20. Comparing Classification Methods
The Heart Disease Model

Bruce E. Shapiro

Getting Started in Machine Learning

Last revised: April 4, 2019

Explore a Heart Disease Data Set

- Look at a single data set
- Classify with variety of methods
  - 1000 train/test splits on each model
  - use default settings for each model (more or less)
  - don't tweak the models
- Compare:
  - Boxplots of classification errors
  - Histograms of classification errors
  - ROC curves
- Left as an exercise:
  - Tweaking parameters (e.g., varying forest size or tree depth)
  - Model variations (e.g., different types of SVM classifiers)
  - Dimensional reduction (PCA, ICA)

Data File Description (1/3)

column	name	description
1	**age**	Age (years)
2*	**sex**	Sex, 1=male, 0 =female
3*	**cp**	Type of chest pain: 1=typical angina; 2=atypical angina; 3=non-anginal pain; 4=asymptomatic
4	**trestbps**	resting blood pressure in mm Hg on admission to hospital
5	**chol**	serum cholesterol, mg/dl
6*	**fbs**	fasting blood sugar: 1 means > 120 mg/dl 0 means ≤ 120 mg/dl.

Data File Description (2/3)

column	name	description
7*	**restecg**	resting electrocardiogram: 0 = normal; 1 = ST-T Wave abnormality; 2 = left ventricular abnormality
8	**thalach**	maximum elevated heart rate
9*	**exang**	exercise induced angina: 1 = yes; 2 = no
10	**oldpeak**	ST depression induced by exercise
11*	**slope**	Slope of peak of ST depression: 1 = positive; 2 = flat; 3 =negative

Data File Description (3/3)

column	name	description
12	**ca**	Number of blood vessels colored by flourosopy, 0 to 3
13*	**thal**	Thalassemia: 3 = normal; 6 = fixed defect; 7 = reversible defect
14	**num**	Diagnosis: 0 = healthy nonzero = heart-disease

Method

- Use first 13 columns as features (**`X`** data)
  - Convert categorical data* to numerical data using 1-Hot encoding
    - For example, **`sex`** has two possible values **`1`** and 0.
    - Create one new feature that is **`1`** whenever **`sex=1`** and **`0`** otherwise.
    - Create a second feature that is **`1`** whenever **`sex=0`** and **`0`** otherwise.
  - Number of new features (columns) is equal to the number of possible values (categories) that the category can have
  - Resulting feature data are orthogonal in feature space
- Use 14'th column as exemplar data (**`Y`**, class data)

Load and Label Data

```
import pandas as pd
DF=pd.read_csv(
"https://archive.ics.uci.edu/ml/machine-learning-databases/
    heart-disease/processed.cleveland.data",
    header=None,na_values="?")
DF=DF.dropna(axis=0)
DF.columns=["age","sex","cp","trestbps","chol","fbs",
        "restecg","thalach","exang","oldpeak","slope",
        "ca","thal","num"]
print(DF[:3])
```

```
    age  sex   cp  trestbps   chol  fbs  restecg  thalach  exang
oldpeak  slope   ca  thal  num
0  63.0  1.0  1.0     145.0  233.0  1.0      2.0    150.0    0.0
2.3    3.0  0.0   6.0    0
1  67.0  1.0  4.0     160.0  286.0  0.0      2.0    108.0    1.0
1.5    2.0  3.0   3.0    2
2  67.0  1.0  4.0     120.0  229.0  0.0      2.0    129.0    1.0
2.6    2.0  2.0   7.0    1
```

Designate the **X** and **Y** data arrays.

```
import numpy as np
num=np.array(DF["num"])
Y=np.array([1 if x>0.5 else 0 for x in num])
```

Create 1-Hot Encodings

```
categorical=["sex","cp","fbs", "restecg","exang","slope",
    "thal"]
X=[]
for col in DF.columns[:-1]:       # don't include "num"
    x=np.array(DF[col]).tolist() # list of col names
    if col in categorical:        # check if category
        unis=np.unique(x)         # get category values
        for category in unis:     # make column for each value
            nextcolumn=[1 if u==category else 0 for u in x]
            X.append(nextcolumn) # save column
    else:
        X.append(x)               # save numerical column
X=np.array(X)                     # convert to array
X=X.T                             # rows to columns
```

