РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

Дисциплина: Интеллектуальный анализ данных

Студент: Гусейнов Вахид Азерович

Группа: НБИбд-01-17

Москва 2020

Вариант № 12

chocardiogram Data Set

Название файла: echocardiogram.data

Ссылка: http://archive.ics.uci.edu/ml/datasets/Echocardiogram

Класс: still-alive (столбец No 2)

1. Считайте заданный набор данных из репозитария UCI.

In [1]:
```
import pandas as pd
import numpy as np

from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import plot_precision_recall_curve, plot_roc_curve, roc_auc_sco

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

seed = 111
```

In [2]:
```
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/echocardiogram/echoc
data = pd.read_csv(url, header=None, error_bad_lines=False, prefix='V')
data.head()
```

```
b'Skipping line 50: expected 13 fields, saw 14\n'
```

Out[2]:

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
0	11	0	71	0	0.260	9	4.600	14	1	1	name	1	0
1	19	0	72	0	0.380	6	4.100	14	1.700	0.588	name	1	0
2	16	0	55	0	0.260	4	3.420	14	1	1	name	1	0
3	57	0	60	0	0.253	12.062	4.603	16	1.450	0.788	name	1	0
4	19	1	57	0	0.160	22	5.750	18	2.250	0.571	name	1	0

1. Если среди меток класса имеются пропущенные значения, то удалите записи с пропущенными метками класса. Если в признаках имеются пропущенные значения, то замените их на медианные значения признака. Если какие-либо числовые признаки в наборе были распознаны неверно, то преобразуйте их в числовые. Оставьте в наборе данных только числовые признаки.

In [3]:
```
data.replace('?', np.nan, inplace=True)
data.isna().sum()
```

Out[3]:
```
V0      1
V1      0
V2      5
V3      0
V4      7
V5     14
V6     10
V7      3
V8      1
V9      3
V10     0
V11    22
V12    57
dtype: int64
```

В метках класса нет пропусков

In [4]:
```
for col in data:
    if col == 'V10':
        continue
    data[col] = data[col].astype(float)
    data[col] = data[col].fillna(np.nanmedian(data[col]))
```

