



# Машинное обучение

---

НИЯУ МИФИ, КАФЕДРА ФИНАНСОВОГО МОНИТОРИНГА

КУРС ЛЕКЦИЙ

В.Ю. РАДЫГИН. Д.Ю. КУПРИЯНОВ

ЛЕКЦИЯ 7

# Часть 1

---

МЕТРИКИ

# Несбалансированный пример

---

Все предыдущие примеры работали с довольно сбалансированным набором данных. В каждом классе было приблизительно одинаковое число наблюдений. Но что будет, если мы возьмём несбалансированный пример? Например, такой как на сайте kaggle: <https://www.kaggle.com/mlg-ulb/creditcardfraud>.

Чтобы пример не был слишком вычислительно сложным, используем только первые 20 000 наблюдений.

# Пример 5. Подготовка данных

```
*example2-1bin.py - E:\Works\Victor\Students\STDB\Term2\example2-1bin.py (3.7.2)*
File Edit Format Run Options Window Help
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import svm

table = pd.read_csv("creditcard.csv").head(20000)

from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()

x = scaler_std.fit_transform(table[['V1', 'V2', 'V3', 'V4', 'V5', 'V6',
                                   'V7', 'V8', 'V9', 'V10', 'V11', 'V12',
                                   'V13', 'V14', 'V15', 'V16', 'V17', 'V18',
                                   'V19', 'V20', 'V21', 'V22', 'Amount']])
```

Ln: 9 Col: 0

# Пример 5. Подготовка данных

---

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import svm

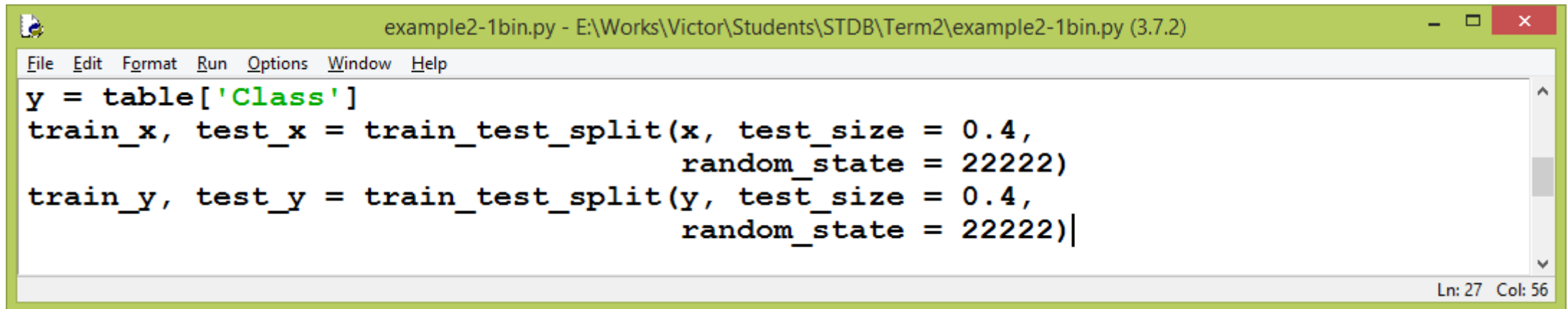
table = pd.read_csv("creditcard.csv").head(20000)
```

```
from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()

x = scaler_std.fit_transform(table[[
    'V1', 'V2', 'V3', 'V4', 'V5', 'V6',
    'V7', 'V8', 'V9', 'V10', 'V11', 'V12',
    'V13', 'V14', 'V15', 'V16', 'V17', 'V18',
    'V19', 'V20', 'V21', 'V22', 'Amount']])
```

# Пример 5. Подготовка данных

---



The image shows a screenshot of a Python IDE window titled "example2-1bin.py - E:\Works\Victor\Students\STDB\Term2\example2-1bin.py (3.7.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

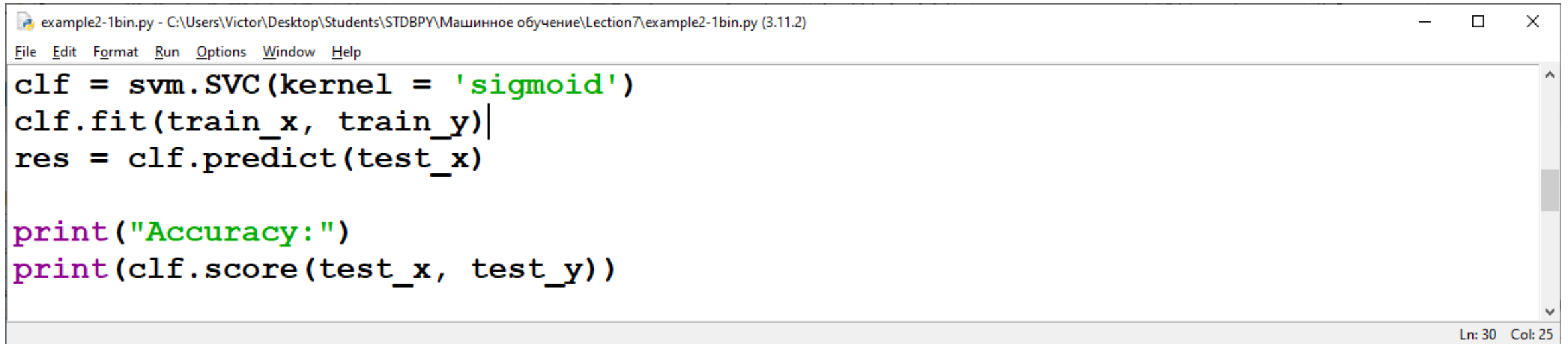
```
y = table['Class']
train_x, test_x = train_test_split(x, test_size = 0.4,
                                   random_state = 22222)
train_y, test_y = train_test_split(y, test_size = 0.4,
                                   random_state = 22222)|
```

The status bar at the bottom right indicates "Ln: 27 Col: 56".



# Пример 5. Обучение и тестирование

---



The image shows a screenshot of a Python IDE window titled "example2-1bin.py - C:\Users\Victor\Desktop\Students\STDBPY\Машинное обучение\Lecture7\example2-1bin.py (3.11.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