Perform Large Number of Train/Test Splits

Return value is list of classification errors

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
def evaluate(X, Y, MODEL, nsplits=100, PRINT=True):
    errs=[]
    for j in range(nsplits):
        XTRAIN,XTEST,YTRAIN,YTEST=train_test_split(X,Y)
        model=MODEL
        model.fit(XTRAIN,YTRAIN)
        YP=model.predict(XTEST)
        errs.append(1-accuracy_score(YTEST,YP))We
    if PRINT:
        print("Mean error=%7.5f std=%7.5f" \
          %(np.mean(errs), np.std(errs)))
    return errs
```

Import all models

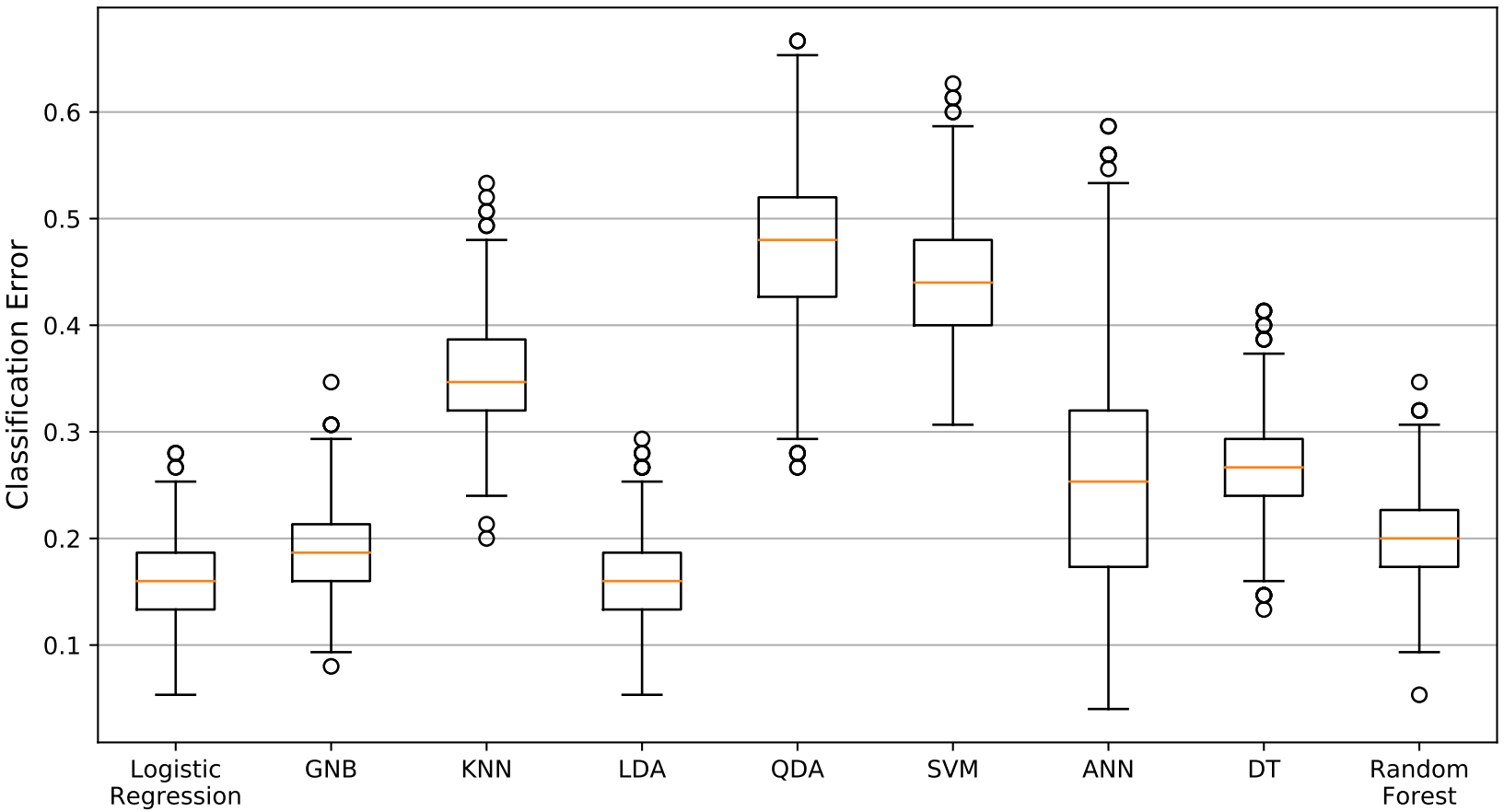
```
from sklearn.linear_model import LogisticRegression as LR
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.discriminant_analysis import \
  LinearDiscriminantAnalysis as LDA
from sklearn.discriminant_analysis import \
  QuadraticDiscriminantAnalysis as QDA
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier as ANN
from sklearn.ensemble import RandomForestClassifier as RF
from sklearn.tree import DecisionTreeClassifier as DT
```

Instantiate Models and Do 1000 Splits of Each