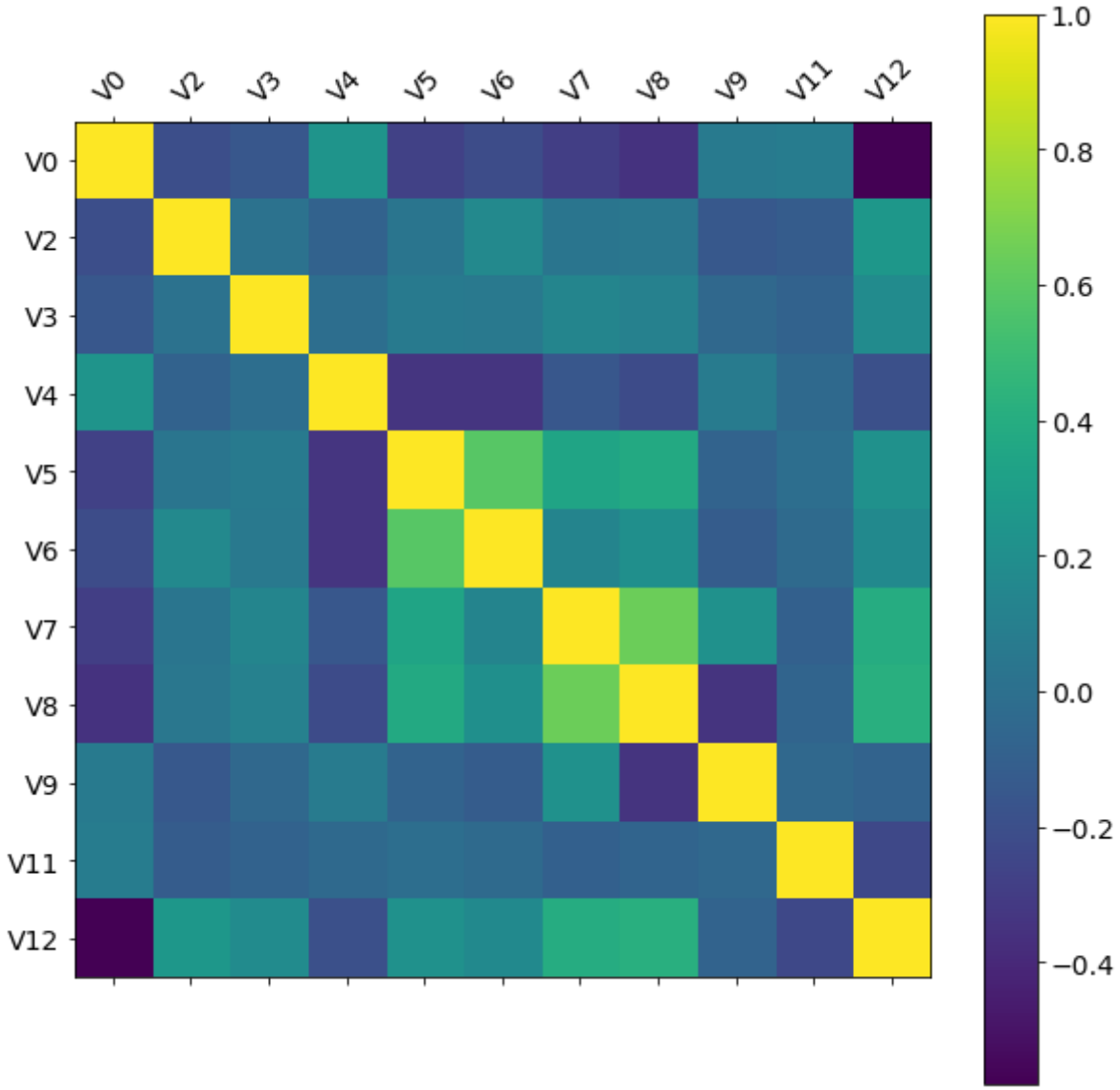
In [5]:
```
data.isna().sum()
```

Out[5]:
```
V0     0
V1     0
V2     0
V3     0
V4     0
V5     0
V6     0
V7     0
V8     0
V9     0
V10    0
V11    0
V12    0
dtype: int64
```

1. Вычислите и визуализируйте матрицу корреляций признаков. Удалите из набора признаки, имеющие высокую корреляцию (близкую к +1 или -1) с другими

признаками.

In [6]:
```
X = data.drop(columns=['V1', 'V10'])
y = data['V1'].values

f = plt.figure(figsize=(10, 10))

plt.matshow(X.corr(), fignum=f.number)
plt.xticks(range(X.shape[1]), X.columns, fontsize=14, rotation=45)
plt.yticks(range(X.shape[1]), X.columns, fontsize=14)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
# plt.title('Correlation Matrix', fontsize=16)
plt.show()
```



In [7]:
```
X.corr()
```

Out[7]:

	V0	V2	V3	V4	V5	V6	V7	V8	V9
V0	1.000000	-0.202778	-0.153772	0.237187	-0.273620	-0.213141	-0.286370	-0.348752	0.062722
V2	-0.202778	1.000000	0.016331	-0.081707	0.036575	0.163159	0.033146	0.046637	-0.144463
V3	-0.153772	0.016331	1.000000	-0.010711	0.067048	0.060320	0.139984	0.115191	-0.044230
V4	0.237187	-0.081707	-0.010711	1.000000	-0.336740	-0.338129	-0.150135	-0.221665	0.074576
V5	-0.273620	0.036575	0.067048	-0.336740	1.000000	0.587310	0.344988	0.377927	-0.074369

	V0	V2	V3	V4	V5	V6	V7	V8	V9
V6	-0.213141	0.163159	0.060320	-0.338129	0.587310	1.000000	0.133137	0.207389	-0.122431
V7	-0.286370	0.033146	0.139984	-0.150135	0.344988	0.133137	1.000000	0.642867	0.217128
V8	-0.348752	0.046637	0.115191	-0.221665	0.377927	0.207389	0.642867	1.000000	-0.342168
V9	0.062722	-0.144463	-0.044230	0.074576	-0.074369	-0.122431	0.217128	-0.342168	1.000000
V11	0.080417	-0.118021	-0.081776	-0.042878	-0.011056	-0.030866	-0.092745	-0.068651	-0.044964
V12	-0.580073	0.258449	0.183801	-0.191549	0.221257	0.172488	0.397863	0.413869	-0.080068

Нет высокой корреляции.