```
clf = svm.SVC(kernel = 'sigmoid')
clf.fit(train_x, train_y)
res = clf.predict(test_x)

print("Accuracy:")
print(clf.score(test_x, test_y))
```

The status bar at the bottom right indicates "Ln: 30 Col: 25".



# Пример 5. Обучение и тестирование

---

```
clf = svm.SVC(kernel = 'sigmoid')
```

```
clf.fit(train_x, train_y)
```

```
res = clf.predict(test_x)
```

```
print("Accuracy:")
```

```
print(clf.score(test_x, test_y))
```

# Результат

---

Метрика accuracy для полученного классификатора будет равна 0,9965. Или 0,996 приблизительно (при 8000 наблюдений и 35 мошеннических).

Но похожий результат можно получить ничего не делая. Пусть мой другой классификатор таков:

`class = 0`

Тогда его точность  $(8000 - 35) / 8000 = 0,995625$  Или 0,996 приблизительно.

Будет ли второй классификатор хорошо? НЕТ! Но метрика одинаковая. Это говорит о некачественности метрики Accuracy для таких задач. Посмотрим, какие ещё бывают метрики.

# Матрица ошибок (confusion matrix)

---

Матрица ошибок – это таблица, где колонки соответствуют истинным классам, строки – предсказанным классам, а в ячейках число наблюдений, у которых должен быть класс, соответствующий колонке, а получился класс, соответствующий строке.

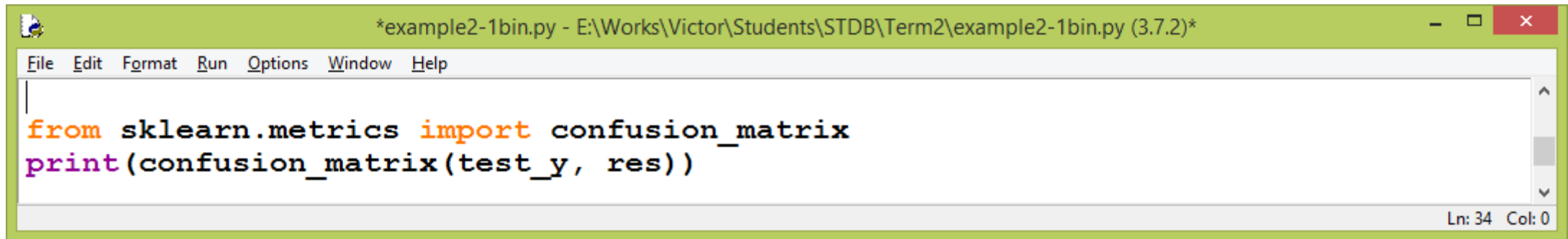
	Класс 1	...	Класс N
Класс 1'	$K_{1',1}$	...	$K_{1',N}$
...	...	...	...
Класс N'	$K_{N',1}$	...	$K_{N',N}$

# Для нашей задачи

---

	Обычная транзакции согласно набору данных	Мошенническая транзакции согласно набору данных
Обычная транзакции согласно предсказанию	$K_{0,0}$	$K_{0,1}$
Мошенническая транзакции согласно предсказанию	$K_{0,1}$	$K_{1,1'}$

# Пример 5. Матрица ошибок

A screenshot of a Python IDE window titled '\*example2-1bin.py - E:\Works\Victor\Students\STDB\Term2\example2-1bin.py (3.7.2)\*'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains two lines of Python code: 'from sklearn.metrics import confusion\_matrix' and 'print(confusion\_matrix(test\_y, res))'. The status bar at the bottom right shows 'Ln: 34 Col: 0'.

```
*example2-1bin.py - E:\Works\Victor\Students\STDB\Term2\example2-1bin.py (3.7.2)*
File Edit Format Run Options Window Help
from sklearn.metrics import confusion_matrix
print(confusion_matrix(test_y, res))
Ln: 34 Col: 0
```

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(test_y, res))
```

Результат получается транспонированным!

## Пример 5. Результат

---

```
[ [7957      8]  
  [ 20     15] ]
```

Как видно из результата только 15 из 35 мошеннических транзакций были опознаны верно.

# Для нашей задачи

---

Обозначим обычную транзакцию за 1, а мошенническую за 0.

	$y = 1$	$y = 0$
$y' = 1$	7957 (Истинно положительные) (True Positive) TP	20 (Ложно положительные) (False Positive) (FP)
$y' = 0$	8 (Ложно отрицательные) (False Negative) (FN)	15 (Истинно отрицательные) (True Negative) (TN)

# Метрики

---

$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN)$  – доля правильно распознанных среди всех. Будем называть данный параметр «Правильность прогнозирования».