```
nsplits=1000
LRerrs=evaluate(X,Y,LR(),nsplits,PRINT=False)
NBerrs=evaluate(X,Y,GaussianNB(), nsplits, PRINT=False)
KNNerrs=evaluate(X,Y, KNN(), nsplits, PRiNt=False)
LDAerrs=evaluate(X,Y,LDA(), nsplits, PRINT=False)
QDAerrs=evaluate(X,Y,QDA(), nsplits, PRINT=False)
SVMerrs=evaluate(X,Y,SVC(kernel="rbf"), nsplits,
    PRINT=False)
ANNerrs=evaluate(X,Y,ANN(hidden_layer_sizes=(7,7,7),
    solver='lbfgs',random_state=1,
    alpha=1e-5), nsplits,PRINT=True)
RFerrs=evaluate(X,Y,RF(), nsplits, PRINT=False)
DTerrs = evaluate(X,Y,DT(), nsplits, PRINT=False)
```

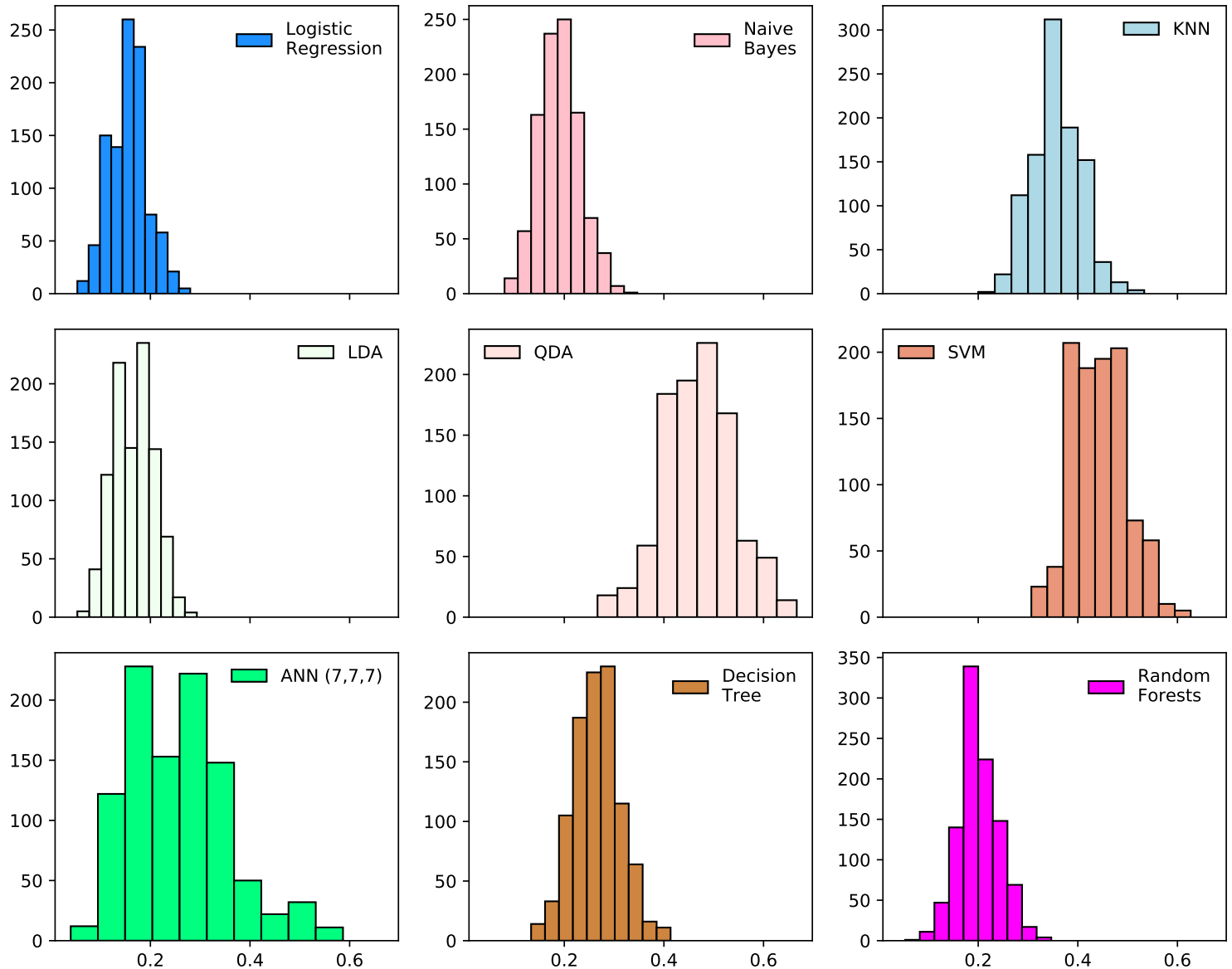
Generate Box Plots of Errors

```
plt.boxplot([LRerrs,NBerrs,KNNerrs, LDAerrs,QDAerrs,
    SVMerrs, ANNerrs, DTerrs,RFerrs])
plt.xticks(range(1,10),["Logistic\nRegression", "GNB",
    "KNN", "LDA", "QDA", "SVM", "ANN", "DT",
    "Random\nForest"]);
plt.ylabel("Classification Error", fontsize=12)
```


Classification Error
0.6
0.5
0.4
0.3
0.2
0.1
Logistic Regression
GNB
KNN
LDA
QDA
SVM
ANN
DT
Random Forest

Generate Histograms of Same Data

```
fig,ax=plt.subplots(nrows=3,ncols=3,sharex=True)
ax[0][0].hist(LRerrs,color="dodgerblue",edgecolor="k",
   label="Logistic\nRegression")
ax[0][1].hist(NBerrs,color="pink",edgecolor="k",
   label="Naive\nBayes")
ax[0][2].hist(KNNerrs,color="lightblue",edgecolor="k",
   label="KNN")
ax[1][0].hist(LDAerrs,color="honeydew",edgecolor="k",
   label="LDA")
ax[1][1].hist(QDAerrs,color="mistyrose",edgecolor="k",
   label="QDA")
ax[1][2].hist(SVMerrs,color="darksalmon",edgecolor="k",
   label="SVM")
ax[2][0].hist(ANNerrs,color="SpringGreen",edgecolor="k",
   label="ANN (7,7,7)")
ax[2][1].hist(DTerrs,color="Peru",edgecolor="k",
   label="Decision\nTree")
ax[2][2].hist(RFerrs,color="fuchsia",edgecolor="k",
   label="Random\nForests")
for i in range(3):
```


Logistic Regression
Naive Bayes
KNN
LDA
QDA
SVM
ANN (7,7,7)
Decision Tree
Random Forests
0.2
0.4
0.6

Function to Obtain ROC Data for a Single Split

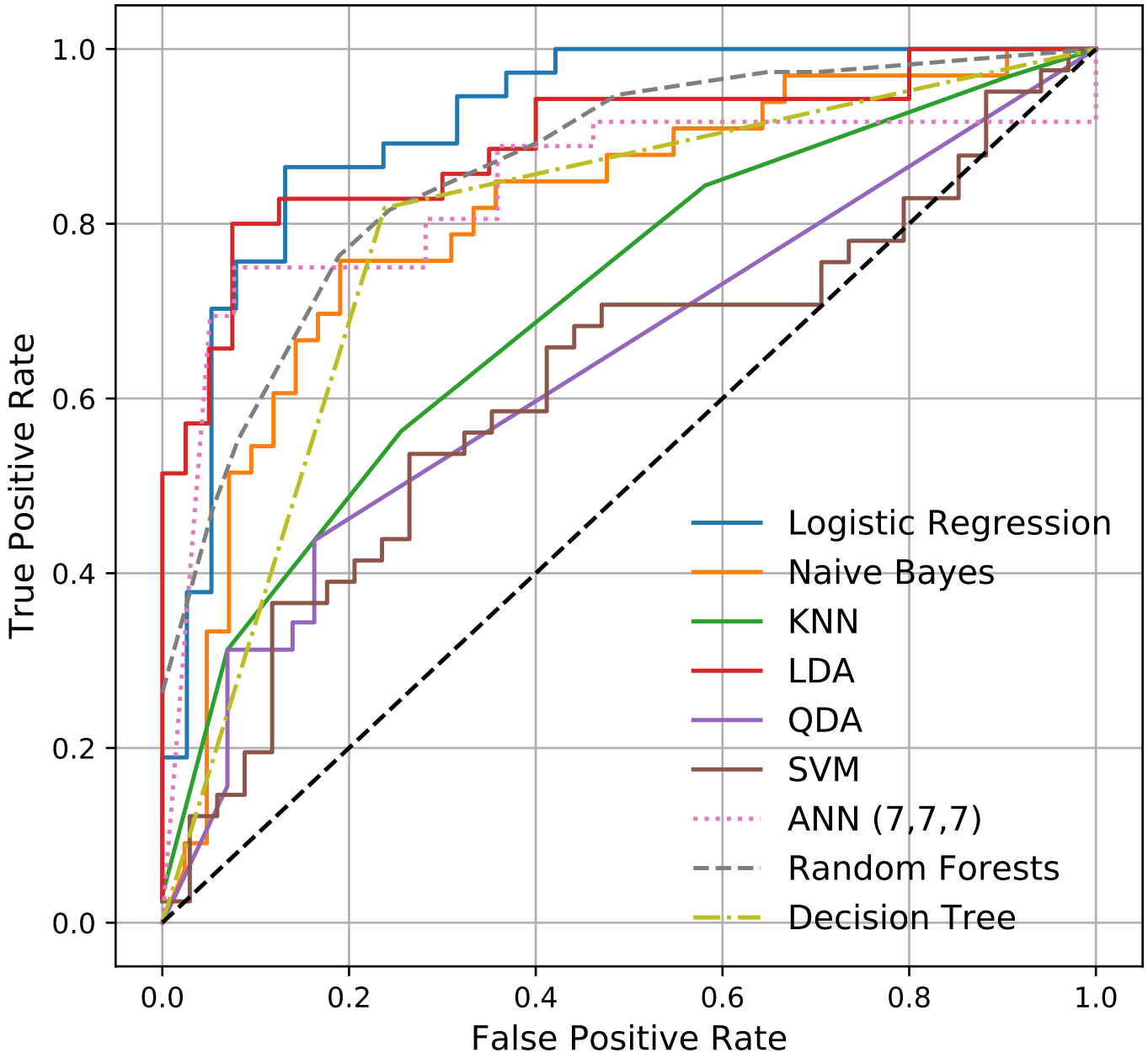
```
def ROC(X, Y, MODEL):
    XTRAIN,XTEST,YTRAIN,YTEST=train_test_split(X,Y)
    model=MODEL
    model.fit(XTRAIN,YTRAIN)
    YPR=model.predict_proba(XTEST)[:,1]
    F,T,THRESH=roc_curve(YTEST,YPR)
    return(F,T)
```

Code to Acquire ROC Data

```
LRF,LRT=ROC(X,Y,LR())           # Logistic
NBF,NBT=ROC(X,Y,GaussianNB()) # Naive Bayes
KNNF,KNNT=ROC(X,Y,KNN())        # KNN
LDAF,LDAT=ROC(X,Y,LDA())        # LDA
QDAF,QDAT=ROC(X,Y,QDA())        # QDA
SVMF,SVMT=ROC(X,Y,SVC(probability=True)) # SVM
AF, AT = ROC(X,Y, ANN(hidden_layer_sizes=(7,7,7),
   solver='lbfgs',random_state=1,
   alpha=1e-5))                 # ANN
RFF, RFT = ROC(X,Y, RF())       # Ran Forest
DTF, DTT = ROC(X,Y, DT())       # Dec Tree
```

Plot Acquired ROC Curves

```
plt.plot(LRF,LRT,label="Logistic Regression")
plt.plot(NBF,NBT,label="Naive Bayes")
plt.plot(KNNF,KNNT,label="KNN")
plt.plot(LDAF,LDAT,label="LDA")
plt.plot(QDAF,QDAT,label="QDA")fig-heart-disease-roc}
plt.plot(SVMF,SVMT,label="SVM")
plt.plot(AF, AT, label="ANN (7,7,7)", ls=":")
plt.plot(RFF, RFT, label="Random Forests", ls="--")
plt.plot(DTF, DTT, label="Decision Tree", ls="-.")
plt.xlabel("False Positive Rate", fontsize=12)
plt.ylabel("True Positive Rate",fontsize=12)
```


True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression
Naive Bayes
KNN
LDA
QDA
SVM
ANN (7,7,7)
Random Forests
Decision Tree

References

1. UCI Machine Learning Repository `http://archive.ics.uci.edu/ml`. Irvine, CA: University of California, School of Information and Computer Science.
2. "Processed Cleveland" data set from `https://archive.ics.uci.edu/ml/datasets/heart+Disease`
3. Detrano R et. al. (1989). "International application of a new probability algorithm for the diagnosis of coronary artery disease." *American Journal of Cardiology*, **64**, 304-310.