**Постановка задачи**: с помощью библиотеки sklearn, keras и matplotlib реализовать 3 метода классификации двумерных данных на языке программирования python

Этапы выполнения задачи

1. ***Выбор трёх методов классификации***

* Метод k ближайших соседей (k-nearest)
* Наивный Байесовский классификатор (naive Bayes classifier)
* Метод опорных векторов (SVM)

1. ***Генерация 5 вариантов исходных данных***:
2. Генерация точек вокруг двух окружностей:

|  |
| --- |
| *from sklearn import datasets*  *n\_samples = 500 seed = 30 noisy\_circles = datasets.make\_circles(n\_samples=n\_samples, factor=0.5, noise=0.05, random\_state=seed) x, y = noisy\_circles* |

1. Генерация точек вокруг двух парабол:

|  |
| --- |
| *noisy\_moons = datasets.make\_moons(n\_samples=n\_samples, noise=0.05, random\_state=seed) x, y = noisy\_moons* |

1. Немного хаотичное распределение:

|  |
| --- |
| *cluster\_std = [1.0, 0.5] varied = datasets.make\_blobs(n\_samples=n\_samples, cluster\_std=cluster\_std, random\_state=seed, centers=2) x, y = varied* |

1. Точки вокруг прямых

|  |
| --- |
| *random\_state = 170 x, y = datasets.make\_blobs(n\_samples=n\_samples, random\_state=random\_state, centers=2) transformation = [[0.6, -0.6], [-0.4, 0.8]] x\_aniso = np.dot(x, transformation) aniso = (x\_aniso, y) x, y = aniso* |

1. Точки в слабо пересекающихся областях

|  |
| --- |
| *blobs = datasets.make\_blobs(n\_samples=n\_samples, random\_state=seed, centers=2) x, y = blobs* |

1. ***Разделение исходных данных на обучающую и тестовую выборки*** (80% исходных данных для обучения, 20% для проверки) с помощью функции *train\_test\_split*:

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42) |

1. ***Создание и обучение моделей классификации данных*** (выбрать 3 варианта по своему желанию)
2. **Метод k ближайших соседей**:

|  |
| --- |
| from sklearn.neighbors import KNeighborsClassifier  model = KNeighborsClassifier(n\_neighbors=3) model.fit(x\_train, y\_train) y\_pred = model.predict(x\_test) |

**2.Наивный Байесовский классификатор**:

|  |
| --- |
| from sklearn.naive\_bayes import GaussianNB  model = GaussianNB() model.fit(x\_train, y\_train) y\_pred = model.predict(x\_test) |

**3. Метод опорных векторов:**

|  |
| --- |
| from sklearn.svm import SVC  model = SVC(kernel='rbf', C=1.0) model.fit(x\_train, y\_train) y\_pred = model.predict(x\_test) |

5) **Создание сетки для построения границы между классами**

1. Генерация сетки по осям х и у:

|  |
| --- |
| temp\_x = np.linspace(x[:, 0].min() - 1, x[:, 0].max() + 1, 100) temp\_y = np.linspace(x[:, 1].min() - 1, x[:, 1].max() + 1, 100) |

1. Создание сетки для построения границы

|  |
| --- |
| xx, yy = np.meshgrid(temp\_x, temp\_y) |

1. Получение прогнозов для всех точек пересечения сетки

|  |
| --- |
| Z = model.predict(np.c\_[xx.ravel(), yy.ravel()]) Z = Z.reshape(xx.shape) |

6) **Визуализация результатов работы моделей на различных вариантах исходных данных**

Для визуализации можно воспользоваться следующим кусочком кода:

|  |
| --- |
| markers = ['x', 'o']  fig, ax = plt.subplots() plt.contourf(xx, yy, Z, alpha=0.3, cmap='bwr')  *# Области классов*  for i in range(len(x\_train)):     ax.scatter(x\_train[i, 0], x\_train[i, 1], marker=markers[y\_train[i]], c='b')  for i in range(len(x\_test)):     m = 'r'     if y\_test[i] == y\_pred[i]:         m = 'g'     ax.scatter(x\_test[i, 0], x\_test[i, 1], marker=markers[y\_test[i]], c=m) plt.show() |

***Итоговая задача***: с помощью matplotlib построить небольшую таблицу, визуализирующую результат работы различных методов классификации на различных вариантах исходных данных. В данной таблице будет 5 строк и 3 столбца, где по строкам будут различные варианты исходных данных, а по столбцам различные методы классификации.