**Отчет по дисциплине:**  Теория машинного обучения

**Язык программирования:** Python

**Тема №2:** Байесовские методы классификации

**Ход работы программы:**

Используя готовые методы из библиотеки sklearn, применить байесовские алгоритмы классификации «наивный байес», линейный дискриминант Фишера и plug-in алгоритм для классификации исходных данных. Данный метод заключается в теореме утверждающей, что если известны плотности распределения классов, то алгорит классификации сожно выписать в явном виде:

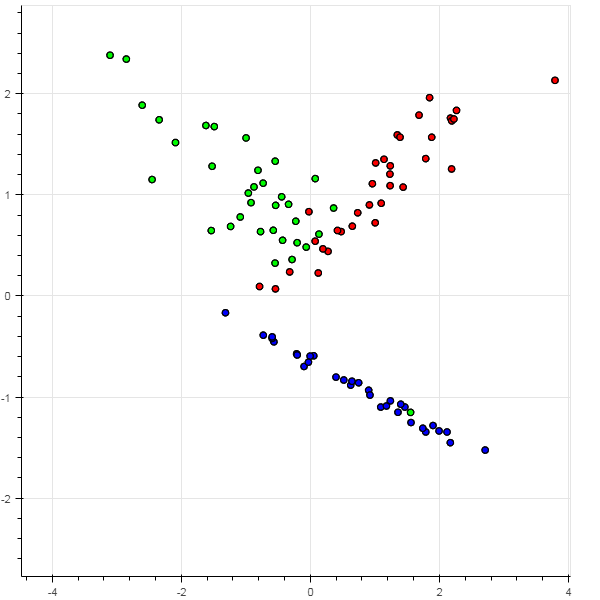
Рассмотрим более детально два метода это Наивный Байесовский и линейный дискриминант Фишера.

**Наивный Байесовский классификатор** основан на гипотезе о том, что если объекты описываются числовыми признаками , и эти признаки независимые случайные величины, то функции правдоподобия классов представлены в виде:

где , a- плотность распределения j-го признака для класса y.

В пакете **sklearn** данный классификатор описан в классе **GaussianNB**

Класс **GaussianNB** позволяет установить априорные вероятности для каждого из исследуемых классов.



Мы не имеем данных по априорным вероятностям классов, можем инициализировать класс **GaussianNB ,** который имеет вид:

x\_steps = 400

y\_steps = 400

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

x\_step\_len = (x\_max-x\_min)/x\_steps

y\_step\_len = (y\_max-y\_min)/y\_steps

**clf = GaussianNB()**

**clf.fit(X,Y)**

classes\_field = []

for y in range(x\_steps):

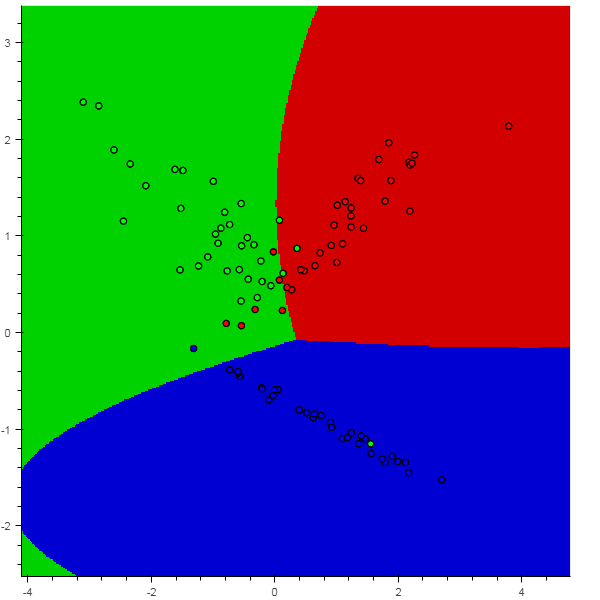
classes\_line = []

for x in range(y\_steps):

classes\_line.append(clf.predict([[x\_min+(x\*x\_step\_len),y\_min+(y\*y\_step\_len)]]))

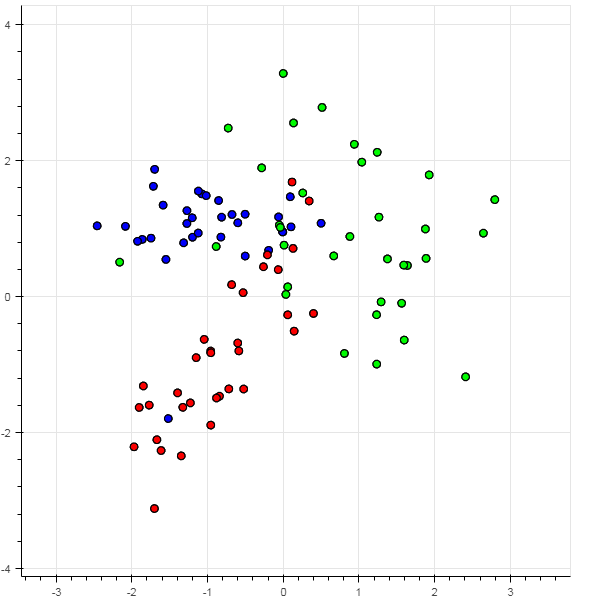
classes\_field.append(classes\_line)

Следовательно имеем результат:



А так же рассмотрим **линейный дискриминант Фишера**. В пакете **sklearn** данный классификатор описан в классе **LinearDiscriminantAnalysis.**

Имеем выборку:

****

Не имея данных по приоритетным классам, инициализируем классификатор следующим образом в коде:

x\_steps = 400

y\_steps = 400

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

x\_step\_len = (x\_max-x\_min)/x\_steps

y\_step\_len = (y\_max-y\_min)/y\_steps

**clf = LinearDiscriminantAnalysis()**

**clf.fit(X,Y)**

classes\_field = []

for y in range(x\_steps):

classes\_line = []

for x in range(y\_steps):

classes\_line.append(clf.predict([[x\_min+(x\*x\_step\_len),y\_min+(y\*y\_step\_len)]]))

classes\_field.append(classes\_line)

Следовательно имеем результат:

