HW9

Juan Vila

Question 1

A

$K_a(x, y) = f(K(x, y))$, where f is a polynomial function. Then:

$$K_a(x, y) = \sum_{i=0}^{d} 1 + K(x, y) + K(x, y)^2 + ... + K(x, y)^d$$

Then we know that $K(x, y)^d$ is a product of kernel and therefore a kernel. On the otherhand, we know that the sum of kernel is a kernel. Also, i we add up constant to a kernel if the constant is positive is also a kernel, the constant in this case is 1 which is positive. Then, $K_a(x, y)$ is a Kernel

B

$$k_b(x, y) = exp(k(x, y))$$

If we use a taylor series which reflects the exponential funcion we can see

$$exp(x) = \sum_{i=1}^{\infty} 1 + \frac{x}{1!} + \frac{x^2}{2!} + ...$$

Since x is kernel we can see that the exponential function is a sum of X, multiplied by itself and multiplied by constants and adding constant which made $k_b(x, y) = exp(k(x, y))$ a Kernel

C

$$k_c(x, y) = f(x)k(x, y)f(y)$$

If we argue that $\phi(x^{'}) = f(x)\phi(x)$ and $\phi(y^{'}) = f(y)\phi(y)$, then:

$$f(x)k(x, y)f(y) = f(x)\phi(x)f(y)\phi(y) = \phi(x^{'})\phi(y^{'}) = k_c(x, y)$$

Which show that k_c is a Kernel

D

$$k_d(x, y) = k(\phi(x), \phi(y))$$

If we argue that $\phi(x^{'}) = \phi(\phi(x))$ and $\phi(y^{'}) = \phi(\phi(y))$, then:

$$\phi(\phi(x))\phi(\phi(y)) = \phi(x^{'})\phi(y^{'}) = k_d(x, y)$$

Which show that k_d is a Kernel

Question 2

A

Let assume that $x^i = \begin{pmatrix} x_1^i \\ x_2^i \end{pmatrix}$, where i can take values of none and $'$, then the $\phi(X)$ is equal to:

$$\phi(x^i) = \begin{pmatrix} 1 & \sqrt{2}x_1^i & \sqrt{2}x_2^i & x_1^{2i} & \sqrt{2}x_1^i x_2^i & x_2^{2i} \end{pmatrix}$$

Then,
$\phi(x)^T\phi(x^{'}) = 1 + 2x_1x_1^{'} + 2x_2x_2^{'} + 2x_1^2x_1^{2'} + 2x_2^2x_2^{2'} + 2x_1x_1^{'}x_2x_2^{'} = (1 + x_1x_1^{'} + x_2x_2^{'})^2 = (X^TX^{'} + 1)^2$

B

Both ϕ vectors are:

$$\phi(x_1, x_2) = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Then the gram matrix is:

$$K(x_1, x_2) = \begin{pmatrix} 3 & 0 & 3 \\ 4 & 0 & 4 \\ 5 & 0 & 5 \end{pmatrix}$$

Question 3

A

Since the gaussian process has a form $y = f(x) + \epsilon$, where $f(x) \sim GP(0, k(x_*, x)$, the the joint distribution is:

$$\begin{pmatrix} y \\ f(x) \end{pmatrix} \sim \left(0, \begin{pmatrix} K(X,X) + \sigma_n^2 I_n & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right)$$

Where X is the train data and X_* is the test data. Then using the joint prior distribution, we can get that the variance of f(x) is:

$$var_n(f(x_*)) = K(X_*, X_*) - K(X_*, X)^T[K(X,X) + \sigma_n^2 I_n]^{-1}K(X, X_*)$$

B

For proofing this we need proof, we need to decomposie the matrix of the second part of the variance since the first part is the same in n or n+1. Then if we use the matrix inversion lemma we get the following results:

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}, X^{-1} = \begin{pmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{pmatrix}$$

