**Выбор классификаторов и данных**

В этой лабораторной работе были выбраны и утверждены следующие классификаторы:

1) Gaussian Process (Гауссовская классификация)

2) Support Vector Machines (kernel=’rbf’) (Метод опорных векторов)

3) Multi-layer Perceptron (Многослойный перцептрон)

и следующий датасет (возрастные рейтинги видеоигр в зависимости от их контента):

<https://www.kaggle.com/datasets/imohtn/video-games-rating-by-esrb>

**Решение**

1) Импорт необходимых библиотек и классов

import pandas

import matplotlib.pyplot as pyplot

import pylab

from sklearn.preprocessing import StandardScaler

from sklearn.gaussian\_process import GaussianProcessClassifier

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

2) Загрузка тренировочных и тестовых данных

#loading datasets

train\_df = pandas.read\_csv('Video\_games\_esrb\_rating.csv')

test\_df = pandas.read\_csv('test\_esrb.csv')

3) Подготовка данных

#preprocessing

def preprocessing(df):

df = df.copy()

df = df.drop('title', axis = 1)

X = df.drop('esrb\_rating', axis = 1)

y = df['esrb\_rating']

X = pandas.DataFrame(X, index = X.index, columns = X.columns)

return X, y

4) Масштабирование данных

#scaling

scaler = StandardScaler()

X\_train, y\_train = preprocessing(train\_df)

X\_train = scaler.fit\_transform(X\_train)

X\_test, y\_test = preprocessing(test\_df)

X\_test = scaler.fit\_transform(X\_test)

5) Использование классификаторов для предсказания возрастного рейтинга видеоигр

#run classifiers

gp = GaussianProcessClassifier(kernel = None, random\_state = 0).fit(X\_train, y\_train)

gp\_predictions = pandas.Series(gp.predict(X\_test))

print('GP Classifier accuracy: ' + str(gp.score(X\_test, y\_test)\*100) + '%')

svm = SVC(kernel = 'rbf').fit(X\_train, y\_train)

svm\_predictions = pandas.Series(svm.predict(X\_test))

print('SVM Classifier accuracy: ' + str(svm.score(X\_test, y\_test)\*100) + '%')

mlp = MLPClassifier(random\_state = 1, max\_iter = 300).fit(X\_train, y\_train)

mlp\_predictions = pandas.Series(mlp.predict(X\_test))

print('MLP Classifier accuracy: ' + str(mlp.score(X\_test, y\_test)\*100) + '%')

6) Вывод круговых графиков

#output pie charts

pylab.subplot(1, 2, 1)

pyplot.pie(y\_test.value\_counts().sort\_index(), labels = sorted(y\_test.unique()), autopct='%1.1f%%')

pyplot.title('Original data')

pylab.subplot(1, 2, 2)

pyplot.pie(gp\_predictions.value\_counts().sort\_index(), labels = sorted(gp\_predictions.unique()), autopct='%1.1f%%')

pyplot.title('GP Predictions')

pyplot.show()

pylab.subplot(1, 2, 1)

pyplot.pie(y\_test.value\_counts().sort\_index(), labels = sorted(y\_test.unique()), autopct='%1.1f%%')

pyplot.title('Original data')

pylab.subplot(1, 2, 2)

pyplot.pie(svm\_predictions.value\_counts().sort\_index(), labels = sorted(svm\_predictions.unique()), autopct='%1.1f%%')

pyplot.title('SVM Predictions')

pyplot.show()

pylab.subplot(1, 2, 1)

pyplot.pie(y\_test.value\_counts().sort\_index(), labels = sorted(y\_test.unique()), autopct='%1.1f%%')

pyplot.title('Original data')

pylab.subplot(1, 2, 2)

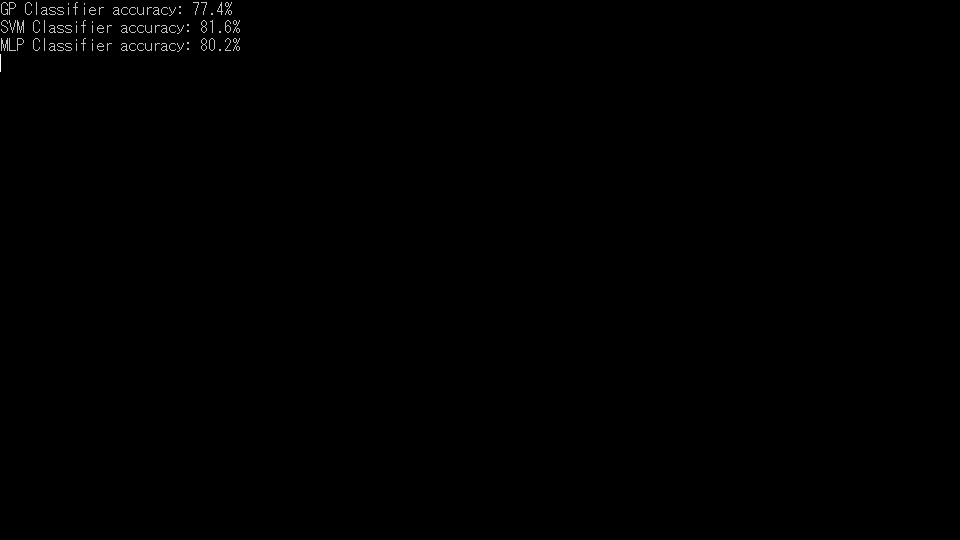
pyplot.pie(mlp\_predictions.value\_counts().sort\_index(), labels = sorted(mlp\_predictions.unique()), autopct='%1.1f%%')

pyplot.title('MLP Predictions')

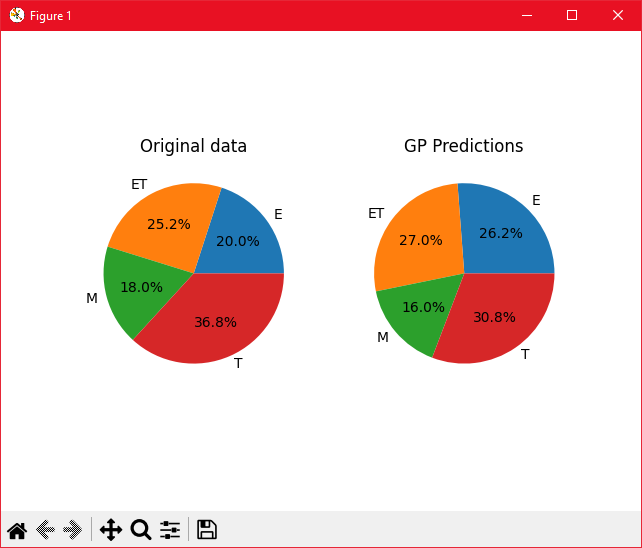
pyplot.show()

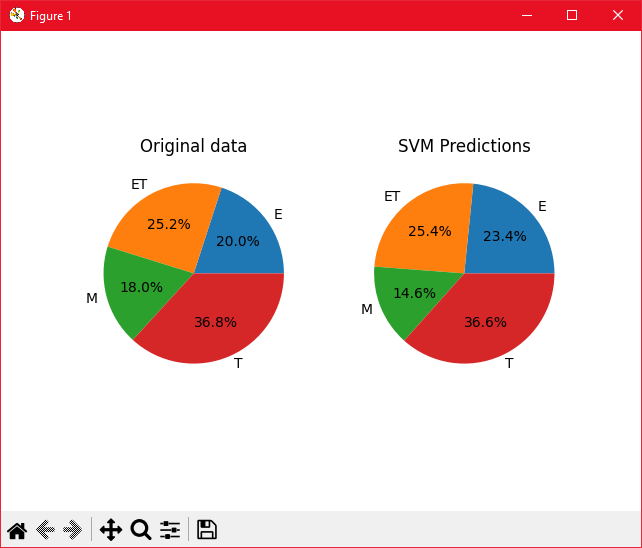
**Результаты**

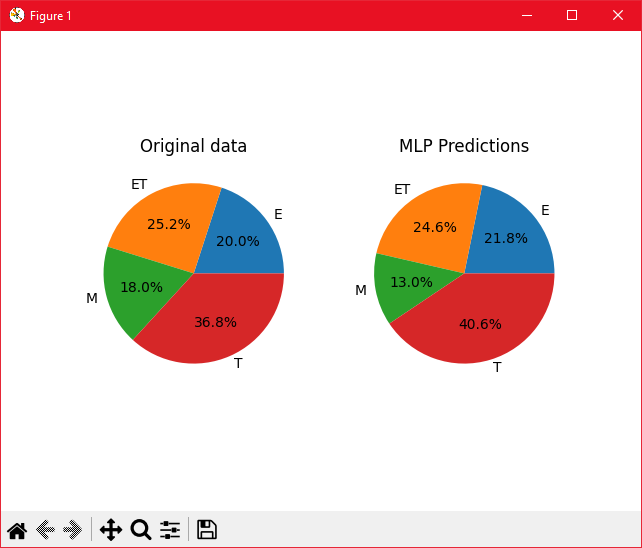
1) Точность использованных классификаторов согласно методу score() библиотеки sklearn:



2) Визуализация данных в виде круговых графиков (% от общего размера тестовой выборки):







Из графиков и посчитанной с помощью метода score точности видно, что метод опорных векторов (Support Vector Machines) справляется с задачей лучше всего (81.6% точности), недалеко за ним идёт многослойный перцептрон (Multi-layer Perceptron; 80.2% точности), а с достаточным отрывом (77.4% точности) наихудшим из рассмотренных классификаторов является Гауссовская классификация (Gaussian Process). Стоит, однако, заметить, что многослойный перцептрон работает гораздо медленнее и вызывает заметную задержку перед выводом его точности в консоль, в отличие от двух других методов, работающих почти моментально.