1. Если столбец с метками классов содержит более двух классов, то объедините некоторые классы, чтобы получить набор для бинарной классификации. Объединяйте классы таким образом, чтобы положительный и отрицательный классы были сопоставимы по количеству точек.

In [8]:
```
np.unique(y).size
```

Out[8]:
```
2
```

1. Используя метод рекурсивного исключения признаков (RFE) и логистическую регрессию, определите и оставьте в наборе наиболее значимые признаки (не менее двух). Если в наборе данных осталось более двух признаков, то определите два признака с наибольшей дисперсией для визуализации.

In [9]:
```
model = LogisticRegression(max_iter=1000, random_state=seed)
rfe = RFE(model)
fit = rfe.fit(X, y)

print("Число признаков: %d" % fit.n_features_)
print("Выбранные признаки: %s" % fit.support_)
print("Ранг признаков: %s" % fit.ranking_)
```

```
Число признаков: 5
Выбранные признаки: [False False  True  True False False False  True  True False  Tr
ue]
Ранг признаков: [2 5 1 1 4 6 3 1 1 7 1]
```

In [10]:
```
X = X.loc[:, fit.support_]
```

In [11]:
```
X.var()
```

Out[11]:
```
V3     0.150793
V4     0.010944
V8     0.202798
V9     0.038720
V12    0.150793
dtype: float64
```

In [12]:
```
X = X.loc[:, ['V3', 'V8']]
```

1. Масштабируйте признаки набора данных на интервал от 0 до 1.

In [13]:
```
scaler = MinMaxScaler()
```

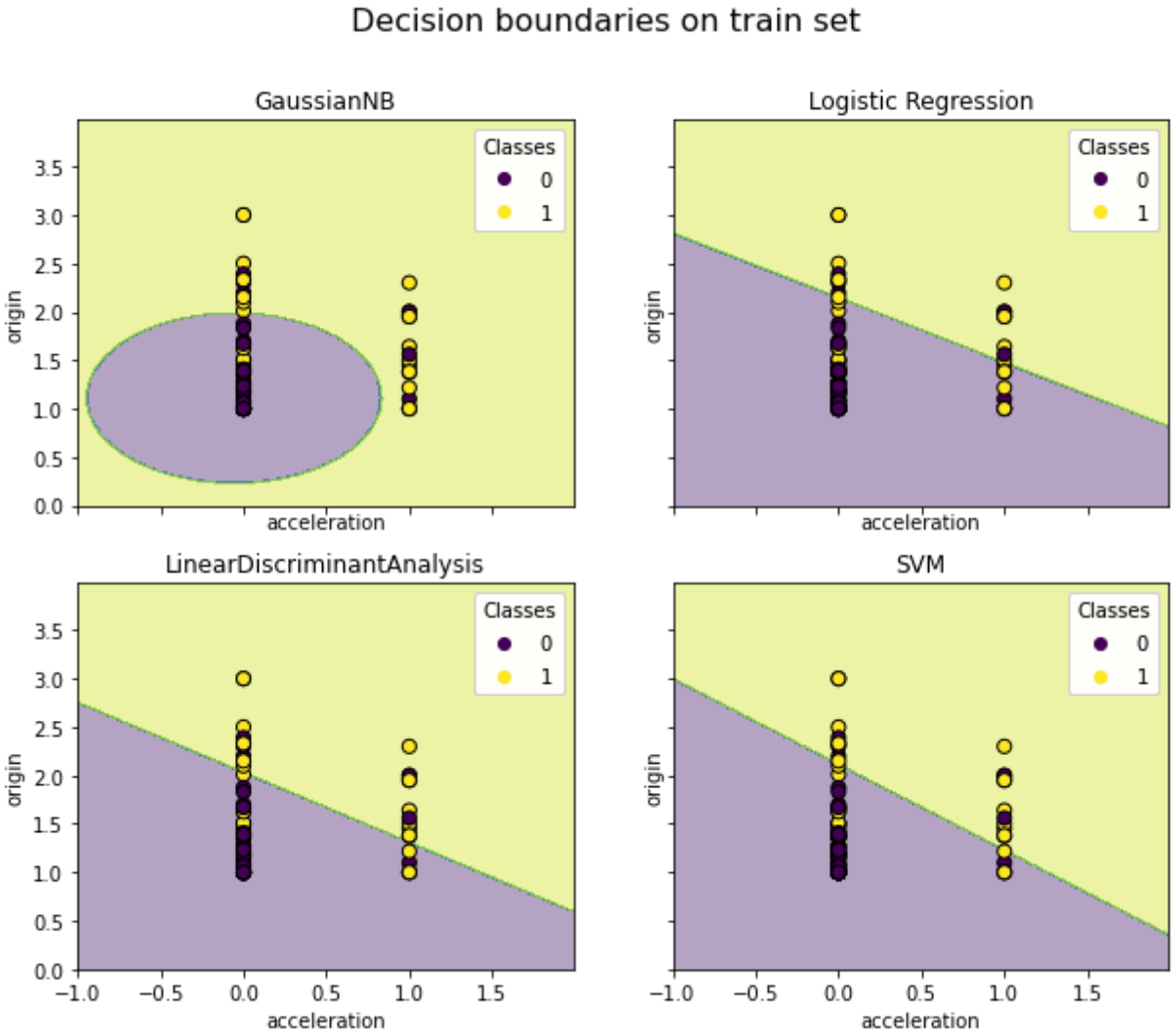
```
X_scaled = scaler.fit_transform(X)
```