Where:

$$\hat{A} = A^{-1} + A^{-1}BZCA^{-1}$$
$$\hat{B} = -A^{-1}BZ$$
$$\hat{C} = -ZCA^{-1}$$

$$\hat{D} = Z$$
$$Z = (D - CA^{-1}B)^{-1}$$

Then using this decompisition we can decompose the gram matrix of the central term:

$$K(X,X) + \sigma_n^2 I_n = \begin{pmatrix} K_{n-1} + \sigma_n^2 I_{n-1} & K_{n-1}(X_*) \\ K_{n-1}(X_*)^T & K(X_*,X_*) + \sigma^2 \end{pmatrix}$$
$$(K(X,X) + \sigma_n^2 I_n)^{-1} = \begin{pmatrix} K_{n-1} + \sigma_n^2 I_{n-1} & K_{n-1}(X_*) \\ K_{n-1}(X_*)^T & K(X_*,X_*) + \sigma^2 \end{pmatrix}^{-1}$$

Then we can use the decomposition, then:

$$\hat{A} = V_{n-1}^{prior} + V_{n-1}^{prior} K_{n-1}(X_*) R K_{n-1}(X_*)^T V_{n-1}^{prior}$$
$$\hat{B} = -V_{n-1}^{prior} K_{n-1}(X_*) R$$
$$\hat{C} = -R K_{n-1}(X_*)^T V_{n-1}^{prior}$$
$$\hat{D} = R$$
$$R = (K(X_*,X_*) + \sigma^2 - K_{n-1}(X_*))(K_{n-1} + \sigma_n^2 I_{n-1})^{-1}$$
$$V_{n-1}^{prior} = (K_{n-1} + \sigma_n^2 I_{n-1})^{-1}$$
$$(K(X,X) + \sigma_n^2 I_n)^{-1} = \begin{pmatrix} V_{n-1}^{prior} + V_{n-1}^{prior} K_{n-1}(X_*) R K_{n-1}(X_*)^T V_{n-1}^{prior} & -V_{n-1}^{prior} K_{n-1}(X_*) R \\ -R K_{n-1}(X_*)^T V_{n-1}^{prior} & R \end{pmatrix}$$

Where $V_{n-1}^{prior} = K_{n-1} + \sigma_n^2 I_{n-1}$

Now we need to decompose K(X_*,X):

$$K(X_*,X) = \begin{pmatrix} K_{n-1}(X_*) \\ K_n(X_*) \end{pmatrix}$$

Then replacing all this expresion into the second term of the variance we get:

$$\begin{pmatrix} K_{n-1}(X_*) \\ K_n(X_*) \end{pmatrix}^T \begin{pmatrix} V_{n-1}^{prior} + V_{n-1}^{prior} K_{n-1}(X_*) R K_{n-1}(X_*)^T V_{n-1}^{prior} & -V_{n-1}^{prior} K_{n-1}(X_*) R \\ -R K_{n-1}(X_*)^T V_{n-1}^{prior} & R \end{pmatrix} \begin{pmatrix} K_{n-1}(X_*) \\ K_n(X_*) \end{pmatrix}$$

Finally if we multiply the whole expression we get:

$$(K(X,X) + \sigma_n^2 I_n)^{-1} = K_{n-1}(X_*)^T V_{n-1}^{prior} K_{n-1}(X_*) + (\beta - K(X,X^*))^2$$

Where β is a constant. The we can see that the second term variance is related to the varaince of n-1 plus a constant this proof it, because if we add in both sides we can see that the variance in n is the variance in n-1 plus a postive term, then we can see that variance in n is larger that the one in n-1.

$$K(X_*,X_*) + (K(X,X) + \sigma_n^2 I_n)^{-1} = K(X_*,X_*) + K_{n-1}(X_*)^T V_{n-1}^{prior} K_{n-1}(X_*) + (\beta - K(X,X^*))^2$$
$$var_n(f(x_*)) = var_{n-1}(f(x_*)) + (\beta - K(X,X^*))^2$$

Question 2

Part A

In [123]:
```
%matplotlib inline

'''
(1) Decision Tree*
(2) Random Forest*
(3) AdaBoost*
(4) Logistic Regression*
(5) Linear Discriminant Analysis
(6) Naive Bayes*
(7) Neural Network
(8) Gaussian Processes1*
(9) Support vector machine*
'''



# General math and plotting modules.
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

import time

# basic sklearn library
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_classification

# You may use existing machine learning libraries

#split dataset into test set, train set and unlabel pool
def split(X, y, train_size, test_size):
    X_train, X_pool, y_train, y_pool = train_test_split(
        X, y, train_size = train_size, random_state=42)
    unlabel, X_test, label, y_test = train_test_split(
        X_pool, y_pool, test_size = test_size, random_state=42)
    return X_train, y_train, X_test, y_test, unlabel, label

def create_dataset():
    X, y = make_classification(n_samples=1250,
                               n_features=2,
                               n_redundant=0,
                               n_informative=2,
                               random_state=1,
                               n_clusters_per_class=1)

    rng = np.random.RandomState(2)
    X += 3*rng.uniform(size=X.shape)
    linearly_separable = (X, y)

    X = StandardScaler().fit_transform(X)

    return X,y

X,y = create_dataset()
```

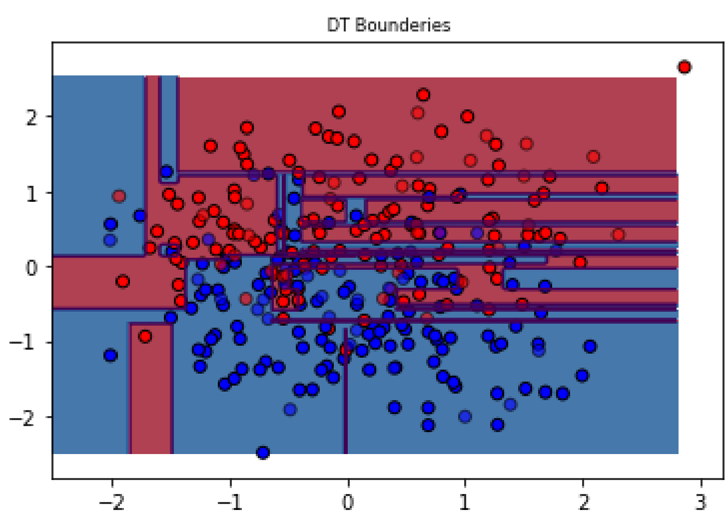
```
num_samples = X.shape[0]
incremental_train_size = 100
init_train_ratio = 0.04
test_ratio = 0.20/(1-init_train_ratio)

X_INIT, Y_INIT, X_TEST, Y_TEST, X_UNLABELED, Y_UNLABELED = split(
            X,y, init_train_ratio, test_ratio)


# Compare performance of different classifiers on the above dataset.
# add your code here

X_tot = np.concatenate((X_INIT,X_TEST), axis =0)
y_tot = np.concatenate((Y_INIT,Y_TEST), axis =0)
X_train, X_test, y_train, y_test = train_test_split(X_tot, y_tot, test_s
ize=0.2, random_state=42)
```

In [124]:

```
### Decision Tree

from sklearn . tree import DecisionTreeClassifier

clf_a = DecisionTreeClassifier(max_depth=None,min_samples_leaf=1)
clf_a=clf_a.fit(X_train,y_train)


def score( clf , X_test , y_test ):
    y_pred = clf.predict_proba( X_test )[:,1]
    acc = sum( np.round( y_pred ) == y_test ) / len( y_test )
    return acc


# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_a . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title (" DT Bounderies", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:28: UserWarning: No contour levels were found within the data
range.
```

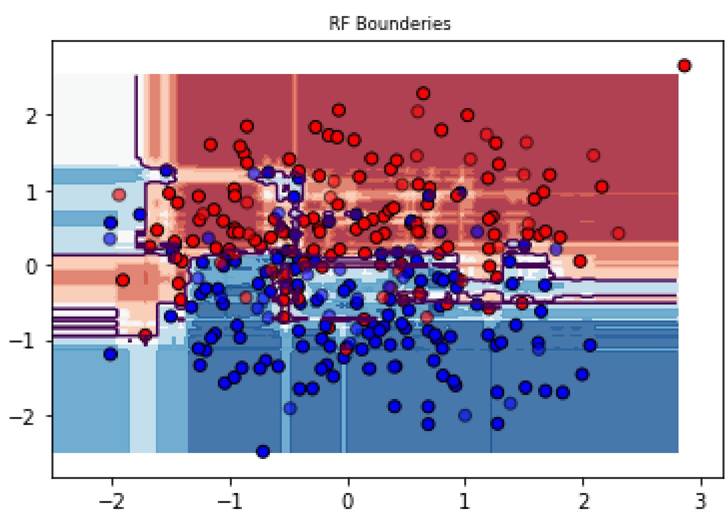
Out[124]: <matplotlib.collections.PathCollection at 0x1a31a6ff50>



In [125]:
```
### Random Forest
from sklearn.ensemble import RandomForestClassifier
clf_b = RandomForestClassifier(max_depth=None, random_state=42)
clf_b.fit(X_train, y_train)


# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_b . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title (" RF Bounderies", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:20: UserWarning: No contour levels were found within the data
range.
```

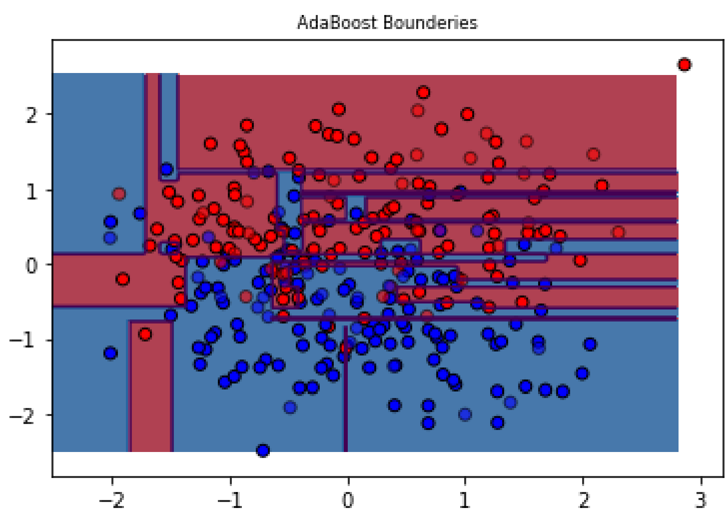
Out[125]: <matplotlib.collections.PathCollection at 0x1a31b9a750>



In [126]:
```
#### AdaBoost
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
clf_c = AdaBoostClassifier(DecisionTreeClassifier(max_depth=None),n_esti
mators=100, random_state=42)
clf_c.fit(X_train, y_train)

# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_c . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title (" AdaBoost Bounderies ", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:20: UserWarning: No contour levels were found within the data
range.
```

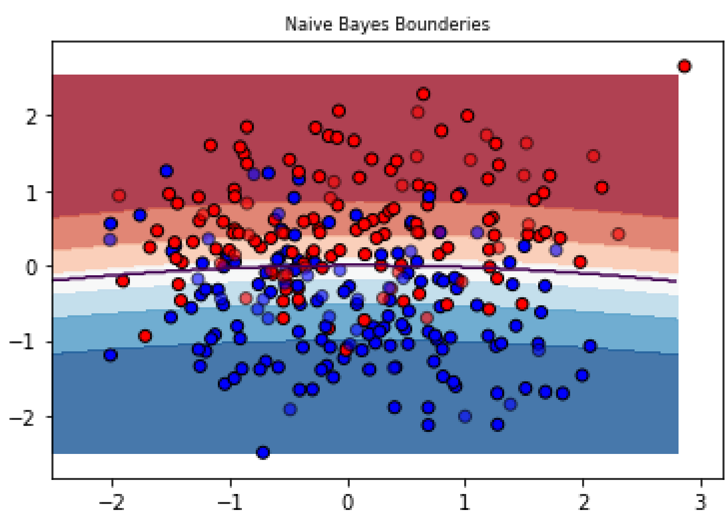
Out[126]: <matplotlib.collections.PathCollection at 0x1a319e0850>



In [127]:
```
from sklearn.naive_bayes import GaussianNB
import pylab as pl
GaussianNB()
clf_d = GaussianNB()
clf_d.fit(X_train, y_train)


# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_d . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title (" Naive Bayes Bounderies ", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:21: UserWarning: No contour levels were found within the data
range.
```

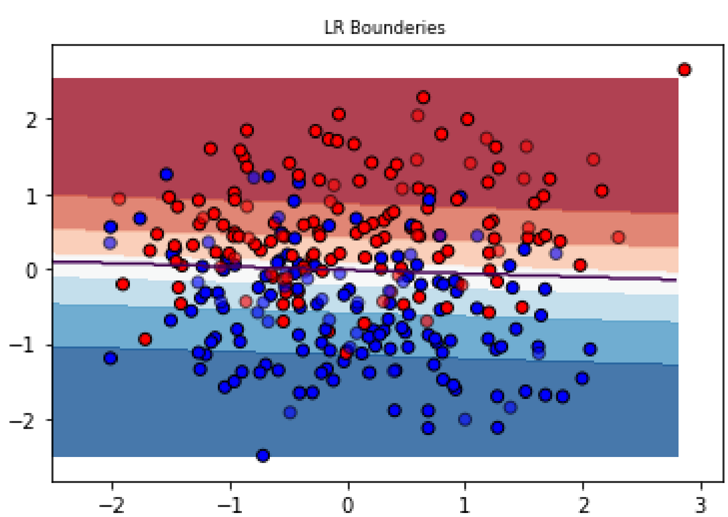
Out[127]: <matplotlib.collections.PathCollection at 0x1a2b4ed810>



In [128]:
```
#### Logistic Regression
from sklearn.linear_model import LogisticRegression

clf_e = LogisticRegression()
clf_e.fit(X_train, y_train)


# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_e . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title ("LR Bounderies ", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:21: UserWarning: No contour levels were found within the data
range.
```

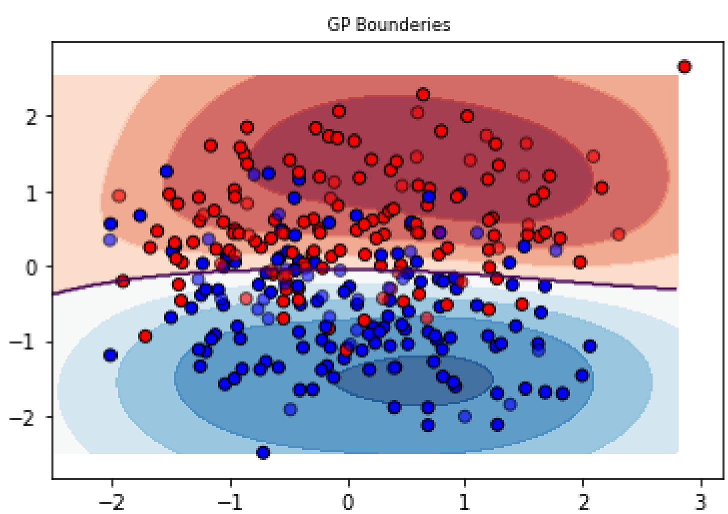
Out[128]: <matplotlib.collections.PathCollection at 0x1a2cb1af10>



In [129]:
```
### Gaussian Process

from sklearn.gaussian_process import GaussianProcessClassifier


clf_f = GaussianProcessClassifier()
clf_f.fit(X_train, y_train)

# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_f . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title (" GP Bounderies", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:22: UserWarning: No contour levels were found within the data
range.
```

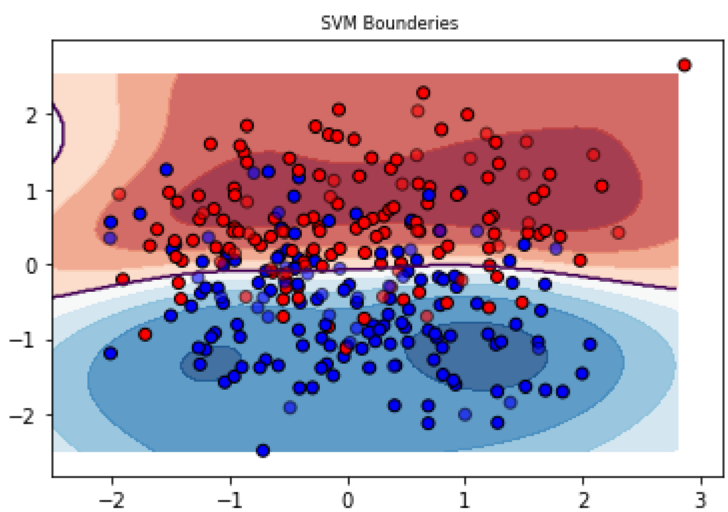
Out[129]: <matplotlib.collections.PathCollection at 0x1a2cfe87d0>



In [130]:
```
### SVM

from sklearn import svm


clf_g = svm.SVC(probability=True)
clf_g.fit(X_train, y_train)

# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_g . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title (" SVM Bounderies", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:22: UserWarning: No contour levels were found within the data
range.
```

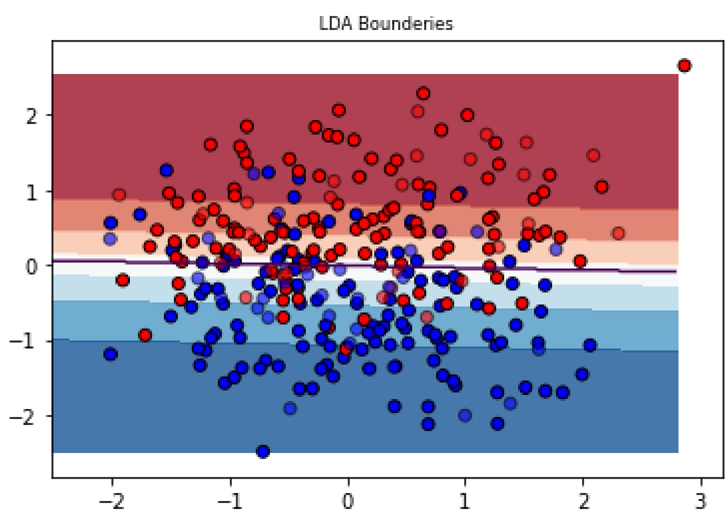
Out[130]: <matplotlib.collections.PathCollection at 0x1a30f59850>



In [133]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

clf_h = LinearDiscriminantAnalysis()
clf_h.fit(X_train, y_train)


# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_h . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title (" LDA Bounderies ", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:20: UserWarning: No contour levels were found within the data
range.
```

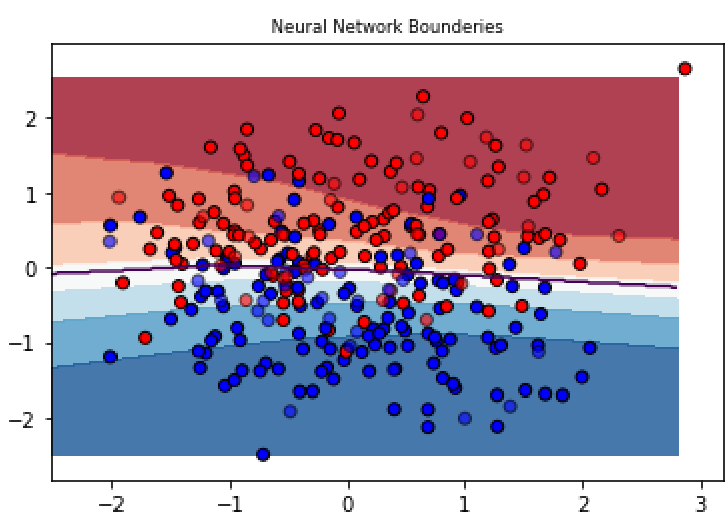
Out[133]: <matplotlib.collections.PathCollection at 0x1a30faf8d0>



In [134]:
```
### Neural Network
from sklearn.neural_network import MLPClassifier
clf_i = MLPClassifier(random_state=1, max_iter=500).fit(X_train, y_train
)

# plotting
x_min , x_max = X_test [: , 0]. min () - .5 , X_test [: , 0]. max () + .
5
y_min , y_max = X_test [: , 1]. min () - .5 , X_test [: , 1]. max () + .
5
h = .02
xx , yy = np . meshgrid ( np . arange ( x_min , x_max , h ) ,
np . arange ( y_min , y_max , h ))
Z = clf_i . predict_proba( np.c_[ xx.ravel() , yy.ravel()])[: , 1]
Z = Z . reshape ( xx . shape )
plt . figure ()
plt . title (" Neural Network Bounderies ", fontsize ="small")
cm = plt . cm . RdBu
cm_bright = ListedColormap ([ "#FF0000", "#0000FF"])
plt . contourf ( xx , yy , Z , cmap = cm , alpha =.8)
plt . contour ( xx , yy , np . round ( Z ) , 0)
plt . scatter ( X_train [: , 0] , X_train [: , 1] , marker ='o',
c = y_train , cmap = cm_bright , edgecolors ='k')
plt . scatter ( X_test [: , 0] , X_test [: , 1] , marker ='o', c = y_tes
t ,
cmap = cm_bright , alpha =0.6 , edgecolors ='k')
```

```
/Users/juanvila1/opt/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:18: UserWarning: No contour levels were found within the data
range.
```

Out[134]: `<matplotlib.collections.PathCollection at 0x1a3197d190>`



Part B

In [135]:
```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classificati
on
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
import time


s_size = [50,100,200,300,400,500,600,700,800,900,1000]
classifiers = [
    SVC(),
    GaussianProcessClassifier(),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1
),
    MLPClassifier(alpha=1, max_iter=1000),
    AdaBoostClassifier(),
    GaussianNB(),
    LinearDiscriminantAnalysis(),
    LogisticRegression()]

scores =dict()
time_m =dict()
for p in classifiers:
    scores.update({str(p): []})
    time_m.update({str(p): []})


for i in s_size:
    for model in classifiers:
        result = []
        result_t = []
        prev_result = scores[str(model)]
        prev_result_t = time_m[str(model)]
        for j in range(10):
            idx = np.arange(X_tot.shape[0])
            selected = np.random.choice(idx, size = i, replace=True)
            X_train = X_tot[selected, :]
            y_train = y_tot[selected,]
            start = time.time()
            model.fit(X_train,y_train)
            end = time.time()
            y_pred = model.predict(X_TEST)
            acc_score = accuracy_score(Y_TEST, y_pred)
            result.append(acc_score)
            result_t.append(end-start)
        result_array= np.asarray(result, dtype=np.float32)
```

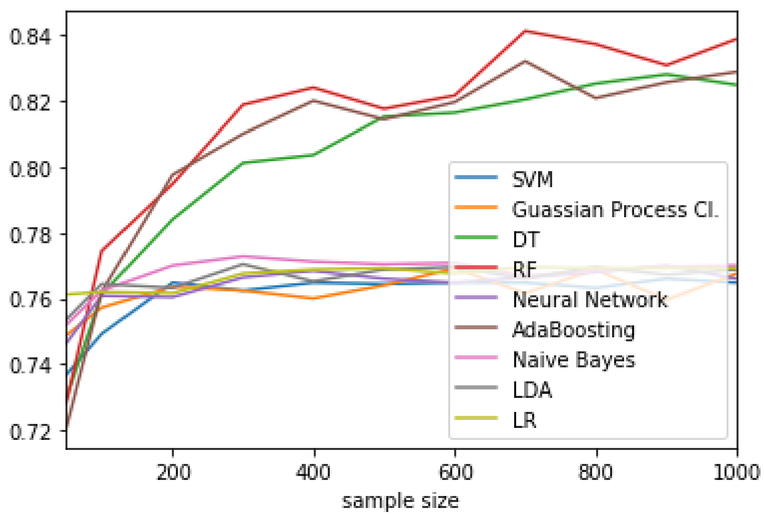
```
        result_t_array= np.asarray(result_t, dtype=np.float32)
        mean = np.mean(result_array)
        mean_t = np.mean(result_t_array)
        prev_result.append(mean)
        prev_result_t.append(mean_t)
        scores[str(model)]= prev_result
        time_m[str(model)]= prev_result_t
```

In [136]:

```
import pandas as pd
sample = {'sample':s_size}
scores.update(sample)
time_m.update(sample)
df_Results = pd.DataFrame(scores)
df_time = pd.DataFrame(time_m)
df_Results.columns=['SVM','Guassian Process Cl.','DT','RF','Neural Netwo
rk',
                    'AdaBoosting','Naive Bayes','LDA','LR', 'sample siz
e']
df_time.columns=['SVM','Guassian Process Cl.','DT','RF','Neural Network'
,
                 'AdaBoosting','Naive Bayes','LDA','LR', 'sample size']
```

In [137]:

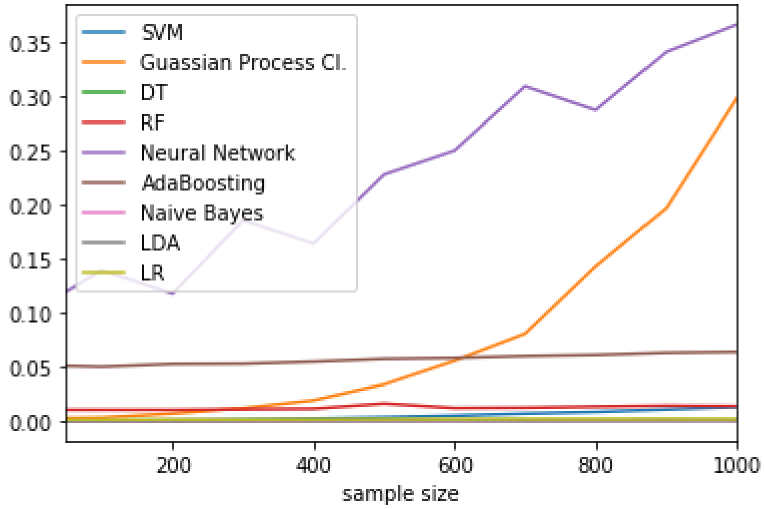
```
df_Results.set_index('sample size').plot();
```



We can see that repect to accuracy the models that have a better result are the trees this is because since the max depth of the tree is set to none, they overfit the data, then thye have a better results. The other result is that SVM and Gaussian Process start with low accuracy and after 200 sample size they manage behave as the other estimators that are in the middle group (i.e. the non tree base classifiers)

Part C

In [138]:
```
df_time.set_index('sample size').plot();
```



We can see that the model that is more time consumming in the fit of the data is the Neural Network as expected. Following by the Gaussian Process, which made since is Baysian base they have to made different iterations to get the result. Finally, in the third model that have a higher estimation time is the AdaBoost, but is greater in level but not increasing in the sample size.