аНАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт до лабораторної роботи №5

з курсу

«Машине навчання»

студента 2 курсу
групи ІТ-02
Макарова Іллі Сергійовича

Викладач:
Оніщенко В.

Тема: Класифікація методом k найближчих сусідів і набір даних Digits

Виконання:

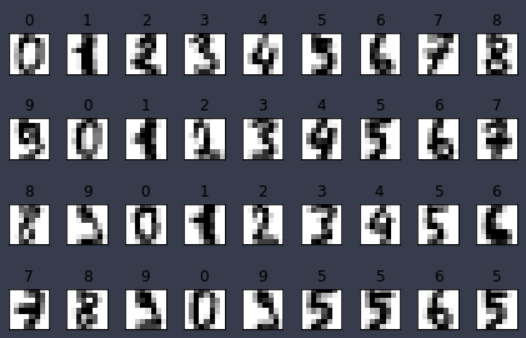
Ок, суть роботы была в классификации рукописных цифр, ну что же, начнем:

Импортируем, что нам понадобиться, и загружаем датасет

```
In [93]: import matplotlib.pyplot as plt
         from sklearn.datasets import load_digits
         from sklearn.model_selection import train_test_split
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.metrics import confusion_matrix, classification_report
         from sklearn.svm import SVC
         from sklearn.naive_bayes import GaussianNB

In [68]: digits = load_digits()
```

Визуализируем 36 цифр, чтоб посмотреть как они вообще выглядят.

```
In [81]: figure, axes = plt.subplots(nrows=4, ncols=9, figsize=(6, 4))

         for axes, image, target in zip(axes.ravel(), digits.images, digits.target):
             axes.imshow(image, cmap=plt.cm.gray_r)
             axes.set_xticks([])
             axes.set_yticks([])
             axes.set_title(target)

         plt.tight_layout()
```



Ок, теперь разобьем нашу выборку на тренировочную и тестовую, та и сразу создадим модель, и научим ее

```
In [75]: x_train, x_test, y_train, y_test = train_test_split(
             digits.data, digits.target, random_state=11, test_size=0.25
         )

In [78]: knn = KNeighborsClassifier()
         knn.fit(x_train, y_train)
         None
```

Пол дела сделано, давайте посмотрим на примере 24 цифр из тестовой выборки, как мы научились собственно классифицировать то:

```
In [83]: y_predicted = knn.predict(x_test)

         for predicted, actual in zip(y_predicted[:24], y_test[:24]):
             print(f'actual value: {actual}, predicted: {predicted}')

actual value: 0, predicted: 0
actual value: 4, predicted: 4
actual value: 9, predicted: 9
actual value: 9, predicted: 9
actual value: 3, predicted: 3
actual value: 1, predicted: 1
actual value: 4, predicted: 4
actual value: 1, predicted: 1
actual value: 5, predicted: 5
actual value: 0, predicted: 0
actual value: 4, predicted: 4
actual value: 9, predicted: 9
actual value: 4, predicted: 4
actual value: 1, predicted: 1
actual value: 5, predicted: 5
actual value: 3, predicted: 3
actual value: 3, predicted: 3
actual value: 8, predicted: 8
actual value: 3, predicted: 5
actual value: 6, predicted: 6
actual value: 9, predicted: 9
actual value: 6, predicted: 6
actual value: 0, predicted: 0
actual value: 6, predicted: 6
```

Ну что, выглядит хайпово, можем тут же и score посчитать с матрицей несоответствия вывести

```
In [84]: print(f'Model score for KNN: {round(knn.score(x_test, y_test), 2)}')

Model score: 0.98

In [86]: confusion_matrix(y_true=y_test, y_pred=y_predicted)

array([[45,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 45,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 54,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 42,  0,  1,  0,  1,  0,  0],
       [ 0,  0,  0,  0, 49,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 38,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 42,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 45,  0,  0],
       [ 0,  1,  1,  2,  0,  0,  0,  0, 39,  1],
       [ 0,  0,  0,  0,  1,  0,  0,  0,  1, 41]])
```

Ну выглядит не плохо, что там дальше, надо еще репорт классификации:

```
In [90]: names = [str(digit) for digit in digits.target_names]
         report = classification_report(y_true=y_test, y_pred=y_predicted, target_names=names)

In [91]: print(report)
```

