CS2TP

April 7, 2019

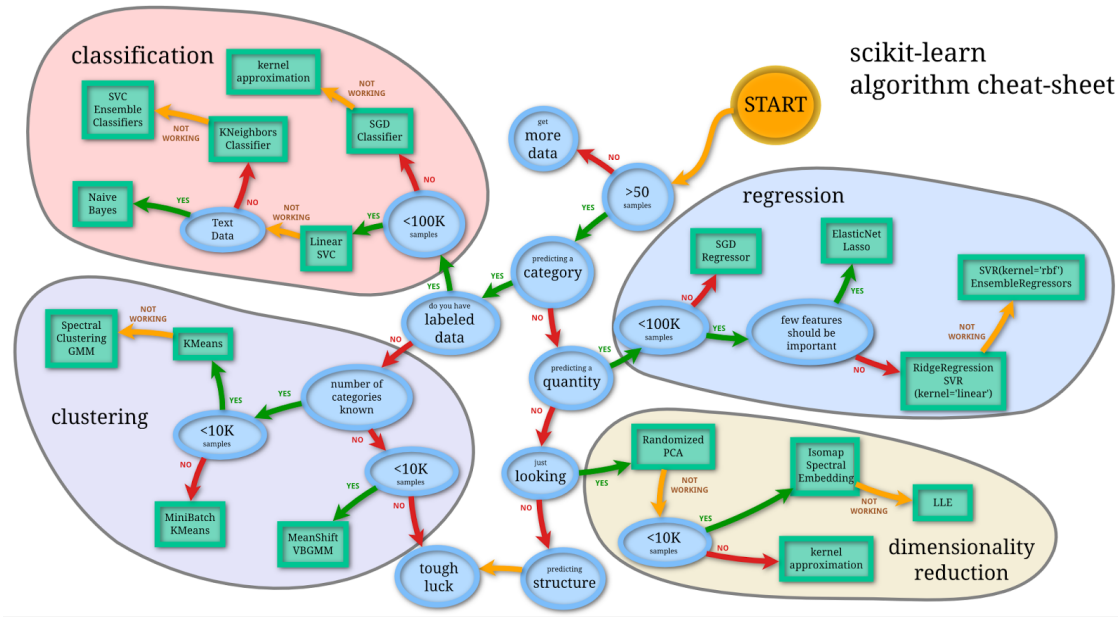
Pour lancer le notebook, aller dans un terminal et taper `jupyter notebook`. Dans le navigateur, ouvrir le fichier CS2TP.ipynb que vous aurez au préalable enregistré. Ensuite, à vous de découvrir `jupyter notebook`, par exemple : http://dichotomies.fr/2016/informatique/info1/cours/debuter-avec-les-notebooks/

Dans ce TP, nous allons utiliser 2 modules spécifiques au cours de data mining :

- `statsmodels` pour les modèles linéaires gaussiens et généralisés,
- `sklearn`pour les modèles d'apprentissage.

```
In [44]: from IPython.display import Image
         Image(filename="/home/labatte/Bureau /MASTER DS 1 ADD/CS2- RD4 AFD/sklearn.png",width=7
```

```
Out[44]:
```



On utilisera aussi les modules `pandas`et `seaborn` pour les manipulations de données statistiques et leurs exploitations graphiques.

Par convention: - X représente l'array des données explicatives, - y la variable à expliquer, - *data* le data frame panda $[X, y]$, - X_train, X_test y_train y_test les échantillons d'appentissage et de test, - y_pred les valeurs prédites par le modèle

1 Exercice 1 : Prise en main de sklearn

On simule deux nuages de points en 3 dimension. Dans chaque nuage de points, les observations sont des vecteurs gaussiens de loi:

- nuage 1 : $\mu 1 = (0,0,0)$ et $\Sigma_1 = \begin{bmatrix} 3 & 1 & 0.5 \\ 1 & 2 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}$
- nuage 2 : $\mu 2 = (3,3,3)$ et $\Sigma_2 = \Sigma_1$.

