plt.plot(clf.coef_[0], a * clf.coef_[0] + b, 'k')
plt.show(w, b)

[51] # Multivariate Gaussian
[52] from sklearn.linear_model import LogisticRegression
LrLogisticRegression()
Lr.fit(X_train,y_train)

LogisticRegression(C=1.0, class_weight=None, dualFalse, fit_intercepttrue,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver'lbfgs', tol=0.0001, verbose=0,
warm_startFalse)

[53] from sklearn.metrics import accuracy_score
y_pred=Lr.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
accuracy

0.9563106796116505

[57] # SVM classifier (using a linear kernel)
from sklearn.svm import SVC
svm=SVC(kernel='linear')
svm.fit(X_train, y_train)

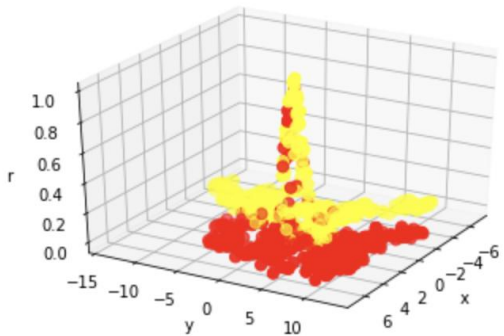
[58] y_pred = svm.predict(X_test)
y_pred

array([1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1

```
[46] r = np.exp(-(points_X ** 2).sum(1))
```

```
from mpl_toolkits import mplot3d
from ipywidgets import interact, interactive, fixed, interact_manual

ax = plt.subplot(projection='3d')
ax.scatter3D(points_X.Variance,points_X.Skewness, r, c=class_y, s=50, cmap='autumn')
ax.view_init(elev=30, azim=30)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('r')

# interact(plot_3D, elev=[-90, 90], azip=(-180, 180),
#          X=fixed(X), y=fixed(y));
```

```
Text(0.5, 0, 'r')
```