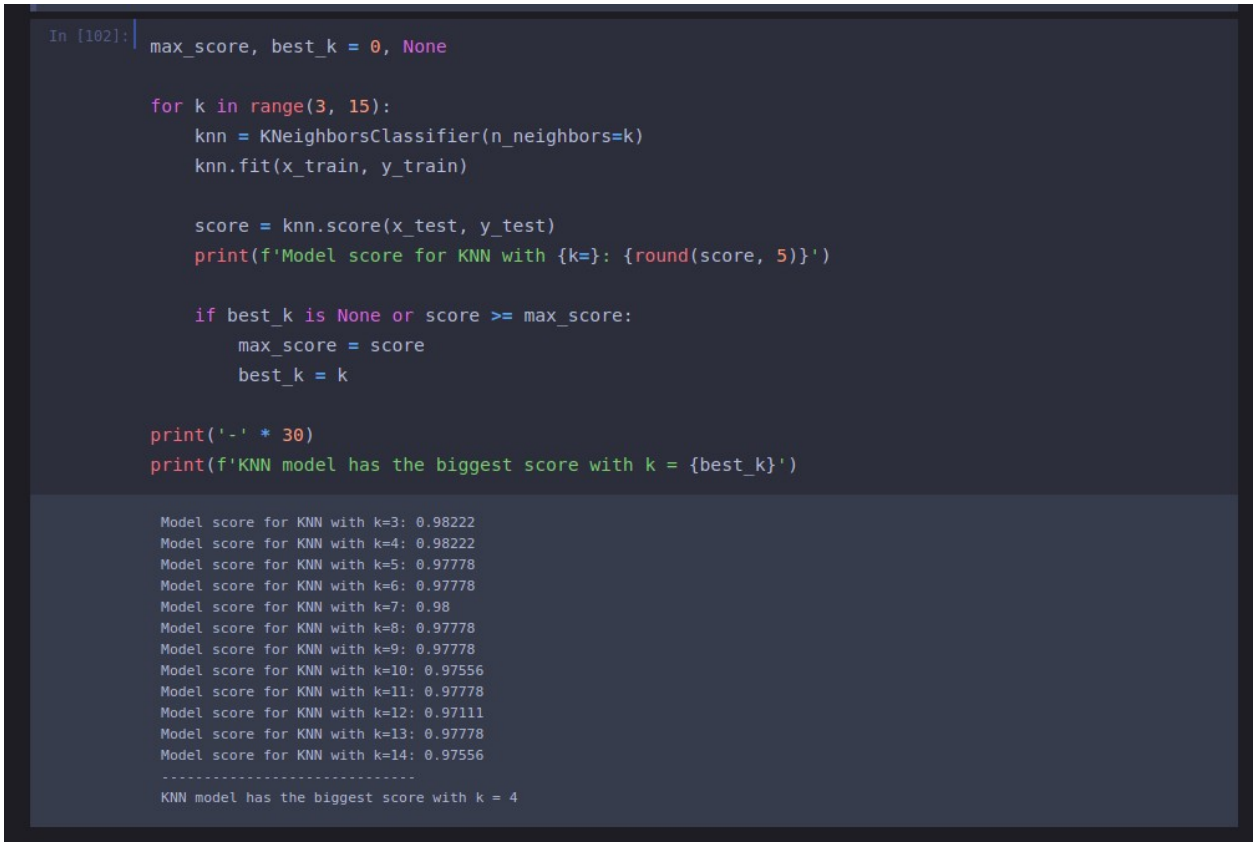
	precision	recall	f1-score	support
0	1.00	1.00	1.00	45
1	0.98	1.00	0.99	45
2	0.98	1.00	0.99	54
3	0.95	0.95	0.95	44
4	0.98	0.98	0.98	50
5	0.97	1.00	0.99	38
6	1.00	1.00	1.00	42
7	0.96	1.00	0.98	45
8	0.97	0.89	0.93	44
9	0.98	0.95	0.96	43
accuracy			0.98	450
macro avg	0.98	0.98	0.98	450
weighted avg	0.98	0.98	0.98	450

Такс, к KNN мы еще вернемся в конце, попробуем гиперпараметр k подобрать, а пока еще затестим SVC и GaussianNB:

```
In [94]: svc = SVC()
         svc.fit(x_train, y_train)

         print(f'Model score for SVC: {round(svc.score(x_test, y_test), 2)}')

         Model score for SVC: 0.99

In [95]: gnb = GaussianNB()
         gnb.fit(x_train, y_train)

         print(f'Model score for GNB: {round(gnb.score(x_test, y_test), 2)}')

         Model score for GNB: 0.87
```

Ну судя по всему метод опорных векторов не много впереди, а вот наивный баес не очень, что собственно не удивительно, он не много не для того создан, как мне кажется.

Итак, финита ля комедия, ну почти, еще нужно, как я уже упоминал подобрать k для нашего KNN



```
In [102]:
max_score, best_k = 0, None

for k in range(3, 15):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train, y_train)

    score = knn.score(x_test, y_test)
    print(f'Model score for KNN with {k=}: {round(score, 5)}')

    if best_k is None or score >= max_score:
        max_score = score
        best_k = k

print('-' * 30)
print(f'KNN model has the biggest score with k = {best_k}')
```

```
Model score for KNN with k=3: 0.98222
Model score for KNN with k=4: 0.98222
Model score for KNN with k=5: 0.97778
Model score for KNN with k=6: 0.97778
Model score for KNN with k=7: 0.98
Model score for KNN with k=8: 0.97778
Model score for KNN with k=9: 0.97778
Model score for KNN with k=10: 0.97556
Model score for KNN with k=11: 0.97778
Model score for KNN with k=12: 0.97111
Model score for KNN with k=13: 0.97778
Model score for KNN with k=14: 0.97556
------------------------------
KNN model has the biggest score with k = 4
```

Ну если верить моему скрипту, то лучше всего наша моделька работает при k = 4, впрочем все еще не дотягивает до опорных векторов чутка.