1. Используя разделение набора данных на обучающую и тестовую выборки в соотношении 70% на 30%, создайте и обучите классификаторы на основе наивного байесовского классификатора, логистической регрессии, линейного дискриминантного анализа и метода опорных векторов.

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X.values, y, test_size=0.3, rand

nb = GaussianNB()
nb.fit(X_train, y_train)

lr = LogisticRegression(max_iter=500, random_state=42)
lr.fit(X_train, y_train)

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

svm = SVC(kernel='linear', probability=True, random_state=42) # for linear SVC
svm.fit(X_train, y_train);
```

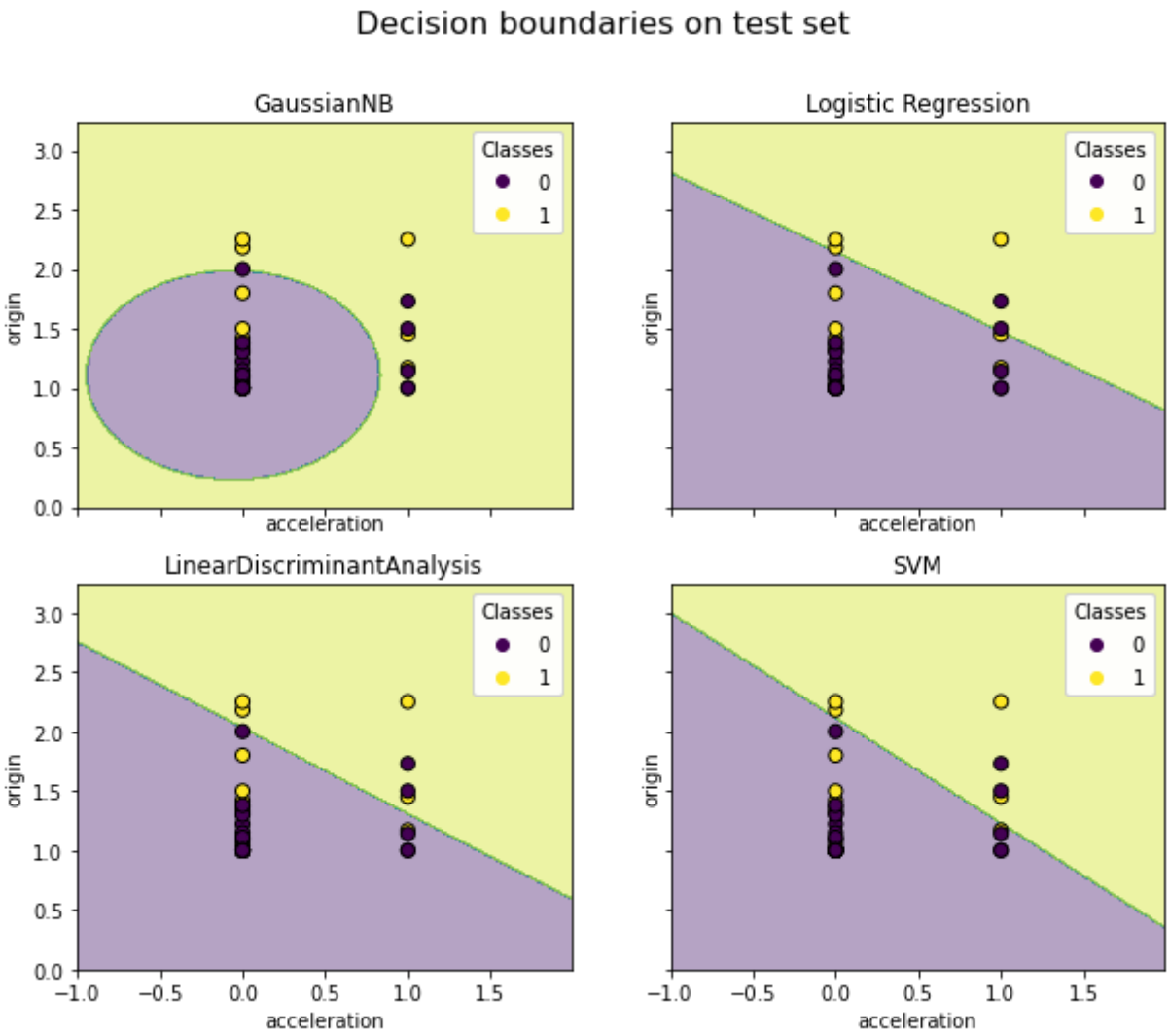
1. Визуализируйте для каждого из классификаторов границу решения, подписывая оси и рисунок и создавая легенду для меток классов набора данных.

In [15]:
```
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                     np.arange(y_min, y_max, 0.01))
f, axes = plt.subplots(2, 2, sharex='col', sharey='row', figsize=(10, 8))
f.suptitle('Decision boundaries on train set', fontsize=16)

