Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY
MNRAS **464**, 3796–3811 (2017) doi:10.1093/mnras/stw2606
Advance Access publication 2016 October 12

Pattern recognition in the ALFALFA.70 and Sloan Digital Sky Surveys: a catalogue of ∼500 000 H I gas fraction estimates based on artificial neural networks

Hossen Teimoorinia,[1★] Sara L. Ellison[1] and David R. Patton[2]

Loading packages

In [1]:
```
%matplotlib inline
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from pandas.plotting import scatter_matrix
from sklearn.neighbors import KNeighborsRegressor
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn import linear_model
import sys
from sklearn.neighbors import KNeighborsClassifier
from scipy.interpolate import interp1d
from matplotlib import pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, model_selection, svm
from sklearn.linear_model import LogisticRegression
print('Done!')
```

```
Done!
```

In [2]:
```
X=np.load('inp_alfalfa.npy')  # Load the input
Y=np.load('tar_alfalfa.npy')  # Load the target

print ('Done!')
```

```
Done!
```

Table 1. Galaxy variables used in training sets.

Input data	Description
M_*	Stellar mass
M_u	u-band absolute magnitude
M_g	g-band absolute magnitude
M_r	r-band absolute magnitude
M_i	i-band absolute magnitude
M_z	z-band absolute magnitude
$g - r$	Observed colour
μ_{*i}	i-band stellar mass density
SFR	Star formation rate
sSFR	Specific star formation rate
$M_{\rm Halo}$	Halo mass
δ_5	Local galaxy density
$\rm rhalf_r$	Half-light radius (kpc) in the r band
B/T	Bulge-to-total fraction in the r band
$\rm rd_{disc}$	Disc radius (kpc) in the r band

In [3]:
```
print(np.shape(X),np.shape(Y)) #Check the size and dimension of the input and
```

```
(13674, 15) (13674,)
```

In [4]:
```
#Y = np.reshape(Y, (-1, 1))
```

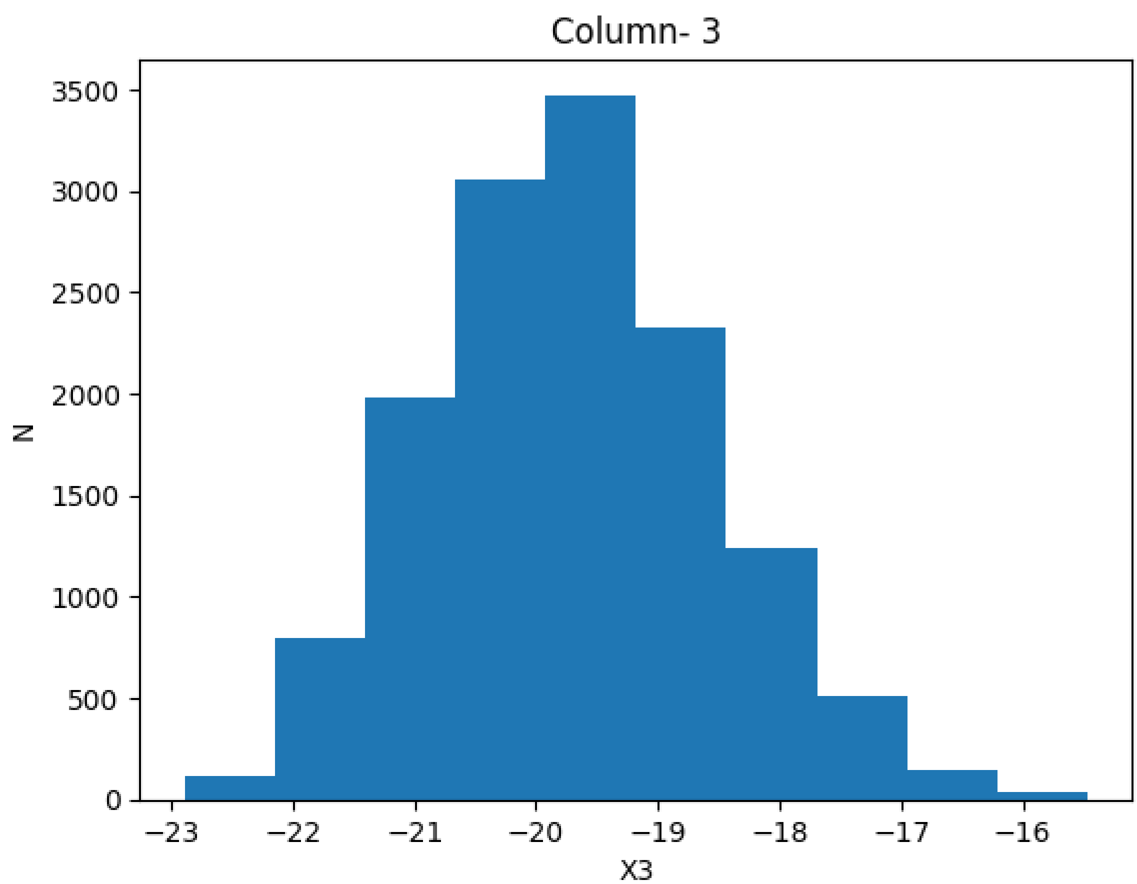
In [5]:
```
#print(np.shape(X),np.shape(Y))
```