$\text{ERR} = (FP + FN) / (TP + FP + FN + TN)$  – доля ошибочно распознанных среди всех. Будем называть данный параметр «Ошибка прогнозирования».

$\text{Precision} = TP / (TP + FP)$  – доля правильно распознанных положительных среди всех принятых за положительные. Будем называть данный параметр «Точность».

$\text{TPR или Recall} = TP / (TP + FN)$  – доля истинно (правильно распознанных) положительных случаев среди всех бывших по настоящему положительными. Будем называть данный параметр «Полнота».

$\text{FPR} = FP / (FP + TN)$  – доля ложноположительных (правильно распознанных отрицательных) случаев среди всех бывших по настоящему отрицательными.



# Метрики

---

$$F_{\beta} = (1 + \beta^2) \times \text{Precision} \times \text{Recall} / (\beta^2 \times \text{Precision} + \text{Recall})$$

При  $\beta = 1$   $F_1$  превращается в среднее гармоническое

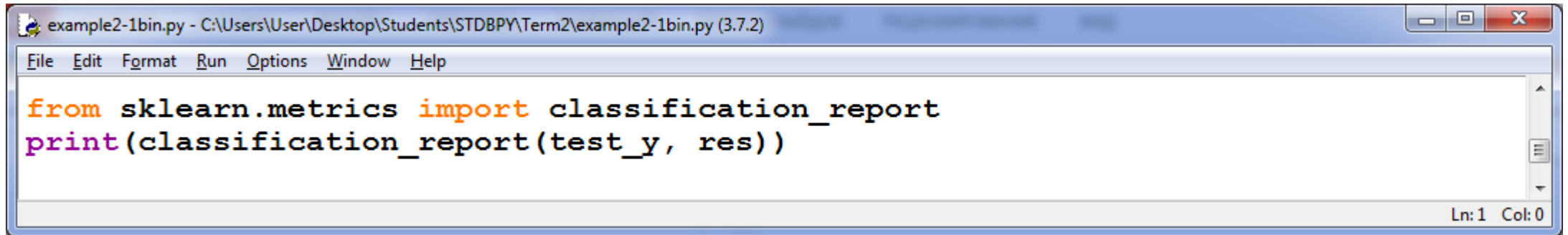
$$F_1 = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall})$$

МСС - коэффициент корреляции Мэтьюза (Matthews Correlation Coefficient, MCC):

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

# Расчёт метрик

---



The screenshot shows a Python IDE window titled "example2-1bin.py - C:\Users\User\Desktop\Students\STDBPY\Term2\example2-1bin.py (3.7.2)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

```
from sklearn.metrics import classification_report  
print(classification_report(test_y, res))
```

The status bar at the bottom right indicates "Ln: 1 Col: 0".

```
from sklearn.metrics import classification_report  
print(classification_report(test_y, res))
```

# Результат

---

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7965
1	0.65	0.43	0.52	35
accuracy			1.00	8000
macro avg	0.82	0.71	0.76	8000
weighted avg	1.00	1.00	1.00	8000

# Вероятностные значения

---

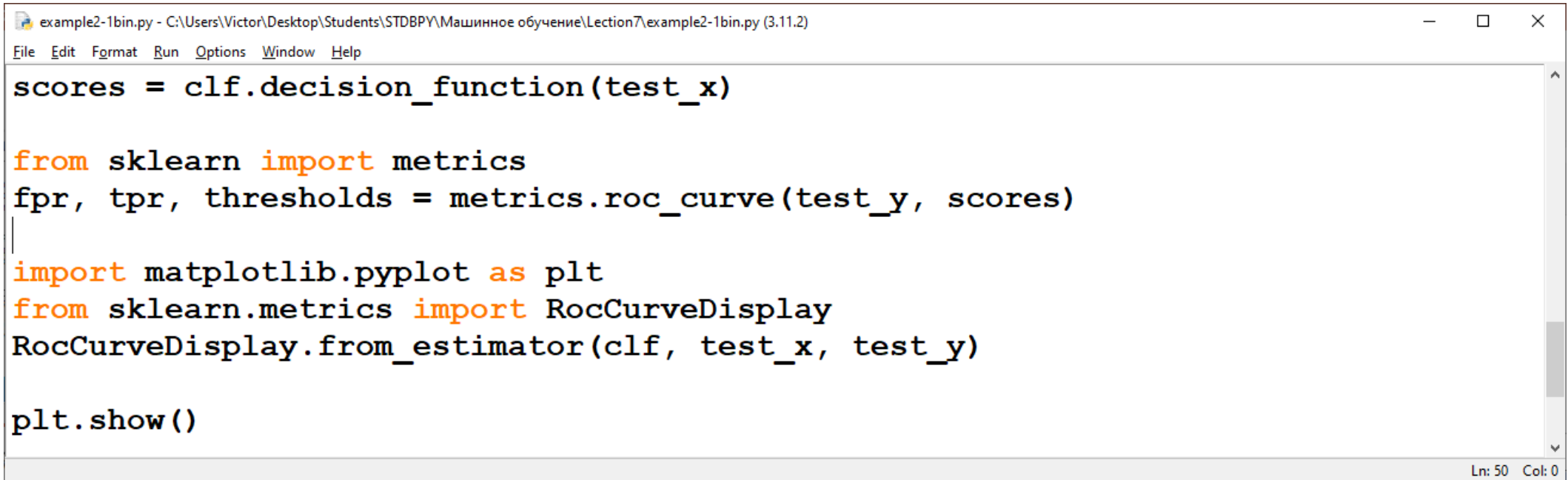
Если классификатор вместо самих классов 0 и 1 будет выдавать вероятность принадлежности классу 1 (от 0 до 1), то для принятия решения, к какому классу отнести строчку, нужно будет вести порог, при достижении которого класс принимается равным 1. Порог может быть разным в зависимости от задачи. Например, можно взять средний порог 0,5 или завышенный порог 0,8 и т.д.

В зависимости от значения порога TPR и FPR будут меняться. График зависимости TPR от значений FPR при изменяющемся от 0 до 1 значении порога называют ROC-кривой (англ. receiver operating characteristic, рабочая характеристика приёмника или характеристическая кривая обнаружения).

Если классификатор идеален, то ROC-кривая идёт строго по сторонам квадрата. Если классификатор не распознает ничего, то ROC-кривая идёт по диагонали.

Площадь части квадрата  $((0, 0), (1, 1))$ , расположенной под ROC-кривой называют AUC-ROC метрикой (англ. area under curve) или просто AUC-метрикой.

# Пример 5. Расчёт ROC-кривой и построение графика

A screenshot of a Python IDE window. The title bar shows the file path: "example2-1bin.py - C:\Users\Victor\Desktop\Students\STDBPY\Машинное обучение\Lecture7\example2-1bin.py (3.11.2)". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains the following Python code:

```
scores = clf.decision_function(test_x)

from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(test_y, scores)

import matplotlib.pyplot as plt
from sklearn.metrics import RocCurveDisplay
RocCurveDisplay.from_estimator(clf, test_x, test_y)

plt.show()
```

The status bar at the bottom right indicates "Ln: 50 Col: 0".

# Пример 5. Расчёт ROC-кривой и построение графика

```
scores = clf.decision_function(test_x)

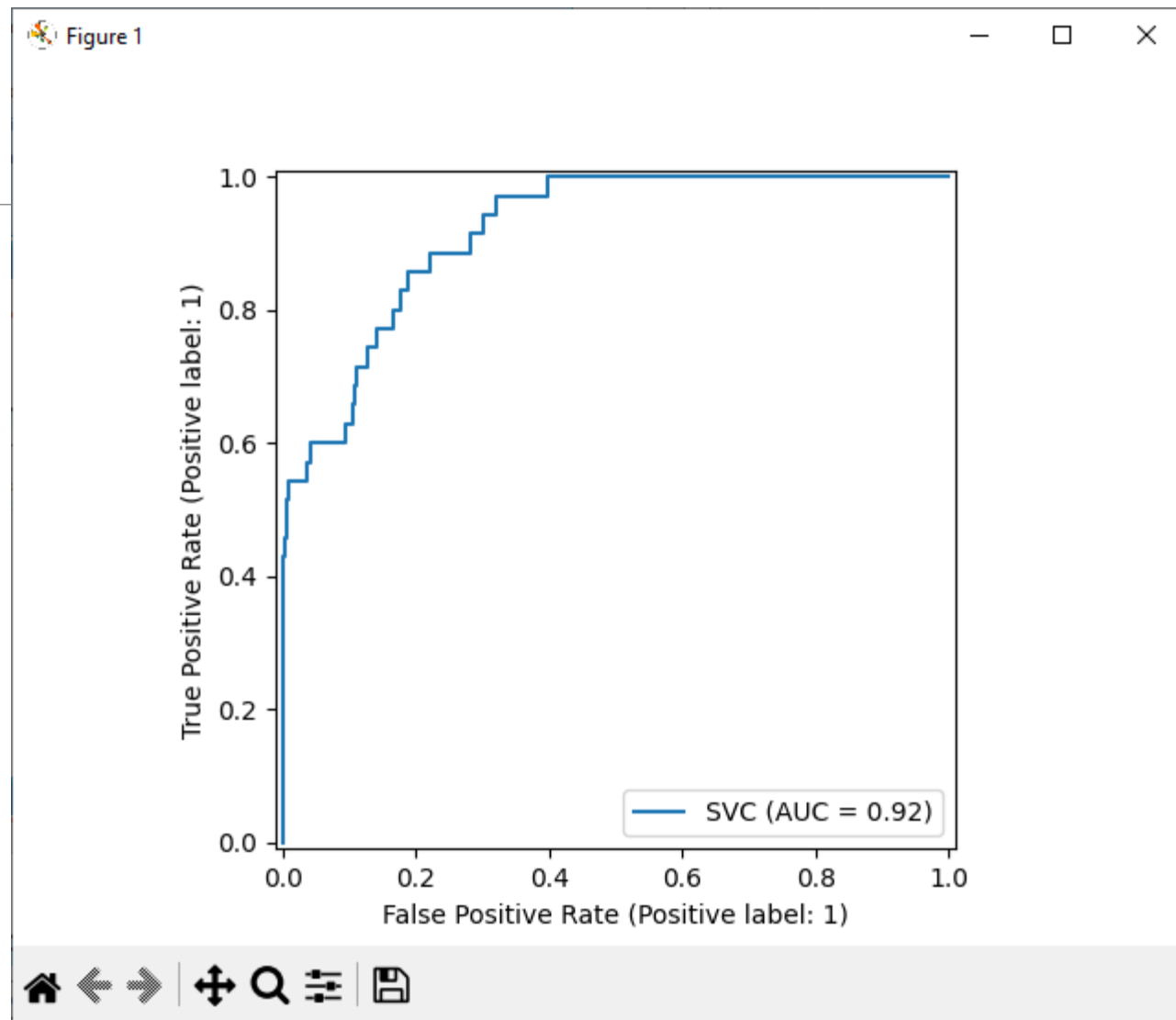
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(test_y, scores)

import matplotlib.pyplot as plt
from sklearn.metrics import RocCurveDisplay
RocCurveDisplay.from_estimator(clf, test_x, test_y)

plt.show()
```

# Результат

На графике мы можем сразу видеть и значение AUC-метрики (0,92).