for ax, clf, tt in zip(axes.ravel(), [nb, lr, lda, svm], ['GaussianNB', 'Logistic Re
                                                          'LinearDiscriminantAnalysis', 'SVM
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.4)
    scatter = ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, s=50, edgecolor='k
    ax.set_title(tt)
    ax.set_xlabel('acceleration')
    ax.set_ylabel('origin')
    legend = ax.legend(*scatter.legend_elements(),
                       loc="upper right", title="Classes")
    ax.add_artist(legend)
plt.show()
```



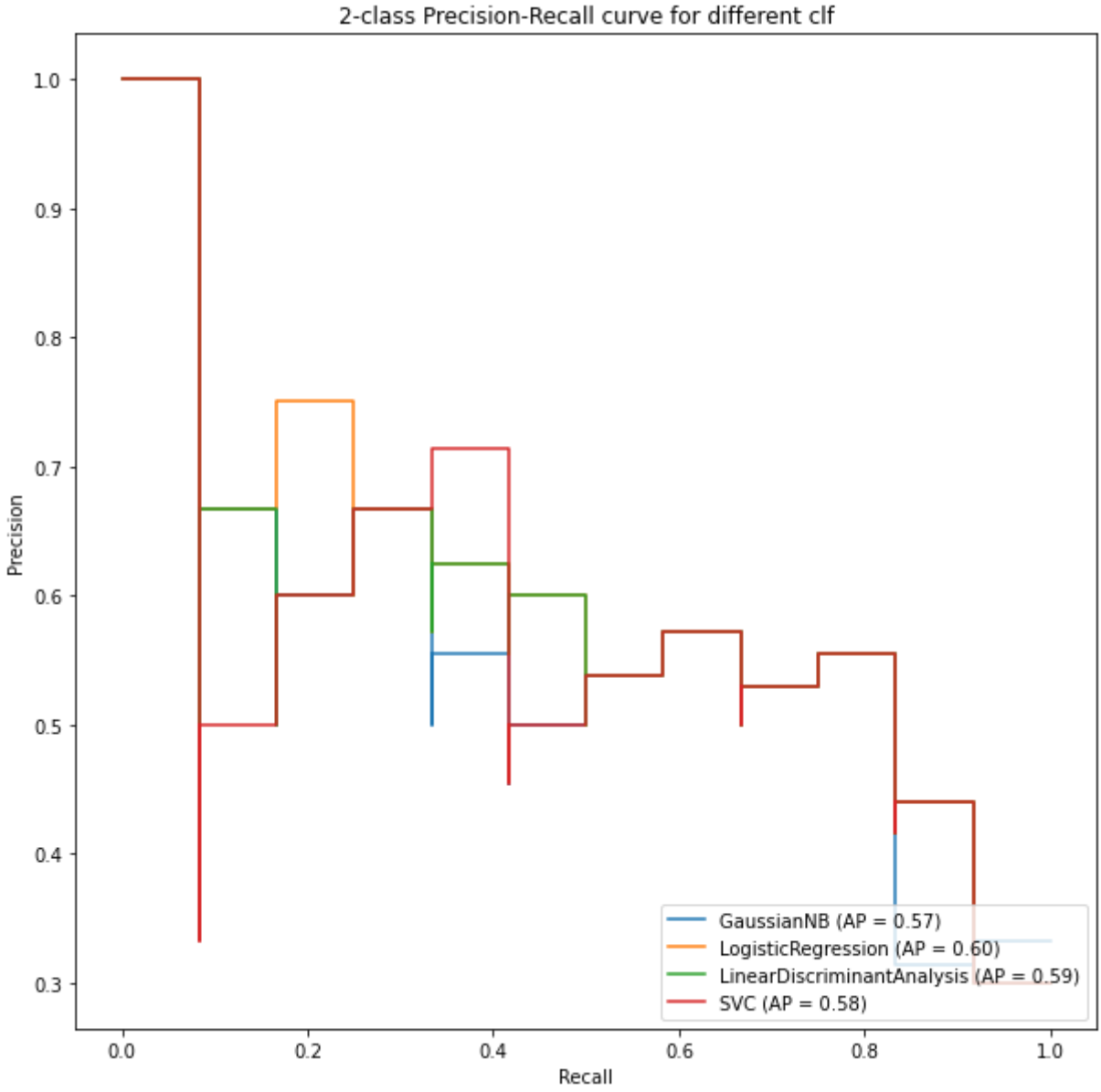
In [16]:

```
x_min, x_max = X_test[:, 0].min() - 1, X_test[:, 0].max() + 1
y_min, y_max = X_test[:, 1].min() - 1, X_test[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                     np.arange(y_min, y_max, 0.01))
f, axes = plt.subplots(2, 2, sharex='col', sharey='row', figsize=(10, 8))
f.suptitle('Decision boundaries on test set', fontsize=16)

for ax, clf, tt in zip(axes.ravel(), [nb, lr, lda, svm], ['GaussianNB', 'Logistic Re
                                                          'LinearDiscriminantAnalysis', 'SVM
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.4)
    scatter = ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, s=50, edgecolor='k',
    ax.set_title(tt)
    ax.set_xlabel('acceleration')
    ax.set_ylabel('origin')
    legend = ax.legend(*scatter.legend_elements(),
                       loc="upper right", title="Classes")
    ax.add_artist(legend)
plt.show()
```



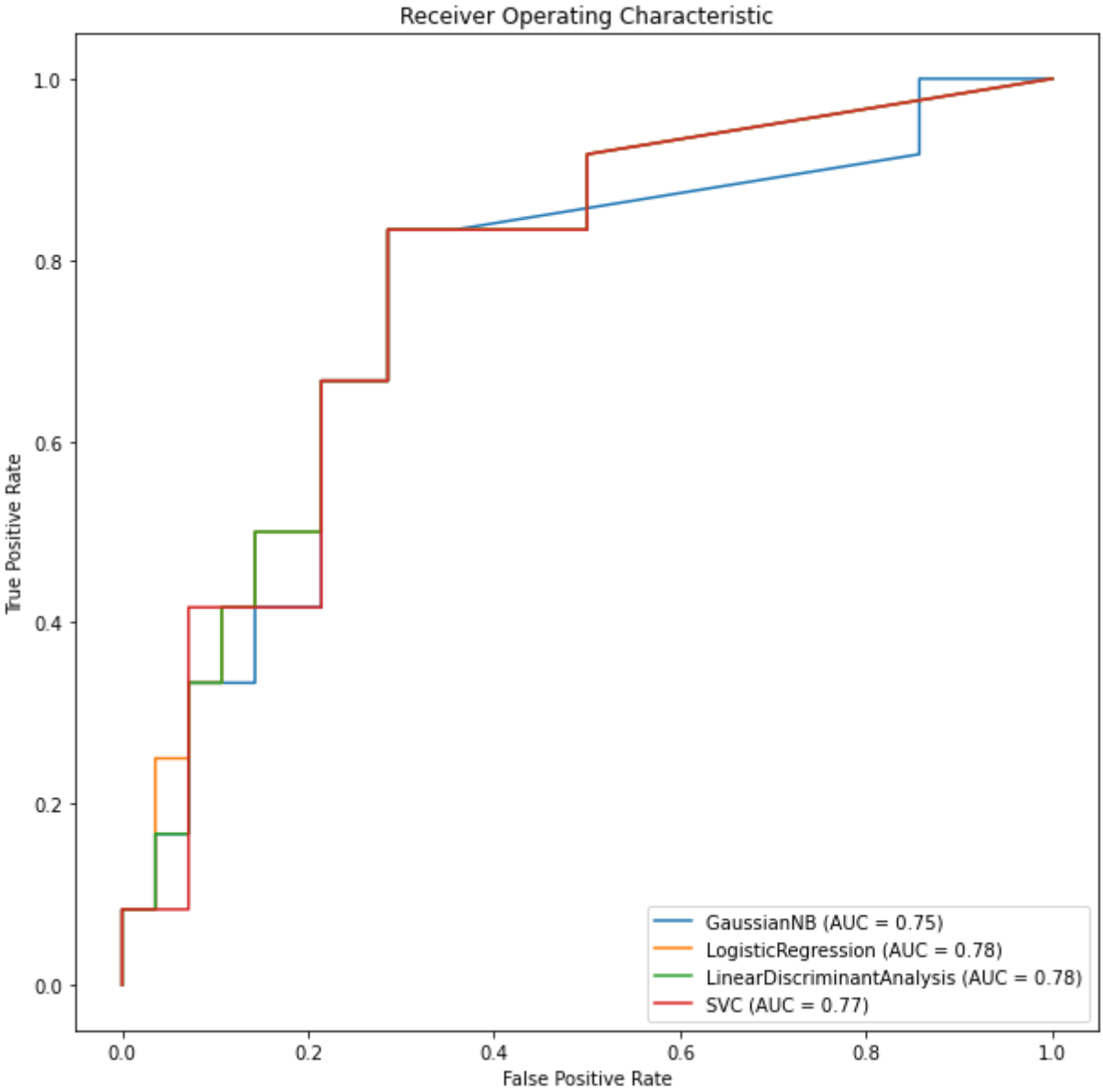
1. Визуализируйте на одном рисунке ROC кривые для каждого из классификаторов, подписывая оси и рисунок и создавая легенду для методов бинарной классификации.

In [17]:
```
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot()
plot_precision_recall_curve(nb, X_test, y_test, ax=ax)