Exploring the data and seeing the distribution of a selected column

In [ ]:
```

```

In [6]:
```
# Choose a column to see the distribution

n_column =3

plt.hist(X[:,n_column])
plt.title('Column- '+str(n_column))
plt.ylabel('N')
plt.xlabel("X"+str(n_column))
```

Out[6]: Text(0.5, 0, 'X3')



In [ ]:

Randomly separate samples to the training set (75%) and validation set (25%). The corresponding targets are also separated.

In [7]:
```
from sklearn.model_selection import train_test_split

X_tr,X_va,Y_tr, Y_va = train_test_split(X,Y ,test_size=0.25 )

print ('training set == ',np.shape(X_tr),np.shape(Y_tr),',, validation set ==
```

```
training set ==  (10255, 15) (10255,) ,, validation set ==  (3419, 15) (341
9,)
```

Normalization.

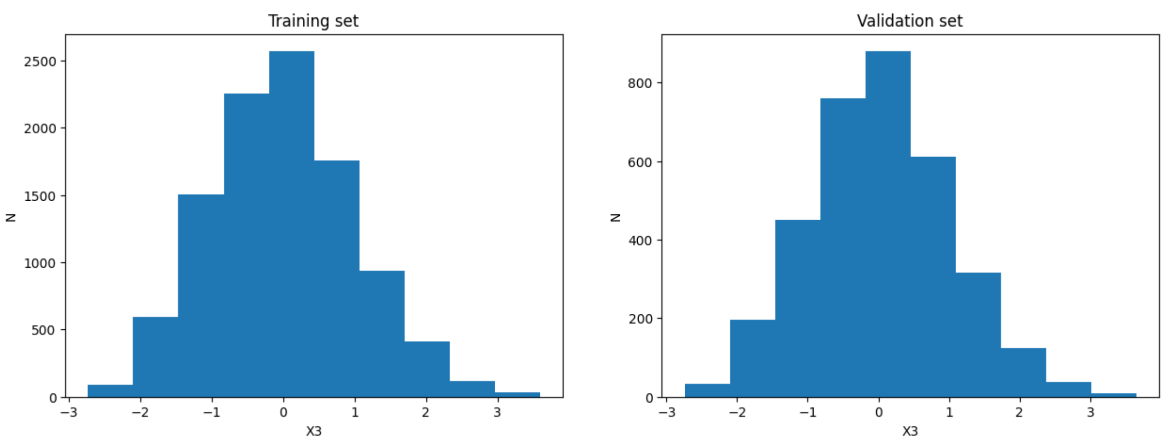
In [8]:
```
#Line #1: Import a model, for normalization, like StandardScaler (https://sci

#Line #2: fitting (finding the parameters of the model based on the training

#Line #3: Predicted (transformed) values for the training set

#Line #4: Predicted (transformed) values for the validation set (using the mo



scaler_S= StandardScaler().fit(X_tr)  # line #2
X_tr_Norm= scaler_S.transform(X_tr) # line # 3

X_va_Norm= scaler_S.transform(X_va)  # Line #4

print('Done!')
```

```
Done!
```

Comparing the distributions from the nomalized training and validation sets

In [9]:
```python
n_column = 3

fig = plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist(X_tr_Norm[:,n_column])
plt.title('Training set')
plt.ylabel('N')
plt.xlabel("X"+str(n_column))


plt.subplot(1, 2, 2)
plt.hist(X_va_Norm[:,n_column])
plt.title('Validation set')
plt.ylabel('N')
plt.xlabel("X"+str(n_column))
```

Out[9]:
```
Text(0.5, 0, 'X3')
```



In [10]:
```python
# Change the shape of the target (if you have a one-component  target )
# Y=np.reshape(Y,-1)
# print(np.shape(X),np.shape(Y))
```

Parameters:	**C : *float, default=1.0*** Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty. **kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'*** Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`. **degree : *int, default=3*** Degree of the polynomial kernel function ('poly'). Ignored by all other kernels. **gamma : *{'scale', 'auto'} or float, default='scale'*** Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. • if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma, • if 'auto', uses 1 / n_features. *Changed in version 0.22:* The default value of `gamma` changed from 'auto' to 'scale'. **coef0 : *float, default=0.0*** Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

The *kernel function* can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma\langle x, x' \rangle + r)^d$, where d is specified by parameter `degree`, r by `coef0`.
- rbf: $\exp(-\gamma\|x - x'\|^2)$, where γ is specified by parameter `gamma`, must be greater than 0.
- sigmoid $\tanh(\gamma\langle x, x' \rangle + r)$, where r is specified by `coef0`.