# Precision-Recall-кривая

---

Метрика AUC-ROC устойчива к несбалансированным классам. Она показывает вероятность того, что случайно выбранное положительное наблюдение будет скорее помечено, как положительное, чем случайно выбранное отрицательное наблюдение.

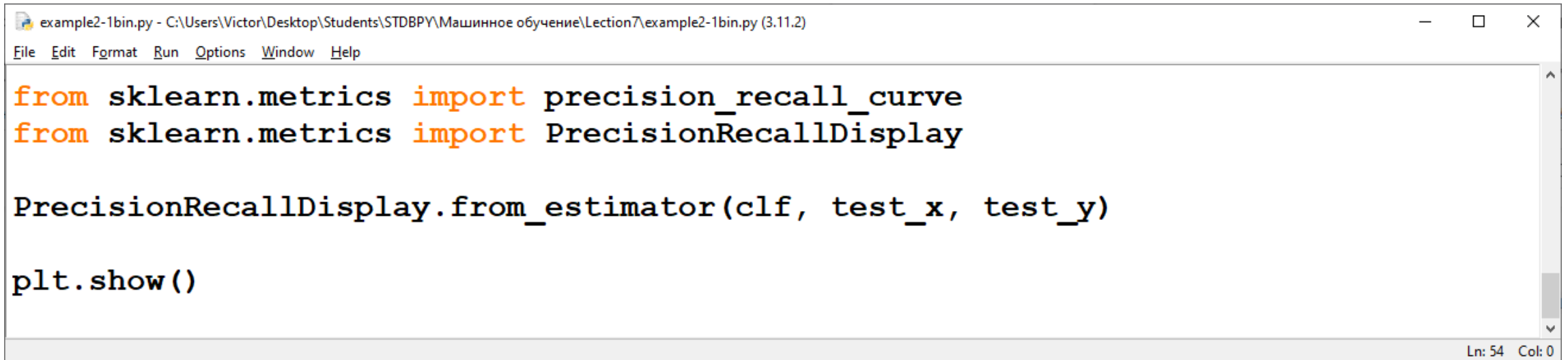
AUC-ROC метрика может дать плохую оценку для ситуации, когда мы сталкиваемся с большим числом наблюдений, неправильно обозначенных положительными, при маленьком положительном классе. В этом случае лучше использовать кривую Precision-Recall и метрику AUC-PR.

Если классификатор идеален, то кривая пройдёт по верхнему и правому ребрам квадрата. Если классификатор не лучше случайного угадывания, то мы получим горизонтальную прямую.



# Пример 5. Расчёт Precision-Recall-кривой и построение графика

---



```
example2-1bin.py - C:\Users\Victor\Desktop\Students\STDBPY\Машинное обучение\Lecion7\example2-1bin.py (3.11.2)
File Edit Format Run Options Window Help

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import PrecisionRecallDisplay

PrecisionRecallDisplay.from_estimator(clf, test_x, test_y)

plt.show()
```

Ln: 54 Col: 0

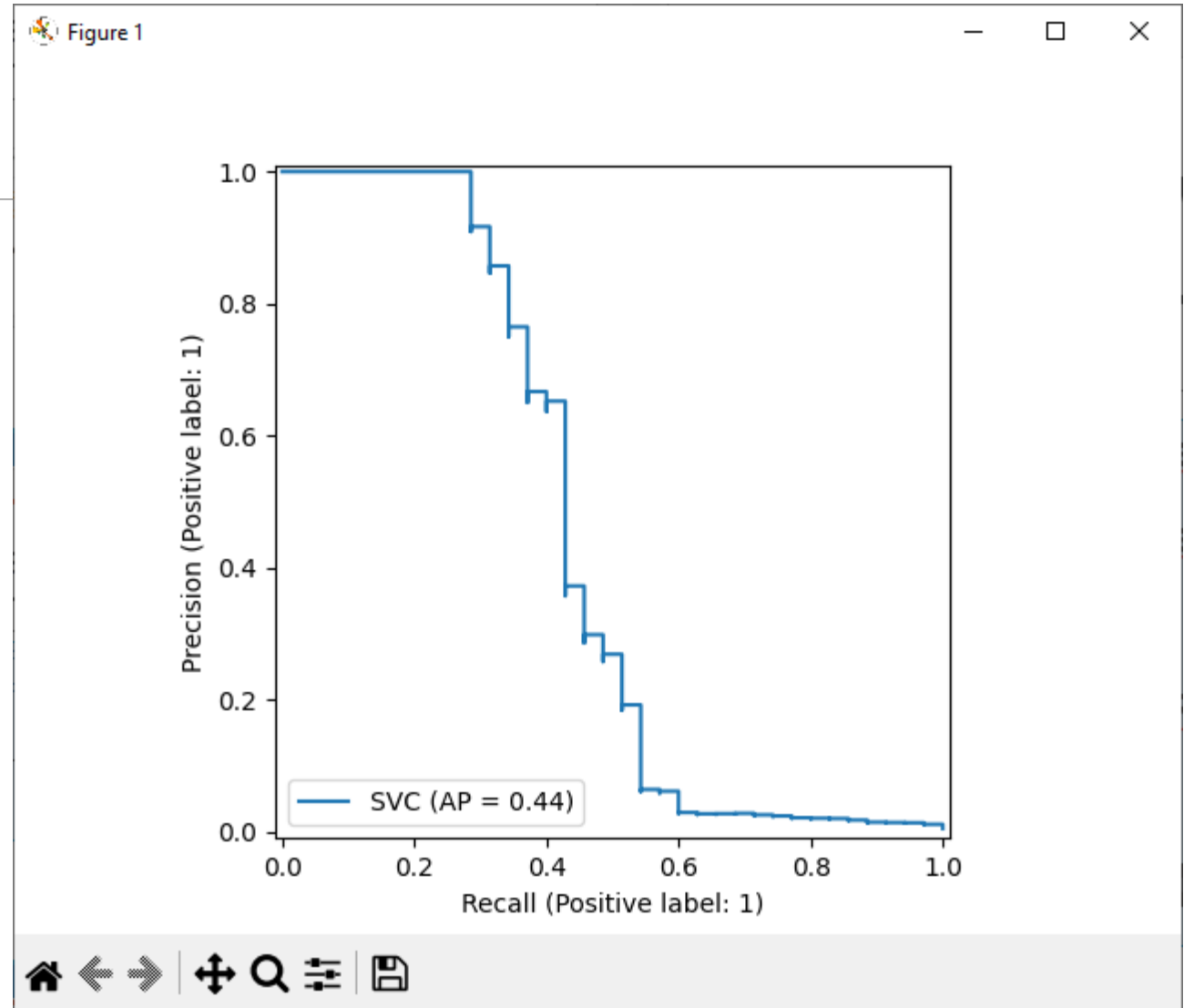
# Пример 5. Расчёт Precision-Recall-кривой и построение графика

---

```
from sklearn.metrics import precision_recall_curve  
from sklearn.metrics import PrecisionRecallDisplay  
  
PrecisionRecallDisplay.from_estimator(clf, test_x, test_y)  
  
plt.show()
```

# Результат

На графике мы можем сразу видеть и значение AUC-PR-метрики (0,44).



# Что делать, если классов больше 2-х?

---

Все рассмотренные метрики были сформулированы, исходя из бинарной классификации. А если классов больше двух? В этом случае рассматривают противопоставление одного класса другим по каждому из классов, а затем усредняют.

Причем бывает два вида усреднения: микро и макро-усреднения.

При микро-усреднении сначала усредняются значения характеристик TP, TN, FP, FN и т.д., а затем рассчитывается метрика. При макро-усреднении усредняются сами значения метрик, рассчитанных для каждого класса.

Для несбалансированных классов предпочтительно макро-усреднение.

# Пример 6

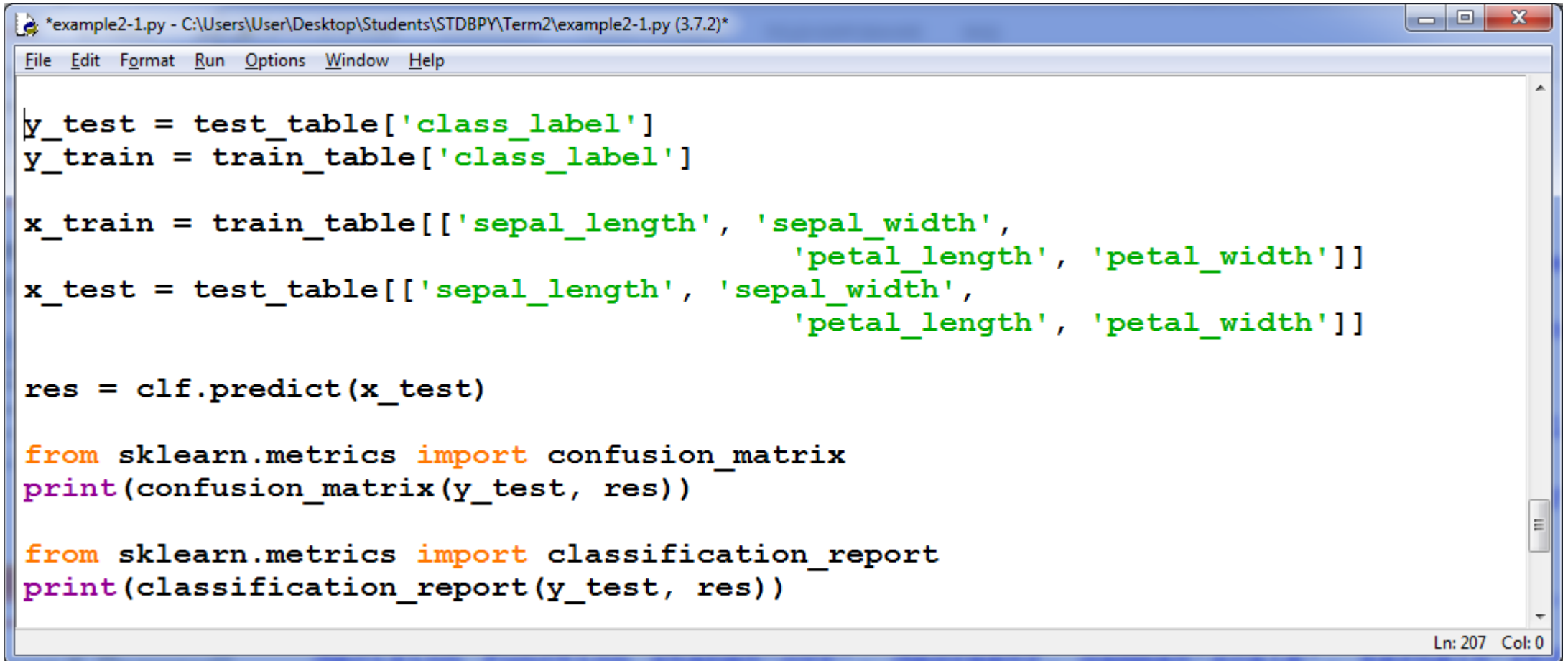
---

Вернёмся к примеру 4 из лекции, посвященной методу SVM. В нём мы в процессе оценки классификаторов получили SVM-классификатор с полиномиальным ядром. Пусть ему соответствует переменная `clf`.

Простейшие метрики `scikit-learn` умеет считать сам, а вот для ROC-AUC придётся что-то сделать.

Тогда используя `OneVsRestClassifier` модуль можно получить много бинарных классификаторов (три) и сделать оценку данного классификатора.

# Пример 6. Подготовка данных и простые метрики



```
*example2-1.py - C:\Users\User\Desktop\Students\STDBPY\Term2\example2-1.py (3.7.2)*
File Edit Format Run Options Window Help

y_test = test_table['class_label']
y_train = train_table['class_label']

x_train = train_table[['sepal_length', 'sepal_width',
                           'petal_length', 'petal_width']]
x_test = test_table[['sepal_length', 'sepal_width',
                        'petal_length', 'petal_width']]

res = clf.predict(x_test)

from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, res))

from sklearn.metrics import classification_report
print(classification_report(y_test, res))

Ln: 207 Col: 0
```

# Пример 6. Подготовка данных и простые метрики

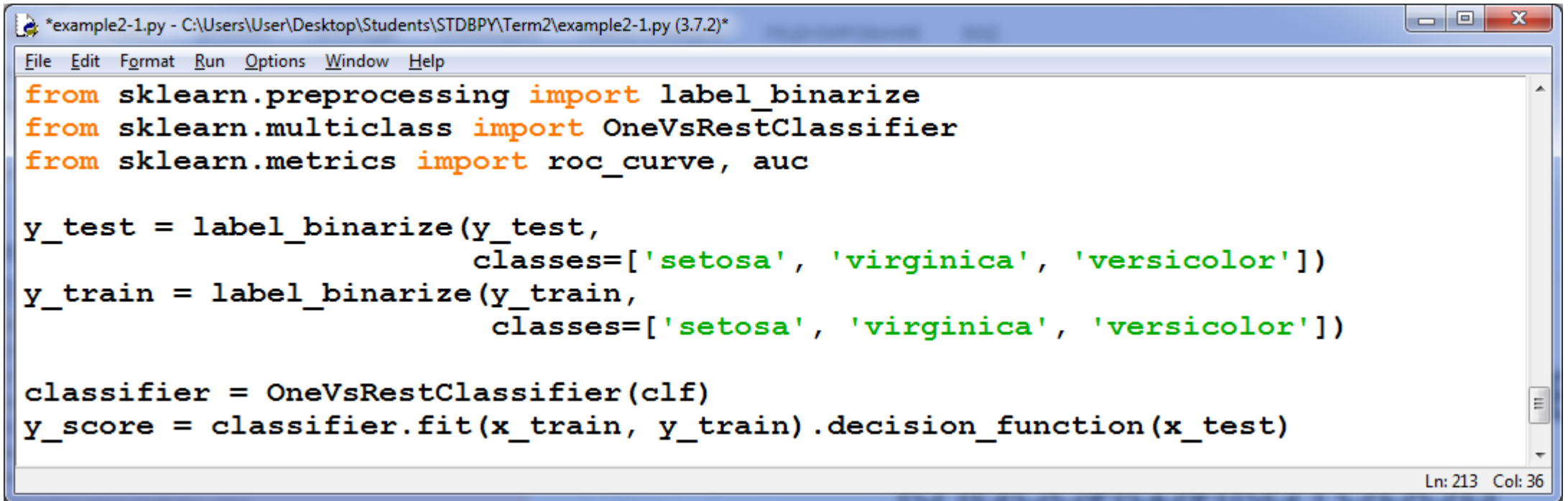
---

```
y_test = test_table['class_label']
y_train = train_table['class_label']
x_train = train_table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
x_test = test_table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
res = clf.predict(x_test)

from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, res))

from sklearn.metrics import classification_report
print(classification_report(y_test, res))
```

# Пример 6. Подготовка 3-х бинарных классификаторов

A screenshot of a Python script editor window. The title bar shows the file path: \*example2-1.py - C:\Users\User\Desktop\Students\STDBPY\Term2\example2-1.py (3.7.2)\*. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code is as follows:

```
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import roc_curve, auc

y_test = label_binarize(y_test,
                        classes=['setosa', 'virginica', 'versicolor'])
y_train = label_binarize(y_train,
                        classes=['setosa', 'virginica', 'versicolor'])

classifier = OneVsRestClassifier(clf)
y_score = classifier.fit(x_train, y_train).decision_function(x_test)
```

The status bar at the bottom right indicates 'Ln: 213 Col: 36'.



# Пример 6. Подготовка 3-х бинарных классификаторов

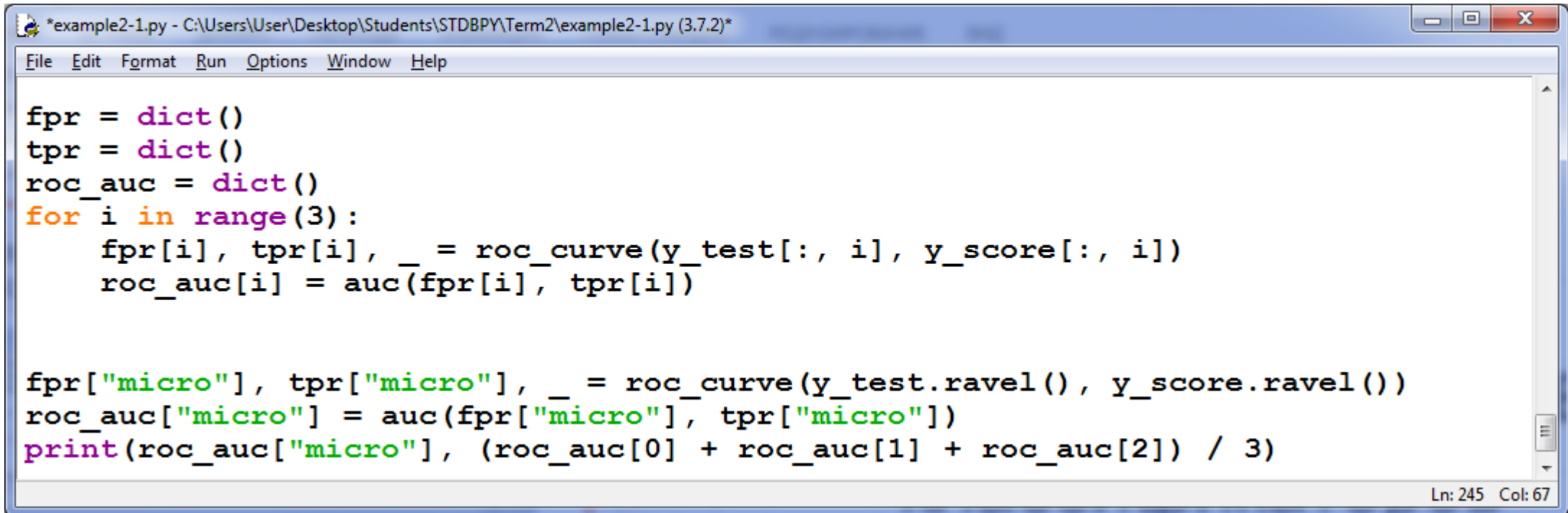
---

```
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import roc_curve, auc

y_test = label_binarize(y_test, classes=['setosa', 'virginica', 'versicolor'])
y_train = label_binarize(y_train, classes=['setosa', 'virginica', 'versicolor'])

classifier = OneVsRestClassifier(clf)
y_score = classifier.fit(x_train, y_train).decision_function(x_test)
```

# Пример 6. Расчёт микро и макро ROC-AUC

A screenshot of a Python IDE window titled '\*example2-1.py - C:\Users\User\Desktop\Students\STDBPY\Term2\example2-1.py (3.7.2)\*'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is as follows:

```
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(3):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
print(roc_auc["micro"], (roc_auc[0] + roc_auc[1] + roc_auc[2]) / 3)
```

The status bar at the bottom right shows 'Ln: 245 Col: 67'.

# Пример 6. Расчёт микро и макро ROC-AUC

```
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(3):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
print(roc_auc["micro"], (roc_auc[0] + roc_auc[1] + roc_auc[2]) / 3)
```

# Интернет ресурсы и литература

---

1. [https://scikit-learn.org/stable/auto\\_examples/plot\\_roc\\_curve\\_visualization\\_api.html#sphx-glz-auto-examples-plot-roc-curve-visualization-api-py](https://scikit-learn.org/stable/auto_examples/plot_roc_curve_visualization_api.html#sphx-glz-auto-examples-plot-roc-curve-visualization-api-py)
2. [https://scikit-learn.org/stable/tutorial/statistical\\_inference/model\\_selection.html](https://scikit-learn.org/stable/tutorial/statistical_inference/model_selection.html)
3. [https://scikit-learn.org/0.22/modules/generated/sklearn.model\\_selection.cross\\_validate.html](https://scikit-learn.org/0.22/modules/generated/sklearn.model_selection.cross_validate.html)
4. <https://scikit-learn.org/0.20/modules/generated/sklearn.metrics.auc.html#sklearn.metrics.auc>
5. <https://habr.com/ru/company/ods/blog/328372/>
6. [https://github.com/esokolov/ml-course-msu/blob/master/ML15/lecture-notes/Sem05\\_metrics.pdf](https://github.com/esokolov/ml-course-msu/blob/master/ML15/lecture-notes/Sem05_metrics.pdf)