plot_precision_recall_curve(lr, X_test, y_test, ax=ax)
plot_precision_recall_curve(lda, X_test, y_test, ax=ax)
plot_precision_recall_curve(svm, X_test, y_test, ax=ax)
ax.set_title('2-class Precision-Recall curve for different clf')
plt.legend(loc='lower right');
```



In [18]:

```
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot()
plot_roc_curve(nb, X_test, y_test, ax=ax)
plot_roc_curve(lr, X_test, y_test, ax=ax)
plot_roc_curve(lda, X_test, y_test, ax=ax)
plot_roc_curve(svm, X_test, y_test, ax=ax)
ax.set_title('Receiver Operating Characteristic')
plt.legend(loc='lower right');
```



1. Определите лучший метод бинарной классификации набора данных по показателю ROC_AUC (площади под ROC кривой).

In [19]:
```
models = [nb, lr, lda, svm]
names = ['GaussianNB', 'Logistic Regression', 'LinearDiscriminantAnalysis', 'SVM']
print('Roc-auc scores:\n')
for model, name in zip(models, names):
    preds = model.predict_proba(X_test)[:, 1]
    score = roc_auc_score(y_test, preds)
    print(f'{name:<30} {score:<10}')
```

```
Roc-auc scores:

GaussianNB                     0.75
Logistic Regression            0.7797619047619048
LinearDiscriminantAnalysis     0.7767857142857143
SVM                            0.7708333333333333
```

Лучший метод - логистическая регрессия.

In [ ]: