Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Отчет

Лабораторные работа № 4
«Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей»

По курсу «Технологии машинного обучения»

ИСПОЛНИТЕЛЬ:
Молева Анастасия
Группа ИУ5-61Б
_____

"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.
_____

"__"_____2020 г.

Москва 2020

Цель лабораторной работы: изучение сложных способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
3. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
4. Постройте модель и оцените качество модели с использованием кросс-валидации.
5. Произведите подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации.

Решение:

Dataset

```
In [104]: X, y = make_moons(n_samples=500, noise=0.30, random_state=42)
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

In [105]: X.shape, X_train.shape, X_test.shape

Out[105]: ((500, 2), (350, 2), (150, 2))
```

KneighborsClassifier

In [106]:
```
k_neigh = KNeighborsClassifier(n_jobs=-1)
k_neigh
```

Out[106]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
```

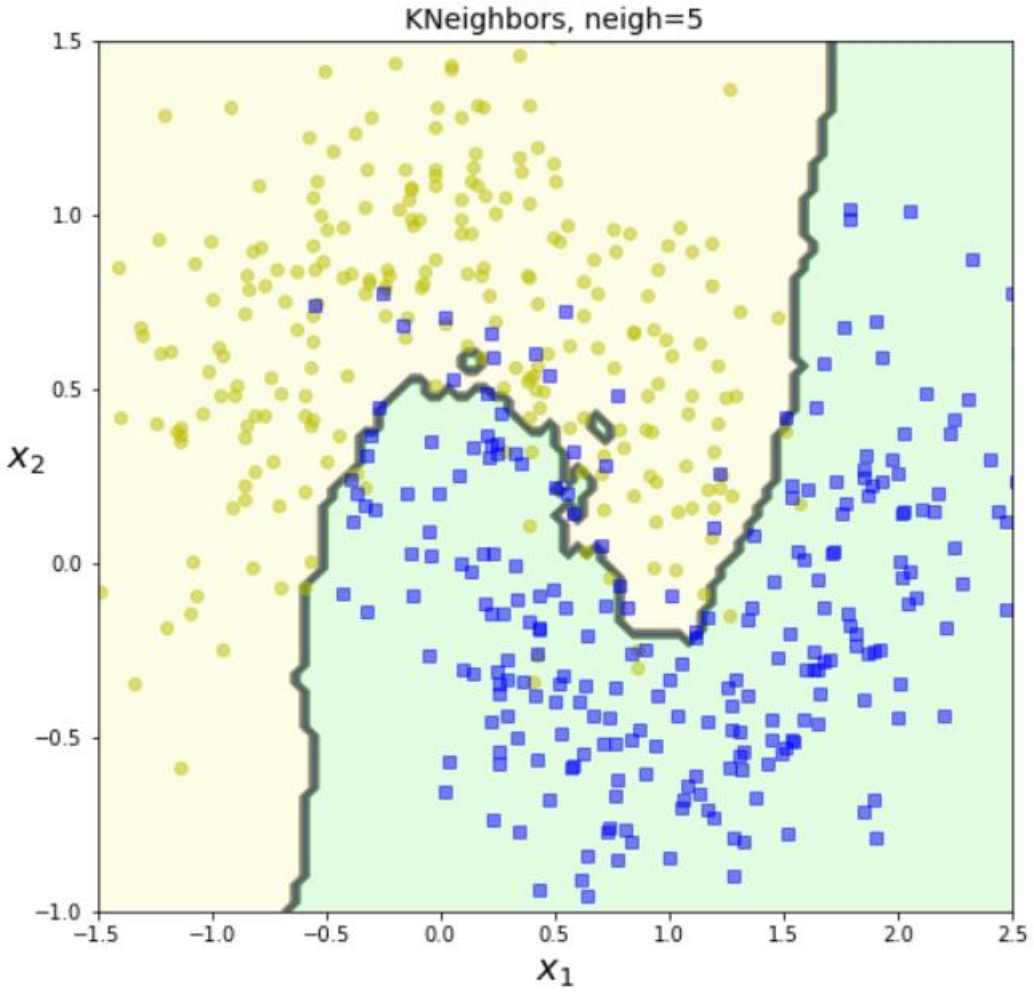
In [42]:
```
import time
import datetime

start_time = datetime.datetime.now()
k_neigh.fit(X_train, y_train)
print("Time elapse:", datetime.datetime.now() - start_time)
```

```
Time elapse: 0:00:00.001008
```

Критерий качества

In [43]:
```
y_pred = k_neigh.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[43]:
```
0.9
```

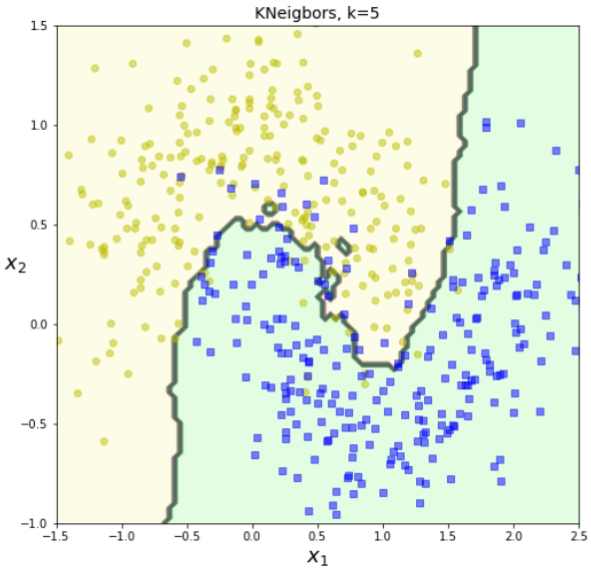


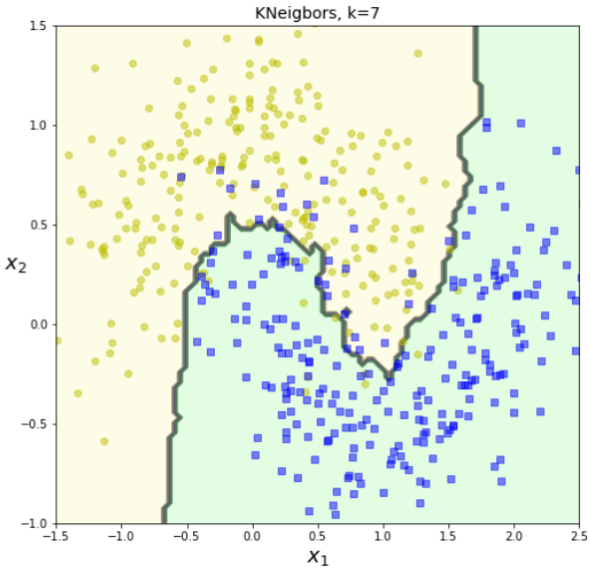
Cross validation

In [56]:
```
max_mean = -1
start_time = datetime.datetime.now()
for i in range(1, 20, 2):
    clf = KNeighborsClassifier(n_jobs=-1, n_neighbors=i)
    cvs = cross_val_score(clf, X, y, cv=kf, scoring='accuracy')
    mean = cvs.mean()
    print(str(i) + ": " + str(mean))
    if mean > max_mean:
        max_mean = mean
        max_n = i
print("Time elapse: ", datetime.datetime.now() - start_time)
print(max_n, max_mean)
```

```
1: 0.858
3: 0.9040000000000001
5: 0.9099999999999999
7: 0.922
9: 0.9200000000000002
11: 0.9199999999999999
13: 0.9219999999999999
15: 0.9179999999999999
17: 0.9119999999999999
19: 0.908
Time elapse:  0:00:00.820541
7 0.922
```

При 7 соседях самая лучшая модель с точностью 0.922





GridSearchCV

In [94]:
```
start_time = datetime.datetime.now()
param_grid = [{'n_neighbors': [3, 5, 7, 9, 11], 'algorithm': ['ball_tree', 'kd_tree', 'brute'],
               'weights': ['uniform', 'distance']}]
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid=param_grid, cv=kf, n_jobs=-1)
grid_search.fit(X_train, y_train)
print("Time elapse: ", datetime.datetime.now() - start_time)
```

```
Time elapse:  0:00:00.188795
```

In [99]:
```
k_best_new = grid_search.best_estimator_
k_best_new.fit(X_train, y_train)
```

Out[99]:
```
KNeighborsClassifier(algorithm='ball_tree', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [100]:
```
y_pred = k_best_new.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[100]:
```
0.9
```

Вывод:

Научился кросс-валидации и перекрестной проверке.