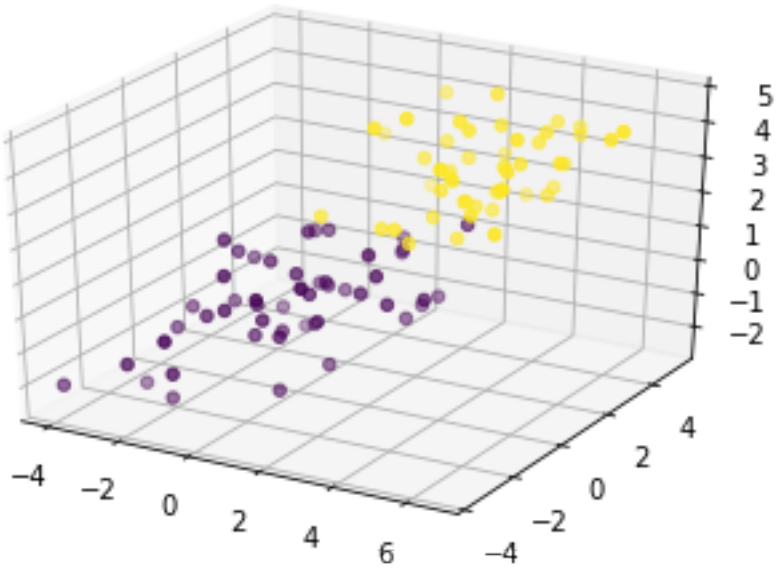
1.1 Question 1 :

Les instruction suivantes permettent la simulation de ces deux nuages. Interpréter les.

In [31]:
```
import matplotlib.pyplot as plt #graphique
import numpy as np # calcul matriciel
import pandas as pd # création de data.frame
import numpy.random as npr # stochastique
import scipy.stats as stats # stat
from math import pi,log,sqrt # fonction math
import seaborn as sns # graphiques avancés avec pandas


''' création de 3 nuages gaussiens en 3 dimensionsde même matrices de covariances
 1 : $\mu_1$ = [0,0,0] $\Sigma_1=$[[3,1,0.5],[1,2,0.5],[0.5,0.5,1]]
 2 : $\mu_1$ = [3,3,3] $\Sigma_2=\Sigma_1=$
'''
mu1=np.array([0,0,0])
mu2=np.array([3,3,3])
S=np.array([[3,1,0.5],[1,2,0.5],[0.5,0.5,1]])
A=np.linalg.cholesky(S)
A.dot(A.T)
X1=npr.randn(50,3).dot(A.T)+mu1
g1=np.mean(X1,0);g1
W1=np.cov(X1,rowvar=False);W1
X2=npr.randn(50,3).dot(A.T)+mu2
g2=np.mean(X2,0);g2
W2=np.cov(X2,rowvar=False);W2
X=np.concatenate((X1, X2))
y=np.concatenate((np.ones(50),np.ones(50)*2));y
from mpl_toolkits.mplot3d import Axes3D # graphique 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:,0],X[:,1], X[:,2], c=y)
```

Out[31]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7f338a59a8d0>
```

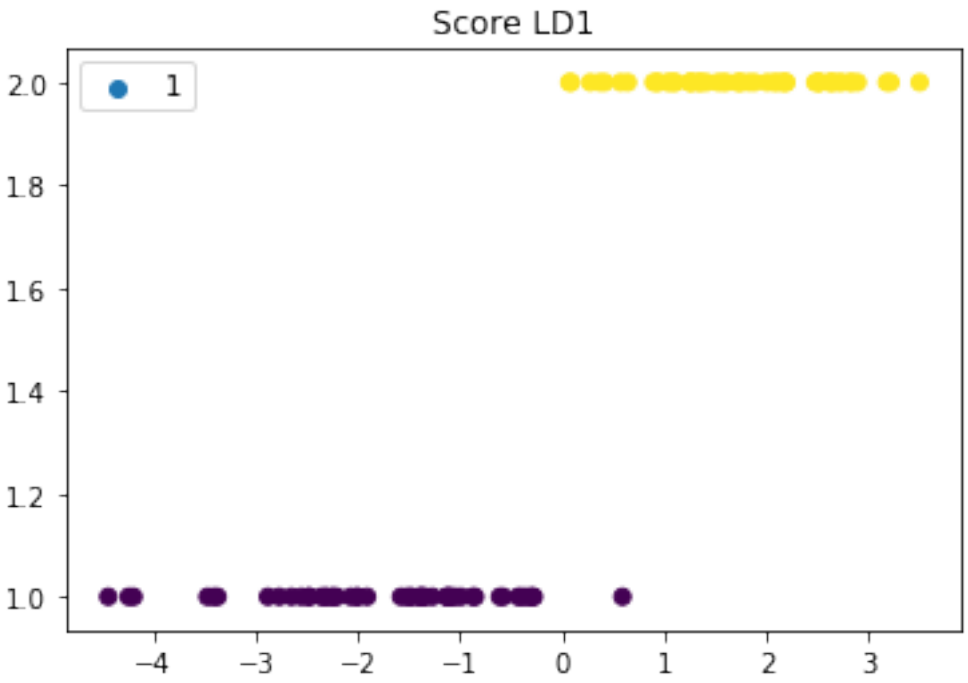


1.2 Question 2 :

On réalise maintenant l'analyse discriminante des données. Combien de variables discriminantes peut-on obtenir? Interpréter les commandes suivantes. Déterminer la fonction discriminante.

```
In [32]: ''' AFD et visualisation LD1
         '''
         from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
         lda = LinearDiscriminantAnalysis()
         lda.fit(X,y) #ajustement du modèle
         LD1=lda.transform(X) # création des LD avec le modèle
         plt.scatter(LD1,y,c=y) #stripchart
         plt.legend(['1','2'])
         plt.title('Score LD1')

Out[32]: Text(0.5, 1.0, 'Score LD1')
```



1.3 Question 3 :

On évalue maintenant la qualité de prédiction. Interpréter les instructions suivantes

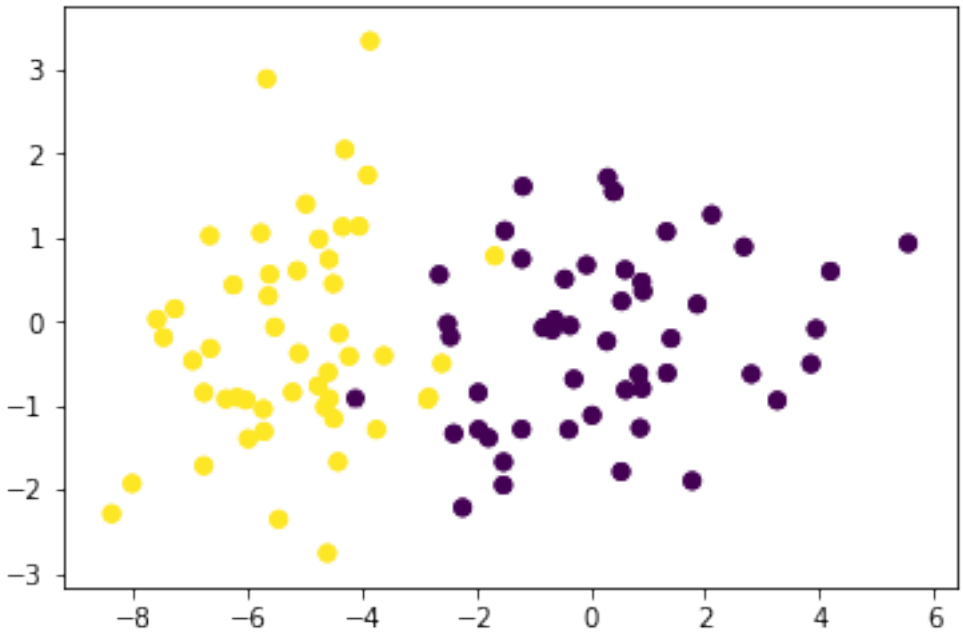
```
In [33]: y_pred=lda.predict(X)
         from sklearn.metrics import confusion_matrix,roc_curve,auc
         confusion_matrix(y,y_pred)
         lda.score(X,y)

Out[33]: 0.99
```

Question 4 : On réalise maintenant l'analyse en composantes principales des données pour une repésentation en 2D. Interpréter les commandes suivantes.

```
In [34]: from sklearn.decomposition import PCA
         from sklearn.preprocessing import scale
         pca = PCA(n_components=2)
         pca.fit(scale(X))
         pca.explained_variance_
         PC=pca.transform(X)
         plt.scatter(PC[:,0],PC[:,1],c=y)
         plt.scatter(PC[y==y_pred,0],PC[y==y_pred,1],c=y[y==y_pred])
         plt.scatter(PC[y!=y_pred,0],PC[y!=y_pred,1],c=y[y!=y_pred],marker='*')

Out[34]: <matplotlib.collections.PathCollection at 0x7f338a49e5c0>
```



1.4 Question 5:

reprendre l'exercice dans les deux cas suivants: - cas 1 : modifier μ_2, - cas 1 : les matrices de variance covariance ne sont plus égales, - cas 2 : 3 groupes au lieu de de deux.

2 Exercice 2 : un exemple concret : wine.

L'exemple wine décrit à l'aide de 13 variables quantitatives 3 classes de vins. Utiliser `wine['DESCR']` pour les légendes.

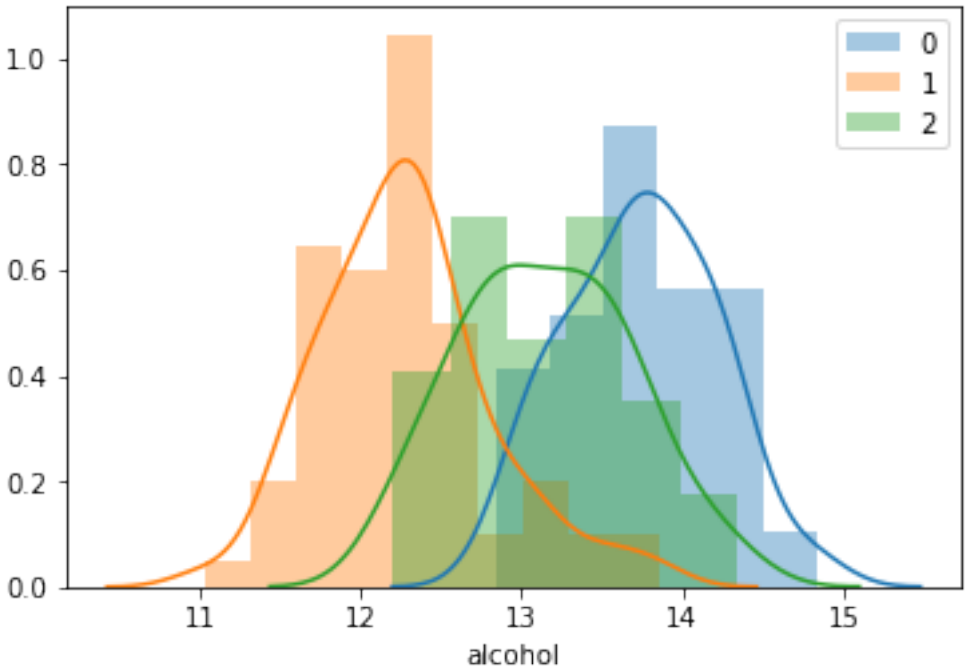
2.1 Question 1 : Données

Interpréter les instructions suivantes. Déterminer la variable initiale discriminant le plus les classes et représenter là.

```
In [45]: import sklearn as sk
         import matplotlib.pyplot as plt
         import numpy as np
         import pandas as pd
         import numpy.random as npr
         import scipy.stats as stats
         from math import pi,log,sqrt
         import seaborn as sns
         from sklearn import datasets
         wine = datasets.load_wine()
```

```
# print(wine['DESCR'])
X=wine.data
y=wine.target
#création d'un data frame pour les graphiques et statsmodels
data = pd.DataFrame(data=wine['data'],columns=wine['feature_names'])
data['target']=pd.Categorical(wine['target']) #transformation en facteur
#data.describe()
#wine['feature_names']
L='alcohol' #L = variable étudiée
for i in data.target.unique():
    F,p=stats.f_oneway(data[L][data.target==0],data[L][data.target==1],\
    data[L][data.target==2])
    sns.distplot(data[L][data.target==i],
                 kde=1,label='{}'.format(i))
plt.legend()
plt.title('Variable {}, F={}'.format(L,round(F)))
```

Out[45]: <matplotlib.legend.Legend at 0x7f338a384e10>



2.2 Question 2 : ANOVA

Interpréter les résultats obtenus pour l'anova.

```
In [ ]: from statsmodels.formula.api import ols
        import statsmodels.api as sm
```

```
lm=ols('alcohol~ C(target)',data=data).fit()
lm.summary()
sm.stats.anova_lm(lm, typ=2)
lm.diagn
stats.shapiro(lm.resid)
stats.levene(lm.resid,data['target']) # équivalent bartlett
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.stats.multicomp import MultiComparison
mc = MultiComparison(data['alcohol'], data['target'])
mc_lm = mc.tukeyhsd()
print(mc_lm)
```

2.3 Question 3 : Echantillon d'apprentissage

A l'aide de la fonction `train_test_split`, construire un échantillon d'apprentissage et un échantillon de test en proportion (0.75,0.25) en respectant les proportions des classes.

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

2.4 Question 4 : Modèle lda, analyse discriminante linéaire

Construire le modèle lda et calculer sa matrice de confusion et son score sur l'échantillon d'apprentissage et de test.

```
In [ ]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
        from sklearn.metrics import confusion_matrix
```

2.5 Question 5 : Modèle knn, plus proches voisins

Estimer le modèle knn et déterminer la valeur du nombre de voisin optimisant la prévision sur l'échantillon test. On étudiera le scrore de la méthode en fonction de différentes valeurs de k, 2 à 12 par exemple pour sélectionner le meilleur choix. En déduire les performances du modèle.

```
In [ ]: from sklearn.neighbors import KNeighborsClassifier
```

2.6 Question 6 : Modèle tree, arbre décisionnel

Utiliser un arbre de décision pour réaliser la prédiction. Pour obtenir un graphique de l'arbre, il faut taper

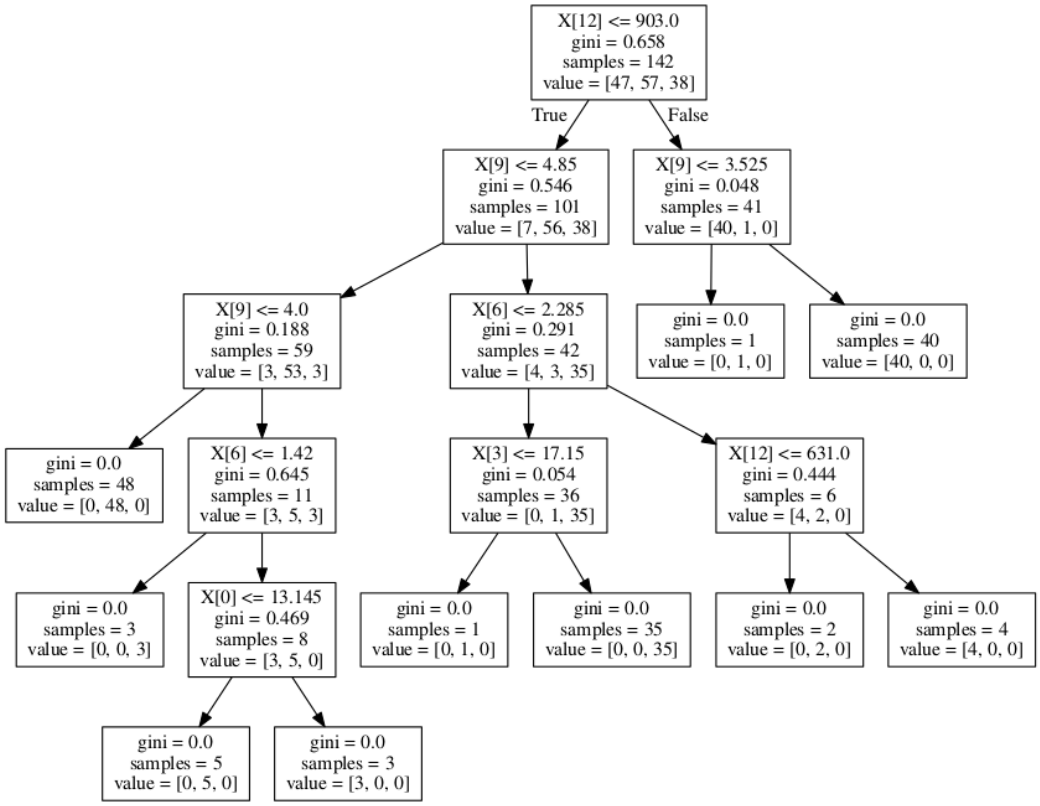
`dot -Tpdf wine.dot -o wine.pdf`

dans un terminal et dans le répertoire de wine.dot. Evaluer les performances du modèle. Que fait arbre2? Quel modèle choisir?

```
In [ ]: from sklearn import tree
        arbre = tree.DecisionTreeClassifier()
        arbre.fit(data_train,label_train)
        with open("wine.dot", 'w') as f:
            f = tree.export_graphviz(arbre, out_file=f)
        arbre2=tree.DecisionTree(max_depth=2)
```

```
In [47]: from IPython.display import Image
         Image(filename="/home/labatte/Bureau /MASTER DS 1 ADD/CS2- RD4 AFD/wine.png",width=700,
```

Out[47]:



2.7 Question 7 : modèle gnb, Naive Bayes

Le modèle repose sur l'hypothèse d'indépendance des variables explicatives, simplifiant fortement le modèle et son estimation. Les modèles les plus classiques sont les modèles gaussiens et multinomiaux. Construire le modèle et l'évaluer.

```
In [ ]: from sklearn.naive_bayes import GaussianNB
        gnb = GaussianNB()
```

3 Exercice 3 : Régression logistique

The datasets, `pima.txt` consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on. L'objectif est de prédire / expliquer l'occurrence du diabète

(variable à prédire) à partir des caractéristiques des personnes (âge, IMC, etc.) http://www.stat.nthu.edu.tw/swcheng/Teaching/stat5230/data/pima.txt http://www.grappa.univ-lille3.fr/torre/Recherche/Expe https://eric.univ-lyon2.fr/~ricco/cours/slides/PJ%20-%20machine%20learning%20avec%20scikit-learn.pdf

```
In [ ]: pima= pd.read_table("/home/labatte/Bureau /MASTER DS 1 ADD/CS2- RD4 AFD/pima.txt",sep="\
        print(pima.columns)
        print(pima.dtypes)
        #le format pandas est utilie pour statsmodels ou les graphiques
        #pour sklearn il faut un format array
        data = pima.as_matrix()
        X=data[:,0:8]
        y=data[:,8]
```

3.1 Question 1 : régression logistique explicative

Interpréter les résultats ci-dessous. L'étude peut être complétée par une étude descriptive variable par variable.

```
In [ ]: import statsmodels.formula.api as sm
        glm = sm.Logit(y,X)
        result = glm.fit()
        result.summary()
```

3.2 Question 2 : Sélection de variables

On procède à la sélection des variables pertinentes. Quelles sont les variables retenues? Expliquer le principe de sélection. Comparer les résultats avec le modèle complet (X) et optimisé (Xs).

```
In [ ]: from sklearn.linear_model import LogisticRegression
        Ir= LogisticRegression()
        from sklearn.feature_selection import RFE
        selecteur = RFE(estimator=Ir)
        sol = selecteur.fit(X,y)
        print(sol.n_features_)
        print(sol.support_)
        print(sol.ranking_)
        Xs=X[:,sol.support_]
```

3.3 Question 3 : Qualité du modèle

Calculer la matrice de confusion sur l'échantillon d'apprentissage et de test avec le modèle complet et simplifié. Calculer les taux d'erreurs.

3.4 Question 4 : Courbe ROC

On construit au préalable un score en fonction duquel les FP et FN sont calculés. Ici on utilise $P(1|X)$. Construire les courbes ROC des deux modèles et comparer les résultats.

```
In [ ]: #Construction d'un score, P(1|x)
        score = glm.predict_proba(X_test)[:,1]


        #Courbe ROC
        from sklearn.metrics import roc_auc_score
        from sklearn.metrics import roc_curve
        glm_roc_auc = roc_auc_score(y_test, glm.predict(X_test))
        fpr, tpr, thresholds = roc_curve(y_test-1, score)
        plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % glm_roc_auc)
        plt.plot([0, 1], [0, 1],'r--')
        plt.xlim([-0.05, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.legend(loc="lower right")
```