In [ ]:

```

```

For more information:

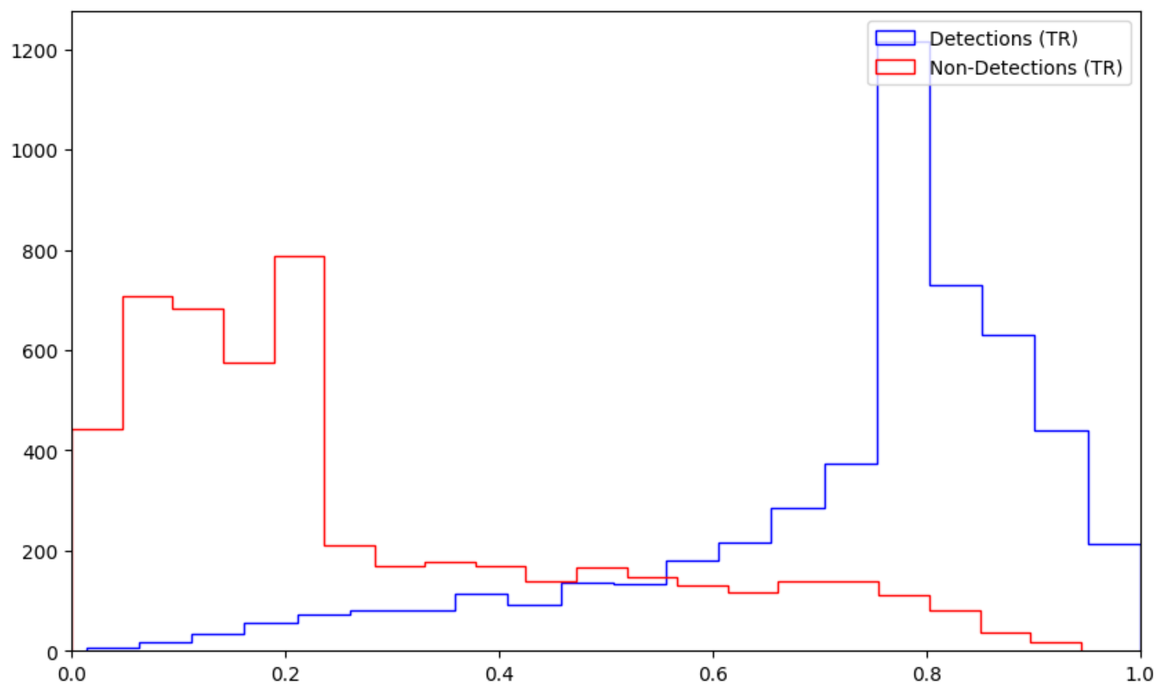
[https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html)](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html)

In [11]:

```
from sklearn.svm import SVC



cls= SVC(C=100, kernel='rbf', degree=3,  probability=True)


cls.fit (X_tr_Norm,Y_tr)  # fit the model with training set



## predict the response for tr and va sets. We can have two outputs: probabil
Y_tr_prob = cls.predict_proba(X_tr_Norm)[:,1]
Y_tr_pred = cls.predict(X_tr_Norm)

Y_va_prob = cls.predict_proba(X_va_Norm)[:,1]
Y_va_pred = cls.predict(X_va_Norm)


idx_tr_1 = (Y_tr==1)
idx_tr_0 = (Y_tr==0)


idx_va_1 = (Y_va==1)
idx_va_0 = (Y_va==0)
```

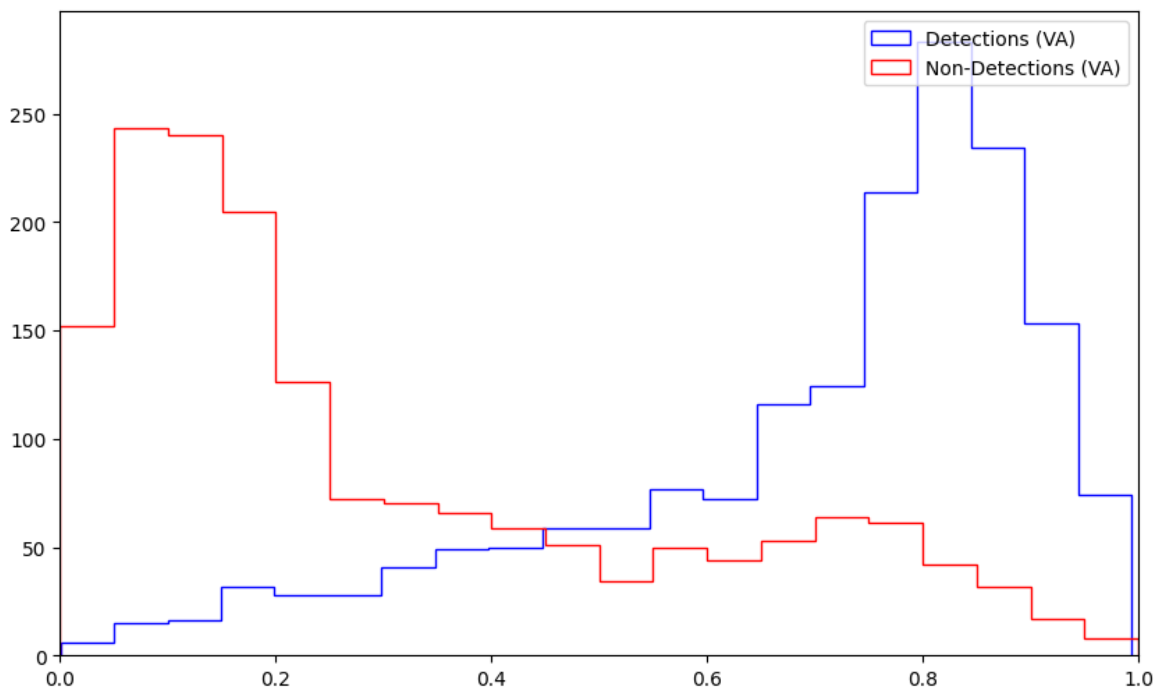
In [12]:
```python
plt.figure(figsize=(10, 6))

plt.figure(1)
plt.hist(Y_tr_prob[idx_tr_1],20,histtype='step',color = "blue", label='Detect

plt.xlim([0,1])
plt.legend()


plt.figure(1)
plt.hist(Y_tr_prob[idx_tr_0],20,histtype='step',color = "red",label='Non-Dete

plt.xlim([0,1])

plt.legend()
```

Out[12]: <matplotlib.legend.Legend at 0x20adf7ce3d0>



In [13]:
```
plt.figure(figsize=(10, 6))

plt.figure(1)
plt.hist(Y_va_prob[idx_va_1],20,histtype='step',color = "blue", label='Detect

plt.xlim([0,1])
plt.legend()


plt.figure(1)
plt.hist(Y_va_prob[idx_va_0],20,histtype='step',color = "red",label='Non-Dete

plt.xlim([0,1])

plt.legend()
```

Out[13]:
```
<matplotlib.legend.Legend at 0x20adf996d60>
```

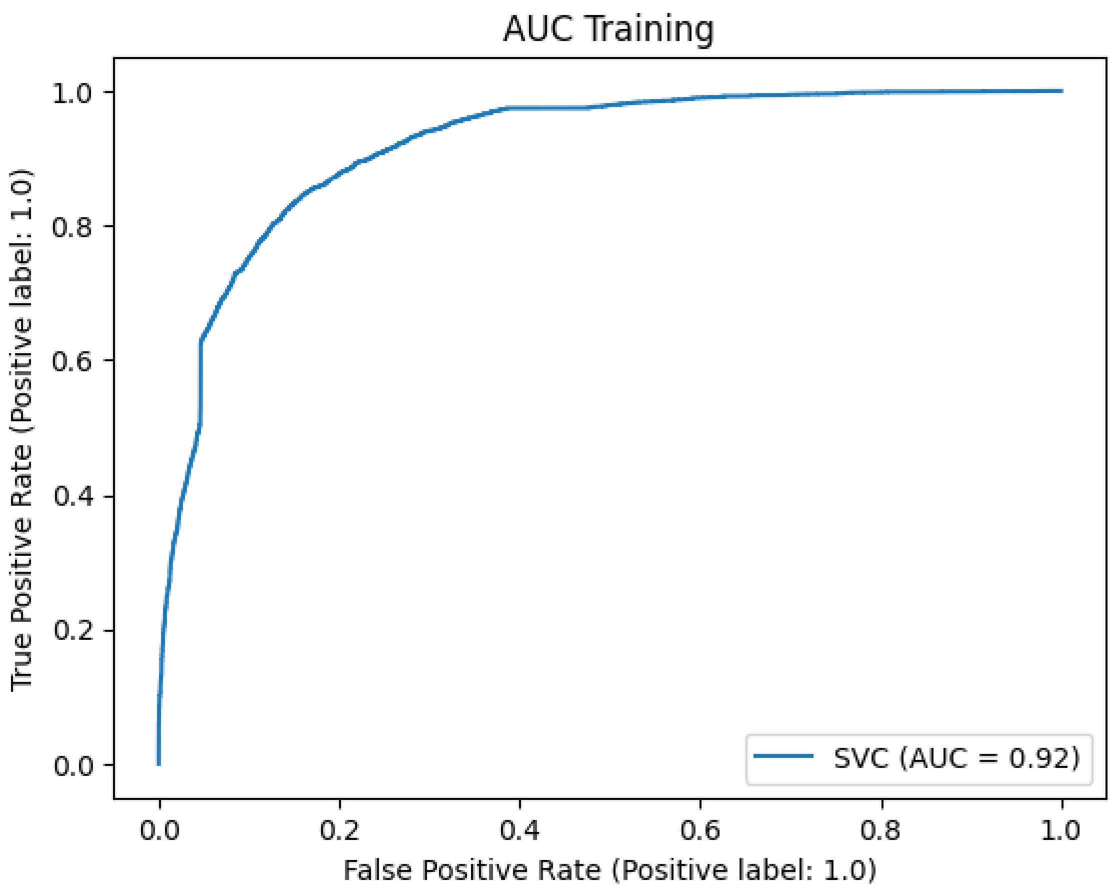


In [14]:

```
metrics.plot_roc_curve(cls, X_tr_Norm, Y_tr)
plt.title('AUC Training')
```

```
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function plot_roc_curve is deprecated; Function :fun
c:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: :meth:`sklearn.metrics.RocCurveDisplay.from_prediction
s` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

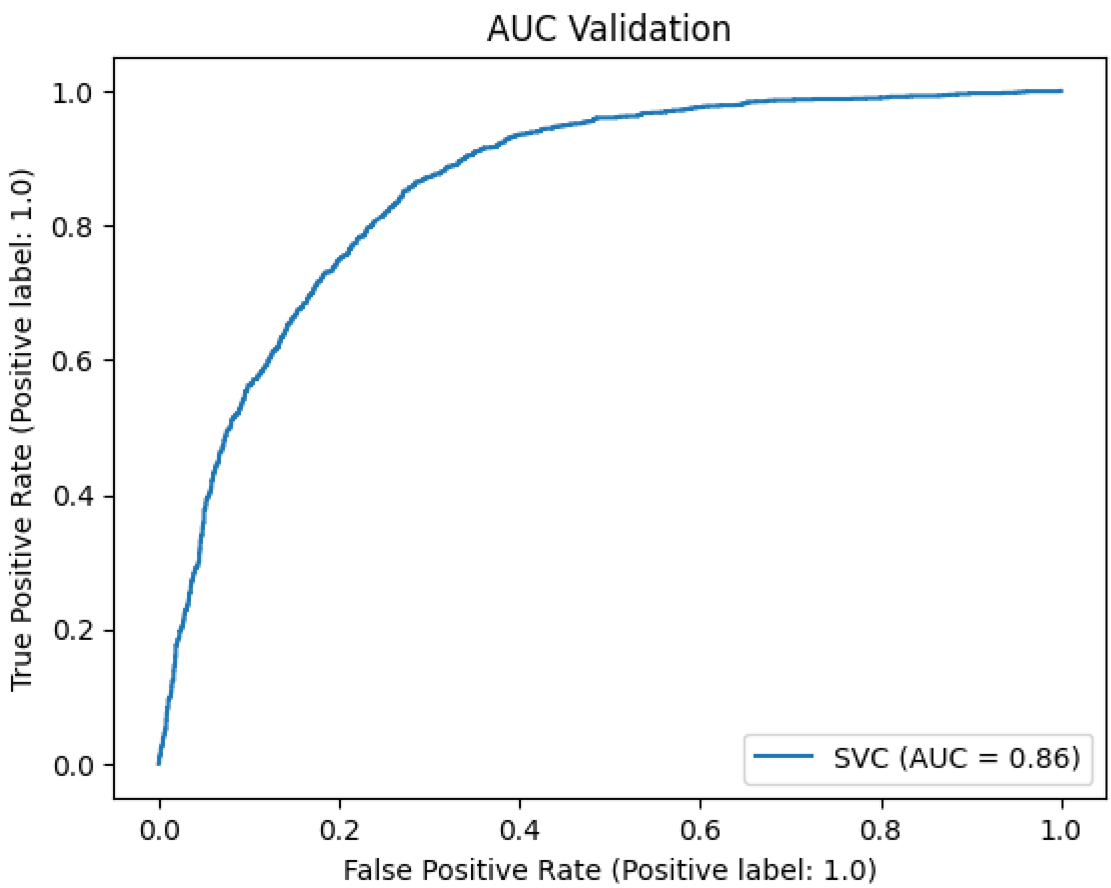
Out[14]: Text(0.5, 1.0, 'AUC Training')



In [15]:
```
metrics.plot_roc_curve(cls, X_va_Norm, Y_va)
plt.title('AUC Validation')
```

```
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function plot_roc_curve is deprecated; Function :fun
c:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: :meth:`sklearn.metrics.RocCurveDisplay.from_prediction
s` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

Out[15]:
```
Text(0.5, 1.0, 'AUC Validation')
```



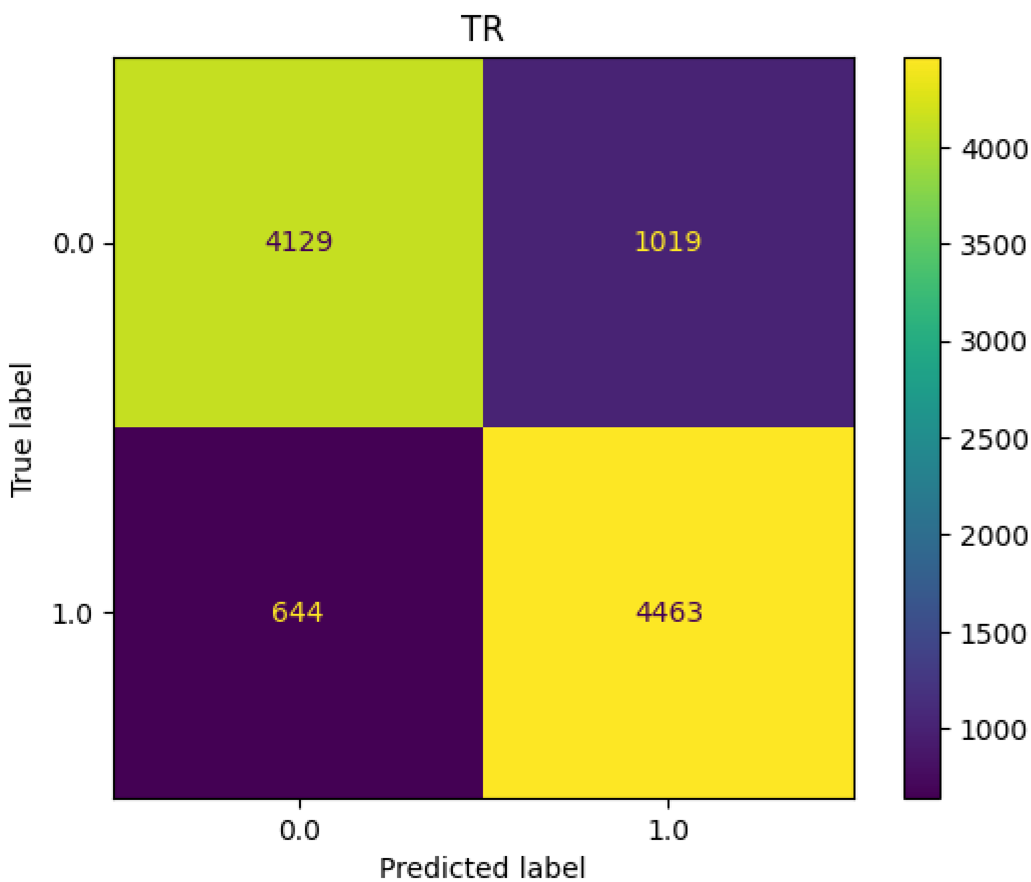
In [16]:

```
plot_confusion_matrix(cls, X_tr_Norm, Y_tr)
plt.title('TR')
```

```
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Functi
on `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2.
Use one of the class methods: ConfusionMatrixDisplay.from_predictions or Con
fusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[16]:

```
Text(0.5, 1.0, 'TR')
```

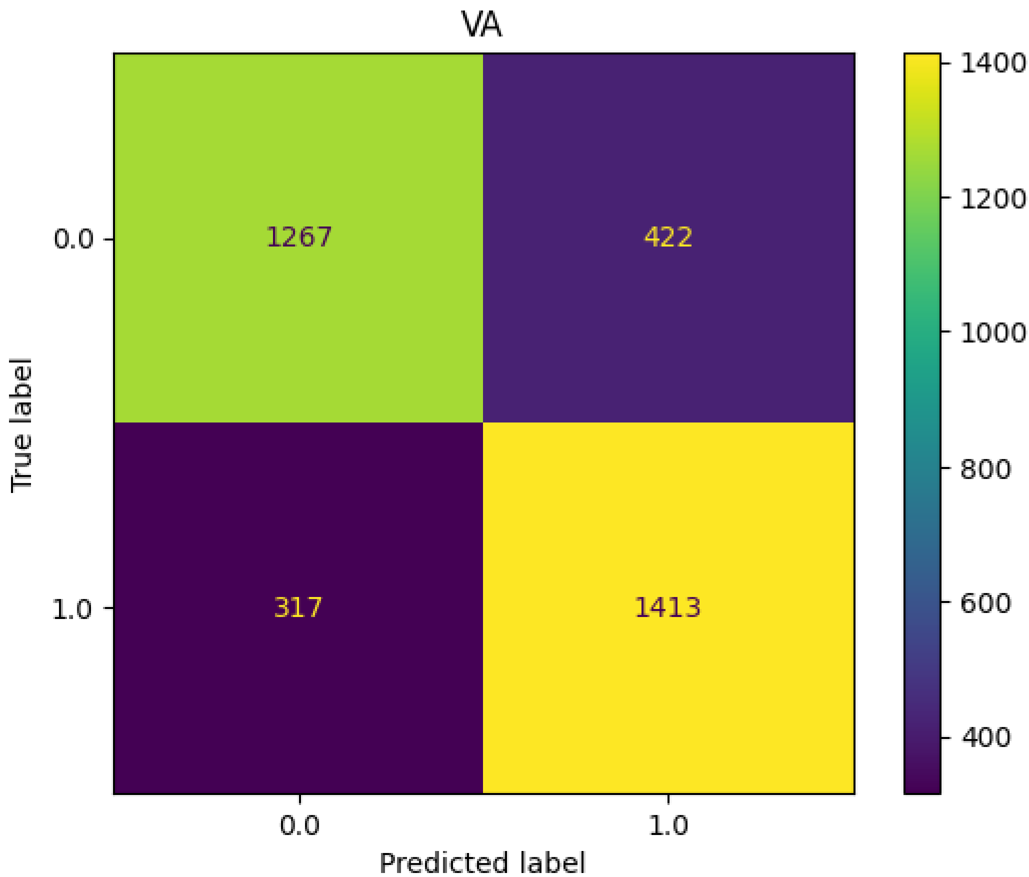


In [17]:

```
plot_confusion_matrix(cls, X_va_Norm, Y_va)
plt.title('VA')
```

```
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Functi
on `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2.
Use one of the class methods: ConfusionMatrixDisplay.from_predictions or Con
fusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

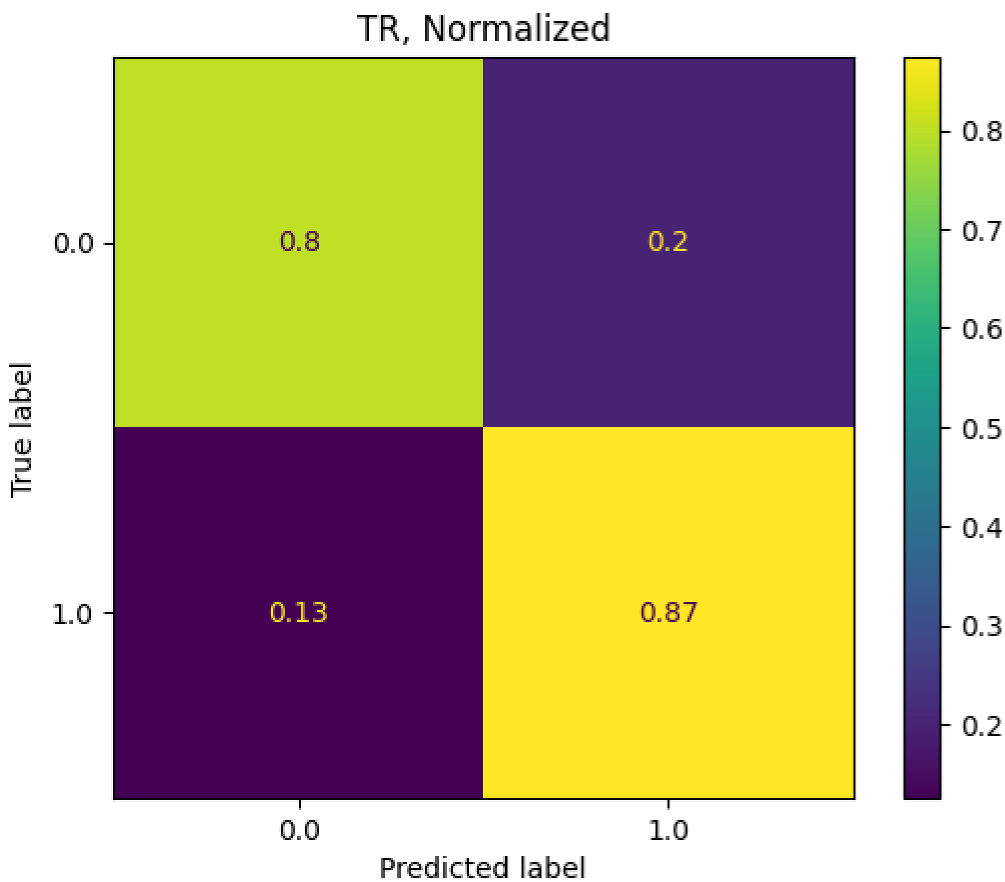
Out[17]: `Text(0.5, 1.0, 'VA')`



In [18]:
```
plot_confusion_matrix(cls, X_tr_Norm, Y_tr,normalize='true')
plt.title('TR, Normalized')
```

```
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Functi
on `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2.
Use one of the class methods: ConfusionMatrixDisplay.from_predictions or Con
fusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

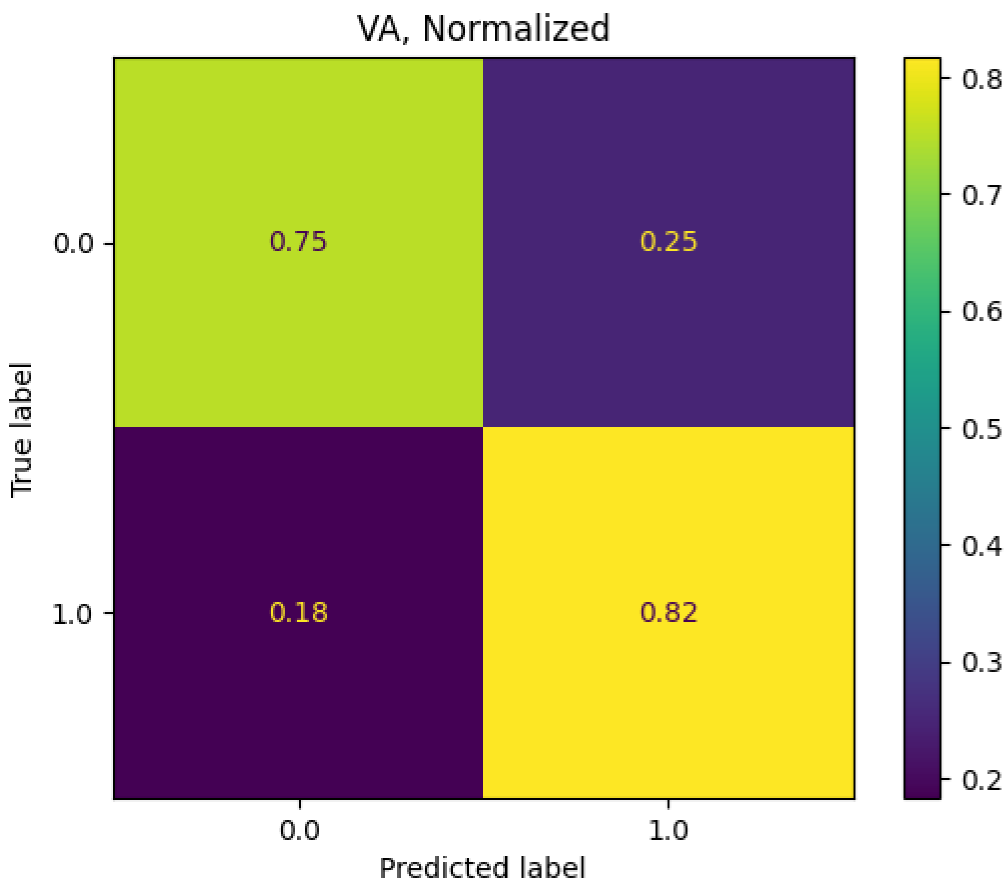
Out[18]:
```
Text(0.5, 1.0, 'TR, Normalized')
```



In [19]:
```python
plot_confusion_matrix(cls, X_va_Norm, Y_va,normalize='true')
plt.title('VA, Normalized')
```

```
C:\Users\hteim\anaconda3\envs\tf\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Functi
on `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2.
Use one of the class methods: ConfusionMatrixDisplay.from_predictions or Con
fusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[19]:
```
Text(0.5, 1.0, 'VA, Normalized')
```



In [20]:
```python
print ("Recall (TR) = ", metrics.recall_score(Y_tr, Y_tr_pred))
print ("Recall (VA) = ", metrics.recall_score(Y_va, Y_va_pred))
```

```
Recall (TR) =  0.8738985705893871
Recall (VA) =  0.8167630057803468
```

In [21]:
```python
print ("Accuracy (TR) = ", metrics.accuracy_score(Y_tr, Y_tr_pred))
print ("Accuracy (VA) = ", metrics.accuracy_score(Y_va, Y_va_pred))
```

```
Accuracy (TR) =  0.8378352023403218
Accuracy (VA) =  0.7838549283416204
```

In [22]:
```
print ("Precision (TR) = ", metrics.precision_score(Y_tr, Y_tr_pred))
print ("Precision (VA) = ", metrics.precision_score(Y_va, Y_va_pred))
```

```
Precision (TR) =  0.8141189346953667
Precision (VA) =  0.7700272479564033
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: